CHAPTER 6

RASTER SUBSYSTEM

The Raster Subsystem contains a Raster Engine chip which is responsible for performing low level pixel rendering operations. The Raster Subsystem also contains two hardware cursor chips which are used to draw block glyph or cross hair cursors. The Raster Engine is instructed by the Geometry Engine to draw lines and horizontal spans. The Geometry Engine also instructs the Raster Engine to perform pixel read and write DMA operations. The Raster Engine provides the control signals for reading and writing the pixel values into the frame buffer bitplanes, the Window ID (WID) bitplanes, the Pop Up (PUP) overlay bitplanes, the User Auxiliary (UAUX) bitplanes and the optional Z buffer bitplanes if installed.

The following sections describe the external interfaces, the major components, the registers, the basic operations and the programming considerations for the Raster Subsystem.

External Interfaces

The Raster Subsystem has external interfaces with the Host System (via the Host Interface Subsystem and the Geometry Subsystem), the Geometry Subsystem and the Display Subsystem. The following paragraphs describe these external interfaces.

Host Interface

The Raster Subsystem has an interface with the Host System over the Utility Bus. The host can read and write the cursor glyph and the cursor registers in the two cursor chips. The host can also have the Geometry Subsystem initiate bitplane pixel DMA operations over the Geometry data bus. The host software can use the GE_LOADRE token to load values into some of the Raster Engine registers.

Geometry Subsystem Interface

The Geometry Subsystem uses the REptr register in the HQ1 chip to address the various Raster Engine registers. The data is then written into or read from the Raster Engine registers using the Geometry data bus. The GE5 microcode can also initiate DMA transfers between the bitplanes and the GE5 data RAM or the host system RAM. The HQ1 chip in the Geometry Subsystem also controls the host data transfers to and from the registers in the two cursor chips.

Display Subsystem Interface

The Raster Subsystem receives timing control signals from the Display State Machine in the Display Subsystem. In response its bitplanes and cursor chips send data to the Display Subsystem to be converted to analog RGB signals for output to the high resolution color monitor.

Major Components

The major components of the Raster Subsystem are shown in the block diagram of Figure 6.1. The Raster Engine 2 (RE2) is a proprietary Silicon Graphics chip which performs the low level pixel rendering operations as instructed by the Geometry Subsystem. The Cursor chips are used to display two single color cursors or a single multicolor cursor. The cursor can be either a user defined glyph pattern or a cross hair pattern.

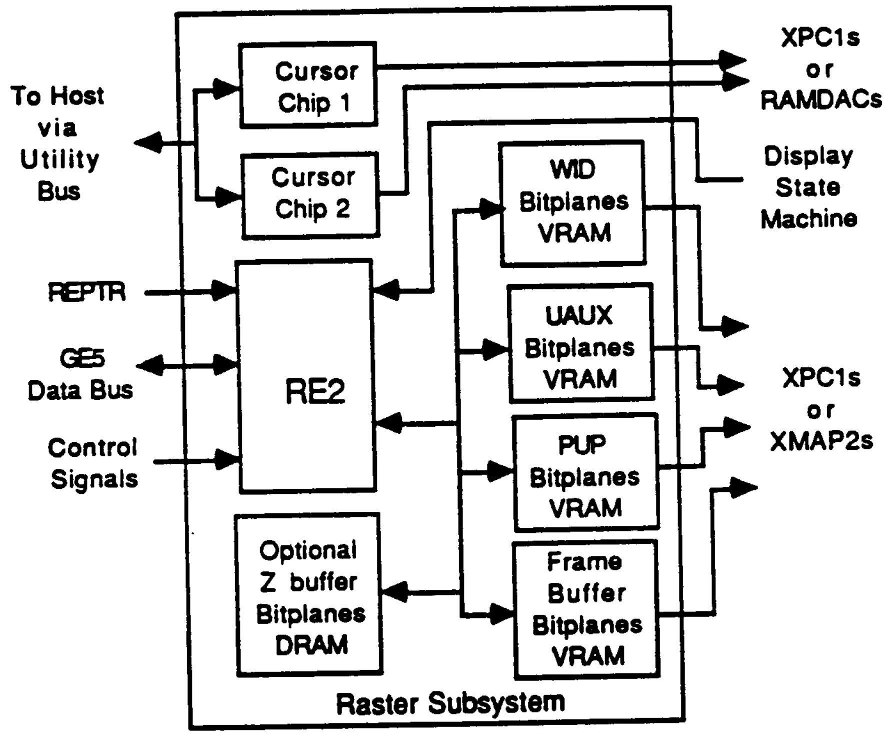


Figure 6.1 Raster Subsystem Block Diagram

The frame buffer bitplanes are used to store the pixel image to be displayed on the monitor. The Window ID (WID) bitplanes are used for pixel write clipping and to index into the mode registers in the XPC1 or XMAP2 chips in the Display Subsystem to control pixel display formats. The PUP and UAUX bitplanes are used to hold overlay and underlay pixel data. The PUP bitplanes are managed by the host window manager software and are used to draw menus and other overlays. The UAUX bitplanes are controlled by the various graphics applications to display overlays or underlays. The Z buffer bitplanes are used for doing Z axis depth comparison for hidden line removal.

The number of bitplanes available on the MGR adapter depends on the whether the base configuration or the enhanced configuration is being used. For the base configuration, the MDE1 daughter board is installed so the base configuration has 8 bitplanes of Frame Buffer VRAM, 2 bitplanes of WID VRAM and 2 bitplanes of PUP VRAM. For the enhanced configuration the MEV2 daughter board is installed so the enhanced configuration has 24 bitplanes of Frame Buffer VRAM, 4 bitplanes of WID VRAM, 2 bitplanes of PUP VRAM and 2 bitplanes of UAUX VRAM. Both the base and enhanced configuration can have the optional MZB1 card installed with the 24 bitplanes of Z buffer DRAM. If the MZB1 daughter card is not installed then no Z buffer hidden line removal is available.

The following paragraphs describe the major components of the Raster Subsystem.

Raster Engine 2 (RE2)

The RE2 performs scan conversion of lines and spans into pixels and controls all memory timings for writing these pixel values into the pixel bitplanes. The GE5, in the Geometry Subsystem, loads a bank of registers on the RE2 chip to indicate what kind of drawing operation is desired. The RE2 then iterates the x,y and z values and the red, green and blue colors to form the individual pixels values which it then writes into the pixel bitplanes. The RE2 can flat shade or Gouraud shade RGB or color index values while at the same time performing pixel write conditioning checks and pixel write mask operations. The RE2 can apply stipple patterns as it draws antialiased lines. The RE2 can also apply patterns as it draws spans. The RE2 provides support for 8, 12 and 24 bit RGB pixels and it also provides support for 4, 8 or 12 bit color index pixels. The RE2 also provides double buffer support for the 12 bit RGB and the 4 or 12 bit color index pixels. The color index pixels and the 12 bit RGB pixels can have a dithering operation performed on the color values which are used to calculate the Frame Buffer pixel values. The RE2 also provides support for 16 different raster operations which are performed on a bit by bit basis. The following paragraphs describe the main features of the RE2 chip.

RE2 Architecture

The RE2 consists of five major units as shown in the block diagram of Figure 6.2. The following paragraphs give a brief description of these five units.

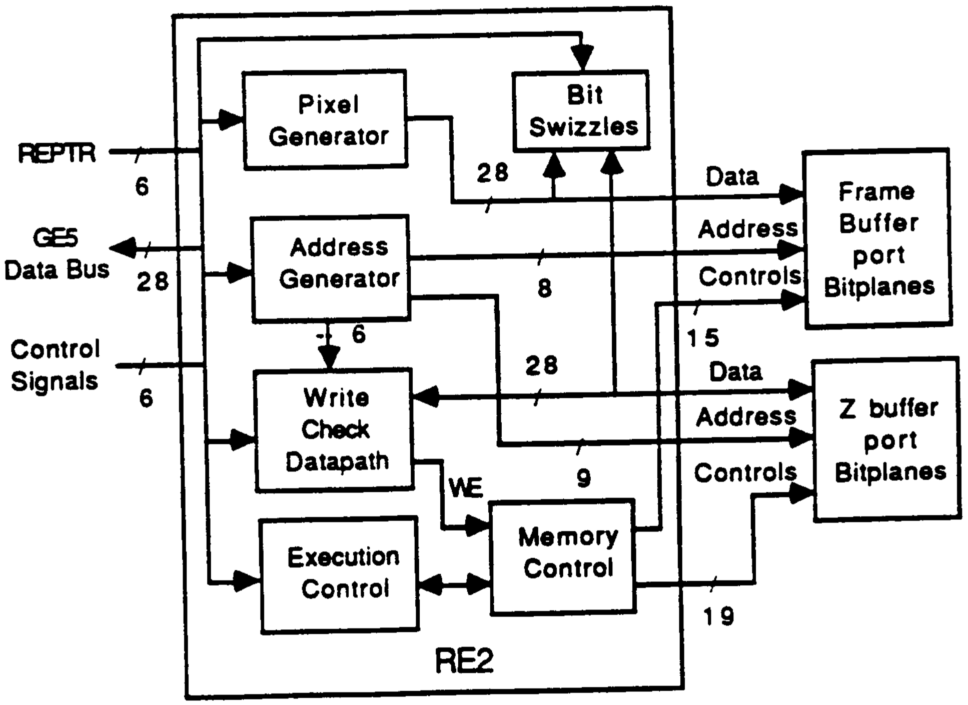


Figure 6.2 RE2 Block Diagram

The Pixel Generator consists of three color components interpolators, a dither matrix and a pixel packing multiplexer. The color component interpolaters contain a digital differential analyzer (DDA) and a dithering circuit. The interpolator for red is also used for the color index values. This unit is responsible for calculating the Frame Buffer pixel values. These values are then used as part of the source data for the Raster operations. The color component interpolators add delta color values to the current color values and then performs the dithering calculations. The resulting color values are then packed based on the current pixel type and is placed in the source buffer for the Raster Operation. The result is then written into the Frame Buffer bitplanes.

The Address Generator consists of 2 DDAs, one for X and one for Y. This unit generates the addresses used to access the various bitplanes. The address generator adds the delta x and y values to the current x and y values to form the pixel address of the next pixel to be accessed.

The Write Check Datapath unit consists of the DDA used for calculating Z values. This unit also contains the circuits for Z value comparisons, for window ID checks and for pattern and stipple checks. The delta Z value is added to the current z value to obtain the new z value. The new z value is then compared to the current z value in the Z buffer bitplanes at the x and y location. If the comparison fails the pixel is not written. The WID checking hardware is used to compare a current WID value with the WID value in the WID bitplanes at the x and y location. If the WID check is enabled and the WID comparison fails then the pixel is not written. The pattern checking can be used to enable or disable pixel writes for shaded spans. The stipple checks can be used to enable or disable pixel writes for lines.

The Execution Control unit determines the type of operation that the Raster Engine is performing. Possible operations include pixel write, Z write, memory refresh and display refresh. This unit controls the operation of the Memory Control unit.

The Memory Control unit generates the memory control signals for the various bitplanes.

The bit swizzles unit is used to shift the various pixel data bits left into the appropriate data bit locations for writing into the bitplanes. It also shifts the bits right as needed to right justify the bits which are read from the various bitplanes.

The RE2 basic operations section of this chapter discusses the operations performed by the RE2 in much greater detail.

Instruction Set

The RE2 has an Instruction Register and 56 microcode visible control registers. The GE5 microcode loads instructions into the Instruction Register to specify the operation that the Raster Engine will perform next. The 56 control registers are loaded with the data needed to perform the operation. When the Instruction Register is loaded by the Geometry Engine, the values in the control registers are loaded into the Raster Engine's execution units and the new instruction begins executing.

The control registers that are frequently loaded have input buffers. These control registers can be loaded while the Raster Engine is drawing. The control registers without input buffers can be loaded only when the Raster Engine is idle. The sequence of steps for issuing an instruction is:

1. Update control registers that have input buffers
2. If necessary, update control registers that do not have input buffers
3. Store instruction opcode in the Instruction Register

This sequence of steps gives the maximum overlap between Raster Engine execution and Geometry Engine execution. The GE5 microcode should try to update control registers that do not have input buffers only while the RE2 is idle. If the RE2 is not idle, the microcode will be stalled until the RE2 becomes idle.

The Registers section of this chapter defines the registers and the RE2 basic operations section of this chapter describes the use of the RE2 instructions and the control registers to perform the RE2 operations.

Pixel Bitplanes

The MGR uses 1 Megabit VRAM chips to create the frame buffer bitplanes, PUP bitplanes, UAUX bitplanes and WID bitplanes. The MGR uses 1 Megabit DRAM chips to create the Z buffer bitplanes. The component timing restrictions require the VRAM and DRAM chips to be organized into five pixel pipelines to create a 1280 x 1024 pixel screen display. The pixels on each scan line are organized in an interleaved manner in groups of five pixels. The first pixel in each group of five pixels is contained in VRAM 0. The second pixel in each group of five pixels in is contained in VRAM 1. The third pixel in each group of five pixels is contained in VRAM 2. The fourth pixel in each group of five pixels is contained in VRAM 3 and finally the fifth pixel in each group of five pixels is contained in VRAM 4. Each VRAM chip provides 256 x 1024 x 4 pixels. The Z buffer pixels are organized in the same manner. The x = 0 and y = 0 pixel location is defined to be the lower left corner of the screen. The RE2 chip controls the storage of pixels into the VRAM chips and also the transfer of pixels out of the VRAM chips to the Display Subsystem.

The RE2 TOPSCAN register is used to specify the top scan line number to be displayed. This allows the VRAM configuration to support multiple different screen formats. The MGR adapter has four different display timings built into the Display State Machine (DSM) in the Display Subsystem. The four timings include the 1280 x 1024 noninterlaced and interlaced timings, the NTSC timing and the PAL timing. The TOPSCAN register must be set appropriately by the host software for each of the different timings.

The DSM controls the display timing for transfering pixels out of the VRAM to the Display Subsystem for display on the attached monitor. For each line on the raster display the DSM issues an transfer request to the RE2. The RE2 then uses the value in the TOPSCAN register to transfer the top scan line in the five VRAMs into their internal shift registers. The TOPSCAN register also contains the number of columns on each scan line. The DSM then generates a shift clock to the VRAM for each pixel on the scan line. Each shift clock pulse causes a pixel to be shifted out of the appropriate VRAM to the display subsystem for display. This process is repeated for each scan line on the screen. After the RE2 causes the current row to be transferred into the shift register in the VRAM it then decrements the TOPSCAN register so that it can access the next scan line row down the screen. After the bottom scan line on the screen has been displayed the DSM generates a vertical retrace to the monitor. After the vertical retrace period has completed the next raster image is scanned in the same manner as before.

The Z Buffer pixels are used only by the RE2 when it performs the Z value depth comparisons and the Z values are not shifted out to the Display Subsystem. This allows the Z buffer pixels to be stored in DRAM chips rather than the dual ported VRAM chips. The following paragraphs discuss the bitplane layouts in the base and the enhanced adapters.

Base Adapter Pixel Bitplanes

The base configuration of the MGR adapter contains 8 Frame Buffer bitplanes, 2 PUP bitplanes and 2 WID bitplanes as shown in Figure 6.3. These bitplanes are contained in fifteen VRAM chips. As described above the VRAMs are interleaved in five pixel groups. Each VRAM provides 256 x 1024 x 4 pixels. The 2 PUP bitplanes and the 2 WID bitplanes are grouped in the same five VRAMs. The chip layout shown in Figure 6.3 does not show the five pixel interleaving well. As described above, each VRAM contains every fifth pixel on each scan line. The first pixel is in VRAM 0 and the fifth pixel is in VRAM 4. This is repeated for each group of five pixels on each scan line.

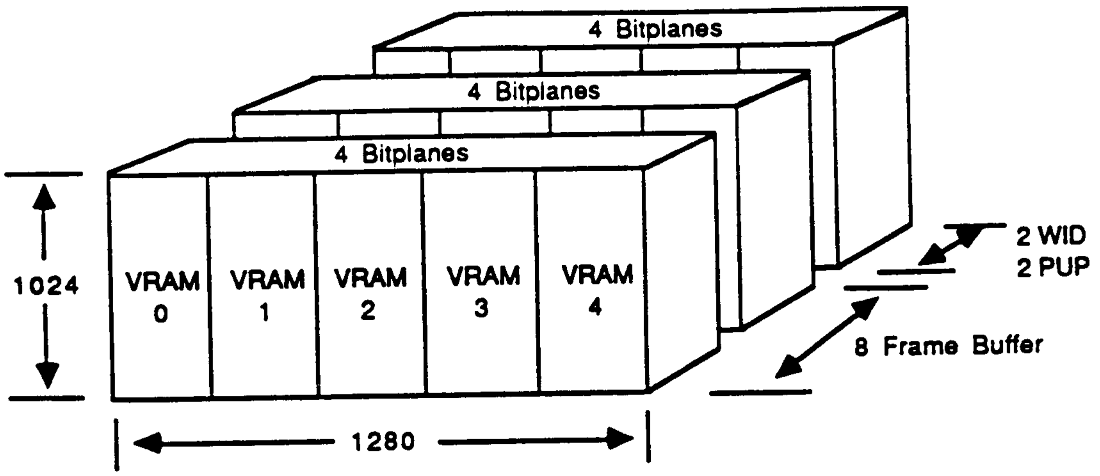


Figure 6.3 Base Configuration VRAM Layout

The base configuration of the adapter can also have an optional 24 bit Z buffer card installed. The Z buffer values are stored in DRAM chips rather than in VRAM chips. The layout of the Z Buffer is shown in Figure 6.4. The Z buffer DRAM chips are organized in the same five pixel interleaving that is used for the VRAM chips. Each DRAM chip contains 256 x 1024 Z values. Each DRAM chips provides 4 bits of the 24 bit Z value. This means that 30 DRAM chips are required for the Z Buffer.

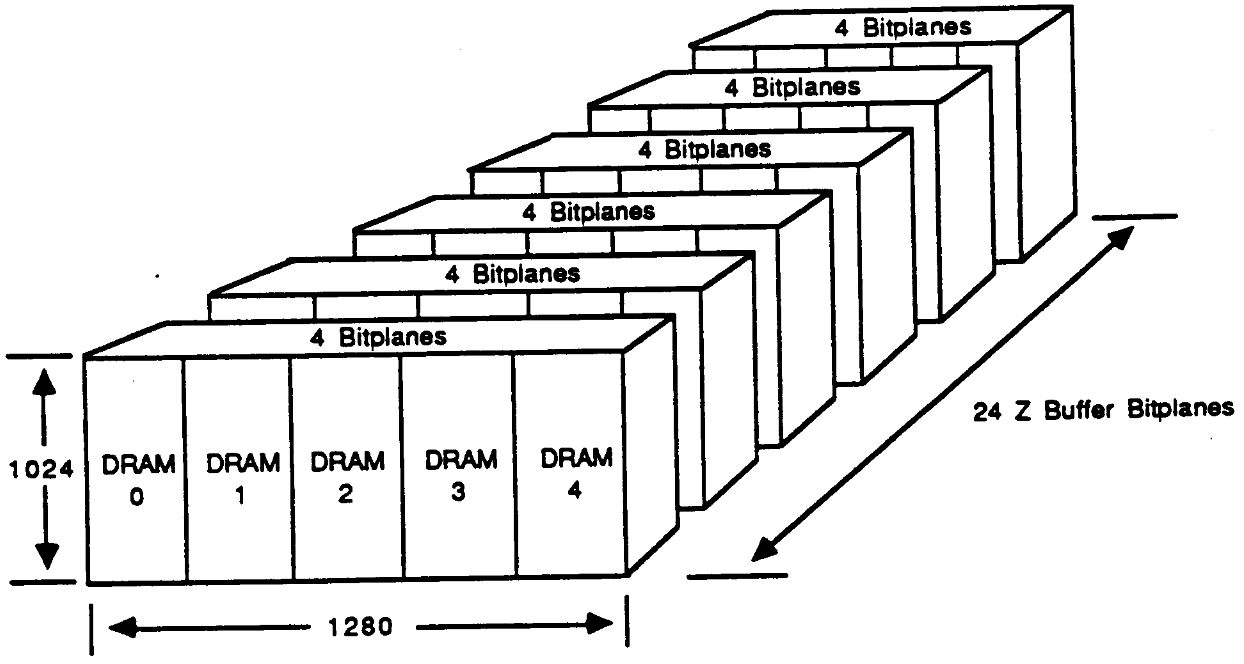


Figure 6.4 Z Buffer DRAM Layout

Enhanced Adapter Pixel Bitplanes

The enhanced configuration of the MGR adapter contains 24 Frame Buffer bitplanes, 2 PUP bitplanes, 2 UAUX bitplanes and 4 WID bitplanes as shown in Figure 6.5. These bitplanes are contained in 40 VRAM chips. As described above the VRAMs are interleaved in five pixel groups.

Each VRAM provides 256 x 1024 x 4 pixels. The 2 PUP bitplanes and the 2 UAUX bitplanes are grouped in the same five VRAMs.

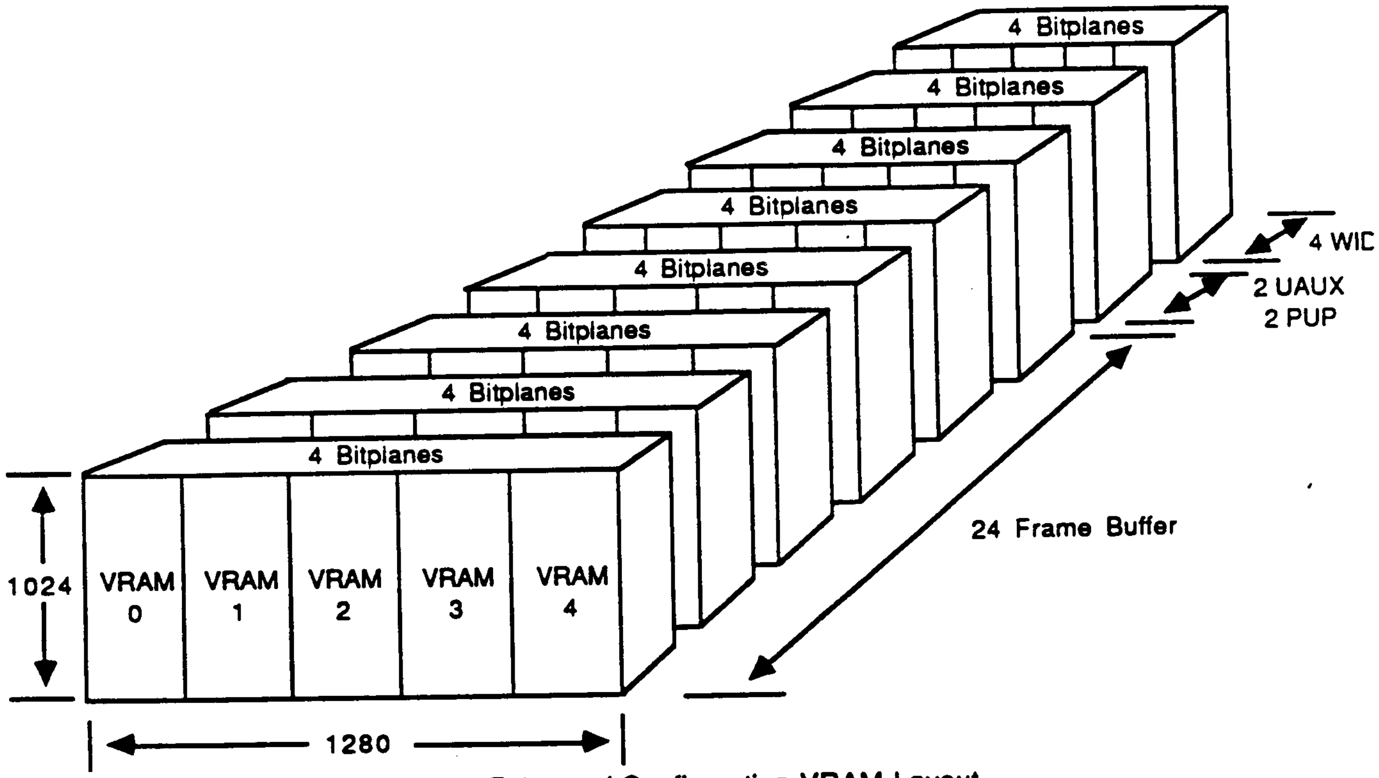


Figure 6.5 Enhanced Configuration VRAM Layout

Frame Buffer Bitplanes

The Frame Buffer bitplanes are used by the host software to hold the normal display pixels which are transferred to the Display Subsystem for display. The RE2 supports both color index and RGB pixel formats as well as single and double buffer formats. The following pixel types are supported:

- 4 bit color index double buffer (both configurations, usually only the base configuration)
- 8 bit color index single buffer (base configuration only)
- 12 bit color index double buffer (enhanced configuration only)
- 8 bit RGB single buffer (base configuration only)
- 12 bit RGB double buffer (enhanced configuration only)
- 24 bit RGB single buffer (enhanced configuration only)

The GE_PIXTYPE token is used to specify the pixel type. For the color index pixels the GE_COLOR or GE_COLORF token is used to specify the color index value. For RGB pixels the GE_RGBCOLOR token is used to specify the red, green and blue color values which form the RGB pixels. The GE_PIXWRITEMASK is used to specify the Frame Buffer pixel write mask. This mask is used to control which bits in the pixel are written. The host software uses the write mask to control whether the front or back buffer is written for double buffer pixels.

PUP Bitplanes

The PUP bitplanes are normally used by the host window manager software to display overlay or underlay pixels. The value written into the PUP bitplanes is used by the Display Subsystem as an index into the overlay color map. The GE_COLOR or GE_COLORF tokens are used to specify the index value. The GE_AUXWRITEMASK token is used to specify the write mask for the PUP bitplanes. Bits 0 and 1 of the aux write mask are used to mask PUP writes.

UAUX Bitplanes

The UAUX bitplanes are normally used by the user graphics application software to display overlay or underlay pixels. The value written into the UAUX bitplanes is used by the Display Subsystem as an index into the overlay color map. The GE_COLOR or GE_COLORF tokens are used to specify the index value. The GE_AUXWRITEMASK token is used to specify the write mask for the UAUX bitplanes. Bits 2 and 3 of the aux write mask are used to mask UAUX writes. The host software can configure the PUP and UAUX bitplanes as 4 UAUX bitplanes.

WID Bitplanes

The Window ID bitplanes are used by the host window manager to control pixel clipping for obscured windows. The LSB bit of the WID on the enhanced adapter is also used to perform Z value invalidation for fast Z clear operations. The Window ID is used by the Display Subsystem as an index into the mode registers of the XPC1 or XMAP2 chips to control pixel formatting operations during the pixel display operations. The GE_COLOR or GE_COLORF tokens are used to specify the window ID value. The GE_AUXWRITEMASK token is used to specify the write mask for the WID bitplanes. Bits 4 through 7 of the aux write mask are used to mask WID bitplane writes.

Z Buffer Bitplanes

The Z buffer bitplanes are used to hold Z values for each pixel. The x and y locations of the pixel are used to select which pixel is written and the z value can be used to perform depth comparison operations. If Z buffer comparisons are enabled then the new Z value is compared to the existing Z value in the bitplanes. If the comparison passes then the new pixel value is written into the Frame Buffer bitplanes, the PUP bitplanes, the UAUX bitplanes and the WID bitplanes subject to the settings of the pixel and aux write masks. The aux write mask bit 8 is used to enable or disable the Z buffer writes. The GE_ZBUFFER token is used to enable or disable Z buffer checks. The GE_ZFUNCTION token is used to specify the comparison function. The GE_AUXWRITEMASK is used to specify the write mask.

Cursor Chips

The Raster Subsystem uses two Brooktree Bt431 cursor controller chips to generate a cursor which can be one of three colors or two separate 1 color cursors. The cursor can be either a user defined 64 x 64 bit glyph pattern or a cross hair pattern. Both the glyph pattern and the cross hair cursor pattern can be displayed simultaneously, with logical OR and exclusive-OR operations supported. Either cursor may be moved off the top, bottom, left or right side of the screen without wrap-around. The cross hair cursor may be implemented as a full screen or full window cross hair cursor.

The cursor controller chip block diagram is shown in Figure 6.6. The cursor chip contains an address register and other control registers. The address register is used to select the other registers and the 64 x 64 bit RAM. The cursor chip contains Cursor X and Cursor Y registers which are loaded with the screen location at which the cursor is to displayed. The Horizontal and Vertical counters count the HSYNC and VSYNC signals from the Display State Machine in the Display Subsystem. The comparators and control logic compare the counter values with the Cursor X and Cursor Y values and when they are equal it enables the 64 x 64 bit RAM and cross hair logic circuits to produce the appropriate outputs. The Format logic is used to generate from 1 to 5 output signals that are used to select cursor colors.

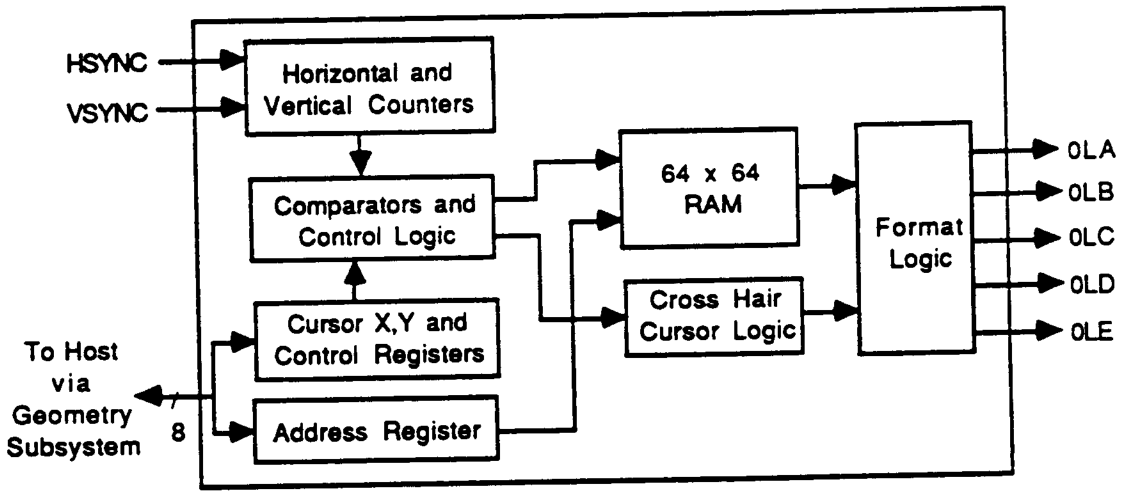

HSYNC
VSYNC
Horizontal and Vertical Counters
Comparators and Control Logic
Cursor X,Y and Control Registers
Address Register
64 x 64 RAM
Cross Hair Cursor Logic
Format Logic
OLA
OLB
OLC
OLD
OLE
To Host via Geometry Subsystem
8

Figure 6.6 Cursor Chip Block Diagram

The MGR uses all five cursor output signals. The connections of the outputs from the two cursor chips is shown in Figure 6.7. In the base configuration, the five outputs from both cursor chips are connected to the five XPC1 chips. The cursor inputs to the XPC1 chips have the highest precedence in determine which color value will be displayed. If either cursor output bit is a 1 then the OVL output of the XPC1 chip processing the current pixel will be set to the two cursor chip output values and the OVL bits will be used by the RGB RAMDAC chip to select the cursor color from the overlay palette. If both cursor outputs are zero then the XPC1 will use the contents of the Frame Buffer bitplanes and the PUP bitplanes to determine the pixel color. In the enhanced configuration, the five outputs from both cursor chips are connected directly to the OL inputs of the three RAMDAC chips. The RAMDACs use the cursor inputs to select a value from the overlay palette. If both cursor inputs are zero then the RAMDACs use the color input byte to select a color palette entry. Refer to the Display Subsystem for a detailed description of the use of the cursor output bits.

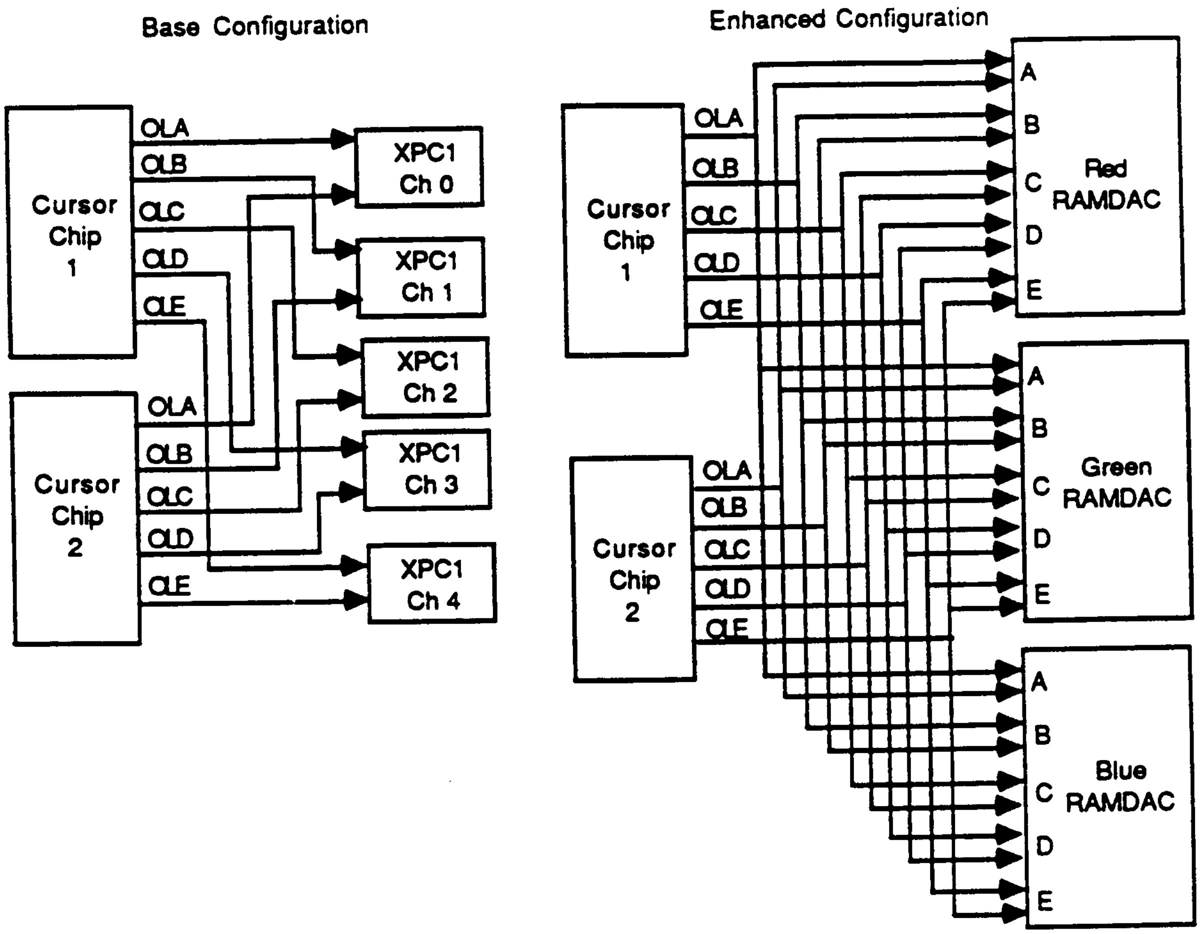


Figure 6.7 Cursor Output Connections

The following paragraphs describe the host access to the cursor chip registers and the cursor chip basic operations.

Host Access to the Cursor Chip Registers and Glyph RAM

The cursor controller chip contains Cursor X and Cursor Y registers which are used to position the center of the cursor on the screen. It also has an address register which is used to access the 64 x 64 RAM and the other control registers. The control registers include a command register, a window X register, a window Y register, a window width register and a window height register. The host address map for these registers and the cursor RAM is shown in Table 6.1. The addresses are the offset address from the base address of the MGR adapter.

Address Register

The address register is a 16 bit wide register which is used to access the other registers and the cursor glyph RAM in the cursor chip. The address register is written as two bytes called address register 0 and address register 1. The host software writes a byte to the addresses shown in Table 6.1 to place the address in the address register. Once the address register has been written the host can then read or write the register or glyph location whose address is in the address register.

Table 6.1 Cursor Host Address Map

Host Address Cursor Chip 0	Host Address Cursor Chip 1	Cursor Address Register 1-0 Value	Source/Destination
560 Hex	580 Hex	XX	Cursor Address Register 0
564 Hex	584 Hex	XX	Cursor Address Register 1
568 Hex	588 Hex	000 - 1FF hex	Cursor Glyph RAM
56C Hex	58C Hex	00	Command Register
56C Hex	58C Hex	01	Cursor X Low Register
56C Hex	58C Hex	02	Cursor X High Register
56C Hex	58C Hex	03	Cursor Y Low Register
56C Hex	58C Hex	04	Cursor Y High Register
56C Hex	58C Hex	05	Window X Low Register
56C Hex	58C Hex	06	Window X High Register
56C Hex	58C Hex	07	Window Y Low Register
56C Hex	58C Hex	08	Window Y High Register
56C Hex	58C Hex	09	Window Width Low Register
56C Hex	58C Hex	0A hex	Window Width High Register
56C Hex	58C Hex	0B hex	Window Heigth Low Register
56C Hex	58C Hex	0C hex	Window Height High Register

HQMMSB Register must be set to 1

Cursor Glyph RAM

Each cursor chip has a 512 byte RAM which is used to store a 64 x 64 bit cursor glyph. This is also referred to as a block cursor. The glyph contains the cursor pixel bit pattern used to form the block cursor. The host software must initialize the cursor glyph after the MGR adapter is reset. The host software loads the byte address in the address register and then reads or writes the RAM location. After the RAM location has been read or written the address register is incremented. This allows the host software to set the address register to 0 and then do 512 writes to fill the RAM. If the address is 0x1FF it wraps to 0x000 after the read or write operation. The glyph RAM layout is shown in Figure 6.8.

The bit 7 location of byte 0 is the upper left corner of the block cursor as it will be displayed on the screen. The 64 bits in the x direction are contained in 8 bytes. The 64 bits in the y direction are contained in 64 rows of 8 bytes.

The upper left corner of the glyph RAM is the 0, 0 position and the lower left corner of the screen is the 0, 0 position for the graphics drawing operations. This means that the host software must reverse the bit pattern of the glyph in the y direction from the way it would define other bit patterns.

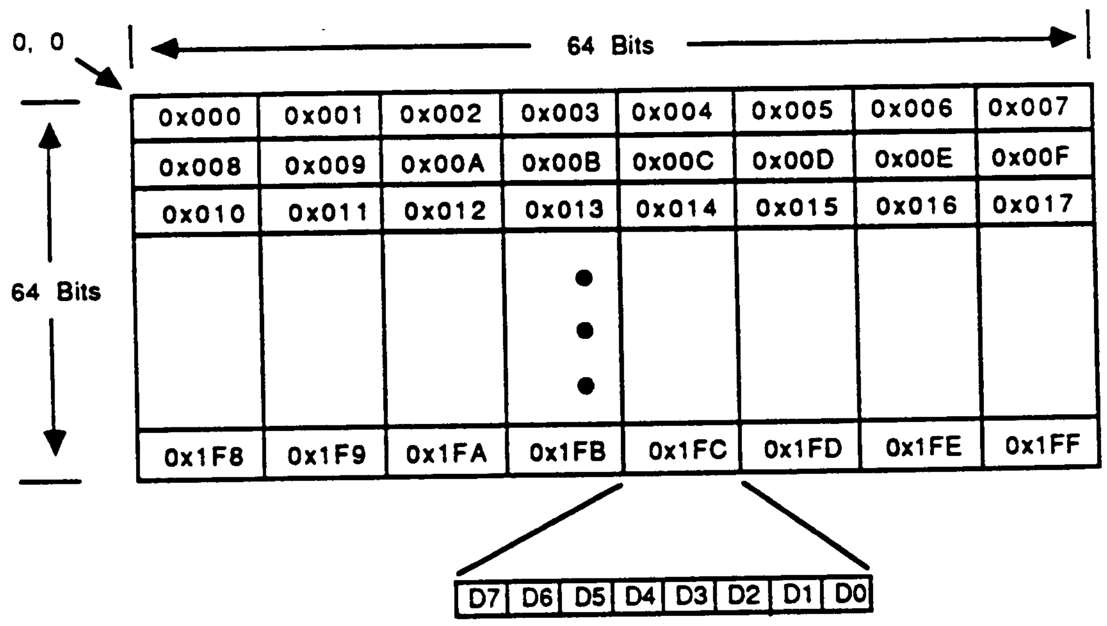


Figure 6.8 Cursor Glyph RAM Layout

Command Register

The command register is used by the host software to control the operation of the cursor chip. The command register is used enable either the block cursor or the cross hair cursor or both cursors. When both the block cursor and the cross hair cursor are enabled the command register is used to specify whether the intersecting pixels in the two cursors are logically ORed or exclusively ORed. The command register is also used to specify the pixel width of the cross hair cursor. The output multiplex control bits in the command register are used to select the number of output bits the cursor chip outputs on each clock cycle. The MGR adapter has the five pixel pipelines so the multiplex control bits should be set to for five outputs. The host reads or writes the command register by writing a 0 into the address registers and then reading or writing the cursor command register.

Cursor X,Y Registers

The cursor X,Y registers are used to specify the X and Y location of the center of the cursor glyph or the location where the cross hair cursor intersects. The cursor X and Y registers are 12 bit wide registers and are read or written as a low byte and a high byte. The host software writes a 1 into the address registers and then reads or writes the low byte of the cursor X register. The host then writes a 2 into the address registers and then reads or writes the high byte of the cursor X register. The host software writes a 3 into the address registers and then reads or writes the low byte of the cursor Y register. The host then writes a 4 into the address registers and then reads or writes the high byte of the cursor Y register. The new cursor location takes effect after the high byte of the cursor Y register is written.

Window X,Y Register

The window X,Y registers are used to specify the X and Y location of the upper left corner of the cross hair cursor window. The cross hair window is used to specify the cursor cross hair cursor clipping rectangle. The window width and height registers are used to specify the window size. The window X and Y registers are 12 bit wide registers and are read or written as a low byte and a high byte. The host software writes a 7 into the address registers and then reads or writes the low byte of the window Y register. The host then writes a 8 into the address registers and then reads or writes the high byte of the window Y register.

Window Width and Height Registers

The window width register is a 12 bit register which is used to specify the width of the cross hair cursor window. The host software writes a 9 into the address registers and then reads or writes the low byte of the window width register. The host then writes an 0xA into the address registers and then reads or writes the high byte of the window width register. The window height register is a 12 bit register which is used to specify the height of the cross hair cursor window. The host software writes an 0xB into the address registers and then reads or writes the low byte of the window height register. The host then writes an 0xC into the address registers and then reads or writes the high byte of the window height register. The new window size takes effect after the high byte of the window height register is written.

Cursor Chip Basic Operations

The cursor chip provides the capability to display either a 64 x 64 bit block cursor or a cross hair cursor. Both the block cursor and the cross hair cursor can be displayed at the same time. The MGR adapter contains two cursor chips which allows either two block or cross hair cursors of the same color to be displayed at different locations on the screen or a single block cursor to be displayed on the screen with one to three colors. When both the cross hair cursor and the block cursor are enabled on the same cursor chip the intersecting pixels can be combined with a logical OR or a logical exclusive OR operation. The cursor X,Y registers are used to specify the screen location of the center of the block cursor and the intersection point for the cross hair cursor.

The cross hair cursor is displayed within the cross hair cursor window. The window width and height registers are used to specify the size of the cross hair cursor window. The window X,Y registers are used to specify the upper left corner of the cross hair cursor window. The thickness of the cross hair cursor can be specified in pixels. The width values are 1,3 , 5 or 7 pixels and are specified by two bits in the command register.

The horizontal and vertical sync pulses generated by the Display State Machine are used by the cursor chip to determine the cursor outputs. The vertical sync marks the start of a new raster scan and resets the cursors y counter to zero. Each horizontal sync pulse resets the the cursors x counter to zero. The DSM generates pixel clock pulses for each pixel on the current scan line. The upper left corner of the screen is the 0,0 position and the positive x direction is to the right and the positive y direction is down the screen. The cursor compares the counter x,y values with the cursor X,Y register values.

For the block cursor the comparison begins at cursor X - 31 and Y - 31 and continues for the next 64 pixel locations in the x direction and the next 64 scan lines in the y direction. When the location comparison passes the cursor chip checks the contents of the cursor RAM to determine the output value. If the cursor RAM contains a zero bit at the X,Y location the five outputs are all set to zero. If the cursor RAM bit is a one then a one is output on all five outputs. For all screen locations outside the block cursor 64 x 64 bit locations the cursor chip generates zeroes on all five outputs.

For the cross hair cursor the comparisons begin at the window boundaries and are controlled by the cursor X, Y values and the thickness setting. For the pixels that are located on the cross hair cursor x and y locations the cursor chip generates a one on all five outputs. For all pixel locations which are not on the cross hair cursor the cursor chip generates zeroes on all five outputs.

Registers

The following paragraphs describe the registers for the following Raster Subsystem components:

- RE2 chip
- Cursor chip

RE2 Registers

The RE2 contains an instruction register and 56 control registers which are programmable by the GE5 microcode. The Instruction Register is loaded with the opcode of the command to be executed by the RE2. The other 56 registers are loaded with data which is used by the RE2 during the execution of the command. The host does not have the ability to directly program any of the RE2 registers. The ENABRGB and the TOPSCAN registers are the only registers that the host needs to program. The host uses the GE_LOADRE token to load these two registers. The other registers are loaded by the microcode and should not be modified by the host software. The following registers have input buffers which allow multiple values to be written into them for multiple command execution:

- ENABRGB register : Specifies if 8 bit RGB pixels are to be used for 8 bitplane configuration
- BIGENDIAN register : Specifies the pixel packing order for the Write Buffer instruction
- FUNC register : Contains the raster op function
- HADDR register : Specifies the starting pixel offset in the first packed word
- NOPUP register : Specifies if 2 PUP and 2 UAUX bits or 0 PUP and 4 UAUX bits are used
- XYFRAC register : Contains the initial XY fractions bits for antialiased lines
- RGB register : Contains initial red, green and blue color values
- YX register : Contains the initial X and Y values
- PUPDATA register : Contains the 2 bit PUP data value
- PATL register : Contains the lower 16 bits of the pattern mask
- PATH register : Contains the upper 16 bits of the pattern mask
- DZI register : Contains the initial integer portion of the Delta Z value
- DZF register : Contains the fraction portion of the Delta Z value
- DR register : Contains the Delta Red value
- DG register : Contains the Delta Green value
- DB register : Contains the Delta Blue value
- Z register : Contains the initial Z value

- R register : Contains the initial Red value
- G register : Contains the initial Green value
- B register : Contains the initial Blue value
- STIP register : Contains the line Stipple pattern
- STIPCOUNT register : Contains the LSB stipple bit repeat count
- DX register : Contains the Delta X value
- DY register : Contains the Delta Y value
- NUMPIX register : Contains the pixel count for the instruction
- X register : Contains the initial X value
- Y register : Contains the initial Y value
- IR register : Contains the instruction to be executed

The following registers do not have input buffers and are generally not changed for each instruction:

- RWDATA register : The Read and Write Buffer instruction data register
- PIXMASK register : Contains the Frame Buffer pixel write mask
- AUXMASK register : Contains the PUP, UAUX, WID and Z buffer bitplanes write mask
- WIDDATA register : Contains the 2 or 4 bit WID data value
- UAUXDATA register : Contains the 2 or 4 bit UAUX data value
- RWMODE register : Specifies the source or destination bitplanes for the Read and Write Buffer instructions
- READBUF register : Selects which frame buffer is read by READBUF instruction
- PIXTYPE register : Selects the frame buffer pixel type
- ASELECT register : Selects the antialiasing weight
- ALIGNPAT register : Controls the pattern alignment
- ENABPAT register : Enables or disables the pattern mask
- ENABSTIP register : Enables or disables the line stipple pattern
- ENABDITH register : Enables or disables color dithering
- ENABWID register : Enables or disables WID checking for shaded span instructions

- CURWID register : Contains the current Window ID
- DEPTHFN register : Specifies the Z or color compare relational function
- REPSTIP register : Specifies the repeat count for all bits in the stipple count except LSB bit
- ENABLWID register : Enables or disables WID checking for Draw Line instructions
- FBOPTION register : Specifies if 8 or 24 bitplanes are installed
- TOPSCAN register : Specifies the number of columns and rows to be displayed on the screen
- ZBOPTION register : Specifies if the Z buffer card is installed or not
- XZOOM register : Specifies the X zoom repeat count for Write Buffer instructions
- UPACMODE register : Specifies the number of pixels/long word for Write Buffer instruction
- YMIN register : Contains the bottom boundary of the hardware screen mask
- YMAX register : Contains the top boundary of the hardware screen mask
- XMIN register : Contains the left boundary of the hardware screen mask
- XMAX register : Contains the right boundary of the hardware screen mask
- COLORCMP register : Controls the selection of the Z compare or the color compare
- MEGOPTION register : Specifies if 1 MEG or 256K VRAMs are installed

ENABRGB Register

This is a one bit register which is used to enable 8 bit RGB pixels. This register is ignored on the enhanced adapter when the FBOPTION register is set to one. When the FBOPTION register is set to zero for the base adapter the value in the ENABRGB register is used by the RE2. When this register is set to one the RE2 does the special 233 processing to form the 8 bit RGB pixels from the red, green and blue color bytes. The RE2 takes the most significant 2 bits of the blue color value and places them in the most significant 2 bits of the 8 bit RGB value. The most significant 3 bits of the green value are placed in the middle 3 bits of the 8 bit RGB value and the three most significant bits of the red value are placed in the three LSB bits of the 8 bit RGB value. When the ENABRGB register is zero the 8 bit RGB processing is not performed on the color values. Instead the 8 bit red value is placed in the 8 bit pixel values. The host software must set this register using the GE_LOADRE token. It can be set at any time after the microcode has been downloaded and the three data parameters have been sent. The format of the 8 bit RGB register is shown in Figure 6.9.

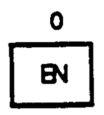


Figure 6.9 ENABRGB Register

REptr index = 4

Buffered Input Register

Bit 0 : Enable 8 bit RGB (Write Only). This bit enables or disables 8 bit RGB mode on the base adapter.

FBOPTION = 0

0 - Disable 8 bit RGB mode
1 - Enable 8 bit RGB mode

FBOPTION = 1

X - ENABRGB register is ignored and normal 24 bit RGB processing is performed

BIGENDIAN Register

This is a 1 bit register that specifies the byte order for packed pixels for the Write Buffer instruction. When this register is set to one then the BIGENDIAN mode is enabled and the pixels are packed in the long words from left to right. When this register is a zero the BIGENDIAN mode is off and the pixels are packed right to left in the long words. Since the Eddy register provides the little endian to big endian byte ordering this register is always set to one for BIGENDIAN mode. The format of this register is shown in Figure 6.10.

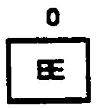


Figure 6.10 BIGENDIAN Register

REptr index = 5

Buffered Input Register

Bit 0 : Enable Big Endian mode (Write Only). This bit enables or disables Big Endian mode.

0 - Disable Big Endian mode
1 - Enable Big Endian mode

FUNC Register

This is a 4 bit register which is used to specify the Raster operation which is to be performed on the pixel data. The raster op is performed between the new pixel data (src) and the pixel data in the bitplanes (dst). The format of this register is shown in Figure 6.11.

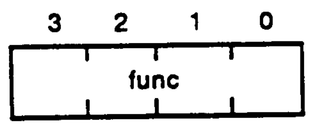


Figure 6.11 FUNC Register

REptr index = 6

Buffered Input Register

Bits 3-0 : func (Write Only). These bits specify the raster op function.

bits	destination value
0000	zero
0001	src AND dest
0010	src AND (COMP dest)
0011	src
0100	(COMP src) AND dest
0101	dest
0110	src XOR dest
0111	src OR dest
1000	(COMP src) AND (COMP dest)
1001	(COMP src) XOR dest
1010	COMP dest
1011	src OR (COMP dest)
1100	COMP src
1101	(COMP src) OR dest
1110	(COMP src) OR (COMP dest)
1111	One

COMP - 1's complement

HADDR Register

This register is used to specify the starting pixel offset in the first long word DMAed to the RE2 to be written to the bitplanes. This register allows the host buffer to start on a byte or short boundary which is not long word aligned. The UPACMODE register specifies how many pixels per long word are in each long word received by the RE2. The format of this register is shown in Figure 6.12

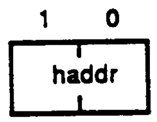


Figure 6.12 HADDR Register

REptr index = 7

Buffered Input Register

Bits 1-0 : haddr (Write Only). These bits specify the starting pixel location in the first packed long word. The following bit definitions assume big endian mode is selected.

00 - lst byte or short pixel is starting pixel
01 - 2nd byte pixel is starting pixel
10 - 3rd byte pixel or 2nd short is the starting pixel
11 - 4th byte pixel is starting pixel

NOPUP Register

This register is used to select either 2 PUP bits and 2 UAUX bits or 0 PUP bits and 4 UAUX bits. This register is only used on the enhanced configuration and is ignored on the base adapter. The format of this register is shown in Figure 6.13.

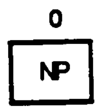


Figure 6.13 NOPUP Register

REptr index = 8

Buffered Input Register

Bit 0 : NP (Write Only). This bit specifies the size of the UAUX bits on the enhanced adapter. When this bit is 0 the overlay bits consist of 2 PUP bits and 2 UAUX bits. When this bit is 1 the overlay bits consist of 0 PUP bits and 4 PUP bits. This bit is ignored on the base adapter. This register is always set to 0 in the MGR adapter since the microcode provides the same functionality.

0 - select 2 PUP and 2 UAUX bits
1 - select 0 PUP and 4 UAUX bits

XYFRAC Register

This register is used to improve the quality of antialiased lines. It is used to specify the initial fraction bits of the x or y axis that is changing slower on the antialiased line. The value in the register affects the value of the antialiasing weight used when drawing the first pixel on the line. This register allows the space between antialiased lines to appear constant on slow moving wireframes with closely spaced lines. The format of this register is shown in Figure 6.14.

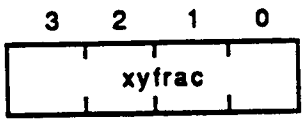


Figure 6.14 XYFRAC Register

REptr index = 9

Buffered Input Register

Bits 3-0 : xyfrac (Write Only). These bits specify the initial xyfrac value used in antialiased lines.

RGB Register

This register is used to hold the initial red, green and blue color values. This register allows the microcode to load the RGB values faster than using the R, G and B registers independently. This register is the same functionally as the separate R, G and B registers. The format of this register is shown in Figure 6.15.

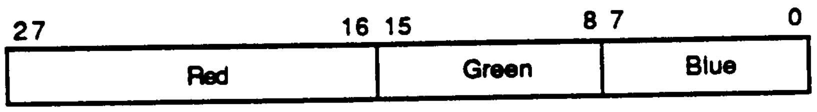


Figure 6.15 RGB Register

REptr index = 0xA

Buffered Input Register

Bits 27-16 : red (Write Only). These bits contain the initial red color.

Bits 15-8 : green (Write Only). These bits contain the initial green color.

Bits 7-0 : blue (Write Only). These bits contain the initial blue color.

YX Register

This register contains both the initial Y and X screen locations of the starting pixel location to be read or written by the next instruction. The X value is in the special DIVMOD format in which the X screen location is divided by 5 and the quotient is stored in the upper nine bits and the remainder is stored in the lower 3 bits of the X part of this register. The format of this register is shown in Figure 6.16.

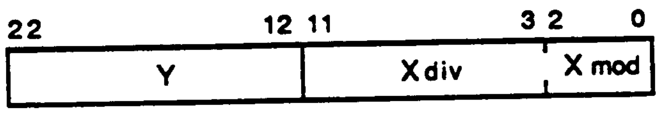


Figure 6.16 YX Register

REptr index = 0xB

Buffered Input Register

Bits 22-12 : Initial Y (Write Only). These bits contain the initial Y screen location.

Bits 11-0 : Initial X (Write Only). These bits contain the initial X screen location.

PUPDATA Register

This register is loaded with the data which is written into the PUP bitplanes. On the enhanced adapter (FBOPTION = 1) the NOPUP register controls whether the 2 PUP bits are used or not. If the NOPUP register is zero then the 2 PUP bits are used as well as 2 bits in the UAUX register. If the NOPUP register is a one then the 2 PUP bits are not used and 4 bits in the UAUX register are used instead. The format of this register is shown in Figure 6.17.

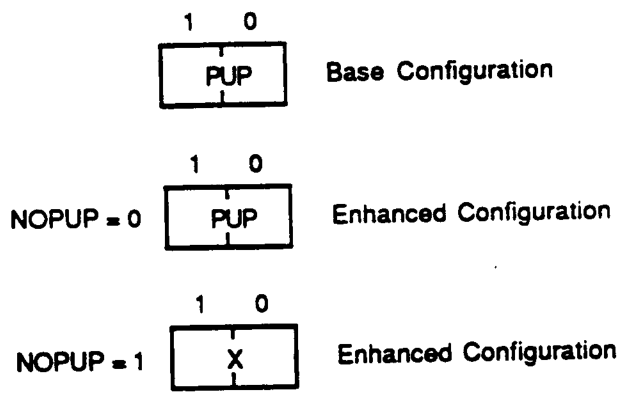


Figure 6.17 PUPDATA Register

REptr index = 0xC

Buffered Input Register

Bits 1-0 : PUP (Write Only). These bits contain the PUP data bits.

PATL Register

This register contains the lower 16 bits of the 32 bit Pattern Mask register used by the Draw Shaded Span instruction to condition pixel writes with a texture pattern. When pattern masking is enabled with the ENABPAT register the pixels whose pattern mask bits are one are written and pixels whose pattern mask bits are zero are not written. If pattern masking is disabled the pattern mask bits are ignored as shaded span pixels are written. The ALIGNPAT register is used to control the pattern to screen pixels. The format of this register is shown in Figure 6.18.

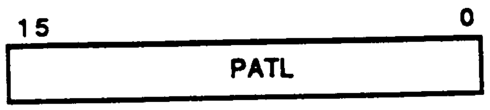


Figure 6.18 PATL Register

REptr index = 0xD

Buffered Input Register

Bits 15-0 : PATL (Write Only). These bits contain the lower 16 bits of the Pattern Mask.

PATH Register

This register contains the upper 16 bits of the 32 bit Pattern Mask register used by the Draw Shaded Span instruction to condition pixel writes with a texture pattern. When pattern masking is enabled with the ENABPAT register the pixels whose pattern mask bits are one are written and pixels whose pattern mask bits are zero are not written. If pattern masking is disabled the pattern mask bits are ignored as shaded span pixels are written. The ALIGNPAT register is used to control the pattern to screen pixels. The format of this register is shown in Figure 6.19.

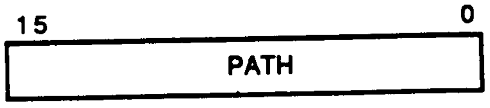


Figure 6.19 PATH Register

REptr index = 0xE

Buffered Input Register

Bits 15-0 : PATH (Write Only). These bits contain the upper 16 bits of the Pattern Mask.

DZI Register

This register contains the integer portion of the delta Z value. This value is added to the Z value after each pixel is written to get the next Z value. The value in the register is a 24 bit 2's complement integer. The format of this register is shown in Figure 6.20.

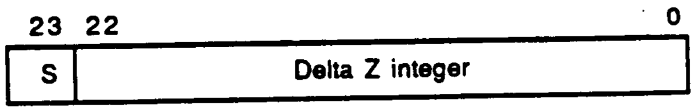


Figure 6.20 DZI Register

REptr index = 0xF

Buffered Input Register

Bit 23 : DZI sign bit (Write Only). This bit contains the sign bit for the integer portion of the delta Z value

Bits 22-0 : DZI (Write Only). These bits contain the integer portion of the delta Z value

DZF Register

This register contains the fraction portion of the delta Z value. This value is added to the fraction portion of the Z value after each pixel is written to get the next fractional part of the Z value. If the result of adding the two fractional parts is greater than 1 then the integer portion of the Z is incremented. The format of this register is shown in Figure 6.21.

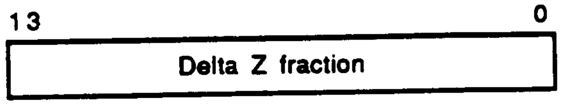


Figure 6.21 DZF Register

REptr index = 0x10

Buffered Input Register

Bits 13-0 : DZF (Write Only). These bits contain the fraction portion of the delta Z value

DR Register

This register contains the delta Red color value which is added to the Red color value after the current pixel is written to get the next Red color value. The integer portion of this register is a 2's complement integer. The integer portion of this register is 12 bits wide because it is used for the color index values for color index pixels. The format of this register is shown in Figure 6.22.

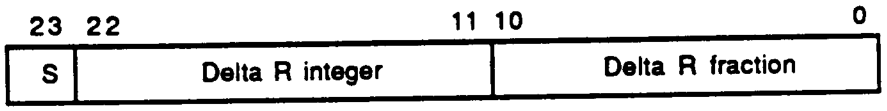


Figure 6.22 DR Register

REptr index = 0x11

Buffered Input Register

Bit 23 : DR sign bit (Write Only). This bit contains the sign bit for the integer portion of the delta R value

Bits 22-11 : DR integer (Write Only). These bits contain the integer portion of the delta R value

Bits 10-0 : DR fraction (Write Only). These bits contain the fraction portion of the delta R value

DG Register

This register contains the delta Green color value which is added to the Green color value after the current pixel is written to get the next Green color value. The integer portion of this register is a 2's complement integer. The format of this register is shown in Figure 6.23.

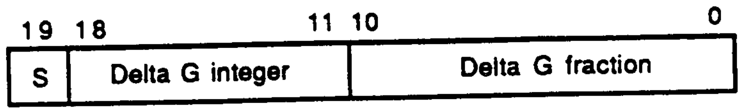


Figure 6.23 DG Register

REptr index = 0x12

Buffered Input Register

Bit 19 : DG sign bit (Write Only). This bit contains the sign bit for the integer portion of the delta G value

Bits 18-11 : DG integer (Write Only). These bits contain the integer portion of the delta G value

Bits 10-0 : DG fraction (Write Only). These bits contain the fraction portion of the delta G value

DB Register

This register contains the delta Blue color value which is added to the Blue color value after the current pixel is written to get the next Blue color value. The integer portion of this register is a 2's complement integer. The format of this register is shown in Figure 6.24.

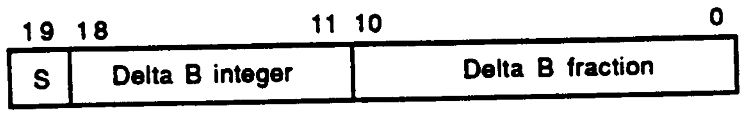


Figure 6.24 DB Register

REptr index = 0x13

Buffered Input Register

Bit 19 : DB sign bit (Write Only). This bit contains the sign bit for the integer portion of the delta B value

Bits 18-11 : DB integer (Write Only). These bits contain the integer portion of the delta B value

Bits 10-0 : DB fraction (Write Only). These bits contain the fraction portion of the delta B value

Z Register

This register contains the integer portion of the initial Z value. This value is used for the Z buffer checks and is written to the Z buffer along with the other pixel bits if the Z check passes. The delta Z value is added to the Z value after each pixel is written to get the next Z value. The value in the register is a 24 bit 2's complement integer. The format of this register is shown in Figure 6.25.

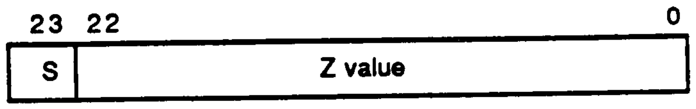


Figure 6.25 Z Register

REptr index = 0x14

Buffered Input Register

Bit 23 : Z sign bit (Write Only). This bit contains the sign bit for the integer portion of the initial Z value

Bits 22-0 : Z (Write Only). These bits contain the integer portion of the initial Z value

R Register

This register contains the initial Red color value which is used as the Red color component of the RGB pixels or as the color index for color index pixels. The delta Red color value is added to the Red color value after the current pixel is written to get the next Red color value. The format of this register is shown in Figure 6.26.

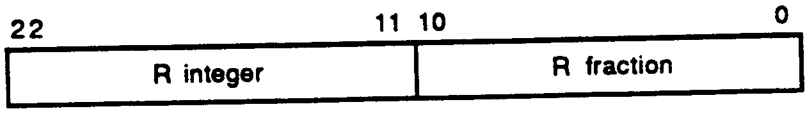


Figure 6.26 R Register

REptr index = 0x15

Buffered Input Register

Bits 22-11 : R integer (Write Only). These bits contain the integer portion of the Red color value

Bits 10-0 : R fraction (Write Only). These bits contain the fraction portion of the Red value

G Register

This register contains the initial Green color value which is used as the green color component of the RGB pixels. The delta Green value is added to the Green color value after the current pixel is written to get the next Green color value. The format of this register is shown in Figure 6.27.

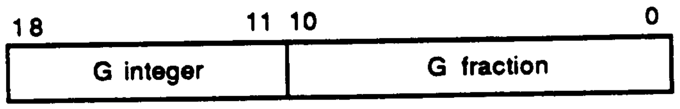


Figure 6.27 G Register

REptr index = 0x16

Buffered Input Register

Bits 18-11 : G integer (Write Only). These bits contain the integer portion of the Green color value

Bits 10-0 : G fraction (Write Only). These bits contain the fraction portion of the Green color value

B Register

This register contains the initial Blue color value which is used as the blue color component of RGB pixels. The delta Blue value is added to the Blue color value after the current pixel is written to get the next Blue color value. The format of this register is shown in Figure 6.28.

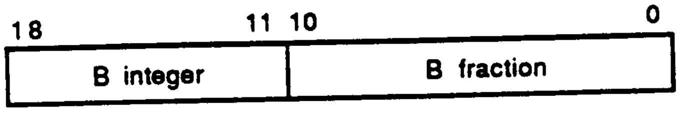


Figure 6.28 B Register

REptr index = 0x17

Buffered Input Register

Bits 18-11 : B integer (Write Only). These bits contain the integer portion of the Blue color value

Bits 10-0 : DB fraction (Write Only). These bits contain the fraction portion of the B color value

STIP Register

This register is used to hold the 16 bit line stipple pattern. When stipple conditioning is enabled with the ENABSTIP register the LSB bit is used to determine if the pixel is written during Draw Line instructions. If the LSB bit in the STIP register is a one the pixel is written if it is a zero the pixel is not written. The STIPCOUNT register determines how many times the initial LSB bit is used before the STIP register is rotated right one bit. For the remaining bits in the STIP register the REPSTIP register contains the repeat count for how many times each of those bits will be used to condition pixel writes before the STIP register is rotated right again. The format of this register is shown in Figure 6.29.

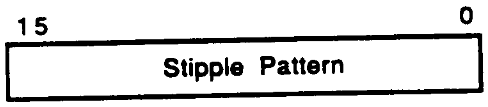


Figure 6.29 STIP Register

REptr index = 0x18

Buffered Input Register

Bits 15-0 : Stipple Pattern (R/W). These bits contain the line stipple pattern.

STIPCOUNT Register

This register contains the repeat count for the LSB bit in the STIP register. When stipple checking is enabled with the ENABSTIP register the STIPCOUNT register determines how many pixel writes will be conditioned by the LSB bit in the STIP register before the STIP register is rotated right. The stipple pattern is used to condition pixel writes for the Draw Line instructions. The format of this register is shown in Figure 6.30.

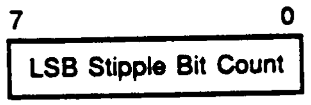


Figure 6.30 STIPCOUNT Register

REptr index = 0x19

Buffered Input Register

Bits 7-0 : LSB Bit Stipple Count (R/W). These bits contain the repeat count for the LSB bit in the stipple pattern register.

DX Register

This register contains the delta X value. This value is added to the X value after each pixel is written to get the next X value. If the result of adding the two fractional parts is greater than 1 then the integer portion of the X value is incremented. The integer portion of this register is a 2 bit 2's complement format. The format of this register is shown in Figure 6.31.

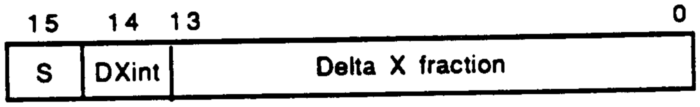


Figure 6.31 DX Register

REptr index = 0x1A

Buffered Input Register

Bit 15 : DX sign bit (Write Only). This bit contains the sign bit for the delta X value

Bit 14 : DX integer (Write Only). This bit contains the integer portion of the delta X value

Bits 13-0 : DX fraction (Write Only). These bits contain the fraction portion of the delta X value

DY Register

This register contains the delta Y value. This value is added to the Y value after each pixel is written to get the next Y value. If the result of adding the two fractional parts is greater than 1 then the integer portion of the Y value is incremented. The integer portion of this register is a 2 bit 2's complement format. The format of this register is shown in Figure 6.32.

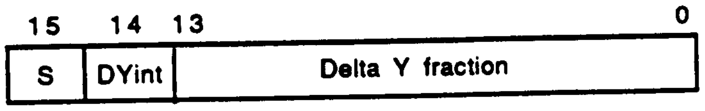


Figure 6.32 DY Register

REptr index = 0x1B

Buffered Input Register

Bit 15 : DY sign bit (Write Only). This bit contains the sign bit for the delta Y value

Bit 14 : DY integer (Write Only). This bit contains the integer portion of the delta Y value

Bits 13-0 : DY fraction (Write Only). These bits contain the fraction portion of the delta Y value

NUMPIX Register

This register is used as a pixel counter by the RE2. The number of pixels to be read or written during the next instruction are written into this register. As each pixel is read or written the NUMPIX register is decremented. When the count becomes zero the instruction execution terminates. The format of this register is shown in Figure 6.33.

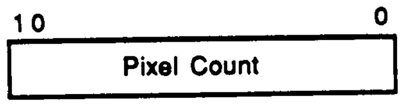


Figure 6.33 NUMPIX Register

REptr index = 0x1C

Buffered Input Register

Bits 10-0 : Pixel Count (Write Only). These bits contain the pixel count for the current instruction being executed.

X Register

This register contains the initial X screen location of the starting pixel location to be read or written by the next instruction. The X value is in the special DIVMOD format in which the X screen location is divided by 5 and the quotient is stored in the upper nine bits and the remainder is stored in the lower 3 bits of this register. The format of this register is shown in Figure 6.34.

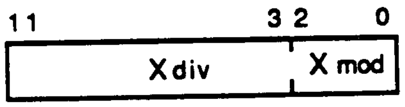


Figure 6.34 X Register

REptr index = 0x1D

Buffered Input Register

Bits 11-0 : Initial X (Write Only). These bits contain the initial X screen location.

Y Register

This register contains the initial Y screen location of the starting pixel location to be read or written by the next instruction. The format of this register is shown in Figure 6.35.

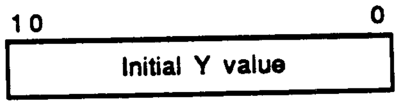


Figure 6.35 Y Register

REptr index = 0x1E

Buffered Input Register

Bits 10-0 : Initial Y (Write Only). These bits contain the initial Y screen location.

IR Register

This register is loaded with the instruction which the microcode wants the RE2 to execute. When the instruction is loaded into the IR register the RE2 will execute it as soon as it completes the previous instruction execution. The format of this register is shown in Figure 6.36.

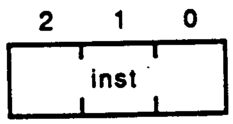


Figure 6.36 IR Register

REptr index = 0x1F

Buffered Input Register

Bits 2-0 : Inst (Write Only). These bits contain the instruction to be executed by the RE2.

000 - Undefined
001 - Draw Shaded Span
010 - Draw 1 x 5 Flat Span
011 - Draw 1 x 20 Flat Span (Block Write Mode)
100 - Draw Top of Antialiased Line
101 - Draw Bottom of Antialiased Line
110 - Read Buffer
111 - Write Buffer

RWDATA Register

This register is the data register used by the read buffer and write buffer instruction. The data to be written into the bitplanes is written by the DMA hardware into this register and the RE2 then takes the value from the RWDATA register and writes it to the bitplanes selected by the RWMODE register. The read buffer instruction reads the selected bitplanes and places the data into the RWDATA register. The DMA hardware then transfers the data from the register to the host buffer or to the pixel buffer in the GE5 data RAM. The format of the register is shown in Figure 6.37.

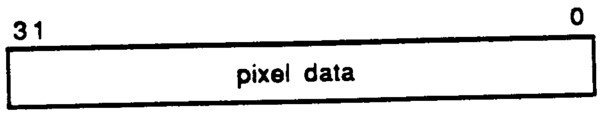


Figure 6.37 RWDATA Register

REptr index = 0x20

Unbuffered Input Register

Bits 31-0 : pixel data (R/W). These bits contain the pixel data read from the bitplanes or to be written to the bitplanes. For pixel writes all 32 bits can be used while only the right most 28 bits will be used for pixel reads.

PIXMASK Register

This register contains the Frame Buffer pixel write mask which controls which bits in the pixel data is written into the Frame Buffer bitplanes. The pixel write mask is used to select either a single buffer write or a double buffer write. For the double buffer write the front or back buffer is selected for writing with the value in the pixel write mask. The format of this register is shown in Figure 6.38.

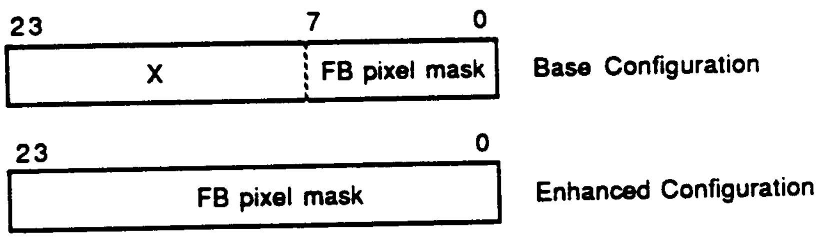


Figure 6.38 PIXMASK Register

REptr index = 0x21

Unbuffered Input Register

Bits 23-0 : pixmask (Write Only). These bits contain the Frame Buffer pixel write mask.

0x0000FF	- write to all the base adapter Frame Buffer bitplanes
0xFFFFFF	- write to all the enhanced adapter Frame Buffer bitplanes
0xFFFFFF	- write to all bits of PIXTYPE = 0 (24 bit RGB)
0x0000FF	- write to all bits of PIXTYPE = 0 (8 bit RGB)
0x0F0F0F	- write to the front buffer with PIXTYPE = 1 (12 bit RGB)
0xF0F0F0	- write to the back buffer with PIXTYPE = 1 (12 bit RGB)
0x000FFF	- write to the front buffer with PIXTYPE = 2 (8 or 12 bit CI)
0xFFF000	- write to the back buffer with PIXTYPE = 2 (8 or 12 bit CI)
0x00000F	- write to the front buffer with PIXTYPE = 3 (4 bit CI)
0x0000F0	- write to the back buffer with PIXTYPE = 3 (4 bit CI)

AUXMASK Register

This register contains the write mask which is used to control which bits are written into the PUP bitplanes, the UAUX bitplanes, the WID bitplanes and the Z Buffer bitplanes. For the PUP bitplanes the UAUX bitplanes and the WID bitplanes the bits in the aux mask are used to mask individual bits. For the Z Buffer bitplanes a single mask bit is used to enable or disable writes into all 24 Z Buffer bitplanes. The format of this register is shown in Figure 6.39.

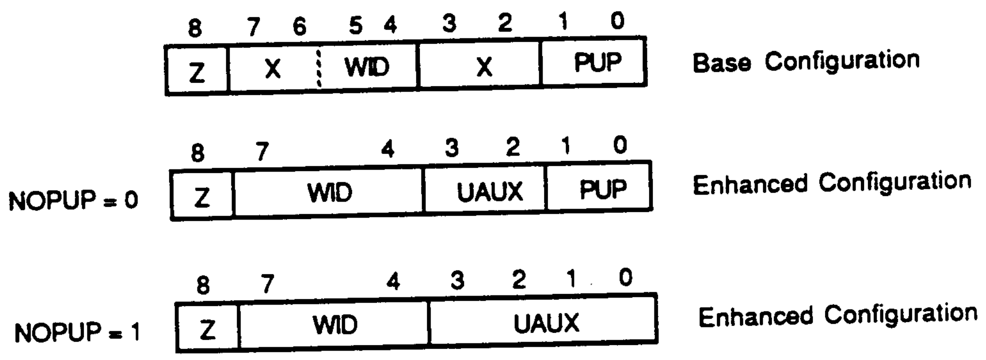


Figure 6.39 AUXMASK Register

REptr index = 0x22

Unbuffered Input Register

Bits 8-0 : auxmask (Write Only). These bits contain the write mask for the PUP bitplanes, the UAUX bitplanes, the WID bitplanes and the Z Buffer bitplanes.

0x003 - to write to just the PUP bitplanes
0x00C - to write to just the UAUX bitplanes
0x00F - to write to just the UAUX bitplanes with NOPUP =1 and FBOPTION = 1
0x0F0 - to write to just the WID bitplanes
0x100 - to write to just the Z Buffer bitplanes

WIDDATA Register

This register contains the WID data which is written into the WID bitplanes if the WID mask bits in the AUXMASK register are set to one. The GE_DRAWMODE token is used to specify the WID draw mode and then the GE_COLOR or GE_COLORF tokens are used to specify the index value to be written into the WIDDATA register. The WID data written into the WID bitplanes is used to condition pixel writes for the shaded span, draw line, and write buffer instructions. The LSB bit of the WID data in the WID bitplanes can also be used as a Z buffer invalidate bit for fast Z clears. This condition is enabled or disabled with bit 3 of the DEPTHFN register. The WID data is also used as an index for the mode registers in the XPC1 or XMAP2 chips in the Display Subsystem. The mode registers control the pixel display formatting thus the WID data bits control the pixel formatting for the various on screen windows. The format of the WIDDATA register is shown in figure 6.40.

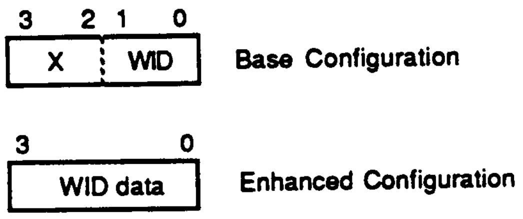


Figure 6.40 WIDDATA Register

REptr index = 0x23

Unbuffered Input Register

Bits 3-0 : WID data (Write Only). These bits contain the Window ID data bits.

UAUXDATA Register

This register contains the UAUX data which is written into the UAUX bitplanes if the UAUX mask bits in the AUXMASK register are set to one. The GE_DRAWMODE token is used to specify the OVER draw mode and then the GE_COLOR or GE_COLORF tokens are used to specify the index value to be written into the UAUXDATA register. The UAUX data written into the UAUX bitplanes is used as an index for the overlay palette registers in the RGB RAMDAC chip or the overlay color map registers in the XMAP2 chips in the Display Subsystem. The index value selects an overlay or underlay color value if the overlay or underlay conditions are satisfied. The format of the UAUXDATA register is shown in figure 6.41.

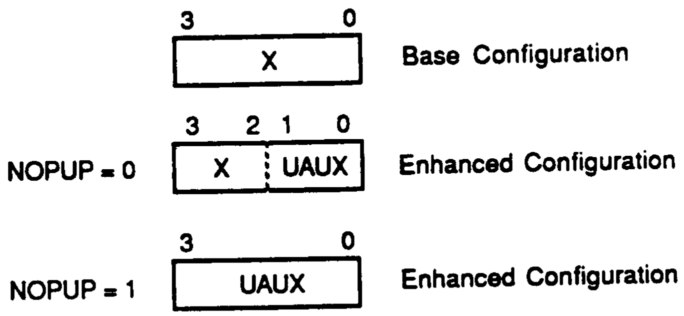


Figure 6.41 UAUXDATA Register

REptr index = 0x24

Unbuffered Input Register

Bits 3-0 : UAUX data (Write Only). These bits contain the UAUX data bits.

RWMODE Register

This register specifies the source bitplanes for the Read Buffer instruction and specifies the destination bitplanes for the Write Buffer bitplanes. The format of this register is shown in Figure 6.42.

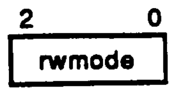


Figure 6.42 RWMODE Register

REptr index = 0x25

Unbuffered Input Register

Bits 2-0 : rwmode (Write Only). These bits specify the source or destination bitplanes to be read or written.

000 - Frame buffer bitplanes
001 - PUP bitplanes
010 - UAUX bitplanes
011 - Z buffer bitplanes
100 - Window ID (WID) bitplanes
101 - invalid
110 - Frame buffer port
111 - Z buffer port

READBUF Register

This register specifies which buffer is read by the Read Buffer instruction when the RWMODE register specifies the Frame Buffer as the read source bitplanes. The format of this register is shown in Figure 6.43.

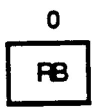


Figure 6.43 READBUF Register

REptr index = 0x26

Unbuffered Input Register

Bit 0 : RB (Write Only). This bit specifies which Frame Buffer bitplane buffer is read by the Read Buffer instruction.

0 - read front buffer
1 - read back buffer

PIXTYPE Register

This register specifies the type of pixel formatting that is performed on the R, G and B registers to form the Frame Buffer pixel data which is written into the Frame Buffer bitplanes. The format of this register is shown in Figure 6.44.

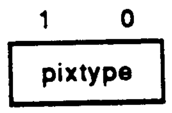


Figure 6.44 PIXTYPE Register

REptr index = 0x27

Unbuffered Input Register

Bits 1-0 : pixtype (Write Only). These bits specify the pixel formatting performed on the color components to form the pixel data written into the Frame Buffer bitplanes.

00 - 8 bit RGB single buffer pixel if FBOPTION = 0
00 - 24 bit RGB single buffer pixel if FBOPTION = 1
01 - 12 bit RGB double buffer pixel (FBOPTION =1 only)
10 - 8 bit CI single buffer pixel (FBOPTION = 0)
10 - 12 bit CI double buffer pixel (FBOPTION = 1)
11 - 4 bit CI double buffer pixel (FBOPTION = 0 or 1)

ASELECT Register

This register contains antialiase weight usage control bits. The 3 most significant bits of the fraction part of the X or Y minor axis (slower changing axis) are used to select a 4 bit antialiase weight value which can be used to replace any nibble in the 24 bit Frame Buffer pixel value. The ASELECT register is used to select which of the six nibbles are replaced by the antialiase weight value. A 1 bit in any of the six bits in the ASELECT register causes the weight to replace the corresponding nibble. The format of the register is shown in Figure 6.45.

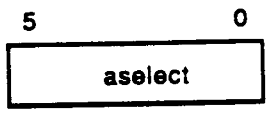


Figure 6.45 ASELECT Register

REptr index = 0x28

Unbuffered Input Register

Bits 5-0 : aselect (Write Only). These bits contain the antialiase weight replacement control bits.

000000 - No nibbles replaced with the weight
000001 - Replace nibble 0 with weight in 24 bit Frame Buffer pixel
000010 - Replace nibble 1 with weight in 24 bit Frame Buffer pixel
000100 - Replace nibble 2 with weight in 24 bit Frame Buffer pixel
001000 - Replace nibble 3 with weight in 24 bit Frame Buffer pixel
010000 - Replace nibble 4 with weight in 24 bit Frame Buffer pixel
100000 - Replace nibble 5 with weight in 24 bit Frame Buffer pixel
111111 - Replace all six nibbles with weight in 24 bit Frame Buffer pixel
000011 - Replace nibbles 0 and 1 with weight for double buffer 4 bit CI pixel
001001 - Replace nibbles 0 and 3 with weight for double buffer 12 bit CI pixel

ALIGNPAT Register

This register controls the pixel alignment of the 32 bit pattern mask. The pattern can be either screen pixel location aligned or relative drawing aligned. The patterned polygons are drawn with screen alignment and characters are drawn with relative alignment. When screen alignment is selected the pattern mask is applied to each group of 32 pixels on the current scan line beginning with the left most pixel. When drawing relative alignment is selected the pattern mask is applied beginning with the starting X, Y pixel location. Each bit in the pattern mask is used by the shaded span instruction to condition a pixel write. If the current bit in the pattern mask is one the pixel is written and if it is zero the pixel is not written. The format of the ALIGNPAT register is shown in Figure 6.46.

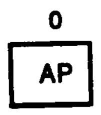


Figure 6.46 ALIGNPAT Register

REptr index = 0x29

Unbuffered Input Register

Bit 0 : alignment (Write Only). This bit controls the pattern mask alignment.

0 - use drawing relative alignment (characters)
1 - use screen alignment (textured polygons)

ENABPAT Register

This register is used to enable or disable the use of the pattern mask for the shaded span and write buffer instructions. When enabled the 32 bit pattern mask contained in the PATH and PATL registers is applied with the alignment specified in the ALIGNPAT register to condition pixel writes. The format of this register is shown in Figure 6.47.

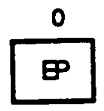


Figure 6.47 ENABPAT Register

REptr index = 0x2A

Unbuffered Input Register

Bit 0 : Enable Pattern (Write Only). This bit is used to enable or disable pattern masking.

0 - disable pattern masking
1 - enable pattern masking

ENABSTIP Register

This register is used to enable or disable the use of the stipple pattern for the draw line instructions. When enabled the 16 bit stipple pattern contained in the STIP register is applied with the repeat counts specified in the STIPCOUNT and REPSTIP registers to condition pixel writes. The format of this register is shown in Figure 6.48.

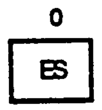


Figure 6.48 ENABSTIP Register

REptr index = 0x2B

Unbuffered Input Register

Bit 0 : Enable Stip (Write Only). This bit is used to enable or disable the use of the stipple pattern when drawing lines.

0 - disable stipple pattern
1 - enable stipple pattern

ENABDITH Register

This register is used to enable or disable the dithering operations during the Frame Buffer pixel calculations. When enabled the dither matrix is indexed by the four MSB fraction bits for color index pixels or the LSB nibble for 12 bit RGB pixels. The dither matrix value is compared to the index value and if the index value is greater than the index value the integer component of the color registers is incremented The format of this register is shown in Figure 6.49.

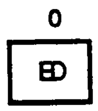


Figure 6.49 ENABDITH Register

REptr index = 0x2C

Unbuffered Input Register

Bit 0 : Enable Dither (Write Only). This bit is used to enable or disable the dithering operations.

0 - disable dithering
1 - enable dithering

ENABWID Register

This register is used to enable or disable the WID checking for the various instructions which allow WID checking. The ENABLWID register also must be enabled for line drawing WID checking. When enabled the value in the WID bitplanes at the current X, Y location are compared to the value in the CURWID register. If the values are the same the WID check passes. If the values are different the WID check fails. The format of this register is shown in Figure 6.50.

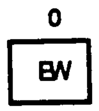


Figure 6.50 ENABWID Register

REptr index = 0x2D

Unbuffered Input Register

Bit 0 : Enable WID check (Write Only). This bit is used to enable or disable WID checking.

0 - disable WID checking
1 - enable WID checking

CURWID Register

This register contains the current WID value which is compared to the WID data in the WID bitplanes when WID checking is enabled. The WID check passes if the WID data value is the same as the value in the CURWID register. If the two values are different the WID check fails. If the WID check passes the pixel write depends on the other pixel write conditioning checks. If the WID check fails the pixel is not written. If WID checking is disabled the WID check automatically passes. The format of this register is shown in Figure 6.51.

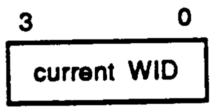


Figure 6.51 CURWID Register

REptr index = 0x2E

Unbuffered Input Register

Bits 3-0 : Current WID (Write Only). These bits contain the current WID value to be compared to the WID bitplane data during WID checking operations.

DEPTHFN Register

This register specifies the relational comparison function which is performed by the Z comparator hardware. If the COLORCMP register is zero the new Z value is compared to the Z value in the Z Buffer. If the COLORCMP register is one the new Frame Buffer pixel value is compared to the color value already in the Frame Buffer. Bit 3 of the DEPTHFN register is used to enable or disable the use of the LSB bit of the WID bitplane data to invalidate the Z value for a fast Z clear operation. The format of this register is shown in Figure 6.52.

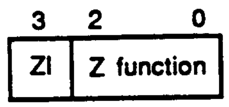


Figure 6.52 DEPTHFN Register

REptr index = 0x2F

Unbuffered Input Register

Bits 3 : Z invalidate (Write Only). This bit specifies if the Z value is valid or not.

0 - disable old Z invalidate
1 - enable old Z invalidate if (WID LSB = 1)

Bits 2-0 : Depth function (Write Only). These bits specify the relational comparison function performed by the Z comparator hardware.

000 - never passes
001 - passes if src < dest
010 - passes if src = dest
011 - passes if src <= dest
100 - passes if src > dest
101 - passes if src <> dest
110 - passes if src >= dest
111 - always passes

REPSTIP Register

This register contains the repeat count for the bits in the STIP register other than the LSB bit. When stipple checking is enabled with the ENABSTIP register the STIPCOUNT register determines how many pixel writes will be conditioned by the LSB bit in the STIP register before the STIP register is rotated right. The count in the REPSTIP register determines how many pixel writes will be conditioned by the each remaining bit in the STIP register before the STIP register is rotated right. The stipple pattern is used to condition pixel writes for the Draw Line instructions. The format of this register is shown in Figure 6.53.

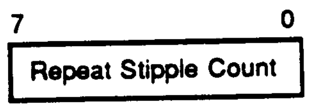


Figure 6.53 REPSTIP Register

REptr index = 0x30

Unbuffered Input Register

Bits 7-0 : Repeat Stipple Count (Write Only). These bits contain the repeat stipple count for all bits in the STIP register except the LSB bit.

ENABLWID Register

This register is used to enable or disable the WID checking for the line drawing instructions. The ENABWID register also must be enabled for the line drawing WID checking to be enabled. When enabled the value in the WID bitplanes at the current X, Y location are compared to the value in the CURWID register. If the values are the same the WID check passes. If the values are different the WID check fails. The format of this register is shown in Figure 6.54.

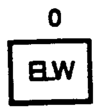


Figure 6.54 ENABLWID Register

REptr index = 0x31

Unbuffered Input Register

Bit 0 : Enable Line WID Checking (Write Only). This bit is used to enable or disable line WID checking. The ENABWID register must be set to one for this register to take effect.

0 - disable line WID checking
1 - enable line WID checking

FBOPTION Register

This register is used to inform the RE2 if the additional Frame Buffer bitplanes card is installed on the adapter. The first data parameter sent down to the microcode after the microcode download is written into the FBOPTION register by the microcode during the adapter initialization. The format of this register is shown in Figure 6.55.

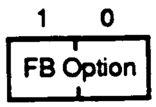


Figure 6.55 FBOPTION Register

REptr index = 0x32

Unbuffered Input Register

Bits 1-0 : FO (Write Only). These bits indicate if the additional Frame Buffer bitplanes card is installed on the adapter.

00 - 8 Frame Buffer, 2 PUP and 2 WID bitplanes installed
01 - 24 Frame Buffer, 2 PUP, 2 UAUX and 4 WID bitplanes installed
10 - Undefined
11 - 16 Frame Buffer, 2 PUP, 2 UAUX and 4 WID bitplanes installed (not supported)

TOPSCAN Register

This register is used to specify how many rows and cols are being displayed for the currently selected Display State Machine timing. The row count specifies the top row on the display and thus the first row which will be displayed. Row 0 is always the bottom row displayed on the screen. The col count specifies the number of columns on each scan line. Column 0 is always the left most column displayed on the screen. The col value is the starting col divided by 5. The host software must use the GE_LOADRE token to load the TOPSCAN register whenever it changes the monitor type in the Display Subsystem. The format of this register is shown in Figure 6.56.

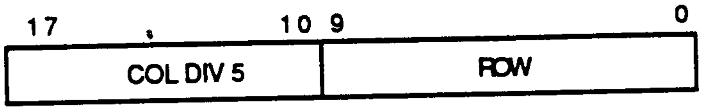


Figure 6.56 TOPSCAN Register

REptr index = 0x33

Unbuffered Input Register

Bits 17-10 : col (Write Only). These bits contain the col number of the first pixel to be displayed.

Bits 9-0 : row (Write Only). These bits contain the row number of the first pixel to be displayed.

monitor	topscan		
type	col	row	
60Hz	0x00	0x3FF	(1023)
30Hz	0x00	0x3FF	(1023)
PAL	0x00	0x23F	(575)
NTSC	0x00	0x1E4	(484)

ZBOPTION Register

This register is used to inform the RE2 if the Z Buffer card is installed on the adapter. The second data parameter sent down to the microcode after the microcode download is written into the ZBOPTION register by the microcode during the adapter initialization. The format of this register is shown in Figure 6.57.

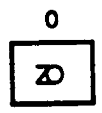


Figure 6.57 ZBOPTION Register

REptr index = 0x36

Unbuffered Input Register

Bit 0 : ZO (Write Only). This bit indicates if the Z Buffer card is installed on the adapter.

0 - Z Buffer card not installed
1 - Z Buffer card installed

XZOOM Register

This register specifies the x zoom factor which is applied to each pixel written during the execution of the Write Buffer instruction. Each pixel will be written x zoom times along the current scan line. The format of this register is shown in Figure 6.58.

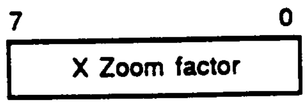


Figure 6.58 XZOOM Register

REptr index = 0x37

Unbuffered Input Register

Bits 7-0 : xzoom (Write Only). These bits contain the x zoom factor which is the number of times each pixel is written during the Write Buffer instruction.

UPACMODE Register

This register specifies the pixel packing used during the Write Buffer instructions. The 2 bit value specifies how many pixels are packed into each long word sent to the RE2 during the DMA operation. The HADDR register is used to specify the starting pixel offset in the first long word. The NUMPIX value will be the number of packed pixels written, not the number of long words sent to the RE2. The format of this register is shown in Figure 6.59.

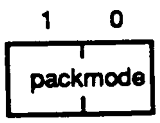


Figure 6.59 UPACMODE Register

REptr index = 0x38

Unbuffered Input Register

Bits 1-0 : packmode (Write Only). These bits specify the number of pixels which are packed into each long word received by the RE2 during a Write Buffer instruction.

00 - 1 pixel/long
01 - 2 pixels/long
10 - undefined
11 - 4 pixels/long

YMIN Register

This register specifies the bottom boundary of the hardware screen mask. Any pixel with a Y screen coordinate less than the value in the YMIN register will be clipped. This register does not affect pixel read operations. The hardware screen mask is always enabled for all pixel writes and can never be disabled. The format is shown in Figure 6.60.

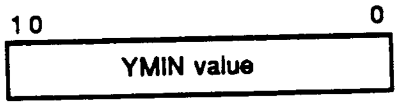


Figure 6.60 YMIN Register

REptr index = 0x39

Unbuffered Input Register

Bits 10-0 : ymin (Write Only). These bits contain the bottom boundary of the hardware screen mask.

YMAX Register

This register specifies the top boundary of the hardware screen mask. Any pixel with a Y screen coordinate greater than the value in the YMAX register will be clipped. This register does not affect pixel read operations. The hardware screen mask is always enabled for all pixel writes and can never be disabled. The format is shown in Figure 6.61.

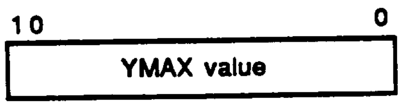


Figure 6.61 YMAX Register

REptr index = 0x3A

Unbuffered Input Register

Bits 10-0 : ymax (Write Only). These bits contain the top boundary of the hardware screen mask.

XMIN Register

This register specifies the left boundary of the hardware screen mask. Any pixel with an X screen coordinate less than the value in the XMIN register will be clipped. This register does not affect pixel read operations. The hardware screen mask is always enabled for all pixel writes and can never be disabled. The format of this register is in the special X DIV 5, X MOD 5 format as are all the X registers in the RE2. The format is shown in Figure 6.62.

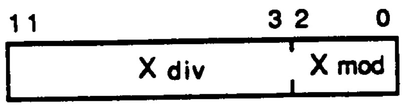


Figure 6.62 XMIN Register

REptr index = 0x3B

Unbuffered Input Register

Bits 11-0 : xmin (Write Only). These bits contain the left boundary of the hardware screen mask.

XMAX Register

This register specifies the right boundary of the hardware screen mask. Any pixel with an X screen coordinate greater than the value in the XMAX register will be clipped. This register does not affect pixel read operations. The hardware screen mask is always enabled for all pixel writes and can never be disabled. The format of this register is in the special X DIV 5, X MOD 5 format as are all the X registers in the RE2. The format is shown in Figure 6.63.

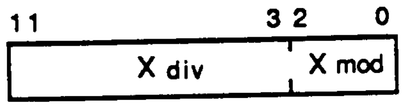


Figure 6.63 XMAX Register

REptr index = 0x3C

Unbuffered Input Register

Bits 11-0 : xmax (Write Only). These bits contain the right boundary of the hardware screen mask.

COLORCMP Register

This register is used to control the sources for the Z comparator hardware in the RE2. The normal usage is to compare new Z values with the current Z value already in the Z Buffer to perform hidden line removal operations. The Z comparator hardware can also be used in antialiased line drawing mode to improve the appearance of intersecting antialiased lines. In this mode the new Frame Buffer pixel value is compared to the pixel data already in the Frame Buffer. Only the brighter of the two pixels will be written at the points of intersection. The DEPTHFN register is used to specify the comparison operation between the selected values. The format of the COLORCMP register is shown in Figure 6.64.

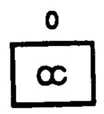


Figure 6.64 COLORCMP Register

REptr index = 0x3D

Unbuffered Input Register

Bit 0 : CC (Write Only). This bit specifies the compare source for the Z comparator hardware.

0 - Compare new Z value with Z value already in Z Buffer
1 - Compare new Frame Buffer pixel value with pixel value already in Frame Buffer

MEGOPTION Register

This register is used to inform the RE2 about the size of the VRAM chips installed on the adapter. The MGR adapter always has 1 MEG VRAMs installed so this register should always be set to one. The third data parameter sent down to the microcode after the microcode download is written into the MEGOPTION register by the microcode during the adapter initialization. The format of this register is shown in Figure 6.65.

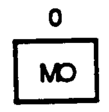


Figure 6.65 MEGOPTION Register

REptr index = 0x3E

Unbuffered Input Register

Bit 0 : MO (Write Only). This bit indicates if the MGR adapter has 256K or 1 MEG VRAM chips. The MGR adapter always has 1 MEG VRAM chips so this register is set to 1 always.

0 - 256K VRAM chips installed
1 - 1 MEG VRAM chips installed (always for the MGR adapter)

Cursor Registers

Each cursor chip contains the following registers:

- address register
- glyph RAM byte
- command register
- cursor X register
- cursor Y register
- window X register
- window Y register
- window width register
- window height register

The following paragraphs describe these registers.

Address Register

The address register contains the address of the glyph RAM bytes or the control registers. It is not initialized at power on or MGR reset. The address register is written as a low byte and a high byte. Only bit 0 in the high byte is used and the other 7 bits should be set to 0. The address register bytes may be written by the host at any time. The format of the address register is shown in Figure 6.66.

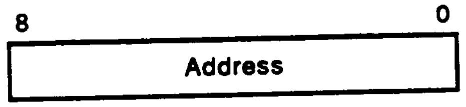


Figure 6.66 Address Register

Bits 8-0 : Address. These bits contain the address of the glyph RAM byte or the control register to be read or written.

Glyph RAM Byte

The bytes in the glyph RAM are used to hold the cursor bits which form the block cursor. The Glyph RAM contains 512 bytes. The bytes are not initialized at power on or MGR reset. They may be written by the host at any time. The format of each byte is shown in Figure 6.67.

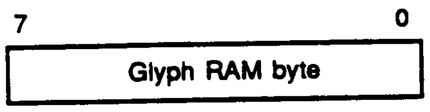


Figure 6.67 Glyph RAM Byte

Bits 7-0 : Glyph RAM byte. These bits contain 8 bits of the glyph RAM .

Command Register

The command register is used to control various functions of the cursor controller chip. It is not initialized at power on or MGR reset. It may be written by the host at any time. The format of the command register is shown in Figure 6.68.

7	6	5	4	3	2	1	0
0	RE	CHE	FC	Multiplex		Thickness	

Figure 6.68 Command Register

Bit 7 : Reserved. This bit should always be zero.

Bit 6 : Cursor Block RAM Enable (RE). This bit is used to disable or enable the user defined cursor Block RAM output.

0 - disables block cursor RAM output
1 - enables block cursor RAM output

Bit 5 : Cross Hair Enable (CHE). This bit is used to disable or enable the cross hair cursor output.

0 - disables cross hair cursor output
1 - enables cross hair cursor output

Bit 4 : Cursor Format Control (FC). This bit specifies the logical operation to be performed to combine the overlapping cursor output pixels if both the RAM cursor and the cross hair cursor are enabled.

0 - exclusive-OR overlapping pixels
1 - OR overlapping pixels

Bits 3-2 : Multiplex Control. These bits are used to select the number of outputs generated by the cursor controller.

00 - 1:1 multiplexing
01 - 4:1 multiplexing
10 - 5:1 multiplexing
11 - reserved

Bits 1-0 : Cross Hair Thickness. These bits specify the horizontal and vertical thickness of the cross hair cursor in pixels.

00 - 1 pixel
01 - 3 pixels
10 - 5 pixels
11 - 7 pixels

Cursor X Register

This register is used to specify the x coordinate of the center of the 64 x 64 block cursor glyph and the x location of the point of intersection of the cross hair cursor. The format of the cursor x register is shown in Figure 6.69. The register is 12 bits wide and is programmed as two bytes. The upper 4 bits of the upper byte are always zero. The upper left corner of the screen is position 0,0.

The cursor x value to be written is calculated as follows:

Cx = desired display screen x position + D + H - P

where

D = skew in pixels between the output cursor data and the external pixel data

H = number of pixels between the first rising edge of CLOCK following the falling edge of HSYNC to active video

P = 37 if 1:1 output multiplexing
= 52 if 4:1 output multiplexing
= 57 if 5:1 output multiplexing

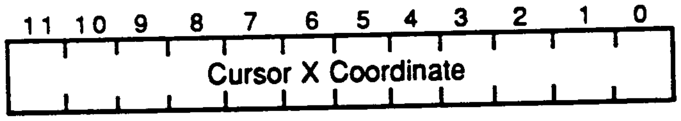


Figure 6.69 Cursor X Register

Bits 11-0 : Cursor X position. These bits specify the x location of the center of the cursor glyph or the point of intersection of the cross hair cursor.

000 - FFF cursor x values

Cursor Y Register

This register is used to specify the y coordinate of the center of the 64 x 64 block cursor or the point of intersection of the cross hair cursor. The format of the cursor y register is shown in Figure 6.70. The register is 12 bits wide and is programmed as two bytes. The upper 4 bits of the upper byte is always zero. After the upper byte has been written the cursor position is updated to the position specified in the Cursor X and Y registers. The upper left corner of the screen is position 0,0.

The cursor y value to be written is calculated as follows:

Cy = desired display screen y position + V - 32

where

V = number of scan lines from the first falling edge of HSYNC that is two or more clock cycles after the falling edge of VSYNC to active video

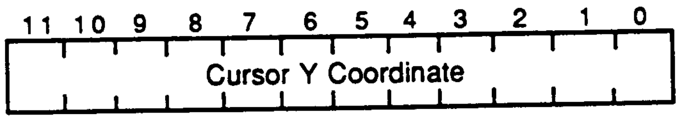


Figure 6.70 Cursor Y Register

Bits 11-0 : Cursor Y position. These bits specify the y location of the center of the cursor glyph or the point of intersection of the cross hair cursor. The 12 bit number is a 2's complement signed value.

0xFC0 - 0xFBF (-64 to +4031) may be loaded into the cursor y values. The negative values (0xFC0 to 0xFFF) are used to position the cursor off the top of the screen.

Window X Register

This register is used to specify the x coordinate of the upper left corner of the cross hair cursor window. The format of the window x register is shown in Figure 6.71. The register is 12 bits wide and is programmed as two bytes. The upper 4 bits of the upper byte are always zero. The upper left corner of the screen is position 0,0.

The window x value to be written is calculated as follows:

Wx = desired display screen x position + D + H - P

where

D = skew in pixels between the output cursor data and the external pixel data

H = number of pixels between the first rising edge of CLOCK following the falling edge of HSYNC to active video

P = 5 if 1:1 output multiplexing
= 20 if 4:1 output multiplexing
= 25 if 5:1 output multiplexing

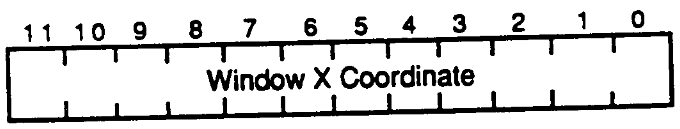


Figure 6.71 Window X Register

Bits 11-0 : Window X position. These bits specify the x location of the upper left corner of the cross hair cursor window.

000 - FFF window x values

Window Y Register

This register is used to specify the y coordinate of the upper left corner of the cross hair cursor. The format of the window y register is shown in Figure 6.72. The register is 12 bits wide and is programmed as two bytes. The upper 4 bits of the upper byte is always zero. After the upper byte has been written the window position is updated to the position specified in the Window X and Y registers. The upper left corner of the screen is position 0,0.

The cursor y value to be written is calculated as follows:

Wy = desired display screen y position + V

where

V = number of scan lines from the first falling edge of HSYNC that is two or more clock cycles after the falling edge of VSYNC to active video

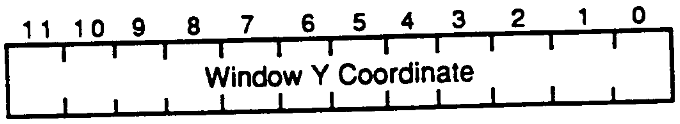


Figure 6.72 Window Y Register

Bits 11-0 : Window Y position. These bits specify the y location of the upper left corner of the cross hair cursor window.

000 - FFF window y values

Window Width Register

This register is used to specify the width of the cross hair cursor in pixels. The window width is written or read as two bytes with the upper four bits of the second byte undefined. The actual window width will be 2, 8 or 10 pixels greater than the window width specified in the register depending on whether 1:1, 4:1 or 5:1 output multiplexing is selected. Since 5:1 output multiplexing is specified for the MGR adapter the desired window width should have 10 subtracted from the value written into the window width register. The window width register is not initialized after power on or reset. The host software can access the register at any time. The format of this register is shown in Figure 6.73.

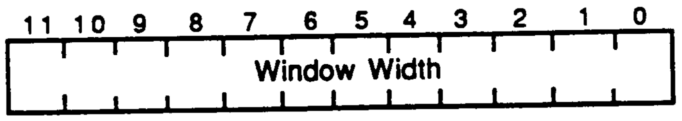


Figure 6.73 Window Width Register

Bits 11-0 : Window width. These bits specify the window width size in pixels.

Window Height Register

This register is used to specify the height of the cross hair cursor in pixels. The window height is written or read as two bytes with the upper four bits of the second byte undefined. The actual window height will be 2, 8 or 10 pixels greater than the window height specified in the register depending on whether 1:1, 4:1 or 5:1 output multiplexing is selected. Since 5:1 output multiplexing is specified for the MGR adapter the desired window height should have 10 subtracted from the value written into the window height register. The window height register is not initialized after power on or reset. The host software can access the register at any time. The format of this register is shown in Figure 6.74.

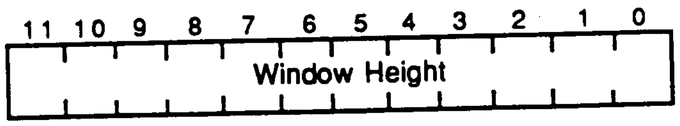


Figure 6.74 Window Height Register

Bits 11-0 : Window height. These bits specify the window height size in pixels.

Interrupts

The Raster Subsystem does not generate any interrupts to the host system.

RE2 Basic Operations

The RE2 provides support for calculating pixel addresses and data for horizontal spans and lines. It also provides DMA read and write support as well as the bitplane read and write control for accessing the Frame Buffer, PUP, UAUX, Z buffer and WID bitplanes. The RE2 can execute the following seven instructions:

- Draw Shaded Span
- Draw Flat 1 Span
- Draw Flat 4 Span
- Draw Top of Antialiased Line
- Draw Bottom of Antialiased Line
- Read Buffer (Read DMA support)
- Write Buffer (Write DMA support)

For each instruction the RE2 calculates pixel addresses for each pixel to be read or written. For the instructions that perform pixel writes the pixel data values for each pixel are calculated. The calculated pixel data can also have logical bitwise raster operations using the source values to be written and the destination values already in the bitplanes. The pixel data writes can then be conditioned with pattern masks, stipple patterns, screen mask clipping, Window ID checks and Z buffer depth comparison checks.

The RE2 is instructed to perform the various instructions by the Geometry Subsystem. The GE5 microcode loads the RE2 control registers with data and then loads an instruction in the Instruction Register. The RE2 executes the specified instruction using the values in the control registers and stores the generated pixel values in the appropriate bitplanes. The following paragraphs describe the RE2 operations involved in calculating the pixel addresses, calculating the pixel data, performing the raster operations and of applying the various pixel write conditioning. The operation of each instruction is described and the differences in the pixel data calculations and the pixel write conditioning are described.

Pixel Address Computation

Each of the RE2 instructions must calculate the pixel addresses from which the pixel data is read or written. For each RE2 instruction the GE5 microcode loads the initial screen position to be used for reading or writing pixels into the X and Y registers. The microcode also loads the X and Y delta values used to calculate new screen positions into the DX and DY registers. For the first pixel to be read or written the RE2 uses the initial pixel address stored by the GE5 microcode in the X and Y registers as the current pixel address. For each successive iteration of the instruction the RE2 adds the delta values in the DX and DY registers to the X and Y registers to get the new pixel address. The delta values to be loaded for each instruction will be discussed in the paragraphs which describe the instructions. Since the MGR adapter is a 3D graphics adapter each pixel also has a Z value associated with each X and Y screen location. The Z values can be used to control depth comparison operations. The Z values are stored in the Z Buffer and the use of these values are described in later paragraphs. In summary a pixels X and Y values control where the pixel is drawn on the screen and the pixels Z value can be used to control whether the pixel is drawn or not.

Raster Subsystem Bitplane Pixel Formats

The bitplane pixel formats and the operation of the RE2 are determined by the settings of the FBOPTION, the ZBOPTION and the MEGOPTION registers. The FBOPTION register specifies if the base configuration (FBOPTION = 0) or the enhanced configuration (FBOPTION = 1) of the adapter is being used. The FBOPTION register is set with the first data parameter sent down after the microcode has been downloaded. The ZBOPTION register is used to inform the RE2 if the optional Z Buffer card is installed. If ZBOPTION is zero then the card is not installed and if it is one then the card is installed. The ZBOPTION register is set with the second parameter sent down after the microcode has been downloaded. Finally the MEGOPTION register determines if the VRAM is composed of 256 K chips (MEGOPTION = 0) or 1 Meg chips (MEGOPTION = 1). The MEGOPTION register is set with the third parameter sent down after the microcode has been downloaded. The MGR adapter always uses 1 Meg VRAM chips so the MEGOPTION register should be set to one.

For each pixel address there can be up to 36 bits/pixel on the base adapter and up to 56 bits/pixel on the enhanced adapter. This pixel data is read and written through two data ports which are called the Frame Buffer port and the Z Buffer port as shown in Figure 6.75. The Z Buffer port always contains the Z Buffer and the WID data so that this data can be read together and used to condition writes from the Frame Buffer port. This arrangement allows the operations to be performed in a parallel manner to increase the performance.

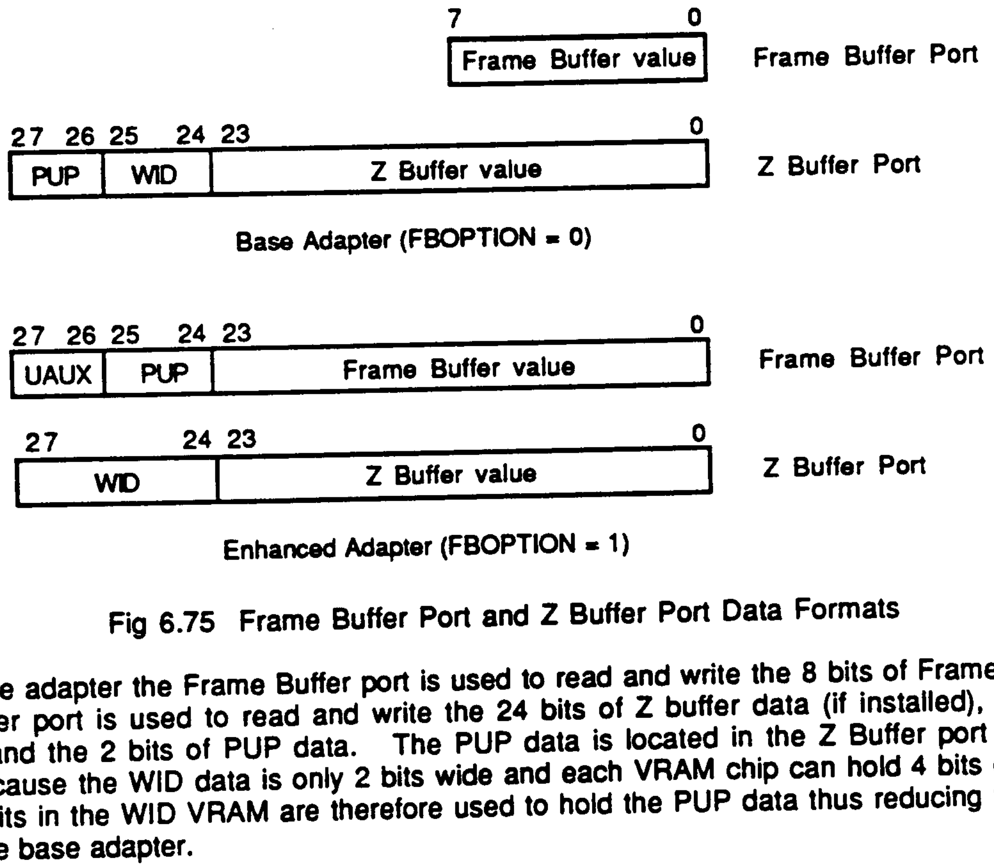


Fig 6.75 Frame Buffer Port and Z Buffer Port Data Formats

For the base adapter the Frame Buffer port is used to read and write the 8 bits of Frame Buffer data. The Z Buffer port is used to read and write the 24 bits of Z buffer data (if installed), the 2 bits of WID data and the 2 bits of PUP data. The PUP data is located in the Z Buffer port for the base adapter because the WID data is only 2 bits wide and each VRAM chip can hold 4 bits of data. The unused 2 bits in the WID VRAM are therefore used to hold the PUP data thus reducing the hardware costs for the base adapter.

For the enhanced adapter the Frame Buffer port is used to read the 24 bits of Frame Buffer data, the 2 bits of PUP data and the 2 bits of UAUX data. The Z Buffer port is used to read and write the 24 bits of Z buffer data (if installed) and the 4 bits of Window ID data.

Write masks are used to control which bits in the bitplanes are actually updated during each pixel write operation. The various pixel writes can be conditioned by the screen mask, pattern mask, line stipple patterns, WID checks and the Z Buffer checks. The write masks and the pixel write conditioning checks are described later.

The following paragraphs describe how the RE2 calculates the pixel data which is loaded into the Frame Buffer port and the Z buffer port for writing to the bitplanes. Pixel data read into the two ports from the bitplanes will of course already be in the appropriate format since it was originally written via these two ports.

RE2 Pixel Data Flows and Calculations

When the RE2 begins the execution of an instruction it uses the data loaded into the various color and data registers by the GE5 microcode to calculate the pixel data which is loaded into the Frame Buffer port and the Z Buffer port to be written into the bitplanes at the current pixel location specified in the X and Y registers. For the shaded span, flat span and line drawing instructions the RE2 writes the data in the Frame Buffer port and it determines when and if the Z Buffer port data is written. The pixel writes can be conditioned with bitplane write masks and various checks which determine if a pixel is written or not. After the current pixel is processed the RE2 updates the pixel location, the pixel color values and the Z value and then processes the next pixel. This continues for the number of iterations specified in the NUMPIX register. The following paragraphs describe the data flow through the RE2 for a single iteration of an instruction. The descriptions of the individual instructions indicate when and if the Z Buffer port is written.

The RE2 registers used to calculate the pixel data are the R, G and B color registers, the PUP, UAUX and WID data registers and the Z register. The R, G and B registers are loaded with the Frame Buffer pixel color values and the PUP and UAUX data registers are loaded with the overlay or underlay color values. The WID data register is loaded with the Window ID value used to perform WID checks and to control the pixel display by the Display Subsystem. The initial Z value is specified by the various tokens used to specify drawing operations and is loaded into the Z register. The host software uses the GE_DRAWMODE token to control the loading of the PUP, UAUX and WID data registers. The GE_COLOR or GE_COLORF tokens are used to specify the data values to be loaded into these registers.

The GE_DRAWMODE token is also used to select the Frame Buffer as the drawing destination for normal drawing operations. For the normal drawing mode the host controls the pixel type to be drawn into the Frame Buffer with the GE_PIXTYPE token. When the pixel type is specified as a color index pixel then the color index value is specified by the host software using the GE_COLOR or the GE_COLORF tokens. Before an instruction is executed the color index value is loaded into the R register. The G and B registers are ignored for color index pixels. If the pixel type is RGB then the three color values specified by the GE_RGBCOLOR token are loaded into the R, G and B registers before an instruction is executed. The following paragraphs describe the pixel data calculations and the data flows for the base and the enhanced adapter.

Base Adapter Data Flows and Calculations

The pixel data calculations and data flow through the RE2 for the base adapter are shown in Figure 6.76. The Frame Buffer port data for the base adapter consists of the 8 bit frame buffer pixel data only. The pixel type determines which type of pixel dithering and formatting operations are to be performed on the color data. The dithering and pixel formatting will be described later.

Once the color data has been formatted it represents the source data for the raster operation specified by the GE_RASTEROP token. The Frame Buffer bitplanes are read through the Frame Buffer port to get the destination input data for the raster operation. The bitwise logical operation is performed between the source and the destination input data. If the instruction is a line drawing instruction the antialiase operation is performed and the result is placed in the Frame Buffer port from which it is written to the bitplanes.

The Z Buffer port on the base adapter contains the 24 bit Z value, the 2 bit WID data and the 2 bit PUP data. The Z value is loaded by the GE5 microcode into the Z register. This value represents the Z depth value calculated by the microcode for the first pixel to be drawn. The Z value is placed unchanged into the Z buffer port. The GE_DRAWMODE token is used to select the PUP or WID data register to receive the index value specified with the GE_COLOR or the GE_COLORF tokens by the host software. These values are specified by the host using the GE_COLOR or the GE_COLORF tokens since they represent index values into the mode registers of the XPC1 or into the overlay color palette in the RGB RAMDAC in the Display Subsystem.

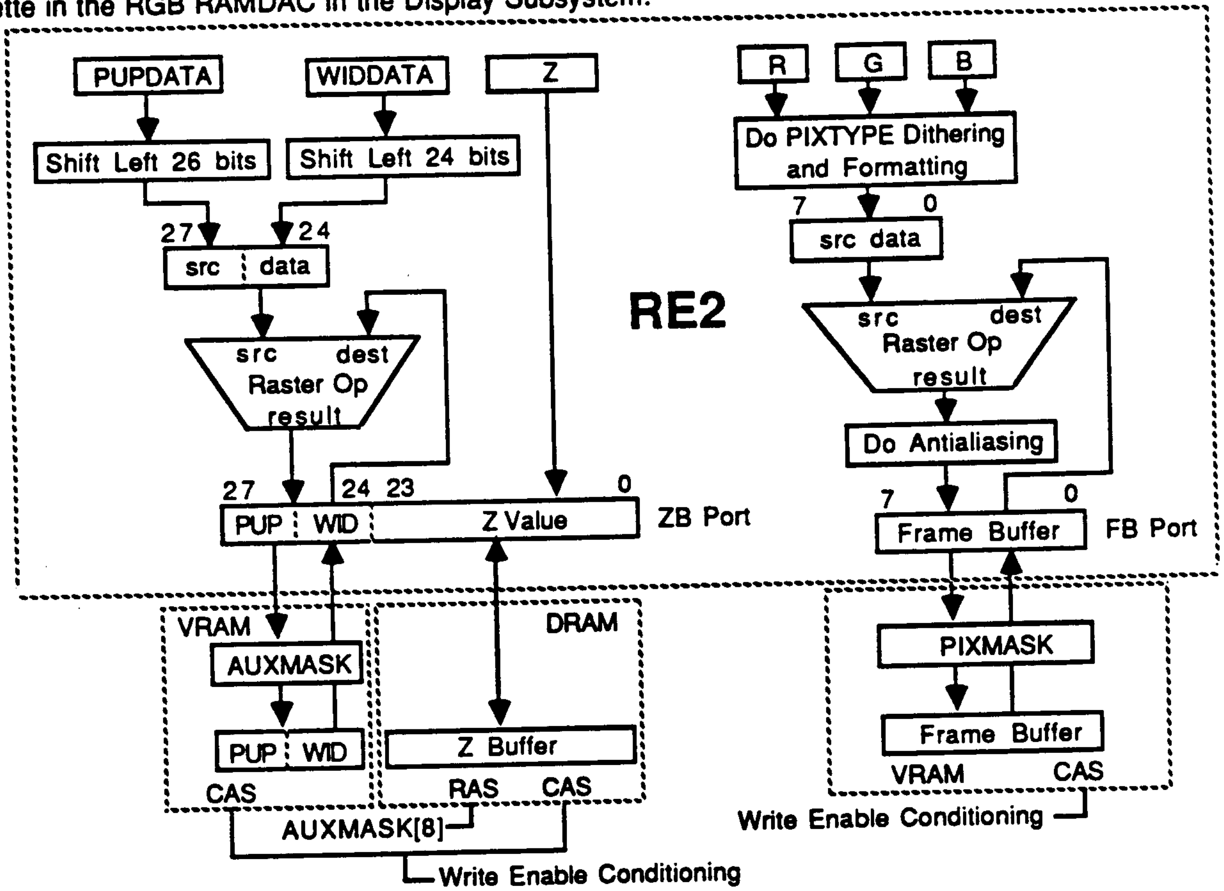


Figure 6.76 Base Adapter Pixel Data Flow

The WID data is shifted left 24 bits and the PUP data is shifted left 26 bits and is then used as the source data for the raster operation. The WID and PUP bitplanes are read through the Z buffer port to get the destination input for the raster operation. The bitwise logical operation is performed between the source and the destination input data and the result is placed in the Z Buffer port from which it and the Z value data are written to the bitplanes.

The Frame Buffer port and the Z Buffer port writes can be conditioned by the screen mask check, the Z buffer check, the WID check, the pattern mask check and the line stipple pattern check. These checks will be described in greater detail in later paragraphs and the instruction descriptions will describe which checks are performed for each instruction.

The initial color and Z values are loaded by the microcode along with the PUPDATA and the WIDDATA which was loaded with the GE_DRAWMODE and GE_COLOR tokens. For each successive iteration of an instruction the color values are updated by adding the color delta values in the DR, DG and DB registers to the current color values in the R, G and B registers. The Z value is updated by adding the delta Z value in the DZI and DZF registers to the current Z register value. The PUPDATA and the WIDDATA values remain the same for each instruction iteration.

Enhanced Adapter Data Flows and Calculations

The pixel data calculations and data flow through the RE2 for the enhanced adapter are shown in Figure 6.77. The Frame Buffer port data for the enhanced adapter consists of the 24 bit frame buffer data, the PUP data and the UAUX data.. The microcode loads the color data into the R, G and B registers. If the current pixel type specified with the GE_PIXTYPE token is color index then the microcode loads just the R register with the color index value specified by the host software using the GE_COLOR or the GE_COLORF tokens. If the pixel type is RGB then the R, G and B registers are loaded by the microcode with the three color values specified by the GE_RGBCOLOR token. The pixel type determines which type of pixel dithering and formatting operations are to be performed on the color data. The dithering and pixel formatting will be described later.

The PUPDATA and UAUXDATA are normally 2 bits each and represent an index into the auxiliary color map in the XMAP2 chips in the display subsystem. The GE_DRAWMODE token is used to select the PUP, UAUX or WID data register to receive the index value specified with the GE_COLOR or the GE_COLORF tokens by the host software. If the NOPUP register is set to one then the PUPDATA register is ignored and the UAUXDATA is 4 bits wide. If the NOPUP register is zero then the PUP data is shifted left 24 bits and the UAUXDATA is shifted left 26 bits and then they are placed in the source data for the raster operation. If the NOPUP register is one then the UAUX data is shifted left 24 bits and the four bits are placed in the source data.

The dithered and formatted color data, the PUP data and the UAUX data represents the source data for the raster operation specified by the GE_RASTEROP token. The Frame Buffer, PUP and UAUX bitplanes are read through the Frame Buffer port to get the destination input data for the raster operation. The bitwise logical operation is performed between the source and the destination input data. If the instruction is a line drawing instruction the antialiase operation is performed and the result is placed in the Frame Buffer port from which it is written to the bitplanes.

The Z Buffer port on the enhanced adapter contains the 24 bit Z value and the 4 bit WID data. The Z value is loaded by the GE5 microcode into the Z register. This value represents the Z depth value calculated by the microcode for the first pixel to be drawn. The Z value is placed unchanged into the Z buffer port. The GE_DRAWMODE token is used to select the WIDDATA register for loading by the host software using the GE_COLOR or the GE_COLORF tokens. The WID data is an index into the mode registers of the XMAP2 chips in the Display Subsystem. The WID data is shifted left 24 bits and is then used as the source data for the raster operation. The WID bitplanes are read through the Z buffer port to get the destination input for the raster operation. The bitwise logical operation is performed between the source and the destination input data and the result is placed in the Z Buffer port from which it and the Z value data are written to the bitplanes. The Frame Buffer port write can be conditioned by the screen mask check, the Z buffer check, the WID check, the pattern mask check and the line stipple pattern check which will be described later.

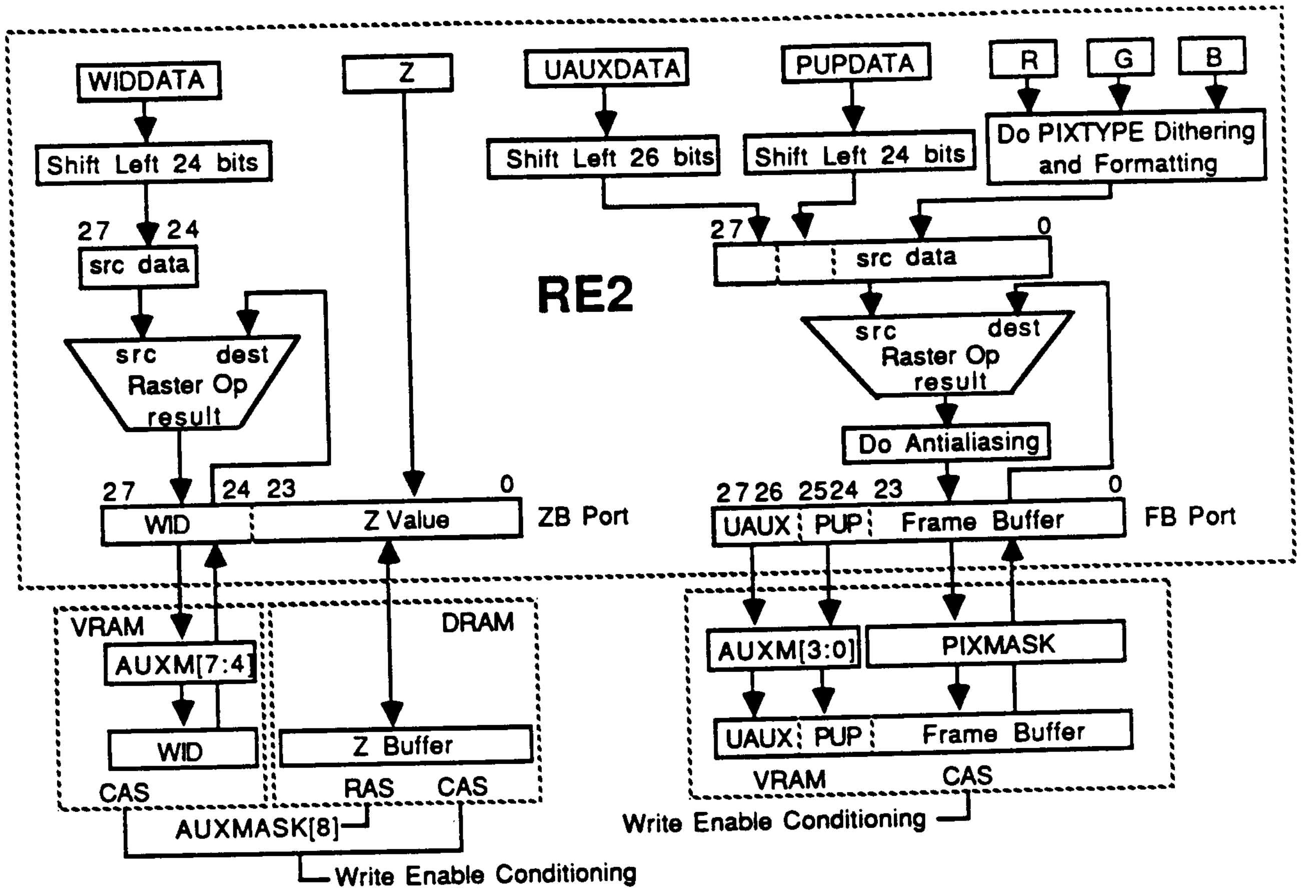


Figure 6.77 Enhanced Adapter Pixel Data Flow

The Frame Buffer port and the Z Buffer port writes can be conditioned by the screen mask check, the Z buffer check, the WID check, the pattern mask check and the line stipple pattern check. These checks will be described in greater detail in later paragraphs and the instruction descriptions will describe which checks are performed for each instruction.

The initial color and Z values are loaded by the microcode along with the PUPDATA, UAUXDATA and the WIDDATA which had been previously loaded with the GE_DRAWMODE and GE_COLOR tokens. For each successive iteration of an instruction the color values are updated by adding the color delta values in the DR, DG and DB registers to the current color values in the R, G and B registers. The Z value is updated by adding the delta Z value in the DZI and DZF registers to the current Z register value. The PUPDATA, UAUXDATA and the WIDDATA values remain the same for each instruction iteration.

The following paragraphs describe the Frame Buffer pixel and the Z value calculations that are performed by the RE2 to get the appropriate values for each pixel written during the execution of an instruction.

Frame Buffer Pixel Data Formation

The Frame Buffer pixel data is formed from the color data placed in the R, G and B color registers. The initial color values are loaded by the microcode along with the delta color values. During the first pixel data calculation the initial color values are used. Then during successive iterations of an instruction the DR, DG and DB delta color values are added to the current R, G and B color values to get the new color values. For each iteration of the instruction being executed the Frame Buffer pixel data values are calculated by performing dithering and various pixel packing operations on the data in the R, G and B registers. The result of these operations is the source data for the raster operations. The result of the raster operation is the input value for the antialiasing operation for line drawing instructions. The results are then placed into the frame buffer bitplanes.

The GE_PIXTYPE token is used to specify the pixel type and this value is loaded into the PIXTYPE register. If the PIXTYPE register is set for a color index pixel type then only the R register is used for the color index value which is specified with the GE_COLOR or the GE_COLORF token. If an RGB pixel type is selected then the R, G and B registers are loaded with the three color values specified with the GE_RGBCOLOR token. The GE5 microcode may modify the specified RGB color values depending on whether lighting mode operations are being performed. In this case the initial color values loaded into the R, G and B registers may be slightly different than those specified by the host. The RE2 supports the following types of pixels:

- PIXTYPE = 00 (Single Buffered Only)
    - 24 bit RGB (enhanced adapter only)
    - 8 bit RGB (base adapter only)
- PIXTYPE = 01 (Double Buffered)
    - 12 bit RGB (enhanced adapter only)
- PIXTYPE = 10 (Double Buffered)
    - 8 bit color index pixels (base adapter only)
    - 12 bit color index pixels (enhanced adapter only)
- PIXTYPE = 11 (Double Buffered)
    - 4 bit color index pixels (both adapters but usually only the base adapter)

The following paragraphs describe the dithering calculations and the pixel formatting operations performed to calculate the Frame Buffer pixel data for each pixel type.

Frame Buffer Pixel Dithering

The host software controls the enabling and disabling of dithering with the GE_ENABDITH operation. The enable/disable flag sent by this token is loaded into the ENABDITH register in the RE2. If the ENABDITH register is set to one then the dithering operation is performed on the pixel color data to adjust the color value in the R, G and B registers. If the ENABDITH register is zero then the dithering operations are not performed. The dithering operation is designed to cause a limited number of pixel colors to appear as more colors. The dithering operation maps the 4 x 4 dither matrix shown in Figure 6.78 to each 4 x 4 group of pixels on the screen. The low 2 bits of the X and the Y pixel address are used to index into the dither matrix. The dither matrix value indexed by the current pixel address is compared with 4 bits from the current pixel colors in the R, G and B registers.

For the 8 or 12 bit RGB pixels the lower nibble of the integer portion of each color component is compared with the indexed dither matrix value. If the lower nibble is greater than the indexed dither matrix value then the upper nibble of the color component will be incremented. For the Color Index pixels the most significant 4 bits of the fractional part of the R register is compared to the indexed dither matrix value. If the 4 fractional color bits are greater than the dither value then the lower 8 bits of the color index value in the integer part of the R register are incremented. The dithering operation causes a percentage of each 4 x 4 group of Frame Buffer pixels to be the unincremented color value and the remainder to be the incremented color value. The percentage of pixels that are incremented depends on the size of the color bits used for the comparison. If the color bits used for the comparison are 8 then a checkerboard pattern will be written with the colors alternating between the original color value and the incremented color value.

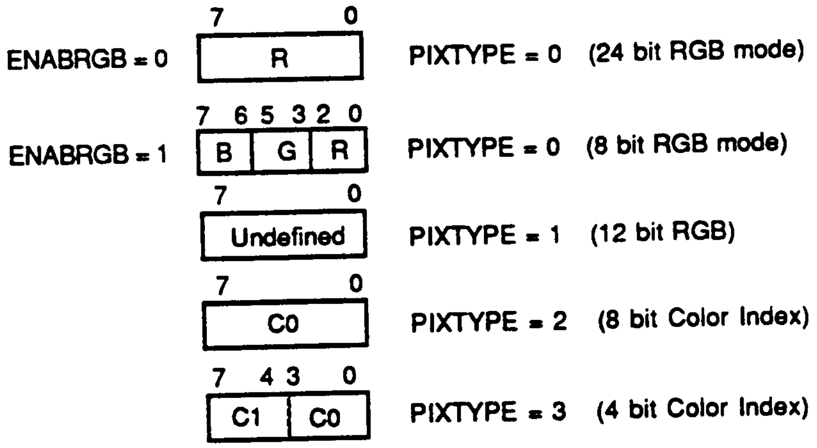
Y[1:0] / X[1:0]	0	1	2	3
0	0	8	2	10
1	12	4	14	6
2	3	11	1	9
3	15	7	13	5

Figure 6.78 RE2 Dither Matrix

The following paragraphs describe the frame buffer pixel data formating operations.

Frame Buffer Pixel Data Operations

The Frame Buffer pixel calculations depend on the setting of the FBOPTION register and the PIXTYPE register. The FBOPTION register specifies whether the base (FBOPTION = 0) or enhanced (FBOPTION = 1) adapter is being used The PIXTYPE register value loaded by the GE_PIXTYPE token determines the type of dithering and formatting operations which are performed on the color data in the R, G and B registers to create the frame buffer pixel data. The pixel formats for the base adapter are shown in Figure 6.79. This configuration supports 8 bit RGB pixels and 4 and 8 bit Color Index pixels.



7 0
ENABRGB = 0
R
PIXTYPE = 0 (24 bit RGB mode)
7 6 5 3 2 0
ENABRGB = 1
B G R
PIXTYPE = 0 (8 bit RGB mode)
7 0
Undefined
PIXTYPE = 1 (12 bit RGB)
7 0
C0
PIXTYPE = 2 (8 bit Color Index)
7 4 3 0
C1 C0
PIXTYPE = 3 (4 bit Color Index)

Figure 6.79 Base Adapter Frame Buffer Pixel Formats

The R, G and B color registers consist of an integer portion and a fraction portion. Only the integer portion of the registers is passed on to the source input to the raster operation. The pixel format diagrams for the base and enhanced adapter only show the integer portion of the Frame Buffer pixel formats since that is all that is written into the Frame Buffer. The Frame Buffer pixel formats for the enhanced adapter are shown in Figure 6.80. This configuration supports 12 and 24 bit RGB pixels and 4 and 12 bit Color Index pixels.

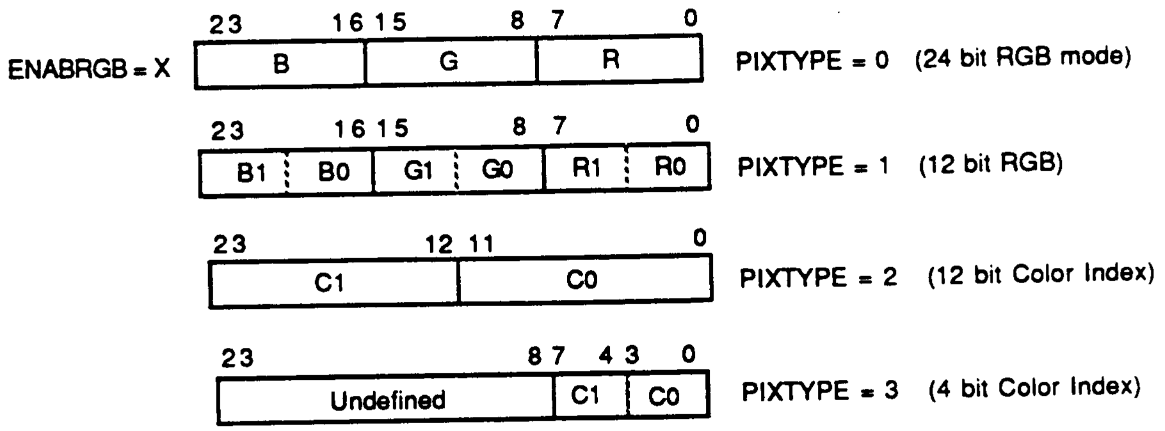


Figure 6.80 Enhanced Adapter Frame Buffer Pixel Formats

The RE2 uses the value in the ENABDITH register specified by the GE_ENABDITH token to determine if dithering will be performed on the color data.The following paragraphs describe the operations performed by the RE2 on the color data in the R, G and B registers to produce these Frame Buffer pixel formats.

PIXTYPE 0 Data Formation

For the base configuration the ENABRGB register determines the pixel formatting done on the color data. If the ENABRGB register is a zero the RE2 will attempt to format the data into 24 bit RGB format. Since the Frame Buffer in the base configuration only has 8 bitplanes only the red color component in the R register will be written to the bitplanes. This mode of operation is not recommended and is only provided for compatibility with the RE1 chip. The normal mode of operation for the base adapter with PIXTYPE 0 is to have the ENABRGB register set to one so that the special 8 bit RGB formatting is performed. Dithering operations are not allowed for the 24 bit RGB pixels. The GE_LOADRE token can be used by the host software to set the ENABRGB register.

When the pixel data is written into the Frame Buffer bitplanes the PIXMASK specified with the GE_PIXWRITEMASK token controls which bits are written. For the 8 bit RGB pixels the PIXMASK should be set to 0x0000FF to allow all 8 bits to be written. For the 24 bit RGB pixels the PIXMASK should be set to 0xFFFFFF to allow all 24 bits to be written.

The dithering and formatting operations for PIXTYPE 0 are shown in Figure 6.81. If FBOPTION is one then the three src_data bytes are written with the values in the R, G and B registers without any changes. If FBOPTION is zero then the dithering operation is performed if ENABDITH is one and then the special 8 bit RGB formatting is performed. The upper 3 bits of the Red color are copied to the lower 3 bits of the src_data byte. The upper 3 bits of the Green color are copied to the middle 3 bits of the src_data byte. Finally the upper 2 bits of the Blue color are copied to the upper 2 bits of the src_data byte. The src_data value is then sent to the source input to the raster operation hardware.

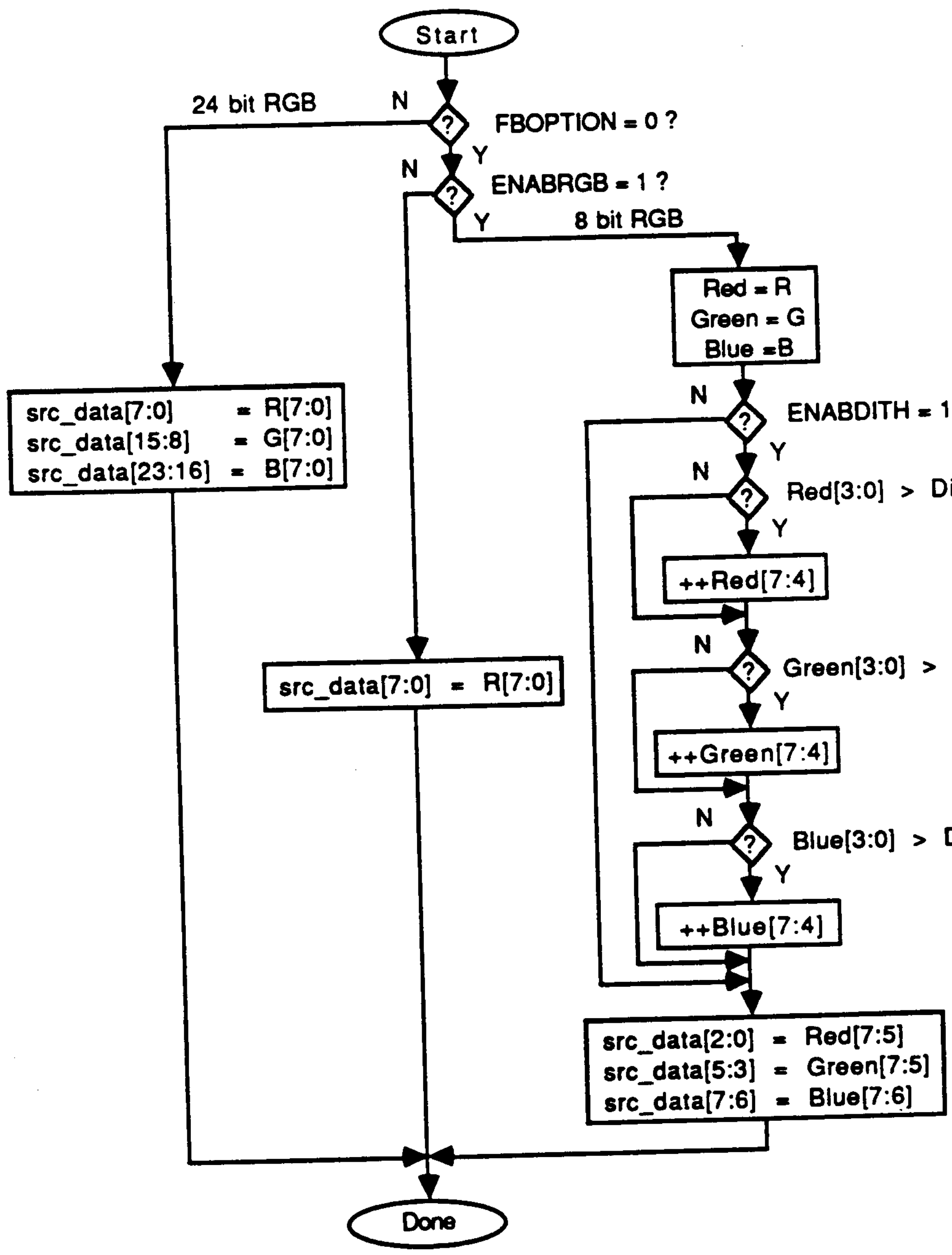


Figure 6.81 PIXTYPE 0 Flow Chart

For the enhanced adapter the RGB data is not changed by the pixel dithering and formatting circuitry. It is passed from the R, G and B registers directly to the source input to the raster operation. For the 8 bit RGB pixels the dithering is performed using the color data from the R, G and B registers as shown in Figure 6.82.

The lower nibble of the integer part of a color component is compared with the indexed dither matrix value. If the color nibble value is greater than the matrix value then the upper nibble of the color value is incremented. If the color nibble value is less than or equal to the matrix value then the upper nibble of the color component is left unchanged. Each of the three color components are dithered in the same manner.

PIXTYPE = 0, FBOPTION = 1 (24 bit RGB mode)

Dithering Invalid for 24 bit RGB pixels

PIXTYPE = 0, FBOPTION = 0 ENABRGB = 1 (8 bit RGB mode)

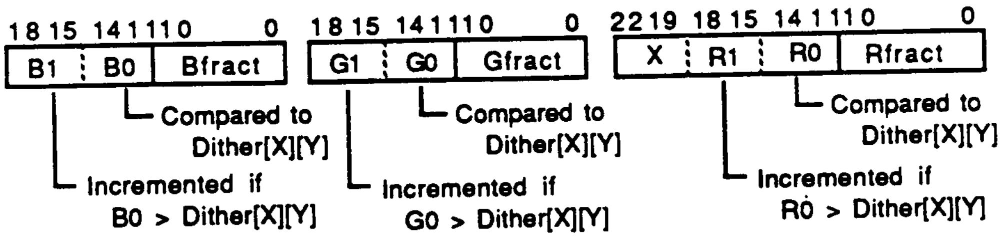


Figure 6.82 PIXTYPE 0 Dithering

PIXTYPE 1 Data Formation

For the base adapter the 12 bit RGB double buffered pixel type is not defined. For the enhanced adapter the dithering operation is performed if the ENABDITH register is set. Dithering is allowed for 12 bit RGB pixels since the number of pixels colors is limited by the double buffering operations used for this pixel. For the 12 bit RGB pixels the dithering is performed using the color data from the R, G and B registers as shown in Figure 6.83.

The lower nibble of the integer part of a color component is compared with the indexed dither matrix value. If the color nibble value is greater than the matrix value then the upper nibble of the color value is incremented. If the color nibble value is less than or equal to the matrix value then the upper nibble of the color component is left unchanged. Each of the three color components are dithered in the same manner.

PIXTYPE = 1, FBOPTION = 1 only (12 bit RGB mode)

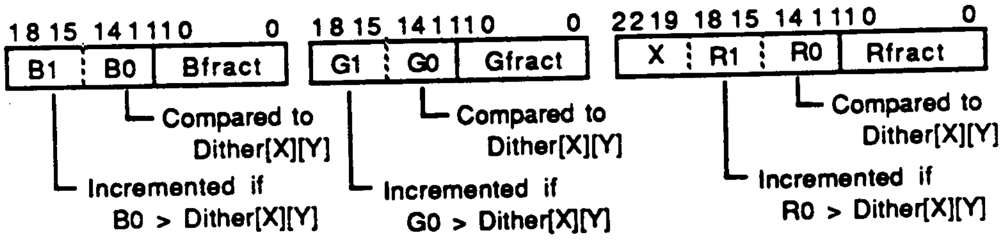


Figure 6.83 PIXTYPE 1 Dithering

The dithering and formatting operations for PIXTYPE 1 are shown in Figure 6.84. If FBOPTION is zero then the src_data byte is undefined. If FBOPTION is one the dither operation is performed if the ENABRGB register is one. The upper nibble of each color component is duplicated into the lower nibble and then the three src_data bytes are written with the three color values which result.

When the pixel data is written into the Frame Buffer bitplanes the PIXMASK specified with the GE_PIXWRITEMASK token controls which bits are written. For the 12 bit RGB double buffered pixels the PIXMASK controls which buffer is written. To write the front buffer pixels the PIXMASK should be set to 0x0F0F0F. To write the back buffer pixels the PIXMASK should be set to

0xF0F0F0. To write to both buffers the PIXMASK should be set to 0xFFFFFF to allow all 24 bits to be written.

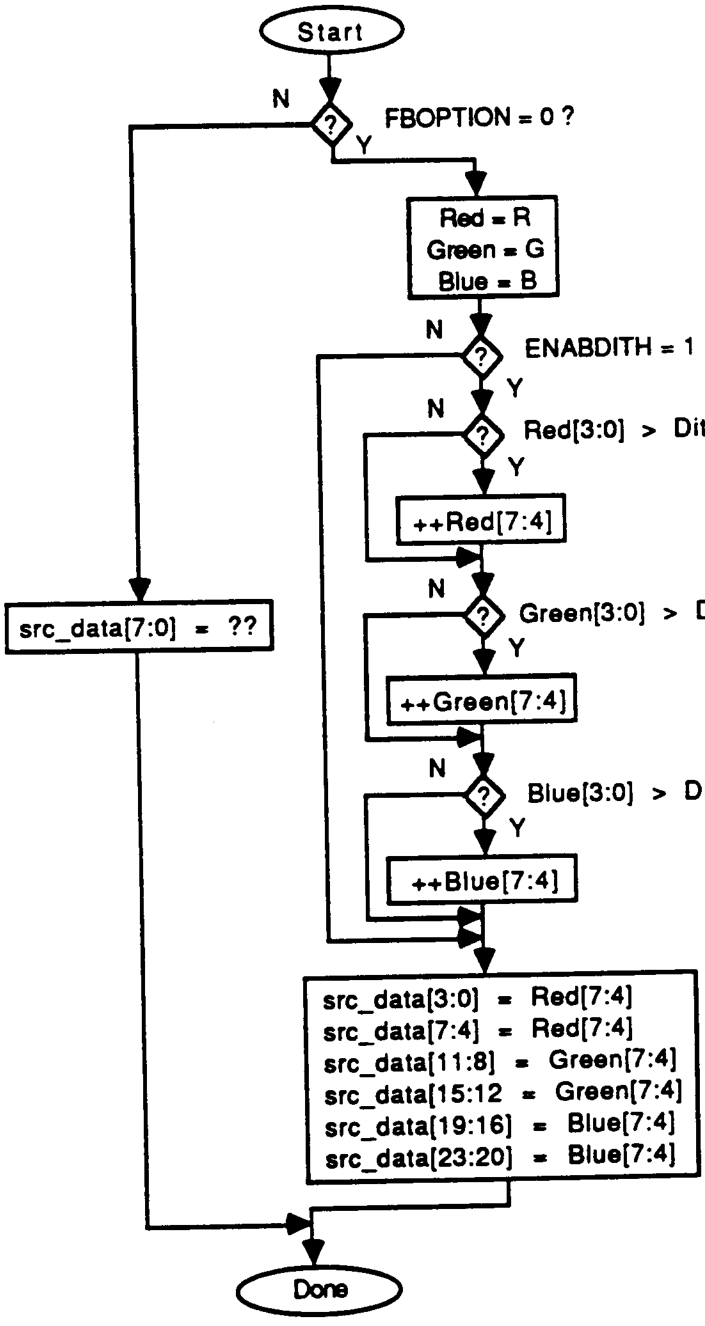


Figure 6.84 PIXTYPE 1 Flow Chart

PIXTYPE 2 Data Formation

For PIXTYPE 2 the color index pixel value is placed in just the R register which has a 12 bit integer portion and an 11 bit fractional part. If the ENABDITH register is one then the color index in the R register is dithered. As shown in Figure 6.85 the most significant 4 bits of the fractional part of the color index is compared to the indexed dither matrix value. If the fractional part is greater than the dither matrix value the lower 8 bits of the integer part of the color index pixel is incremented. The upper 4 bits of a 12 bit color index pixel are never affected by the dithering operation.

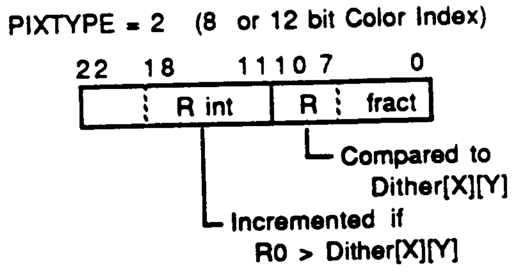

PIXTYPE = 2 (8 or 12 bit Color Index)
22
18
11
10
7
0
R int
R
fract
Compared to Dither[X][Y]
Incremented if R0 > Dither[X][Y]

Figure 6.85 PIXTYPE 2 Dithering

The dithering and data flow for PIXTYPE 2 are shown in Figure 6.86. For the base adapter the 8 bit color index value can be dithered before it is placed in src_data. For the enhanced adapter the 12 bit color index value is dithered and then the 12 bit color index is duplicated into both the lower and upper 12 bits of the src_data. The src_data is then passed to the source input of the raster operation.

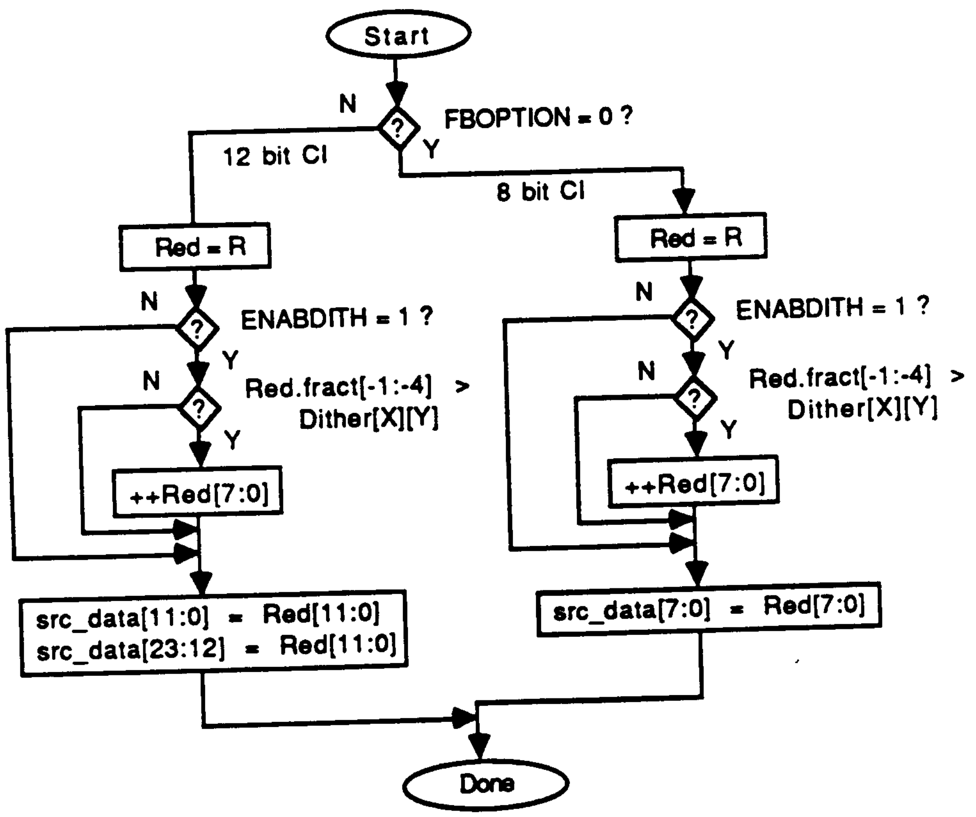


Figure 6.86 PIXTYPE 2 Flow Chart

When the pixel data is written into the Frame Buffer bitplanes the PIXMASK specified with the GE_PIXWRITEMASK token controls which bits are written. For the 8 bit CI pixels the PIXMASK should be set to 0x0000FF to allow all 8 bits to be written. For the 12 bit CI double buffered pixels the write mask is used to control which buffer is written. To write the front buffer pixel data the PIXMASK should be set to 0x000FFF. To write the back buffer pixel data the PIXMASK should be set to 0xFFF000. To write to both buffers the PIXMASK should be set to 0xFFFFFF to allow all 24 bits to be written.

PIXTYPE 3 Data Formation

For PIXTYPE 3 the color index pixel value is placed in just the R register which has a 12 bit integer portion and an 11 bit fractional part. The 4 bit color index pixel is used on the base adapter and it could be used on the enhanced adapter but the 12 bit color index pixel provides greater color resolution.

The dithering and data flow for PIXTYPE 3 are shown in Figure 6.87. For both the base and enhanced adapter the 4 bit color index value is dithered and then the 4 bit color index is duplicated into both the lower and upper nibbles of the right most byte of src_data. For the enhanced adapter the upper 16 bits of src_data will be undefined. The src_data is then passed to the source input of the raster operation.

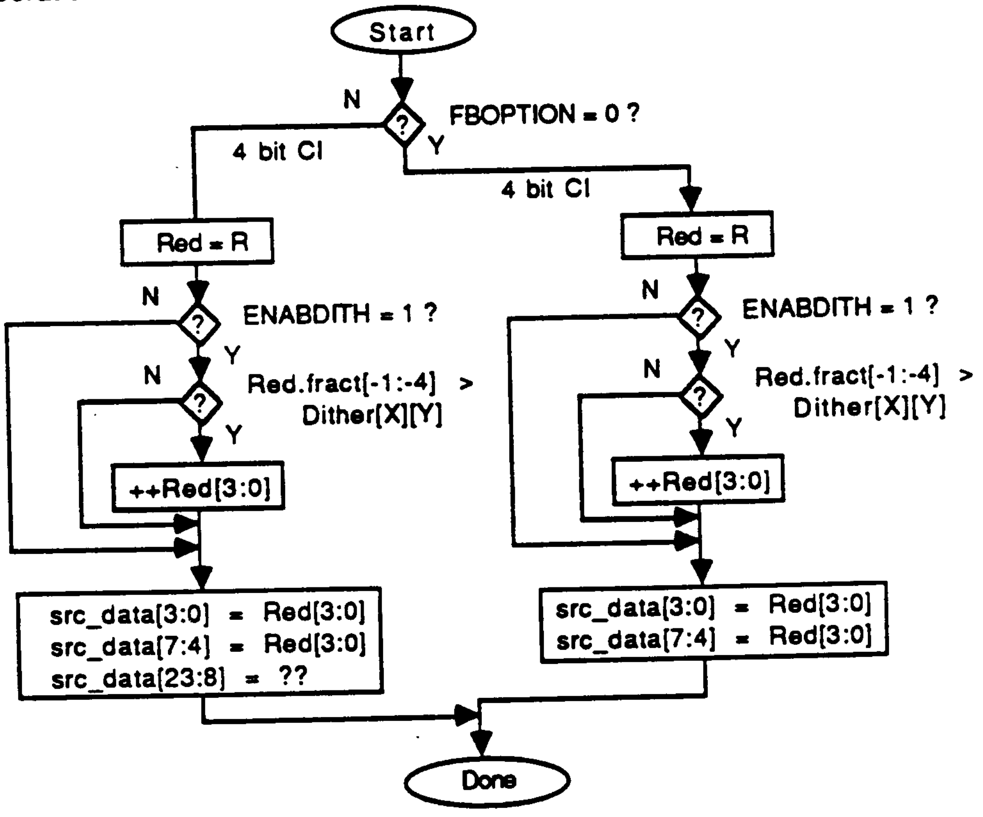


Figure 6.87 PIXTYPE 3 Flow Chart

If the ENABDITH register is one then the color index in the R register is dithered. As shown in Figure 6.88 the most significant 4 bits of the fractional part of the color index is compared to the indexed dither matrix value. If the fractional part is greater than the dither matrix value the lower 4 bits of the integer part of the color index pixel is incremented.

When the pixel data is written into the Frame Buffer bitplanes the PIXMASK specified with the GE_PIXWRITEMASK token controls which bits are written. For the 4 bit CI double buffered pixels the write mask is used to control which buffer is written. To write the front buffer pixel data the PIXMASK should be set to 0x00000F. To write the back buffer pixel data the PIXMASK should be set to 0x0000F0. To write to both buffers the PIXMASK should be set to 0x0000FF to allow all 8 bits to be written.

PIXTYPE = 3 (4 bit Color Index)

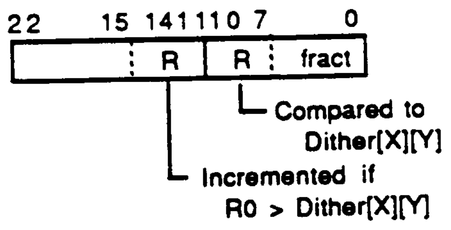


Figure 6.88 PIXTYPE 3 Dithering

Z Buffer Pixel Data Computation

The data written into the Z buffer represents the Z value of the pixel location. As mentioned previously the X and Y values of the pixel are used to determine the pixel location on the screen while the Z value is used in depth comparisons to control whether pixels are updated. The initial Z value is specified by the host software as part of the data sent with the various pixel drawing commands. The initial Z value is loaded into the Z register in the RE2 by the GE5 microcode. The microcode also determines the rate at which the Z value changes and loads the integer portion of the delta Z value into the DZI register and loads the fractional portion of the delta Z value into the DZF register. The Z register represents the integer portion of the Z value. The RE2 has an initial fractional Z register which is always set to zero at the start of an instruction. The fractional part of the Z register is not accessible by the microcode. The integer portion of the Z value is a 24 bit 2's complement number.

For the first iteration of an instruction the RE2 uses the value in the Z register and loads this value directly into the Z Buffer port so it can be written to the Z Buffer. For the following iterations of an instruction the RE2 adds the delta Z values in the DZI and DZF registers to the current value in the Z register. This new Z value is then loaded into the Z Buffer port for possible writing to the Z Buffer. The individual instruction description will indicate when and if the Z buffer port is written.

The Z Buffer writes can be conditioned by the Z Buffer write mask bit which is bit 8 of the AUXMASK register. If this bit is zero then the Z Buffer bitplanes will not be written even if the Z Buffer port is written. The Z Buffer port writes can be conditioned with the pattern mask checks, the stipple pattern checks, the WID checks, the Z Buffer checks and the screen mask checks. These checks are described in the next section. The Z Buffer port writes will not be performed if the upper five bits of the AUXMASK are all zero. This feature is provided to improve performance since Z Buffer port writes slow down the execution of the instructions which write pixels.

Pixel Data Raster Operations

The RE2 provides the capability of performing raster operations between the source data to be written into the bitplanes and the destination data already in the bitplanes. The bitwise logical operation to be performed between the source and the destination data is specified with the GE_RASTEROP token whose data parameter is stored in the FUNC register in the RE2. The legal values for the FUNC register are shown in Table 6.2.

The raster operations can be performed on the source and destination data for the Frame Buffer, PUP, UAUX and WID bitplanes. The Z Buffer bitplanes are not affected by the raster operation.

When the raster operation is set to any value other than 3 (Copy) the RE2 operates slower than when the raster operation is set to 3. This is caused by the RE2 having to first read the destination value from the bitplanes and then write the result value back to the bitplanes. This disrupts the pipelining operations in the RE2. When the RE2 is reset the FUNC register is set to three.

Table 6.2 Raster Operation Values

FUNC	Name	Result of Raster Op
0	Clear	0
1	And	src AND dst
2	And Reverse	src AND NOT dst
3	Copy	src
4	And Inverted	NOT src AND dst
5	Noop	dst
6	Xor	src XOR dst
7	Or	src OR dst
8	Nor	NOT src AND NOT dst
9	Equiv	NOT src XOR dst
10	Invert	NOT dst
11	Or Reverse	src OR NOT dst
12	Copy Inverted	NOT src
13	Or Inverted	NOT src OR dst
14	Nand	NOT src OR NOT dst
15	Set	1

NOT - 1's Complement

All sixteen raster operations can be used with the Shaded Span instruction, the Top and Bottom Line instructions and the Write Buffer instruction. For the Flat 1 Span and Flat 4 Span instructions the raster operation must be set to 3. The raster operation has no affect on the Read Buffer instruction. For the Write Buffer instruction the raster operation must be set to 3 if pixel packing or x zooming are performed.

Antialiasing Calculations

An antialiased line is drawn by drawing two adjacent lines with the draw top of antialiased line and the draw bottom of antialiased lines. Each line has antialiasing calculations performed on each pixel data value. After the Frame Buffer pixel data raster operation has produced a result the value is placed into the Frame Buffer pixel data portion of the Frame Buffer port to be written to the Frame Buffer bitplanes. If the RE2 is executing a line drawing instruction it performs the antialiasing calculations on the raster operation result data before it is placed into the frame buffer port. The RE2 determines the minor axis from the size of the value in the DX or DY register. The minor axis is the axis which has the smallest delta value which means that axis changes the slowest. The upper

3 bits of the fraction portion of the X or Y register which is the minor axis is used as an index into the antialiase weight table shown in Table 6.3.

Table 6.3 Antialiase Weights

3 MSB fraction bits	Top Line Weight	Bottom Line Weight
000	0010	1101
001	0100	1011
010	0110	1001
011	0111	1000
100	1000	0111
101	1001	0110
110	1011	0100
111	1101	0010

The value selected by the index depends on whether the instruction is a draw top of line instruction or a draw bottom of line instruction. A large value for the weight indicates that the line covers a large portion of the pixel. The selected 4 bit antialiase weight is then placed into the nibbles selected by the ASLECT register. Each bit in the ASELECT register corresponds to a nibble in the 24 bit frame buffer value. For each bit in ASELECT which is a one the corresponding nibble in the frame buffer value is replaced with the selected antialiase weight value. The result of the antialiase calculations are then placed into the Frame Buffer port and are written to the Frame Buffer bitplanes.

For RGB pixels the antialiased lines can be drawn on a black background with a line that has a primary color, a secondary color or white. The limitation on antialiased lines for RGB pixels is that the color components of the background color that are non-zero must be equal to the corresponding color components in the line color. The SGI Graphics Library does not support antialiased lines for RGB pixels.

For 12 bit color index pixels the antialiase weight could be used to select the LSB of the nibble with the remaining 8 bits representing a line color value index. This requires the color map to be programmed as 256 different color ramps. Each color ramp contains 16 entries which provide 16 different shades for the current color. This allows the antialiase weight to select different color shades to smooth the line drawing. The same approach could be used for 8 bit and 4 bit color index values. The 8 bit color index values are not double buffered while the 12 bit and 4 bit color index values are double buffered. The GE_ANTIALIASE token is used to specify the value to be written into the ASELECT register. For the 8 bit and 12 bit color index pixels the value is 9 (001001b) which causes the least significant nibble for each buffer to be replaced with the antialiase weight selected with the 3 MSB bits of the fraction portion of the minor axis value. For the 4 bit color index pixels the value specified with the GE_ANTIALIASE token is 3 (000011b) which causes the 4 bit color index to the replaced with the antialiase weight for both buffers.

Wide antialiased lines can be drawn as a sandwich of 1 or more non-antialiased lines between the top and bottom half of an an antialiased line. Depth-cued antialiased lines can be drawn in color index mode by using 4 bits of the color index for the antialiasing weight and 3 or 4 bits of the index for the depth-cued intensity. The color map would contain the depth-cued intensity scaled by the antialiasing weight blended with the background color.

The GE_ANTIALIASE token data value should be 0 to disable antialiasing. With this value no nibbles are replaced with the antialiase weight value. Either draw line instruction can be used to draw non-antialiased lines.

The COLORCMP register allows the Z comparator hardware to be used to compare the new Frame Buffer pixel value with the pixel value already in the Frame Buffer hardware. This comparison is enabled when the COLORCMP register is set to one. This should only be done when antialiased lines are being drawn. The GE_ZSOURCE token is used to set the COLORCMP register. The DEPTHFN register should be set to greater than or equal when colors are being compared. This causes the brightest pixels to be drawn at the points of intersection when antialiased lines are drawn. The GE_ZFUNCTION token is used to specify the relational function which is loaded into the DEPTHFN register.

The XYFRAC register can be used to specify an initial fraction bits for the minor axis. The fraction bits in the XYFRAC register are used to select the antialiase weight for the first pixel in the line. This makes the space between antialiased lines appear constant on slow moving wireframes with closely spaced lines.

Pixel Data Write Conditioning Checks

Pixel writes by the RE2 can be conditioned both on a bit by bit basis and on a complete pixel basis. The PIXMASK and the AUXMASK write masks are provided to mask the writing of the individual bits during a pixel write. Five pixel write conditioning checks are provided which enable or disable the writing of the entire pixel. These checks include the Screen Mask check, the Pattern Mask check, the Line Stipple Pattern check, the Z Buffer check and the WID check. The following paragraphs describe the bitplane write masks and the pixel write conditioning checks performed by the RE2.

Bitplane Pixel Write Masks

The two bitplane write masks provided in the RE2 include the PIXMASK and the AUXMASK. The PIXMASK is used to mask Frame Buffer bitplane writes and the AUXMASK is used to mask PUP, UAUX, WID and Z Buffer bitplane writes. The following paragraphs describe these two write masks.

PIXMASK Write Mask

The PIXMASK register holds the Frame Buffer pixel write mask. The PIXMASK value is specified by the GE_PIXWRITEMASK token which causes the microcode to load the specified pixel mask into the PIXMASK register. Before the Frame Buffer data in the Frame Buffer port is written out to the Frame Buffer bitplanes the PIXMASK register is output to the Frame Buffer VRAM. The VRAM then prevents writes to the bitplanes whose PIXMASK bits are zero and enables writes to the bitplanes whose PIXMASK bits are one. For the base adapter the PIXMASK is 8 bits wide and for the enhanced adapter the PIXMASK is 24 bits wide.

AUXMASK Write Mask

The AUXMASK register holds the write mask for the PUP, UAUX, WID and Z Buffer bitplanes as shown in Figure 6.89. The AUXMASK value is specified by the GE_AUXWRITEMASK token which causes the microcode to load the specified write mask into the AUXMASK register. Before the PUP, UAUX or WID data in the Frame Buffer port or the Z Buffer port is written out the appropriate AUXMASK bits are written out to the VRAM. The VRAM then prevents writes to the bitplanes whose AUXMASK bits are zero and enables writes to the bitplanes whose AUXMASK bits are one. The Z Buffer mask bit is bit 8 of the AUXMASK register. Since the Z Buffer is comprised of DRAM instead of VRAM they do not provide the built bit masking feature. The Z mask bit is connected to the Row Address Strobe (RAS) bits of the DRAM. When the Z mask bit is zero it prevents writes into all 24 of the Z Buffer bitplanes and when the Z mask is a one then it allows writes into all of the Z Buffer bitplanes.

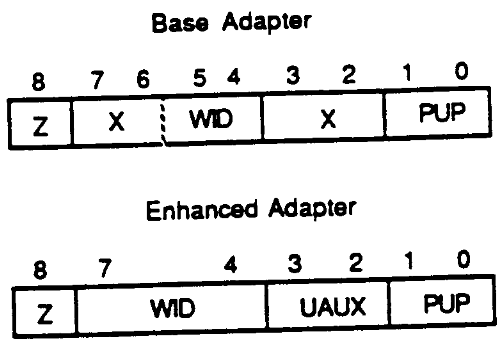


Figure 6.89 AUXMASK Bit Definitions

Pixel Write Conditioning Checks

The following paragraphs describe the five pixel write conditioning checks which are used to either enable or disable the writing of a complete pixel.

Screen Mask Check

The screen mask is applied to every pixel write operation that is performed by the RE2. The screen mask is specified by the GE_SCRMASK token and is used to define a rectangular clipping region on the screen. The microcode loads the data parameters sent with the GE_SCRMASK token into the XMIN, YMIN, XMAX and YMAX registers. If the current X or Y location of the pixel address is outside the screen mask rectangle then the pixel is clipped and the pixel write does not occur. If the pixel location is not outside the screen mask rectangle then the other pixel write checks determine if the pixel is written or not. If all of the other checks pass then the pixel is written and if any of them fail the pixel is not written. This is shown in the flow chart of Figure 6.90.

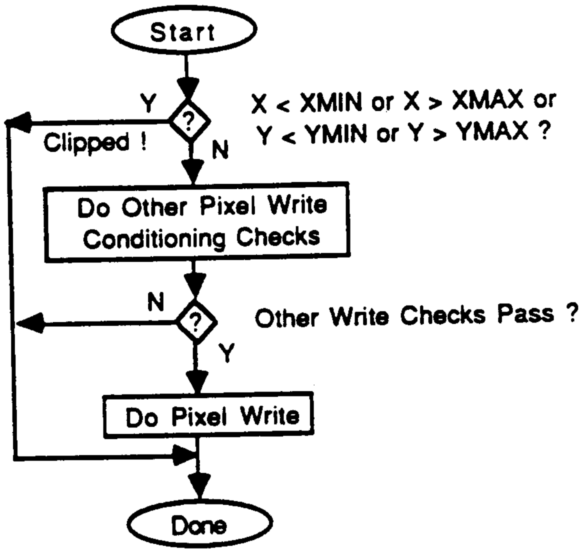


Figure 6.90 Screen Mask Check Flow Chart

As stated above the screen mask clipping is performed for all pixel write operations by all of the RE2 instructions. The screen mask check does not affect the operation of the Read Buffer instruction. The following four write checks are performed for some but not all of the instructions that write pixels. They can also be selectively enabled or disabled by the host software.

Pattern Mask Check

Two of the RE2 instructions can perform pattern masking of pixels as they are written into the bitplanes. The Draw Shaded Span instruction and the Write Buffer instruction can use the pattern masking capability. The Draw Flat Span instructions, the Draw Line instructions and the Read Buffer instruction cannot use the pattern mask capability. The microcode does not support the use of pattern masking for the Write Buffer instruction even though the RE2 supports it.

The host software uses the GE_SETPATTERN token to load a 32 x 32 pattern table into the GE5 data RAM. The GE_PATTERN token is used to instruct the microcode to use the pattern table to setup the RE2 registers to perform pattern masking or to disable RE2 pattern masking. The pattern mask is currently only used for pattern filled polygons and for text character drawing. If pattern mask usage is enabled then the microcode accesses the pattern table to get the appropriate line based on the current scan line location. The 32 bits from the selected line of the pattern table are loaded into the PATL and PATH register in the RE2. The ENABPAT register is used to enable or disable the use of the pattern mask by the RE2. When the pattern mask is enabled the mask is applied to 32 adjacent pixels along the current scan line. After the 32nd pixel is drawn the mask is then applied to the next group of 32 pixels and so on until the end of the scan line is reached or until the specified number of pixels are drawn for the current instruction. The flow chart in Figure 6.91 shows the pattern mask check operations performed by the RE2 for a single pixel write.

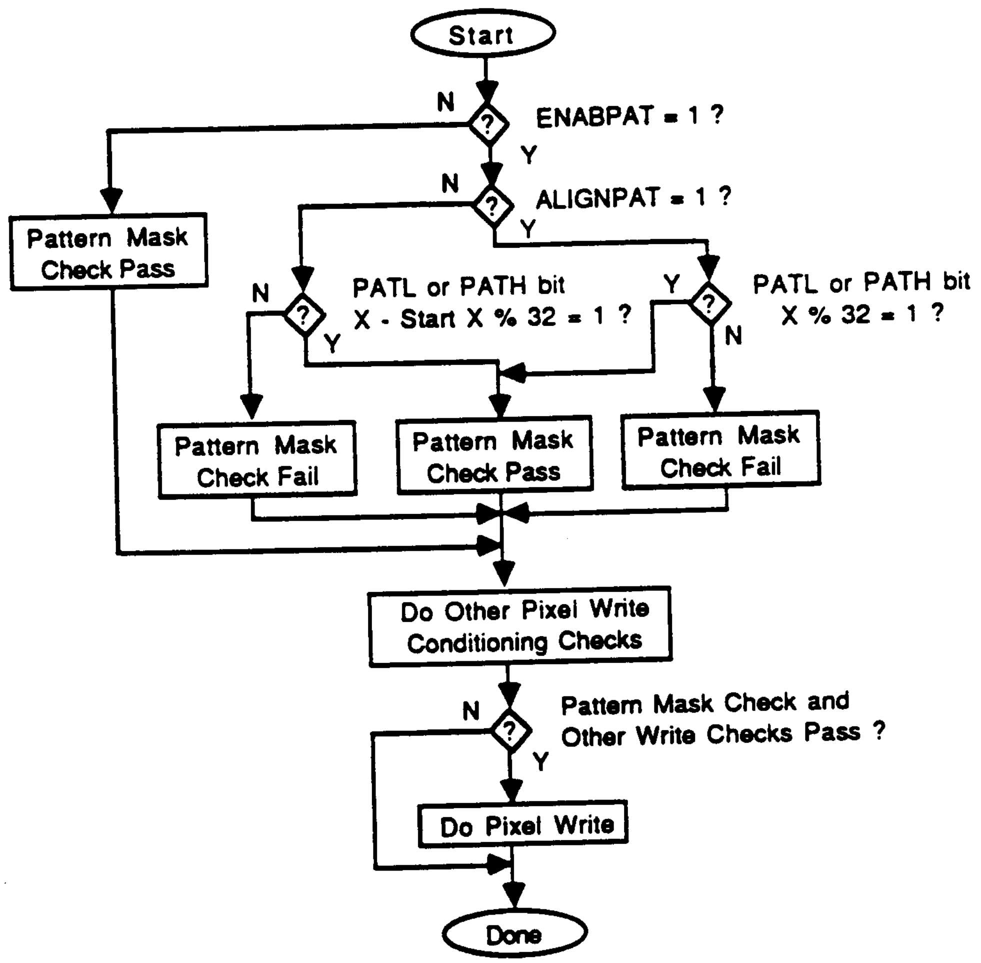

Start
N
ENABPAT = 1 ?
Y
N
ALIGNPAT = 1 ?
Y
Pattern Mask Check Pass
N
PATL or PATH bit X - Start X % 32 = 1 ?
Y
Y
PATL or PATH bit X % 32 = 1 ?
N
Pattern Mask Check Fail
Pattern Mask Check Pass
Pattern Mask Check Fail
Do Other Pixel Write Conditioning Checks
N
Pattern Mask Check and Other Write Checks Pass ?
Y
Do Pixel Write
Done

Figure 6.91 Pattern Mask Check Flow Chart

The ALIGNPAT register determines if the pattern mask is applied with a screen alignment or a drawing relative alignment. When screen alignment is selected the 32 bit pattern mask is applied to each group of 32 pixels beginning with the pixel at the leftmost edge of the screen. Therefore the current X location on the current scan line modulo 32 is used to index into the pattern mask registers to get the appropriate pattern mask bit. When the drawing relative alignment is selected the 32 bit pattern mask is applied beginning with the starting X location of the first pixel to be drawn. The current X minus the starting X on the current scan line modulo 32 is used to index into the pattern mask registers to get the appropriate mask bit. If the selected pattern mask is a one then the pattern mask check passes. If it is zero then the pattern mask check fails. If the pattern mask check fails then the pixel is not written regardless of the other pixel write checks. If the pattern mask check passes then the all the other write checks must also pass for the pixel to be written.

The alignment selection is controlled by the microcode and is not controllable by the host software. The microcode will select screen alignment while doing pattern filled polygons and will select drawing relative alignment when drawing text characters. For the text character drawing the font bitmap for a character is sent down to the microcode using the GE_DRAWCHAR token. For each row of bits in the font bitmap the pattern mask registers are loaded and a Draw Shaded Span instruction is executed.

Line Stipple Pattern Check

The Line Stipple Pattern Check is used only by the Draw Top of Line and the Draw Bottom of Line instructions. None of the other RE2 instructions are affected by the stipple pattern checks. The stipple check operation is shown in the flow chart of Figure 6.92.

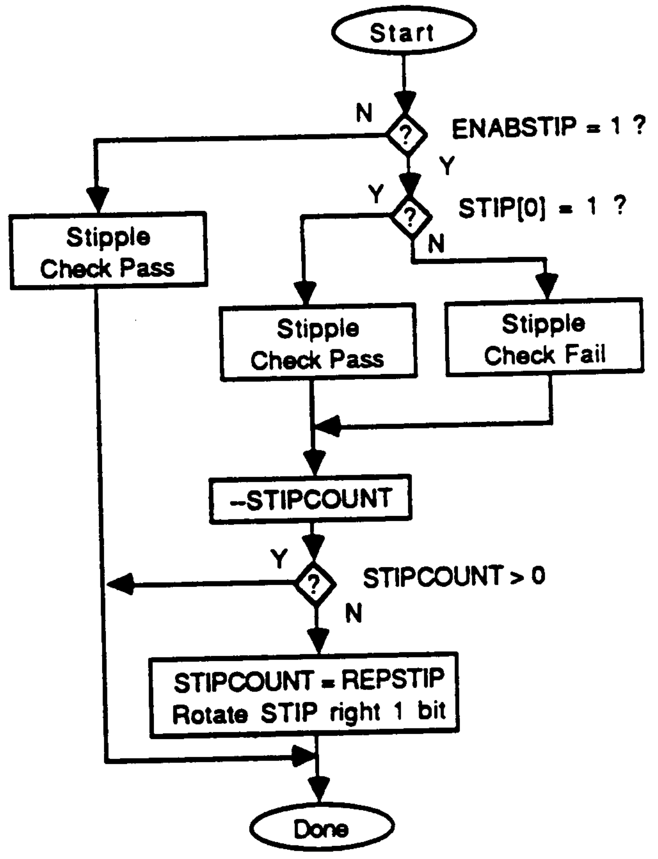


Figure 6.92 Stipple Pattern Check Flow Chart

This flow chart is intended to show the stipple check operation performed for each pixel as the line is being drawn. The ENABSTIP register controls whether the stipple pattern check is performed or not. If the register is a one then the stipple pattern check is enabled and if it is zero then the stipple pattern is ignored which can be considered as a pass condition. The STIP register contains the stipple pattern specified with the GE_LINESTYLE token. The stipple pattern allows the drawing of solid lines or patterned lines such as a dashed line. The LSB bit in the STIP register determines whether the stipple check for the current pixel passes or fails. If the LSB bit is a one then the stipple check passes and the pixel is written if the other write checks also pass. If the LSB bit is zero then the stipple check fails and the pixel is not written regardless of whether the other write checks pass.

The count value in the STIPCOUNT register determines how many times the current LSB bit in STIP register is used to perform the stipple check. After each stipple check is performed for the current pixel the STIPCOUNT register is decremented and when it becomes zero the stipple pattern in the STIP register is rotated right one bit and the value in the REPSTIP register is loaded into the STIPCOUNT register. This means that when the Draw Line instruction is started the count in the STIPCOUNT register determines how many times the initial LSB bit is used and the value in the REPSTIP register determines how many times each successive bit in the stipple pattern is used. The GE_LSREPEAT token is used to specify the value which is loaded by the microcode into the REPSTIP and the STIPCOUNT registers.

To get a continuous stipple across segments of a polyline, the microcode should only load the STIP, STIPCOUNT and REPSTIP register for the first Draw Line instruction. For the remaining Draw Line instructions the microcode should not change the values in the STIP register and the STIPCOUNT register. This causes the current stipple to be continued for each of the line segments in the polyline. To restart the stipple pattern the STIP, STIPCOUNT and REPSTIP register should be loaded with the original values.

Z or Color Comparison Check

The RE2 has a hardware comparator that can be used to perform Z comparisons for hidden line removal or color comparisons for antialiased lines. The COLORCMP register is used to control which type of comparison is performed. When the COLORCMP register is set to 0 the comparator is used to compare the Z register value with the Z bitplane value. When the COLORCMP register is set to 1 the comparator is used to compare the Frame Buffer pixel value with the data in the Frame Buffer bitplanes.

The normal mode of operation is set the COLORCMP register to 0 to perform Z Buffer depth comparisons. The result of the comparison is used to condition the pixel writes. The Z Buffer checking is performed for the Shaded Span, Draw Line and Write Buffer instructions. When the Write Buffer instruction is being executed the Z Buffer check is not performed for writes to the Z Buffer port. The Z Buffer checking is not done for the Flat Span or the Read Buffer instructions. The color compare mode is only used when antialiased lines are being drawn with the Draw Line instructions. The operations performed by the comparator hardware is shown in Figure 6.93.

The DEPTHFN register determines the type of relational comparison which is performed by the comparator hardware. As shown in Table 6.4, bits 0, 1 and 2 control the type of comparison which is performed between the new Z value and the Z Buffer bitplane data or between the Frame Buffer pixel color value and the Frame Buffer bitplanes. If DEPTHFN bits 0, 1 and 2 are all set to one then the comparison always passes and the pixel is written. This has the affect of disabling the Z Buffer checking or color comparison checking.

When the COLORCMP register is 0 the default setting for these three bits is a 3 which means that a new pixel is written only if it's Z value is <= to the previous Z value already written into the Z buffer at the current pixel location. For the left handed screen coordinate system pixels further from the viewer have greater Z values. With the DEPTHFN register set to 3 this means that only pixels with a Z value closer to the viewer are written. This allows objects to be drawn in any order and hidden pixels are not drawn.

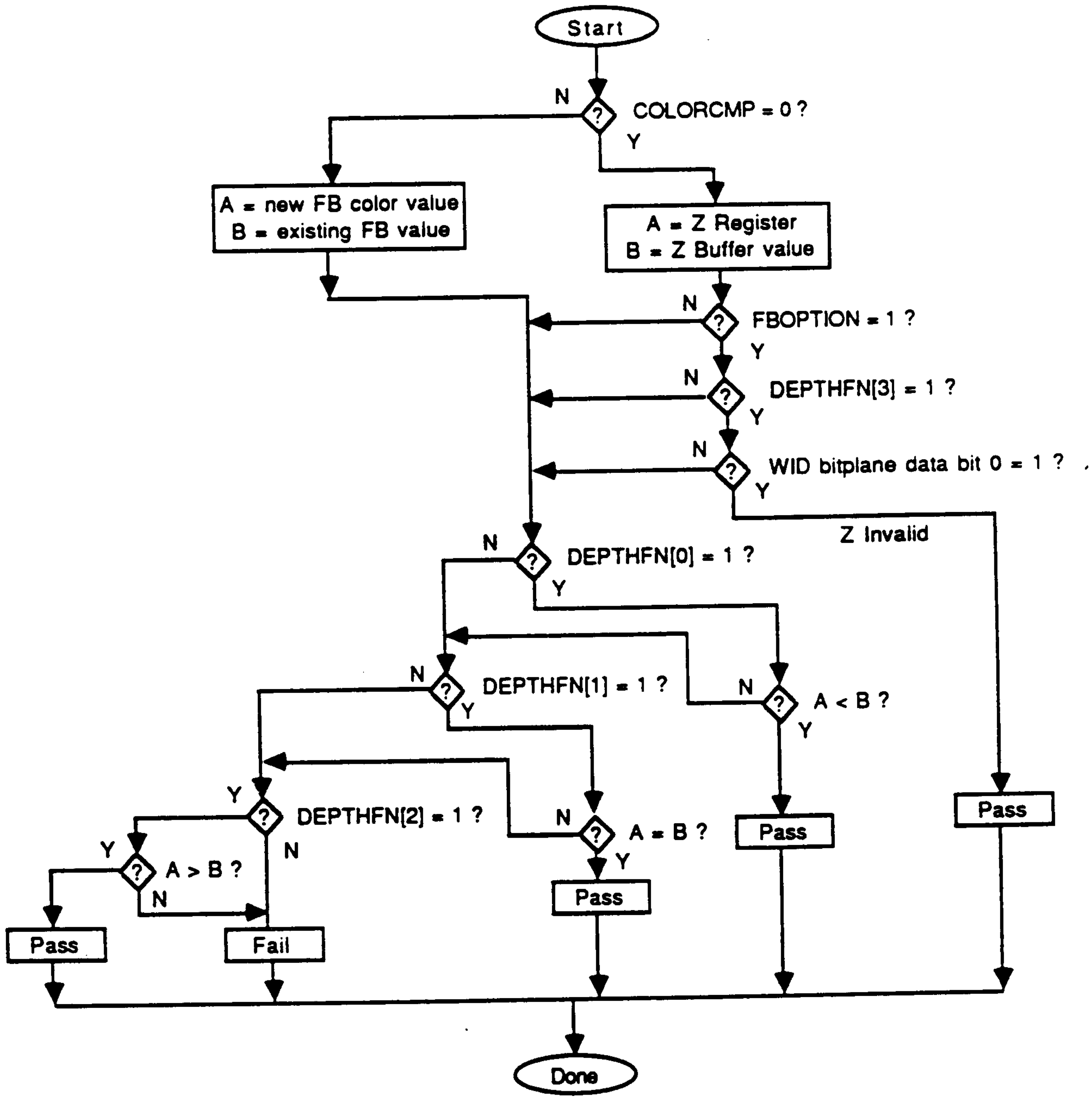


Figure 6.93 Z or Color Comparison Flow Chart

When the COLORCMP register is set to 1 the comparator is used to compare the Frame Buffer pixel values and the Frame Buffer bitplanes. This mode is only used when drawing antialiased lines so that the appearance of the intersecting lines is improved. The DEPTHFN register should be set to 6 in this mode so that only the brighter pixels at the points of intersection are drawn.

A special Z invalidate mode is controlled by bit 3 of the DEPTHFN register. This mode is only used when the COLORCMP register is 0 for Z Buffer checking mode. If bit 3 is zero then the Z Buffer value is valid and the lower 3 bits of the DEPTHFN register control the Z check comparison. If Bit 3 of DEPTHFN is 1 then the LSB bit of the WID data is checked to see if the current Z Buffer value is valid or invalid. If the LSB bit of the WID data is a 0 then the Z value is valid and the normal Z check is performed. If the LSB bit of the WID data is 1 then the Z value is invalid and the Z check automatically passes. This mode is provided to allow a fast Z clear operation in which only the LSB bit of the WID bitplanes is set. This allows the Z clear to be four times faster than if the Z Buffer bitplanes were actually cleared. The Z value is invalidated and is replaced by a new Z value as the pixel data is written.

Table 6.4 DEPTHFN Register Bit Definitions

DEPTHFN Bit	Meaning if bit set
0	Z compare passes if new Z < old Z
1	Z compare passes if new Z = old Z
2	Z compare passes if new Z > old Z
3	Enable old Z invalidate mode

The trick is that the LSB bit of the WID data is cleared to 0 as the new Z value is written and therefore only the first Z value at this pixel location is made invalid. The following pixel writes to the current pixel location will have valid Z values to compare against. The fast Z clear is described in greater detail in a later paragraph which describes the clearing of the various bitplanes. The Z invalidate mode is only valid for the enhanced adapter. For the base adapter the WID checking operation uses both WID bits and thus prevents the Z invalidate mode from being used.

Window ID Check

The base configuration of the MGR adapter contains 2 Window ID bitplanes and the enhanced configuration contains 4 Window ID bitplanes. These bitplanes are used to store the window ID's of on screen windows. The WID values are used to condition pixel writes to the current window and to select a mode register in the XPC1 or XMAP2 chips in the Display Subsystem. The mode register determines the pixel display formatting. The WID checking is provided to allow pixel write clipping for obscured windows which consist of more than one rectangular piece. If the current window is not obscured and consists of a single rectangular piece then the hardware screen mask can be used to clip pixel writes to that on screen rectangle. When the window is obscured and consists of two or more on rectangular pieces or when the window is not rectangular the WID checking allows pixel clipping of the pixel writes to just those pixels which are part of the current window.

The WID checking limits pixel writes to only those pixels whose WID bitplane data bits match the current window ID in the CURRENTWID register which is set by the GE_CURRENTWID token. WID checking will only be performed after it has been be enabled and it can only be enabled for some instructions and not for others. The Draw Shaded Span (IR = 1), the Draw Lines (IR = 4 or 5) and the Write Buffer (IR = 7) instructions can be WID checked while the Draw Flat Span and the Read Buffer instructions cannot be WID checked. The ENABWID register is set by the GE_ENABWID token and is used to enable or disable WID checking for the Draw Shaded Span and the Write Buffer instructions. The ENABLWID register is set by the GE_ENABLWID token and is used along with the ENABWID register to enable or disable WID checking for the Draw Line instructions. The WID checking operations are shown in Figure 6.94.

Since the Frame Buffer port and the Z buffer port are not the same for the base and enhanced configurations of the adapter the RE2 uses the FBOPTION register to determine which adapter configuration is present. The FBOPTION register is set with the number of bitplanes data parameter sent down after the microcode was downloaded. If the FBOPTION register is set to one then the enhanced adapter configuration is being used and if it is a zero then the base adapter is being used. For the enhanced adapter the frame buffer port has the Frame Buffer, PUP and UAUX bitplane pixel data written through it. The Z buffer port has the Z buffer, WID and PUP bitplanes written through it.

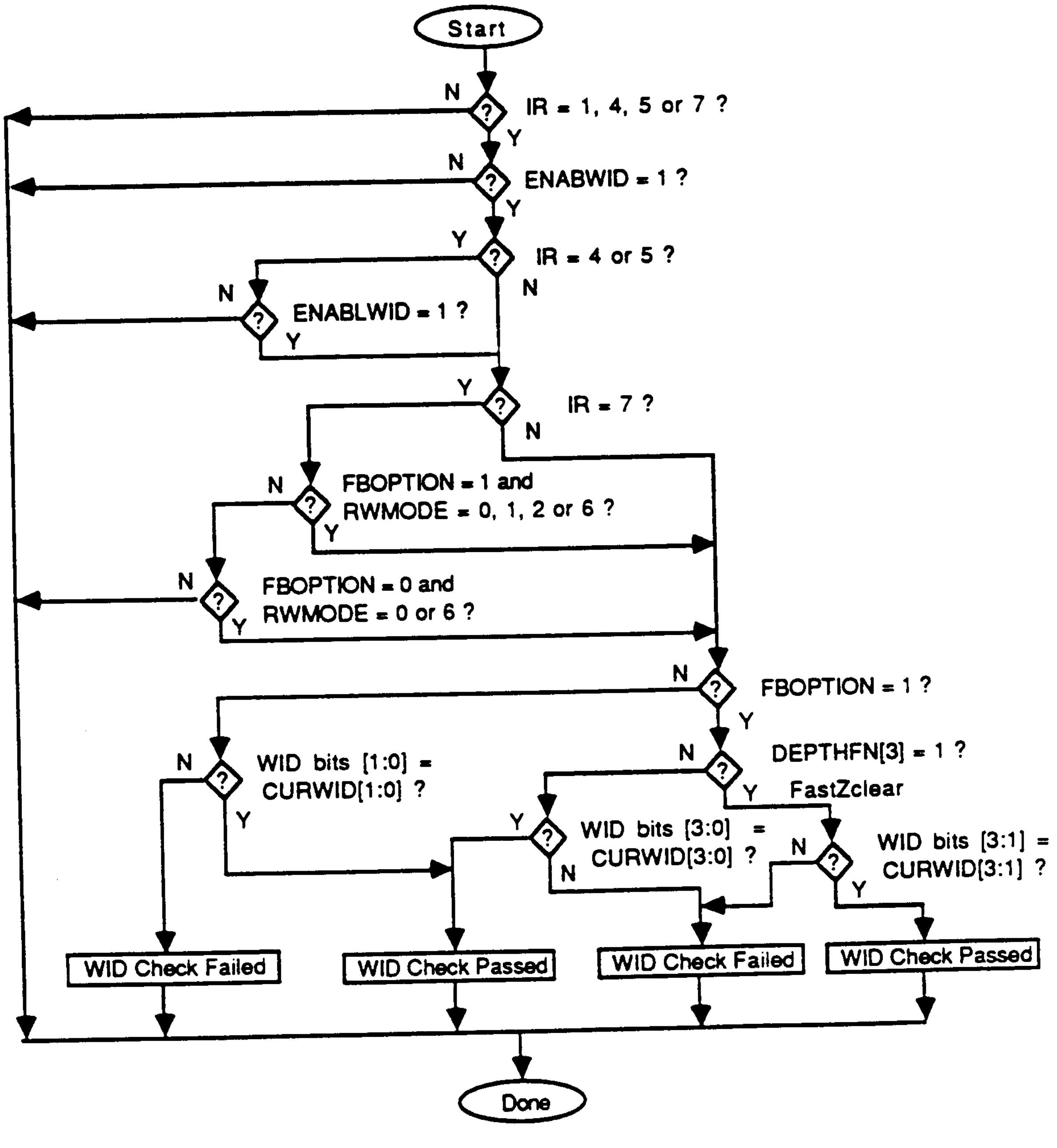


Figure 6.94 WID Check Flow Chart

The Shaded Span and Draw Line instructions can have the data written through both the Frame Buffer and Z Buffer ports WID checked. For the Write Buffer instruction the WID checking is only performed for the bitplanes written through the Frame Buffer port. The bitplanes written through the Z buffer port are not WID checked for the Write Buffer instruction. For the Write Buffer instruction the flow chart shows the RE2 does not even do a WID check for the Z Buffer writes. The pixels are simply written in this case.

For the Write Buffer instructions on the enhanced configuration if the RWMODE is 0, 1, 2 or 6 then a WID check is performed if the ENABWID register is set to one. The Z buffer port has the Z buffer and WID bitplanes written through it so if the RWMODE is 3, 4 or 7 then a WID checked is not performed even if the ENABRGB register is one. For the base configuration the frame buffer port has just the Frame Buffer bitplane pixel data written through it. This means that if the RWMODE is 0 or 6 then a WID check is performed if the ENABWID register is set to one and if the RWMODE is 1,3, 4 or 7 then a WID checked is not performed even if the ENABRGB register is one.

The WID check operation depends on the adapter configuration. For the enhanced adapter if the DEPTHFN register bit 3 is set to 1 then the Fast Z clear mode is enabled. In this mode the LSB bit of the WID bitplanes is used as a Z value validation control bit. Refer to the paragraphs on Z check operation for more details. When Fast Z clear mode is enabled then only the upper 3 bits of the WID bitplanes are compared to the upper 3 bits of the CURWID register. If the values are equal then the WID check passes and the pixel is written. If the values are different then the WID check fails and the pixel is not written. If the Fast Z clear mode is disabled (DEPTHFN[3] = 0) then all four bits of the WID bitplanes are compared to the four bits in the CURWID register. Once again if the values are equal then the WID check passes and the pixel is written. If the values are different then the WID check fails and the pixel is not written.

For the base configuration the two bits of the WID bitplanes are compared to the LSB two bits in the CURWID register. If the values are equal then the WID check passes and the pixel is written. If the values are different then the WID check fails and the pixel is not written.

This completes the discussion of the pixel write conditioning capabilities. The following paragraphs describe the instructions which the RE2 can execute.

Draw Shaded Span Instruction

The Draw Shaded Span instruction is used to draw Gouraud shaded or flat shaded horizontal spans while doing fill operations on polygons and other geometric objects. The shaded spans can be Z Buffer and WID checked as well as have the pattern mask applied to the pixel writes. As with all pixel writes the hardware screen mask is used to clip the pixel writes which are outside of the screen mask rectangle. The initial color values are loaded into the R, G and B registers for RGB pixels or just in the R register for color index pixels. The DR, DG, and DB registers are loaded with the delta color values which are used to vary the color value for Gouraud shaded spans. For flat shaded spans the delta color values are all set to zero. The Draw Shaded Span instruction is used to draw flat spans when they need to be Z Buffer checked, WID checked, pattern masked or a raster operation other than copy needs to be performed. If none of these checks or operations is needed then the Draw Flat Span instruction should be used.

The following paragraphs describe the register usage and the execution flow for the Draw Shaded Span instruction.

Register Usage

The following list of RE2 registers are the registers which are normally specified and used each time a new Shaded Span instruction is executed.

- X	Contains the starting x screen location
- Y	Contains the starting y screen location
- Z	Contains the starting z value
- DX = +1	Contains the change in x which is added to the current x location to get the next x location
- DY = 0	Contains the change in y which is added to the current y location to get the next y location
- DZI	Contains the integer part of the delta Z value
- DZF	Contains the fraction part of the delta Z value
- NUMPIX	Contains a pixel count for the number of pixels to be written
- PIXTYPE	Specifies the type of pixels being written
- R	Contains the initial red color value or the initial color index value
- G	Contains the initial green color value
- B	Contains the initial blue color value
- DR	Contains the delta value for the red color
- DG	Contains the delta value for the green color
- DB	Contains the delta value for the blue color
- PATL, PATH	Contains the pattern mask to be applied for the current scan line

The following registers affect the operation of the Shaded Span instruction but are usually written by the microcode as it executes other tokens and then they remain in effect during the execution of multiple Shaded Span instructions as well as other instructions.

- XMIN	Lower left x coordinate of the hardware screen mask rectangle
- YMIN	Lower left y coordinate of the hardware screen mask rectangle
- XMAX	Upper right x coordinate of the hardware screen mask rectangle
- YMAX	Upper right y coordinate of the hardware screen mask rectangle
- FUNC	Contains the Logical Operation to be applied to each pixel
- ENABWID	Determines if the WID checking is performed for each pixel
- CURWID	Specifies the Current WID for the WID checking operations
- FBOPTION	Specifies the adapter type (0 = base, 1 = enhanced)

- ENABRGB	Specifies that 8 bit RGB pixels are to be written on the base adapter
- NOPUP	Specifies if 2 or 4 UAUX bitplanes are available on the enhanced adapter
- DEPTHFN	Bit 3 controls the FastZclear mode and affects WID and Z Buffer checking
- COLORCMP	Specifies Z or color compare (Should be set to 0 for Shaded Spans)
- PIXMASK	Determines which bits are written to the Frame Buffer bitplanes
- AUXMASK	Determines which bits are written to the PUP, UAUX, WID and Z bitplanes
- WIDDATA	Contains the data to be written to the WID bitplanes
- PUPDATA	Contains the data to be written to the PUP bitplanes
- UAUXDATA	Contains the data to be written to the UAUX bitplanes
- ENABPAT	Specifies if the pattern mask is to be applied
- ALIGNPAT	Specifies the type of pattern alignment used (Should be 1 for Shaded Spans)
- ENABDITH	Determines if the Frame Buffer pixel color values are dithered

Once the RE2 registers have been set up and the Draw Shaded Span instruction has been written to the IR register the RE2 will draw the number of pixels specified in the NUMPIX register. During the execution of a Shaded Span instruction only pixels on the current scan line specified by the Y register can be written. The first pixel to be written on the selected scan line is specified by the X register. The following pixel writes then proceed to the right along the scan line.

Execution Flow

The Draw Shaded Span instruction has two modes of operation depending on the raster operation specified in the FUNC register. These two modes are described in the following sections.

No Raster Operation Mode

The flow of execution for a raster operation of 3 (copy) is shown in Figure 6.95. When the raster operation is set for copy the RE2 does not have to read the Frame Buffer bitplanes which allows it to operate nine times faster than for any other raster operation. This is the equivalent to doing no raster operation. In this mode the RE2 does the Draw Shaded Span instructions by writing the Frame Buffer port bitplanes on a first pass and then going back and writing the Z Buffer port bitplanes on a second pass. The Z Buffer port bitplanes are initially read to get the Z Buffer value and the WID bitplane data. The Z Buffer check and the WID check are performed. The Z Buffer check depends on the setting of the DEPTHFN register to determine if the Z compare passes. If the ENABWID register is set to one then the WID bitplane data is compared to the CURWID register.

The RE2 will proceed along doing the Z Buffer and WID checks until both pass. As long as either one fails neither the Frame Buffer port nor the Z Buffer port is written. When both the Z Buffer check and the WID check pass the RE2 saves the current X, Y and Z values so that it can use these as the starting point for the second pass. The RE2 then begins the first pass where only the Frame Buffer port bitplanes are written. When either the Z Buffer or the WID check fails the first pass ends and the second pass is done using the saved X, Y and Z values as the starting point. A Z count is incremented during the first pass and it determines the number of Z Buffer port bitplane writes to be performed during the second pass. Once the second pass is completed the RE2 continues from the pixel location where it left off. It once again begins doing the Z Buffer and WID checks until both pass. The first and second pass are then done and the entire procedure is continued until the NUMPIX register has finally been decremented to zero. For each pixel in which either the Z Buffer or WID check fails the NUMPIX register is decremented. The NUMPIX register is also decremented for each pixel written during the first pass.

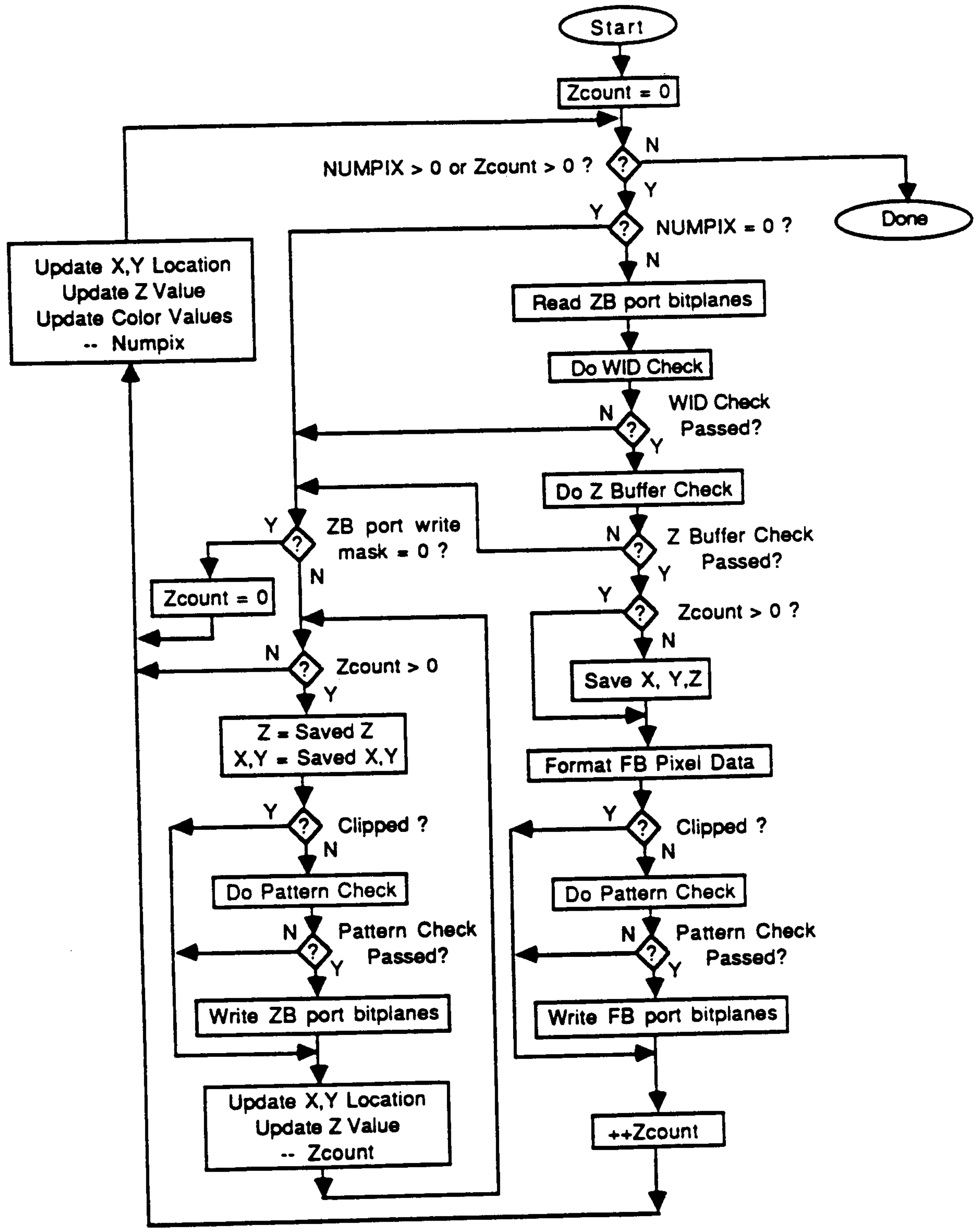


Figure 6.95 No Raster Op Draw Shaded Span Instruction Flow Chart

During the first pass when both the Z Buffer and WID checks are passing the Frame Buffer port bitplanes are written if the pixel location is not outside the screen mask rectangle and the appropriate pattern mask bit is a 1 if the ENABPAT register is set to 1. The Frame Buffer pixel value formatting depends on the color values in the R, G and B registers and the settings of the PIXTYPE and ENABDITH registers. For the base configuration the Frame Buffer bitplanes are written and the bits which are affected depend on the setting of the PIXMASK register. For the enhanced configuration the Frame Buffer bitplanes, the PUP bitplanes and the UAUX bitplanes are written. The bits which are actually written depends on the setting of the PIXMASK and AUXMASK registers. After the Frame Buffer pixels are written the delta color values are added to the current color values and the delta pixel location values are added to the current pixel location values.

During the second pass the saved X, Y and Z values are used to specify the starting pixel location. If the AUXMASK bits are set to 0 so that nothing will be written to the Z Buffer bitplanes the second pass is skipped allowing the Draw Shaded Span instruction to operate almost twice as fast. For the base configuration the PUP, WID and Z mask bits in the AUXMASK must all be zero for the second pass to be skipped. For the enhanced configuration the WID and Z mask bits must be zero for the second pass to be skipped. If any of the mask bits is not zero then the second pass is performed. The Z Buffer port bitplanes are written for each of the pixel locations written during the first pass. For the base configuration the PUP, WID and Z bitplanes are written and for the enhanced configuration the WID and Z bitplanes are written. The bits which are affected depends on the setting of the AUXMASK register. The Z Buffer port bitplane writes are also conditioned by the screen mask and the pattern mask if it is enabled.

Raster Operation Mode

The raster operation mode is performed for any raster operation other than copy. In this mode the RE2 must read and write both the Z Buffer port and the Frame Buffer port for each pixel. This causes the RE2 to not be able to use it's internal pipelining in an efficient manner and so this mode is nine times slower than the no raster operation mode. For each pixel the WID check, Z buffer check, pattern mask and screen mask can condition the pixel writes. The Frame Buffer pixel formatting depends on the PIXTYPE register and whether dithering is enabled. The raster operation is performed on the Frame Buffer pixel data as well as the PUP, UAUX and WID data. The raster operation is performed on a bit by bit basis between the source bitplane data and the bitplane data which was read from the bitplanes. Once all the pixel formatting and raster operations have been performed the Frame Buffer port and Z Buffer port bitplanes are all written if all of the write conditions are satisfied. The bits in the various bitplanes which are written depend on the settings of the PIXMASK and AUXMASK registers. The execution flow for this mode is shown in Figure 6.96.

After the pixel data is written the color values are updated by adding the delta color values to the current color values. The Z value is updated by adding the delta Z value to the current Z value. The current X and Y location is updated by adding the delta X and Y values to the current X and Y values. The DY value must be zero and the DX value must be +1 so that the next pixel to the right of the current pixel is selected. The NUMPIX register is then decremented and this process continues until the NUMPIX register becomes zero.

If Gouraud shaded fills are being performed the delta color values should be set for the appropriate change in the color as needed. If a flat shaded fill is being performed the delta color values should all be zero so that the initial color values do not change. The flat shaded fill should only be performed by the Draw Shaded Span instruction if it needs to be Z Buffer checked, WID checked, pattern masked or have a raster operation other than copy performed. If none of these checks or operations are required then the Draw Flat Span instructions should be used instead since they are much faster than the Draw Shaded Span instruction.

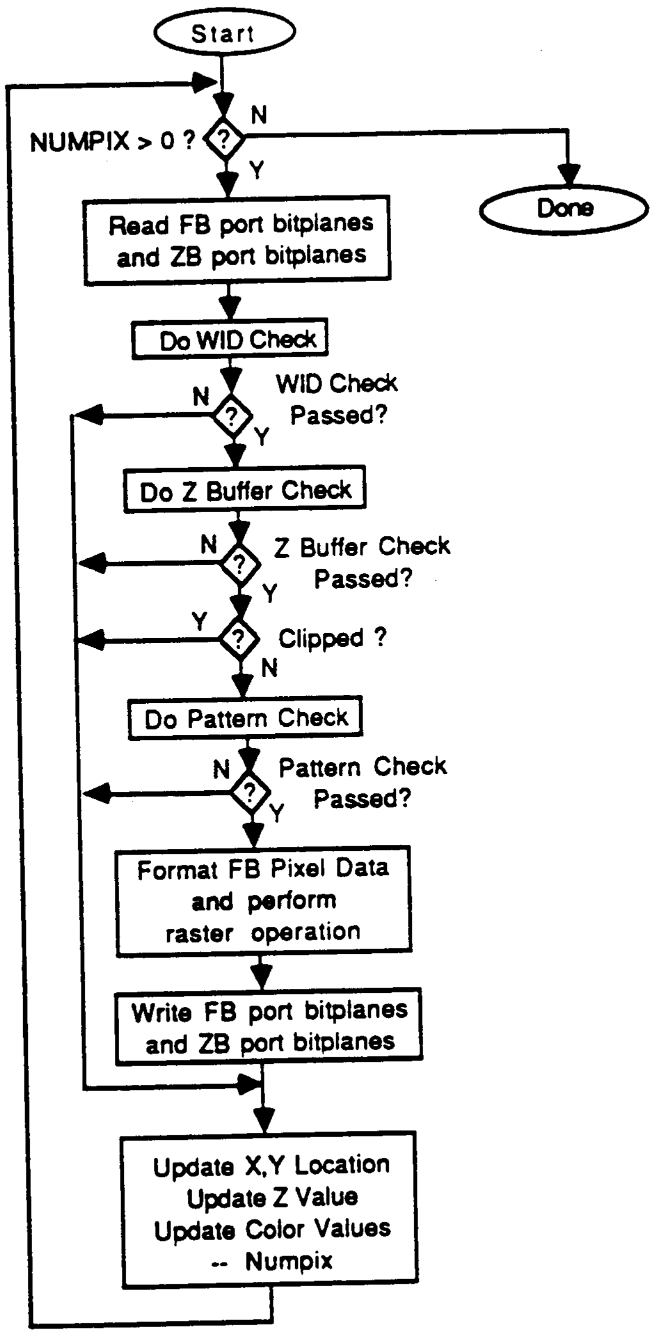


Figure 6.96 Raster Op Draw Shaded Span Instruction Flow Chart

Draw Flat Span Instructions

The RE2 has two forms of the Draw Flat Span instruction which are used to draw flat shaded horizontal spans while doing fill operations on various geometric objects. The two forms are the Draw Flat 1 Span and the Draw Flat 4 Span. The only use of the Draw Flat 1 Span instruction would be to clear the Z Buffer bitplanes which are DRAM chips and do not have the block write feature of the 1 Meg VRAM chips. The Draw Flat 4 Span instruction uses the block write mode of the VRAM chips. The flat spans cannot be Z Buffer checked, WID checked or have the pattern mask applied to the pixel writes. The FUNC register must be set to 3 for a no raster operation mode of copy. As with all pixel writes the hardware screen mask is used to clip the pixel writes which are outside of the screen mask rectangle. The bitplane write masks are also used to determine which bits in the bitplanes are written. The initial color values are loaded into the R, G and B registers for RGB pixels or just in the R register for color index pixels. The DR, DG, and DB registers are always loaded with zero.

The following paragraphs describe the register usage and the execution flow for the Draw Flat Span instruction.

Register Usage

The following list of RE2 registers are the registers which are normally specified and used each time a new Flat Span instruction is executed.

- X	Contains the starting x screen location
- Y	Contains the starting y screen location
- Z	Contains the starting z value
- DX = 0	Contains the change in x which is added to the current x location to get the next x location
- DY = 0	Contains the change in y which is added to the current y location to get the next y location
- DZI	Contains the integer part of the delta Z value
- DZF	Contains the fraction part of the delta Z value
- NUMPIX	Contains a pixel count for the number of pixels to be written
- PIXTYPE	Specifies the type of pixels being written
- R	Contains the initial red color value or the initial color index value
- G	Contains the initial green color value
- B	Contains the initial blue color value
- DR = 0	Contains the delta value for the red color
- DG = 0	Contains the delta value for the green color
- DB = 0	Contains the delta value for the blue color

The following registers affect the operation of the Flat Span instruction but are usually written by the microcode as it executes other tokens and then they remain in effect during the execution of multiple Shaded Span instructions as well as other instructions.

- XMIN	Lower left x coordinate of the hardware screen mask rectangle
- YMIN	Lower left y coordinate of the hardware screen mask rectangle
- XMAX	Upper right x coordinate of the hardware screen mask rectangle
- YMAX	Upper right y coordinate of the hardware screen mask rectangle
- FUNC = 3	Contains the Logical Operation to be applied to each pixel
- FBOPTION	Specifies the adapter type (0 = base, 1 = enhanced)
- ENABRGB	Specifies that 8 bit RGB pixels are to be written on the base adapter
- NOPUP	Specifies if 2 or 4 UAUX bitplanes are available on the enhanced adapter

- PIXMASK	Determines which bits are written to the Frame Buffer bitplanes
- AUXMASK	Determines which bits are written to the PUP, UAUX, WID and Z bitplanes
- WIDDATA	Contains the data to be written to the WID bitplanes
- PUPDATA	Contains the data to be written to the PUP bitplanes
- UAUXDATA	Contains the data to be written to the UAUX bitplanes
- ENABDITH	Determines if the Frame Buffer pixel color values are dithered

Once the RE2 registers have been set up and the Draw Flat Span instruction has been written to the IR register the RE2 will draw the number of pixels specified in the NUMPIX register. During the execution of a Flat Span instruction only pixels on the current scan line specified by the Y register can be written. The first pixel to be written on the selected scan line is specified by the X register. The following pixel writes then proceed to the right along the scan line.

Execution Flow

The execution flow for the Flat Span is designed to be very simple so that it is very fast and is shown in Figure 6.97. The Frame Buffer pixel formatting is performed based on the PIXTYPE selected and whether dithering is enabled. The formatted pixel data is placed in the Frame Buffer port since the raster operation must be 3 which is a copy raster operation. If the current pixel location is not outside of the screen mask the current pixel data will be written from both the Frame Buffer and Z Buffer ports. The number of pixels which are written depends on whether the Draw Flat 1 Span instruction or the Draw Flat 4 Span instructions. The Draw Flat 1 Span instruction can write 1 to 5 adjacent pixels while the Draw Flat 4 Span instruction can write 1 to 20 adjacent pixels. The Draw Flat 4 Span instruction uses the block write mode of the 1 Meg VRAM chips to write up to 20 pixels.

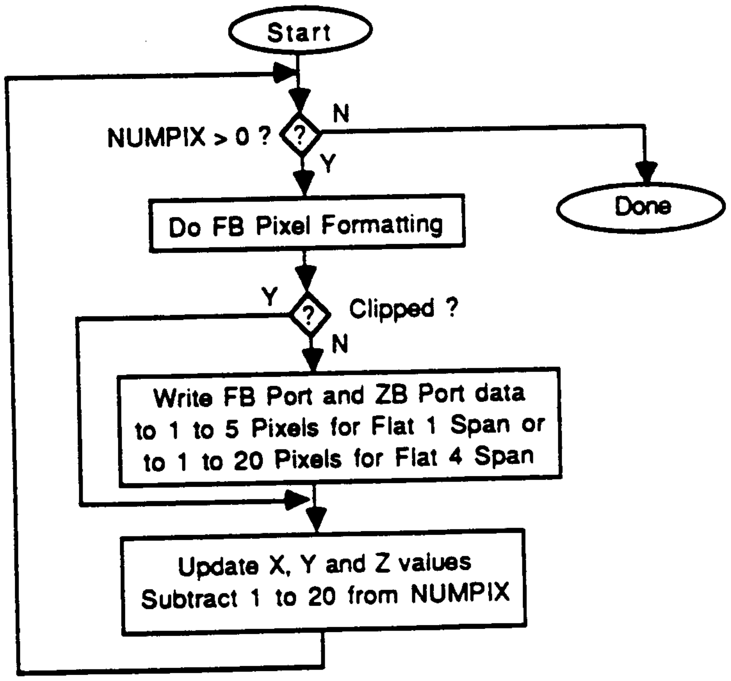

Start
NUMPIX > 0 ?
N
Y
Done
Do FB Pixel Formatting
Y
Clipped ?
N
Write FB Port and ZB Port data to 1 to 5 Pixels for Flat 1 Span or to 1 to 20 Pixels for Flat 4 Span
Update X, Y and Z values Subtract 1 to 20 from NUMPIX

Figure 6.97 Flat Span Instruction Flow Chart

The number of adjacent pixels which are written depends on the current X location. The RE2 can write 1 to 4 pixels, groups of 5 pixels and groups of 20 pixels. If the current X is on a 20 pixel group boundary then all 20 pixels will be written for the Draw Flat 4 Span instruction. If the starting X location is not on a 20 pixel or a 5 pixel boundary then the RE2 will write from 1 to 4 pixels to get to the next 5 pixel boundary. It will then write as many 5 pixel groups as possible until it reaches a 20 pixel boundary. At that time it will write a 20 pixel group assuming that at least 20 pixels are left to write. The reverse procedure is used for the last group of pixels which is less than 20. First as many 5 pixel groups as possible are written and then the final 4 to 1 pixels are written.

For each group of pixels which are written the same data from the Frame Buffer and Z Buffer port are written to each pixel. The Z value is then updated and a new Frame Buffer pixel format calculation are performed for the next pixel. Since the delta color values are zero and dithering should be disabled the Frame Buffer pixel value will not change. For each pixel which is written the PIXMASK and AUXMASK registers determine which bits are actually written.

Draw Line Instructions

The RE2 provides two Draw Line instructions which are the Draw Bottom of antialiased line instruction and the Draw Top of antialiased line instruction. These instructions as their names imply are used to draw antialiased lines. The only difference between these instructions is in the antialiase weight selection from the antialiase weight table. The three most significant fraction bits of the minor axis (slower changing axis) are used as an index into the antialiase weight table. The bottom weight entries are used for the Draw Bottom Line instruction and the top weight entries are used for the Draw Top Line instruction. Either line can be used to draw non-antialiased lines which is the mode when the ASELECT register is set to zero.

The initial color values are loaded into the R, G and B registers for RGB pixels or just in the R register for color index pixels. The DR, DG, and DB registers are loaded with the delta color values which are used to vary the color value for depth-cued lines. For flat lines the delta color values are all set to zero. The delta X and Y values can be set to any integer and fraction values which define the slope of the line. The values can be negative. One of the entries should be set to + or -1 and the other entry to the slope of the line.

The following paragraphs describe the register usage and the execution flow for the Draw Line instructions.

Register Usage

The following list of RE2 registers are the registers which are normally specified and used each time a new Draw Line instruction is executed.

- X	Contains the starting x screen location
- Y	Contains the starting y screen location
- Z	Contains the starting z value
- DX	Contains the change in x which is added to the current x location to get the next x location
- DY	Contains the change in y which is added to the current y location to get the next y location
- DZI	Contains the integer part of the delta Z value
- DZF	Contains the fraction part of the delta Z value
- NUMPIX	Contains a pixel count for the number of pixels to be written
- PIXTYPE	Specifies the type of pixels being written
- R	Contains the initial red color value or the initial color index value
- G	Contains the initial green color value
- B	Contains the initial blue color value
- DR	Contains the delta value for the red color
- DG	Contains the delta value for the green color
- DB	Contains the delta value for the blue color
- STIP	Contains the stipple pattern to be applied for the current scan line
- STIPCOUNT	Contains the repeat count for the initial LSB bit in STIP
- REPSTIP	Contains the repeat count for all other bits in STIP

The following registers affect the operation of the Draw Line instruction but are usually written by the microcode as it executes other tokens and then they remain in effect during the execution of multiple Shaded Span instructions as well as other instructions.

- XMIN	Lower left x coordinate of the hardware screen mask rectangle
- YMIN	Lower left y coordinate of the hardware screen mask rectangle

- XMAX	Upper right x coordinate of the hardware screen mask rectangle
- YMAX	Upper right y coordinate of the hardware screen mask rectangle
- FUNC	Contains the Logical Operation to be applied to each pixel
- ENABWID	Determines if the WID checking is performed for each pixel
- CURWID	Specifies the Current WID for the WID checking operations
- FBOPTION	Specifies the adapter type (0 = base, 1 = enhanced)
- ENABRGB	Specifies that 8 bit RGB pixels are to be written on the base adapter
- NOPUP	Specifies if 2 or 4 UAUX bitplanes are available on the enhanced adapter
- DEPTHFN	Bit 3 controls the FastZclear mode and affects WID and Z Buffer checking
- COLORCMP	Specifies Z or color compare (Should be set to 0 for Shaded Spans)
- PIXMASK	Determines which bits are written to the Frame Buffer bitplanes
- AUXMASK	Determines which bits are written to the PUP, UAUX, WID and Z bitplanes
- WIDDATA	Contains the data to be written to the WID bitplanes
- PUPDATA	Contains the data to be written to the PUP bitplanes
- UAUXDATA	Contains the data to be written to the UAUX bitplanes
- ENABSTIP	Specifies if the stipple pattern is to be applied
- ENABDITH	Determines if the Frame Buffer pixel color values are dithered

Once the RE2 registers have been set up and the Draw Line instruction has been written to the IR register the RE2 will draw the number of pixels specified in the NUMPIX register. During the execution of a Draw Line instruction pixels on any scan line can be written beginning with the scan line specified by the initial Y. The pixel on that scan line specified by the X register is written and then pixels on the same scan line or other scan lines are written depending on the slope values in the DX and DY registers.

Execution Flow

The Draw Line instructions have two modes of execution which are the fast line mode and the slow line mode. As shown in Figure 6.98 the mode is determined by checking the Z Buffer mode, the WID enable register and the raster operation FUNC register. If Z Buffer checking, WID checking and raster operations are disabled then fast line mode is used. The color compare must also be disabled during fast line mode. If any of these checks or operations are enabled then slow line mode is used. The Z Buffer checking is disabled by setting the DEPTHFN register to 7 and clearing the AUXMASK bits for the Z Buffer port bitplanes. For the base adapter this means the PUP, WID and Z mask bits. For the enhanced adapter this means the WID and Z mask bits. The WID checking is disabled by setting the ENABWID register to 0. The raster operation is disabled by setting the FUNC register to 3 for copy mode. The color compare is disabled by setting the COLORCMP register to 0. The following paragraphs describe the fast and slow line modes.

Fast Line Drawing Mode

The fast line mode is a simple mode which is designed to draw lines quickly. In this mode the Frame Buffer pixel formatting is done and then the antialiase operation is done. The resulting Frame Buffer pixel data is placed into the Frame Buffer port and the Frame Buffer and Z Buffer ports are written if the pixel location is not outside the screen mask rectangle and if the LSB bit in the STIP register is a one if the ENABSTIP register is a 1. The bits which are written are determined by the PIXMASK and the AUXMASK registers. The X, Y and Z registers are updated by adding the DX, DY and DZ registers to them. The R, G and B color values are updated by adding the DR, DG and DB registers to them. The NUMPIX register is decremented and the next pixel is then formatted and written as before. This process continues until the NUMPIX register reaches zero at which time the instruction execution is completed and the RE2 reads the IR register to get the next instruction to be executed.

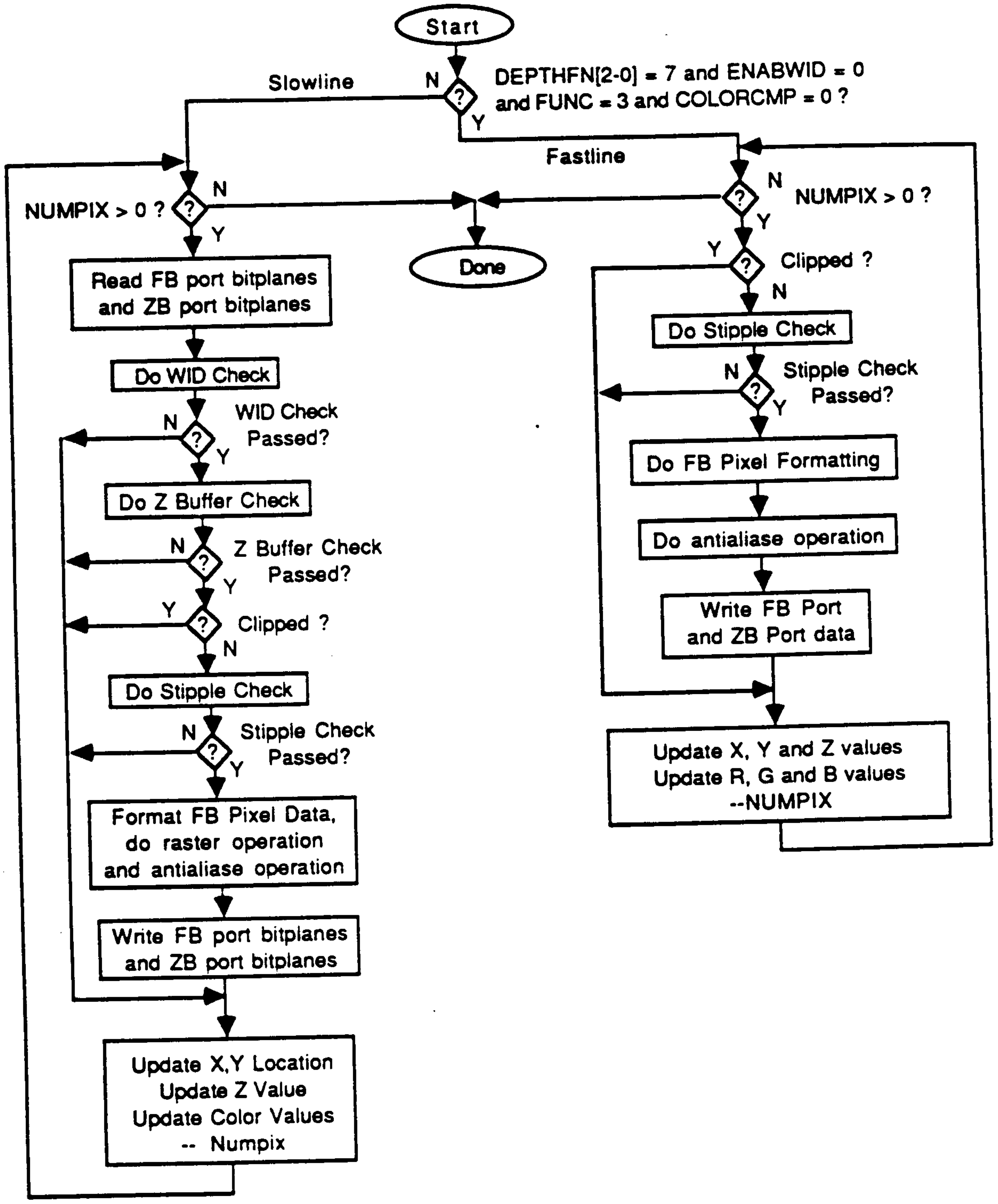


Figure 6.98 Draw Line Instruction Flow Chart

Slow Line Drawing Mode

This drawing mode allows each pixel to be conditioned by the Z Buffer or Color comparison checking, the WID checking and to have a raster operation other than copy performed on the pixel data. The pixel writes can also be conditioned by the stipple pattern if the ENABSTIP register is set to 1. As with all pixel writes the pixel location must be on or inside the screen mask rectangle or the pixel write is clipped. For each pixel location the Z Buffer port bitplanes are read and the Z Buffer and WID checks are performed. If these checks and the stipple check all pass the pixel can be written if it is not clipped. The Frame Buffer pixel data formatting depends on the PIXTYPE and ENABDITH register settings. Once the pixel formatting has been performed the raster operation and the antialiasing operation are performed and the Frame Buffer data is placed into the Frame Buffer port. The Frame Buffer port and the Z Buffer port are then written to the bitplanes if all of the enabled write conditioning checks pass. The bits which are written depend on the settings of the PIXMASK and AUXMASK registers.

After the pixel is written the X, Y and Z values are updated as well as the R, G and B values. The NUMPIX register is decremented and if it is still greater than zero the next pixel is processed and written. If the NUMPIX register is zero then the instruction execution has completed and the RE2 reads the IR register to get the next instruction to execute. If no instruction is in the IR register the RE2 waits for the IR register to be written to before beginning the next instruction execution.

Bitplane DMA Support

The RE2 supports both read and write DMA operations. The DMA can be between the host and the RE2 bitplanes or between the GE5 data RAM and the RE2 bitplanes. To perform a DMA transfer the microcode must first load the appropriate parameters into various RE2 registers. The data to be loaded into the registers will have been saved in the GE5 data RAM by the execution of previous tokens or it will be sent down with the token which causes the DMA operation to be performed. After the RE2 registers have been loaded the Read Buffer or the Write Buffer instruction is loaded into the IR register to start the DMA operation.

The DMA source or destination bitplanes is controlled by the RWMODE register in the RE2, as shown in Table 6.5. The host uses the GE_READSOURCE token to specify the source bitplanes for a read DMA transfer. The bitplanes used for the write destination are specified with the GE_RWMODE token. The microcode will use the values specified by these tokens to load the appropriate value into the RWMODE register depending on whether it is doing a read DMA or a write DMA. In the case of the GE_RECTCOPY token the microcode performs both read and write DMA operations so it will use the value specified by GE_READSOURCE when DMAing a line out of the source bitplanes and will use the value specified by GE_RWMODE when writing the line into the destination bitplanes.

Table 6.5 RWMODE Register Contents

RWMODE	Width of Data (bits)	Data Source or Destination
0	4, 8, 12 or 24	Frame Buffer bitplanes
1	0 or 2	PUP bitplanes
2	0, 2 or 4	UAUX bitplanes
3	24	Z buffer bitplanes
4	2 or 4	WID bitplanes
5	None	Undefined
6	28	Frame Buffer port
7	28	Z buffer port

The following paragraphs describe the DMA support for both reading from and and writing to the specified bitplanes.

DMA Support for Bitplane Reads

The Read Buffer instruction is used for DMA reads from the bitplanes in the Raster Subsystem. Each DMA cycle will transfer a 32 bit word from the RWDATA register of the RE2 to a host buffer or to the GE5 data RAM. Each 32 bit word contains a single pixel in an unpacked format with the pixel being right justified in the word. The following paragraphs describe the format of the pixels read from the selected bitplanes as well as the Read Buffer instruction used to read the bitplanes.

Pixel Formats for Read DMA Transfers

When a read pixel DMA transfer is initiated by the host software the host data buffer will receive pixel data from the selected source bitplanes. The source bitplanes are selected with the GE_READSOURCE token. When a token is sent to the microcode which uses read pixel DMA transfers the microcode loads the source bitplane value sent with the GE_READSOURCE token into the RWMODE register in the RE2. The pixels read from the source bitplanes are placed in the 32 bit words in the host buffer right justified. Multiple pixels are never packed into a single host word. Any bits to the left of the pixel data will be undefined in the the 32 bit host buffer words. The screen mask and write masks do not affect the bitplane reads and the reads can be from any valid pixel location in the source bitplanes. For pixel reads from the Frame Buffer bitplanes the GE_READBUF token is used to specify which buffer the pixel data will be read from. The buffer value sent with the GE_READBUF token will be placed in the READBUF register of the RE2 by the microcode. The appropriate pixel data from the selected buffer is then placed into the host buffer. The following paragraphs describe the pixel formats for the base and the enhanced adapter.

Base Adapter Pixel Formats

The base adapter has 8 frame buffer bitplanes, 2 PUP bitplanes, 2 WID bitplanes and optionally 24 Z buffer bitplanes. These bitplanes can all be read from using DMA transfers. The pixel formats for the base adapter DMA reads are shown in Figure 6.99 and the following paragraphs describe the pixel formats of the data words which will be placed in the host buffer.

Frame Buffer Pixel Formats

For pixel reads from the frame buffer the PIXTYPE determines the type of action that the RE2 will take when reading the pixel data. For pixels written with PIXTYPE set to binary 00 the RE2 will have formatted the pixel data in the frame buffer in a special 8 bit RGB format. When these pixels are read they will be returned to the host in the same 8 bit RGB format. Bits 0 through 2 will represent the most significant three bits of the original 8 bit Red color component. Bits 5 through 3 will represent the most significant three bits of the original 8 bit Green color component. Bits 7 and 6 represent the most significant two bits of the original 8 bit Blue color component. This special 233 format for 8 bit RGB pixels right justified in the 32 bit word placed in the host buffer.

The base adapter does not support the use of 12 bit RGB pixels. This means that double buffered RGB mode is not supported on the base adapter and therefore a PIXTYPE equal to binary 01 is invalid on the base adapter and pixels of this format should never be read back from the Frame Buffer bitplanes of the base adapter.

For pixels written with PIXTYPE set to binary 10 the RE2 uses the right most 8 bits of each word it receives as an 8 bit Color Index (CI) value and the 8 bits are written into the frame buffer bitplanes. When pixels of this type are read the 8 bit color index value will be placed in the host buffer 32 bit word right justified.

For pixels written with PIXTYPE set to binary 11 the RE2 duplicates the 4 bit color index value into the upper nibble and the PIXMASK determines which buffer is written. The host specifies which of the buffers it wishes to read using the GE_READBUF token to specify either the front or back buffer for reads. The READBUF register is loaded by the GE5 microcode with the value specified by the GE_READBUF token. The RE2 returns either the right most nibble for the front buffer or the left most nibble for the back buffer. The 4 bit color index value contained in the selected nibble is placed in the host buffer 32 bit word right justified.

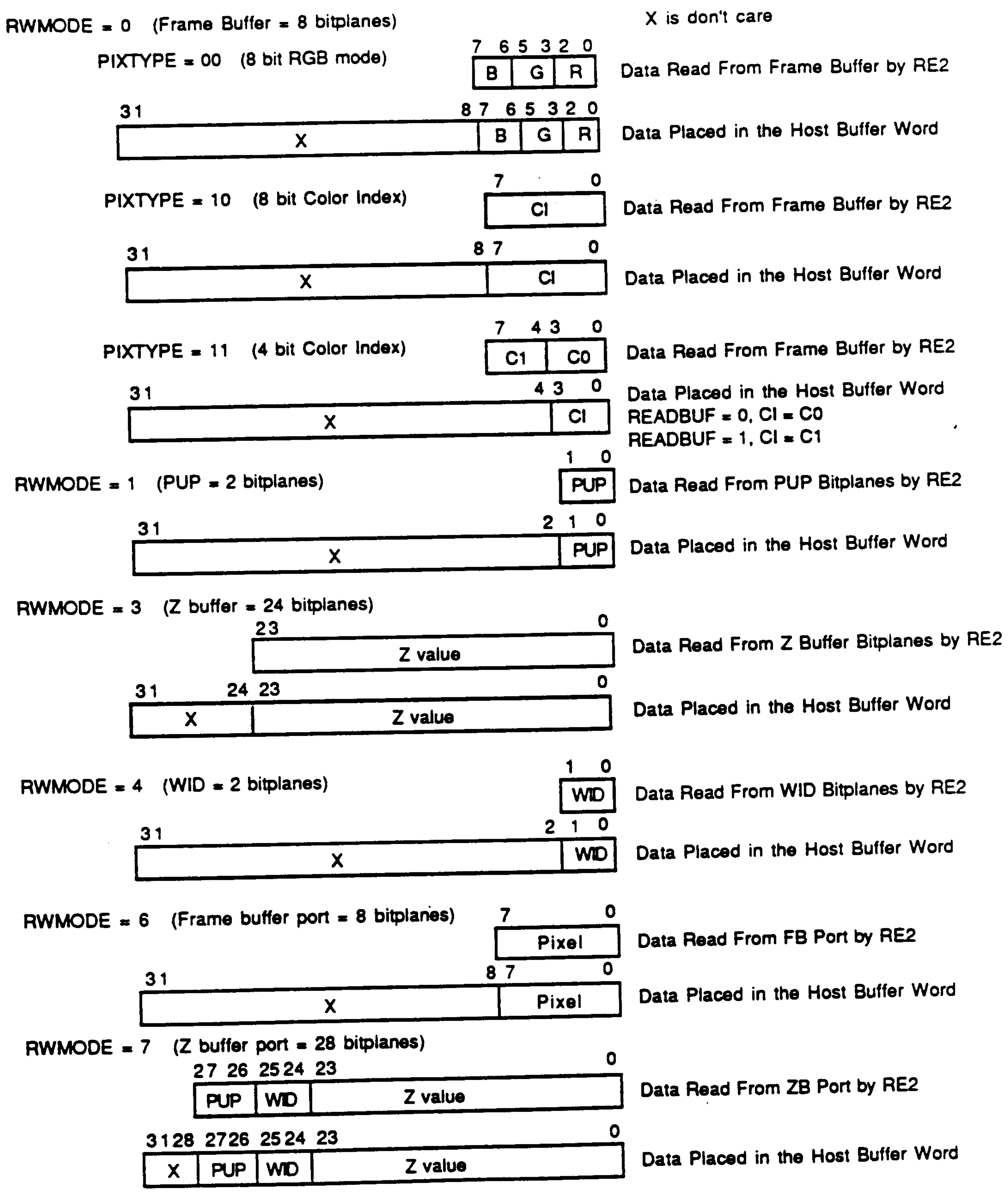

X is don't care
RWMODE = 0 (Frame Buffer = 8 bitplanes)
PIXTYPE = 00 (8 bit RGB mode)
7 6 5 3 2 0
B G R
Data Read From Frame Buffer by RE2
31 8 7 6 5 3 2 0
X B G R
Data Placed in the Host Buffer Word
PIXTYPE = 10 (8 bit Color Index)
7 0
CI
Data Read From Frame Buffer by RE2
31 8 7 0
X CI
Data Placed in the Host Buffer Word
PIXTYPE = 11 (4 bit Color Index)
7 4 3 0
C1 C0
Data Read From Frame Buffer by RE2
31 4 3 0
X CI
Data Placed in the Host Buffer Word
READBUF = 0, CI = C0
READBUF = 1, CI = C1
RWMODE = 1 (PUP = 2 bitplanes)
1 0
PUP
Data Read From PUP Bitplanes by RE2
31 2 1 0
X PUP
Data Placed in the Host Buffer Word
RWMODE = 3 (Z buffer = 24 bitplanes)
23 0
Z value
Data Read From Z Buffer Bitplanes by RE2
31 24 23 0
X Z value
Data Placed in the Host Buffer Word
RWMODE = 4 (WID = 2 bitplanes)
1 0
WID
Data Read From WID Bitplanes by RE2
31 2 1 0
X WID
Data Placed in the Host Buffer Word
RWMODE = 6 (Frame buffer port = 8 bitplanes)
7 0
Pixel
Data Read From FB Port by RE2
31 8 7 0
X Pixel
Data Placed in the Host Buffer Word
RWMODE = 7 (Z buffer port = 28 bitplanes)
27 26 25 24 23 0
PUP WID Z value
Data Read From ZB Port by RE2
31 28 27 26 25 24 23 0
X PUP WID Z value
Data Placed in the Host Buffer Word

Figure 6.99 Base Adapter Pixel Formats for DMA Reads

PUP Pixel Formats

The base adapter contains 2 Pop Up (PUP) bitplanes which can be used to display either overlay or underlay colors. When the PUP bitplanes are read the 2 bit PUP color index value is placed in the host buffer 32 bit words right justified.

WID Pixel Formats

The base adapter contains 2 Window ID (WID) bitplanes which are used as an index into the XPC1 mode registers. When the WID bitplanes are read the 2 bit WID mode register index value is placed in the host buffer 32 bit words right justified.

Z Buffer Pixel Formats

The base adapter can have an optional Z buffer card installed which contains 24 Z buffer bitplanes which are used to perform depth (Z) comparisons to enable or disable pixels from being updated. When the Z buffer pixel bitplanes are read the 24 bit Z values is placed in the host buffer 32 bit word right justified.

Frame Buffer Port Pixel Formats

The base adapter has a Frame Buffer port which only contains the frame buffer pixel and no other pixel data. When the frame buffer port is read the 8 bits of pixel data contained in the Frame Buffer bitplanes are returned unmodified. Both buffers are returned for the double buffered pixels rather than just the buffer selected by the READBUF register. The 8 bit data will be right justified in the 32 bit words placed in the host buffer.

Z Buffer Port Pixel Formats

The base adapter has a Z buffer port which allows the host to access the Z buffer, WID and PUP bitplanes all at once. The 24 bit Z value is placed in bits 0 to 23, the WID data in bits 24 and 25 and the PUP data in bits 26 and 27 of the host buffer. The 28 bits of data will be right justified in the 32 bit words returned to the host buffer.

Enhanced Adapter Pixel Formats

The enhanced adapter has 24 frame buffer bitplanes, 2 PUP bitplanes, 2 UAUX bitplanes, 4 WID bitplanes and optionally 24 Z buffer bitplanes. These bitplanes can all be read from using DMA transfers. The pixel formats for the enhanced adapter DMA reads are shown in Figure 6.100 and the following paragraphs describe the pixel formats of the data words which will be placed in the host buffer.

Frame Buffer Pixel Formats

For pixel reads from the frame buffer the PIXTYPE determines the type of action that the RE2 will take when reading the pixel data. For pixels written with PIXTYPE set to binary 00 the RE2 will have written the three 8 bit RGB color components into the bitplanes with the Red color component in the right most byte, the Green color component in the middle byte and the Blue color component in the left most byte. When this type of pixel is read the same BGR bytes are read from the bitplanes and are placed in the host buffer right justified.

For pixels written with PIXTYPE set to binary 01 the RE2 uses the right most 4 bits of each of the three color components and duplicates that nibble into the lower nibble of the corresponding color component. This PIXTYPE is used normally in a double buffer mode and the PIXMASK controls which buffer receives the color component nibbles. The host specifies which of the buffers it wishes to read using the GE_READBUF token to specify either the front or back buffer for reads. The READBUF register is loaded by the GE5 microcode with the value specified by the GE_READBUF token. The RE2 always returns 24 bit RGB values for this pixel type. If the front buffer is selected then the right most nibble of each color component will be duplicated into the upper nibble of each of the color components in the word returned to the host. If the back buffer is selected then the left most nibble of each color component will be duplicated into the lower nibble of each of the color components in the word returned to the host. The 24 bit RGB pixel data is placed in the host buffer 32 bit word right justified.

For pixels written with PIXTYPE set to binary 10 the RE2 duplicates the 12 bit color index into upper 12 bits and the PIXMASK determines which buffer receives the pixel data. When this pixel type is read the READBUF register determines which of the 12 bit color index values will be returned to the host. If the front buffer is selected then the right most 12 bit color index will be returned. If the back buffer is selected then the left most 12 bit color index will be returned. The returned 12 bit color index value will be right justified in the host buffer word.

For pixels written with PIXTYPE set to binary 11 the RE2 duplicates the 4 bit color index value into the upper nibble and the PIXMASK determines which buffer is written. The host specifies which of the buffers it wishes to read using the GE_READBUF token to specify either the front or back buffer for reads. The READBUF register is loaded by the GE5 microcode with the value specified by the GE_READBUF token. The RE2 returns either the right most nibble for the front buffer or the left most nibble for the back buffer. The 4 bit color index value contained in the selected nibble is placed in the host buffer 32 bit word right justified.

PUP Pixel Formats

The enhanced adapter contains 2 Pop Up (PUP) bitplanes which can be used to display either overlay or underlay colors. When the PUP bitplanes are read the 2 bit PUP color index value is placed in the host buffer 32 bit words right justified. If the NOPUP register is set then the PUP bitplanes become part of the UAUX bitplanes and are read with the UAUX bitplanes as the selected source for the read.

UAUX Pixel Formats

The enhanced adapter contains either 2 or 4 User Auxiliary (UAUX) bitplanes which can be used to display either overlay or underlay colors. When the UAUX bitplanes are read the 2 or 4 bit UAUX color index value is placed in the host buffer 32 bit words right justified. If the NOPUP register is 0 then their are 2 PUP bitplanes and 2 UAUX bitplanes. If the NOPUP register is 1 then there are no PUP bitplanes and 4 UAUX bitplanes.

WID Pixel Formats

The enhanced adapter contains 4 Window ID (WID) bitplanes which are used as an index into the XMAP2 mode registers and to control pixel writes into the various bitplanes. When the WID bitplanes are read the 4 bit WID mode register index value is placed in the host buffer 32 bit words right justified.

Z Buffer Pixel Formats

The enhanced adapter can have an optional Z buffer card installed which contains 24 Z buffer bitplanes which are used to perform depth (Z) comparisons to enable or disable pixels from being updated. When the Z buffer pixel bitplanes are read the 24 bit Z values is placed in the host buffer 32 bit word right justified.

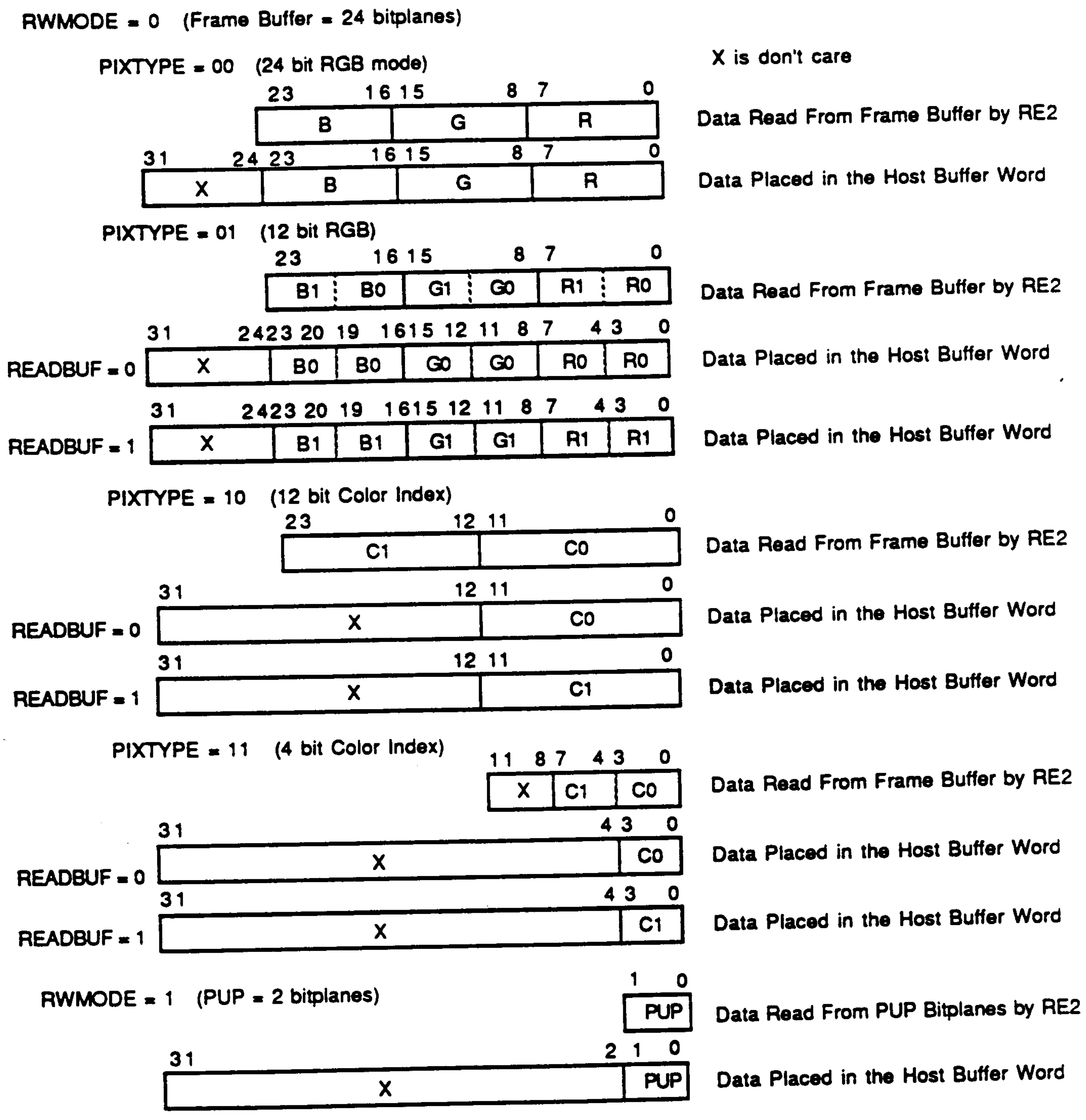

RWMODE = 0 (Frame Buffer = 24 bitplanes)
X is don't care
PIXTYPE = 00 (24 bit RGB mode)
23 16 15 8 7 0
B G R
Data Read From Frame Buffer by RE2
31 24 23 16 15 8 7 0
X B G R
Data Placed in the Host Buffer Word
PIXTYPE = 01 (12 bit RGB)
23 16 15 8 7 0
B1 B0 G1 G0 R1 R0
Data Read From Frame Buffer by RE2
READBUF = 0
31 24 23 20 19 16 15 12 11 8 7 4 3 0
X B0 B0 G0 G0 R0 R0
Data Placed in the Host Buffer Word
READBUF = 1
31 24 23 20 19 16 15 12 11 8 7 4 3 0
X B1 B1 G1 G1 R1 R1
Data Placed in the Host Buffer Word
PIXTYPE = 10 (12 bit Color Index)
23 12 11 0
C1 C0
Data Read From Frame Buffer by RE2
READBUF = 0
31 12 11 0
X C0
Data Placed in the Host Buffer Word
READBUF = 1
31 12 11 0
X C1
Data Placed in the Host Buffer Word
PIXTYPE = 11 (4 bit Color Index)
11 8 7 4 3 0
X C1 C0
Data Read From Frame Buffer by RE2
READBUF = 0
31 4 3 0
X C0
Data Placed in the Host Buffer Word
READBUF = 1
31 4 3 0
X C1
Data Placed in the Host Buffer Word
RWMODE = 1 (PUP = 2 bitplanes)
1 0
PUP
Data Read From PUP Bitplanes by RE2
31 2 1 0
X PUP
Data Placed in the Host Buffer Word

Figure 6.100 Enhanced Adapter Pixel Formats for DMA Reads

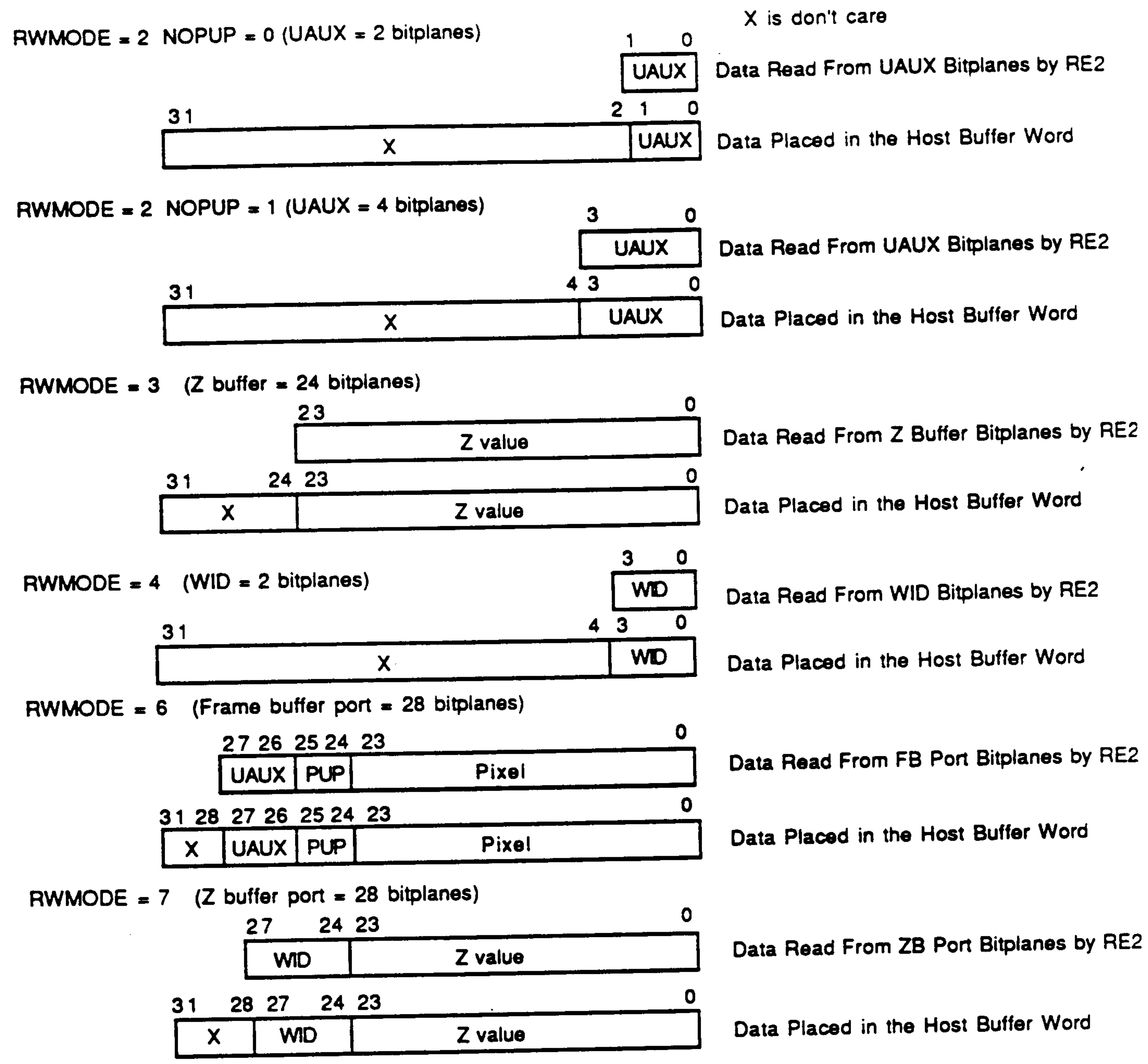

X is don't care
RWMODE = 2 NOPUP = 0 (UAUX = 2 bitplanes)
1 0
UAUX
Data Read From UAUX Bitplanes by RE2
31 2 1 0
X
UAUX
Data Placed in the Host Buffer Word
RWMODE = 2 NOPUP = 1 (UAUX = 4 bitplanes)
3 0
UAUX
Data Read From UAUX Bitplanes by RE2
31 4 3 0
X
UAUX
Data Placed in the Host Buffer Word
RWMODE = 3 (Z buffer = 24 bitplanes)
23 0
Z value
Data Read From Z Buffer Bitplanes by RE2
31 24 23 0
X
Z value
Data Placed in the Host Buffer Word
RWMODE = 4 (WID = 2 bitplanes)
3 0
WID
Data Read From WID Bitplanes by RE2
31 4 3 0
X
WID
Data Placed in the Host Buffer Word
RWMODE = 6 (Frame buffer port = 28 bitplanes)
27 26 25 24 23 0
UAUX PUP Pixel
Data Read From FB Port Bitplanes by RE2
31 28 27 26 25 24 23 0
X UAUX PUP Pixel
Data Placed in the Host Buffer Word
RWMODE = 7 (Z buffer port = 28 bitplanes)
27 24 23 0
WID Z value
Data Read From ZB Port Bitplanes by RE2
31 28 27 24 23 0
X WID Z value
Data Placed in the Host Buffer Word

Figure 6.100 Enhanced Adapter Pixel Formats for DMA Reads (Cont.)

Frame Buffer Port Pixel Formats

The enhanced adapter has a Frame Buffer port which contains the 24 bit frame buffer pixel, the 0 or 2 PUP bitplanes and the 2 or 4 UAUX bitplanes. When the frame buffer port is read the 24 bits of pixel data contained in the Frame Buffer bitplanes are returned unmodified. Both buffers are returned for the double buffered pixels rather than just the buffer selected by the READBUF register. The 24 bit frame buffer pixel data will be in bits 0 to 23, the PUP pixel data will be in bits 24 and 25 and the UAUX pixel bits will be in bits 26 and 27. If the NOPUP register is 1 then the UAUX pixel will be in bits 24 to 27. The 28 bits of data will be right justified in the 32 bit words placed in the host buffer.

Z Buffer Port Pixel Formats

The enhanced adapter has a Z buffer port which allows the host to access the Z buffer and WID bitplanes at the same time. The 24 bit Z value is placed in bits 0 to 23 and the WID data in bits 24 to 27. The 28 bits of data will be right justified in the 32 bit words returned to the host buffer.

Read Buffer Instruction

To perform a bitplane read DMA transfer the microcode must set up the proper RE2 registers and then issue the Read Buffer instruction to the RE2. This causes the RE2 to read the pixel data from the selected source bitplanes and to place the pixel data in the RWDATA register. The read DMA transfer is controlled by the HQ1 chip in the Geometry Subsystem. The HQ1 executes a microcode instruction which instructs it to perform the necessary hardware operations to transfer data words from the RWDATA register in the RE2 to the Host Buffer or to the GE5 Data RAM. The number of word transfers is specified by the count that the microcode had previously loaded into the HQ1 DMA Count register. The RE2 does not do any screen mask clipping, WID checking or pattern mask checking when doing the pixel reads. The following paragraphs describe the RE2 register usage and execution flow of the Read Buffer instruction.

Register Usage

The following list of RE2 registers are the registers which are normally specified and used each time a new Read Buffer instruction is executed.

- X	contains the starting x screen location
- Y	Contains the starting y screen location
- DX = +1	Contains the change in x which is added to the current x location to get the next x location
- DY = 0	Contains the change in y which is added to the current y location to get the next y location
- NUMPIX	Contains a pixel count for the number of pixels to be read
- PIXTYPE	Specifies the type of pixels being read if the frame buffer is the source
- RWMODE	Specifies the source bitplanes to be read
- READBUF	Specifies the frame buffer pixel data to be read
- RWDATA	The output data port of the RE2 for the Read Buffer instruction

The following registers affect the operation of the Read Buffer instruction but are usually written by the microcode as it executes other tokens and then they remain in effect during the execution of multiple Read Buffer instructions as well as other instructions.

- FBOPTION	Specifies the adapter type (0 = base, 1 = enhanced)
- ENABRGB	Specifies that 8 bit RGB pixels are to be used on the base adapter
- NOPUP	Specifies if 2 or 4 UAUX bitplanes are available on the enhanced adapter

Once the RE2 registers have been set up and the Read Buffer instruction issued then the RE2 will read the pixel data and place the pixel data in the RWDATA register so that the HQ1 chip will transfer the data to the Host Buffer or to the GE5 Data RAM. During the execution of a Read Buffer instruction only pixels on the current scan line specified by the Y register can be read. The first pixel to be read on the selected scan line is specified by the X register. The following pixel reads then proceed to the right along the scan line. The number of pixels to be read is specified in the NUMPIX register.

Execution Flow

A logical representation of the Read Buffer instruction execution flow is shown in the flow diagram shown in Figure 6.101. The NUMPIX register contains the count for the number of pixels to be read by the Read Buffer instruction. The main loop is executed until the number of pixels in the NUMPIX register have been read. For each pass through the main loop the following major operations are performed.

- read the pixel data at the current pixel location into the Z Buffer and the Frame Buffer Ports
- if frame buffer pixels are being read then use READBUF to determine which buffer is used
- right justify the pixel data which is being read
- wait for the previously read pixel to be removed from the RWDATA register
- place the pixel data in the RWDATA register
- update the pixel location and decrement NUMPIX

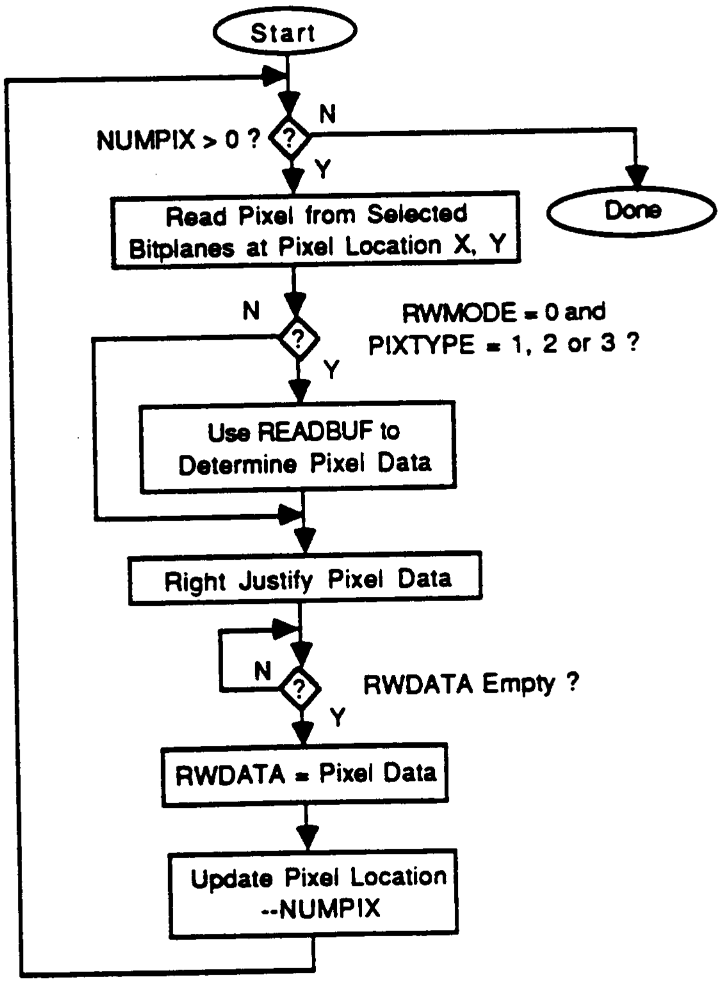

Start
N
NUMPIX > 0 ?
?
Y
Read Pixel from Selected Bitplanes at Pixel Location X, Y
Done
N
?
RWMODE = 0 and PIXTYPE = 1, 2 or 3 ?
Y
Use READBUF to Determine Pixel Data
Right Justify Pixel Data
N
?
RWDATA Empty ?
Y
RWDATA = Pixel Data
Update Pixel Location --NUMPIX

Figure 6.101 Read Buffer Instruction Flow Chart

For each pixel location to be read the RE2 reads all of the bitplanes for the current pixel location specified by the X and the Y registers into the Frame Buffer port and the Z Buffer port. The appropriate field is selected by the source specified in the RWMODE register. If the Frame Buffer pixel data is the source and the PIXTYPE is a double buffered pixel type then the READBUF register determines the part of the data that is returned to the host. For 12 bit RGB pixels (PIXTYPE = 01) the selected nibble in each color component is duplicated into the unselected nibble and the 24 bit RGB value is returned. For the color index pixel types the selected buffer is returned. For the 24 bit RGB pixel type the data is returned unmodified. On the base adapter the 8 bit RGB pixels are returned in the special 233 format. Refer to the pixel format paragraphs above for greater details.

The selected pixel data is shifted to the right if necessary so that it is always returned to the host buffer right justified. After the pixel data has been appropriately shifted it is placed into the RWDATA register. The RE2 will wait for the HQ1 to empty the RWDATA register before placing the next word in it. The pixel location is updated by adding the DX register to the X register and the DY register to the Y register. Since the DX register is +1 and the DY register is 0 the next pixel is to the right of the current one. Finally the NUMPIX register is decremented and the loop is executed again. This process continues until the NUMPIX register becomes zero and the Read Buffer instruction terminates.

DMA Support for Bitplane Writes

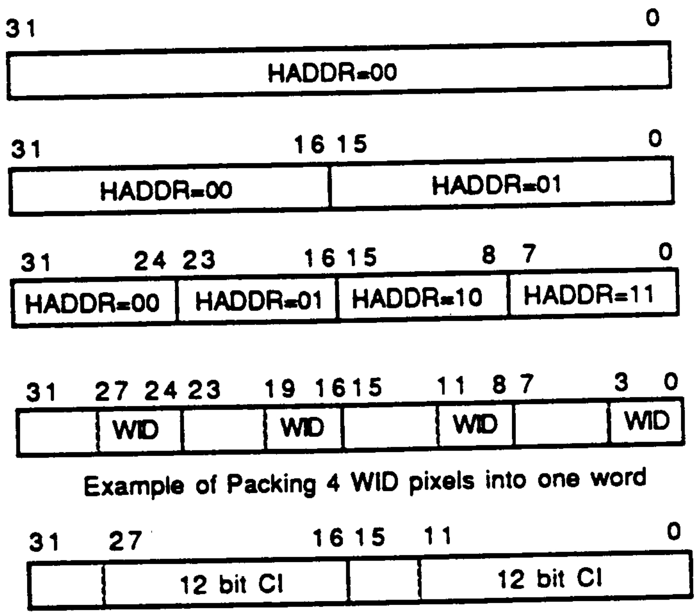
The Write Buffer instruction is used for DMA writes into the bitplanes in the Raster Subsystem. Each DMA cycle will transfer a 32 bit word from the host or the GE5 data RAM to the RWDATA register of the RE2. The 32 bit word can contain pixels in a packed or unpacked format. The format of the pixels must match the pixel format of the destination bitplanes specified by the RWMODE register. If the destination is the Frame Buffer bitplanes then the pixel value must match the current pixel type specified with the GE_PIXTYPE token. The pixels can also have a zoom factor applied to them to cause each pixel to be written into multiple adjacent pixels in the x and y directions. The following paragraphs describe the pixel packing, pixel formats and pixel zooming in greater detail.

Pixel Packing and Unpacking

The host software can pack multiple pixel values into each 32 bit word that is DMAed to the selected bitplanes. The GE_WRITEPIXELS, GE_RECTWRITE and GE_WRITEBLOCK tokens support the use of packed pixels. Each of these tokens requires the host to send down the upac and haddr parameters as part of the data parameters sent with the token. The microcode then places the values in the UPACMODE and the HADDR registers in the RE2 before doing the Write Buffer instruction

During the execution of the Write Buffer instruction each 32 bit word to be written to the bitplanes is transferred from the GE5 Data RAM or the Host Buffer to the RWDATA register. The RE2 UPACMODE register is used to specify the number of pixel values packed into each 32 bit word received in the RWDATA register. Each 32 bit word can can be thought of as an array which can contain either a single 32 bit pixel, two 16 bit pixels or four 8 bit pixels as shown in Figure 6.102. If the size of the pixel value being packed is less than the above sizes they will be right justified within the 8, 16 or 32 bit spaces allocated for each pixel. For example if four 4 bit WID pixel values are being packed into a 32 bit word each 4 bit WID pixel will occupy 8 bits in the word and the 4 bits will be right justified in the 8 bits. The other unused four bits in each byte would be ignored.

UPACMODE	Pixels/32 bit word
00	1
01	2
10	Undefined
11	4



Example of Packing 4 WID pixels into one word

Example of Packing two 12 bit CI pixels into one word

Figure 6.102 UPACMODE and HADDR Register Usage

The HADDR register is loaded with the pixel offset of the first pixel in the first word received by the RE2. This allows the host the freedom of defining buffers on byte boundaries even though the DMA transfer will be done on 32 bit word boundaries. The pixels are packed into the host buffer words in a left to right order. The RE2 uses the pixel at offset HADDR as the first pixel to be drawn at the starting X and Y location. It then proceeds to the right to the next pixel location in the first word until it has used all the pixels in that word. It will then use all of the pixels in the following words starting with the left most pixel and proceeding to the right. The exception is that after the NUMPIX number of pixels have been processed any unused pixels in the last word will be ignored.

Pixel Formats for Write DMA Transfers

The host software must place the appropriately formatted data into the host data buffer before the data is DMAed to the RE2 for writing into the bitplanes which have been selected with the GE_RWMODE token. The pixel data must also be right justified in the host data buffer. If pixels are packed each pixel must be right justified in the 8 or 16 bits in which the pixel is packed. For example if two 12 bit color index pixels are packed into a 32 bit word each pixel uses 16 bits of the word and each pixel will be right justified in the 16 bits which hold it. If the destination bitplanes are the Frame Buffer the pixel data format must correspond to the pixel type selected with the GE_PIXTYPE token. In the following discussion the figures that are referenced show a PIXMASK and an AUXMASK of all ones to show which bitplanes the RE2 would attempt to write for each pixel format.

Base Adapter Pixel Formats

The base adapter has 8 frame buffer bitplanes, 2 PUP bitplanes, 2 WID bitplanes and optionally 24 Z buffer bitplanes. These bitplanes can all be written to using DMA transfers. The pixel formats for the base adapter DMA writes are shown in Figure 6.103 and the following paragraphs describe the pixel formats of the data which must be placed in the host buffer. They also describes how the data is modified by the RE2 before it is written into the selected bitplanes.

Frame Buffer Pixel Formats

When PIXTYPE is set to 00 the RE2 formats the pixel data in the frame buffer in a special 8 bit RGB format. For the 8 bit RGB mode to be enabled the RE2 ENABRGB register must be set to 1 during the adapter initialization. The GE_LOADRE token is used to set the ENABRGB register. For PIXTYPE equal to 00 the host buffer must place 24 bit RGB formatted data in the host buffer. The RE2 converts each of the 24 bit RGB data words it receives to 8 bit RGB format. The conversion consists of taking the 2 most significant bits of the Blue color component and placing them in bits 7-6 of the 8 bit RGB byte. The 3 most significant 3 bits of the Green color component are placed in bits 5-3 of the 8 bit RGB byte. Finally the most significant 3 bits of the Red color component are placed in bits 0-2 of the 8 bit RGB byte and then the 8 bit RGB byte is written into the bitplanes. This means that the original 24 bit RGB data has become 8 bit RGB data in the special 233 format. The PIXMASK would normally be set to 0xFF to write the 8 bit RGB value into the bitplanes. On the base adapter if the ENABRGB register is 0 and PIXTYPE is equal to 00 then only the Red color component would be written to the frame buffer bitplanes. This is not considered a valid mode so make sure the ENABRGB register is set to 1. Since the host buffer words contain 24 bit RGB data pixel packing cannot be used. The 24 bit RGB pixel value must be right justified in the 32 bit word.

The base adapter does not support the use of 12 bit RGB pixels. This means that double buffered RGB mode is not supported on the base adapter and PIXTYPE equal to 01 is invalid on the base adapter.

When PIXTYPE is set to 10 the RE2 uses the right most 8 bits of each word it receives as an 8 bit Color Index (CI) value and the 8 bits are written into the frame buffer bitplanes. This pixtype is usually used for single buffer data. The PIXMASK would normally be set to 0xFF to write the 8 bit CI value into the bitplanes. If pixel packing is not used then the single 8 bit CI pixel must be right justified in the word received by the RE2. If pixel packing is done then a maximum of four 8 bit CI pixels can be packed into the 32 bit word and each 8 bit pixel will be in a byte.

When PIXTYPE is set to 11 the RE2 uses the right most 4 bits of each word it receives as a 4 bit Color Index value. The RE2 duplicates the 4 bit CI value into the upper nibble and writes the two nibbles to the frame buffer bitplanes. This PIXTYPE is used normally in a double buffer mode. The PIXMASK would be set to 0xOF to allow only the front buffer to be updated and it would be set to 0xF0 to allow only the back buffer to be written. As shown in Figure 6.103, if the PIXMASK is set to 0xFF the 4 bit CI value would be written into both buffers. If pixel packing is not used then the single 4 bit CI pixel must be right justified in the word received by the RE2. If pixel packing is done then a maximum of four 4 bit CI pixels can be packed into the 32 bit word and each 4 bit CI pixel will be right justified in the byte that it is in.

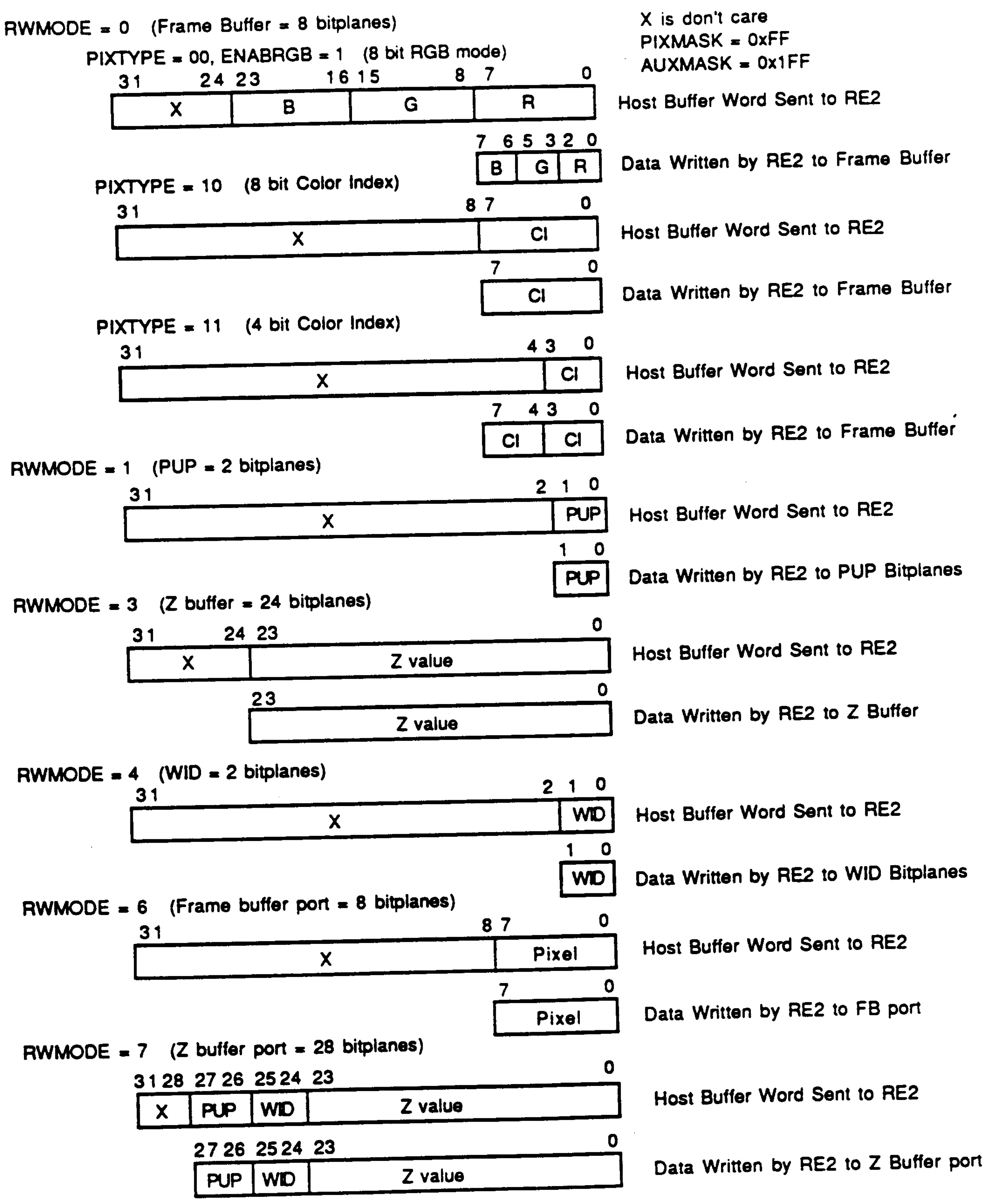

X is don't care
PIXMASK = 0xFF
AUXMASK = 0x1FF
RWMODE = 0 (Frame Buffer = 8 bitplanes)
PIXTYPE = 00, ENABRGB = 1 (8 bit RGB mode)
31 24 23 16 15 8 7 0
X B G R
Host Buffer Word Sent to RE2
7 6 5 3 2 0
B G R
Data Written by RE2 to Frame Buffer
PIXTYPE = 10 (8 bit Color Index)
31 8 7 0
X CI
Host Buffer Word Sent to RE2
7 0
CI
Data Written by RE2 to Frame Buffer
PIXTYPE = 11 (4 bit Color Index)
31 4 3 0
X CI
Host Buffer Word Sent to RE2
7 4 3 0
CI CI
Data Written by RE2 to Frame Buffer
RWMODE = 1 (PUP = 2 bitplanes)
31 2 1 0
X PUP
Host Buffer Word Sent to RE2
1 0
PUP
Data Written by RE2 to PUP Bitplanes
RWMODE = 3 (Z buffer = 24 bitplanes)
31 24 23 0
X Z value
Host Buffer Word Sent to RE2
23 0
Z value
Data Written by RE2 to Z Buffer
RWMODE = 4 (WID = 2 bitplanes)
31 2 1 0
X WID
Host Buffer Word Sent to RE2
1 0
WID
Data Written by RE2 to WID Bitplanes
RWMODE = 6 (Frame buffer port = 8 bitplanes)
31 8 7 0
X Pixel
Host Buffer Word Sent to RE2
7 0
Pixel
Data Written by RE2 to FB port
RWMODE = 7 (Z buffer port = 28 bitplanes)
31 28 27 26 25 24 23 0
X PUP WID Z value
Host Buffer Word Sent to RE2
27 26 25 24 23 0
PUP WID Z value
Data Written by RE2 to Z Buffer port

Figure 6.103 Base Adapter Pixel Formats for DMA Writes

PUP Pixel Formats

The base adapter contains 2 Pop Up (PUP) bitplanes which can be used to display either overlay or underlay colors. The 2 bitplanes are masked by the LSB 2 bits of the AUXMASK register. The value placed in the PUP bitplanes is a color index into the overlay palette in the RGB RAMDAC. Refer to the Display Subsystem chapter for additional information on the use of the PUP bitplanes. If pixel packing is not used then the single PUP pixel must be right justified in the word received by the RE2. If pixel packing is done then a maximum of four PUP pixels can be packed into the 32 bit word and each 2 bit pixel will be right justified in the byte that it is in.

WID Pixel Formats

The base adapter contains 2 Window ID (WID) bitplanes which are used as an index into the XPC1 mode registers. The mode registers specify the display modes that a particular window will have. The WID values can also be used to prevent drawing outside of a specified window. The 2 WID bitplanes are masked by bits 4 and 5 of the AUXMASK register. Refer to the Display Subsystem chapter for additional information on the use of the WID bitplanes. If pixel packing is not used then the single WID pixel must be right justified in the word received by the RE2. If pixel packing is done then a maximum of four WID pixels can be packed into the 32 bit word and each 2 bit pixel will be right justified in the byte that it is in.

Z Buffer Pixel Formats

The base adapter can have an optional Z buffer card installed which contains 24 Z buffer bitplanes which are used to perform depth (Z) comparisons to enable or disable pixels from being updated. Normally the host only accesses the Z buffer bitplanes to set an initial depth value. After that the RE2 updates the values in the Z buffer bitplanes. The Z buffer bitplanes are masked by bit 8 in the AUXMASK. Since the host buffer words contain 24 bit Z data pixel packing cannot be used. The 24 bit Z value must be right justified in the 32 bit word.

Frame Buffer Port Pixel Formats

The base adapter has a Frame Buffer port which only contains the frame buffer pixel and no other pixel data. The RE2 does not perform the same pixel format manipulations as described above for the frame buffer bitplanes. This means that the host buffer must format the frame buffer pixel data in the same manner that the RE2 would do if the frame buffer bitplanes were being written (RWMODE = 0). For example if the host wanted to write 4 bit CI pixels then it would have to duplicate the 4 bit CI value into the upper nibble of the byte and then place the byte in the host word right justified. The PIXMASK value would determine which buffer actually had the 4 bit CI value written into it. The bitplanes written through the Frame Buffer port can be WID checked. This means that on the base adapter only the frame buffer pixel data can be WID checked.

Z Buffer Port Pixel Formats

The base adapter has a Z buffer port which allows the host to access the Z buffer, WID and PUP bitplanes all at once. The 24 bit Z value is placed in bits 0 to 23, the WID data in bits 24 and 25 and the PUP data in bits 26 and 27 of the host buffer. The format of these pixels is the same as described above with the exception that the WID and PUP pixel bits must be placed in the indicated bit positions rather than being right justified. The bitplanes written through the Z Buffer port cannot be WID checked. This means that the Z buffer, WID and PUP pixels cannot be WID checked on the base adapter.

Enhanced Adapter Pixel Formats

The enhanced adapter has 24 frame buffer bitplanes, 2 PUP bitplanes, 2 UAUX bitplanes, 2 WID bitplanes and optionally 24 Z buffer bitplanes. These bitplanes can all be written to using DMA transfers. The pixel formats for the base adapter DMA writes are shown in Figure 6.104 and the following paragraphs describe the pixel formats of the data which must be placed in the host buffer. They also describes how the data is modified by the RE2 before it is written into the selected bitplanes.

Frame Buffer Pixel Formats

When the PIXTYPE is set to 00 on the enhanced adapter the ENABRGB register setting is ignored and the pixel type is 24 bit RGB. The host buffer must contain data in a 24 bit RGB format and the entire 24 bits is written into the frame buffer. PIXTYPE equal to 00 is normally used as a single buffer mode and PIXMASK is set to 0xFFFFFF. Since the host buffer words contain 24 bit RGB data pixel packing cannot be used. The 24 bit RGB pixel value must be right justified in the 32 bit word.

When the PIXTYPE is set to 01 the 24 bit RGB pixel data is handled by the RE2 as 12 bit RGB pixels. The most significant nibble of each of the three color components is used. The RE2 duplicates the upper nibble into the lower nibble for each color component and writes that value out to the bitplanes. This PIXTYPE is used normally in a double buffer mode. The PIXMASK would be set to 0xOFOFOF to allow only the front buffer to be updated and it would be set to 0xF0F0F0 to allow only the back buffer to be written. As shown in Figure 6.104, if the PIXMASK is set to 0xFFFFFF the upper nibble of each color component would be written into both buffers. Since the host buffer words contain 24 bit RGB data pixel packing cannot be used. The 24 bit RGB pixel value must be right justified in the 32 bit word.

When PIXTYPE is set to 10 the RE2 uses the right most 12 bits of each word it receives as a 12 bit Color Index (CI) value. The 12 CI value is duplicated into the upper 12 bits and the 24 bits are written into the frame buffer bitplanes. This pixtype can be used for single buffer data or double buffer data. For single buffer use the PIXMASK would normally be set to 0x000FFF to write the 12 bit CI value into only the front buffer. For double buffer applications the PIXMASK would be set to 0x000FFF for the front buffer and to 0xFFF000 for the back buffer. As shown in Figure 6.104, if the PIXMASK is set to 0xFFFFFF the 12 bit CI value would be written into both buffers. If pixel packing is not used then the single 12 bit CI pixel must be right justified in the word received by the RE2. If pixel packing is done then a maximum of two 12 bit CI pixels can be packed into the 32 bit word and each 12 bit CI pixel must be right justified in the 16 bits which hold it..

When PIXTYPE is set to 11 the RE2 uses the right most 4 bits of each word it receives as a 4 bit Color Index value. The RE2 duplicates the 4 bit CI value into the upper nibble and writes the two nibbles to the frame buffer bitplanes. This PIXTYPE is not normally used on the enhanced adapter since PIXTYPE 10 provides greater color selection. If pixel packing is not used then the single 4 bit CI pixel must be right justified in the word received by the RE2. If pixel packing is done then a maximum of four 4 bit CI pixels can be packed into the 32 bit word and each 4 bit CI pixel will be right justified in the byte that it is in.

PUP Pixel Formats

The enhanced adapter contains 2 Pop Up (PUP) bitplanes which can be used to display either overlay or underlay colors. The 2 bitplanes are masked by the LSB 2 bits of the AUXMASK register. The value placed in the PUP bitplanes is a color index into the auxiliary color map in the XMAP2 chips. Refer to the Display Subsystem chapter for additional information on the use of the PUP bitplanes. These bitplanes are usually accessed by the window manager. The PUP bitplanes can be used as UAUX bits if the NOPUP register is set. If pixel packing is not used then the single PUP pixel must be right justified in the word received by the RE2. If pixel packing is done then a maximum of four PUP pixels can be packed into the 32 bit word and each 2 bit pixel will be right justified in the byte that it is in.

UAUX Pixel Formats

The enhanced adapter contains 2 or 4 User Auxiliary (UAUX) bitplanes which can be used to display either overlay or underlay colors. Normally the adapter is configured to use the 4 auxiliary bitplanes as 2 bits of UAUX and 2 bits of PUP. The NOPUP register controls the configuration and if it is set to 1 then the 4 auxiliary bitplanes are used as 4 UAUX bitplanes. If the NOPUP register is 0 then the normal configuration of 2 UAUX and 2 PUP bitplanes is used. The host uses the GE_LOADGE token to set the UAUX_4BIT flag to instruct the GE5 microcode on how to set the NOPUP register. The UAUX bitplanes are masked by bits 2 and 3 of the AUXMASK if NOPUP is 0 and by bits 0 to 3 if NOPUP is 1. The value placed in the UAUX bitplanes is a color index into the auxiliary color map in the XMAP2 chips. Refer to the Display Subsystem chapter for additional information on the use of the UAUX bitplanes. If pixel packing is not used then the single UAUX pixel must be right justified in the word received by the RE2. If pixel packing is done then a maximum of four UAUX pixels can be packed into the 32 bit word and each UAUX bit pixel will be right justified in the byte that it is in.

WID Pixel Formats

The enhanced adapter contains 4 Window ID (WID) bitplanes which are used as an index into the XMAP2 mode registers. The mode registers specify the display modes that a particular window will have. The WID values can also be used to prevent drawing outside of a specified window. The 4 WID bitplanes are masked by bits 4 to 7 of the AUXMASK register. Refer to the Display Subsystem chapter for additional information on the use of the WID bitplanes. If pixel packing is not used then the single WID pixel must be right justified in the word received by the RE2. If pixel packing is done then a maximum of four WID pixels can be packed into the 32 bit word and each 4 bit pixel will be right justified in the byte that it is in.

Z Buffer Pixel Formats

The enhanced adapter can have an optional Z buffer card installed which contains 24 Z buffer bitplanes which are used to perform depth (Z) comparisons to enable or disable pixels from being updated. Normally the host only accesses the Z buffer bitplanes to set an initial depth value. After that the RE2 updates the values in the Z buffer bitplanes. The Z buffer bitplanes are masked by bit 8 in the AUXMASK. Since the host buffer words contain 24 bit Z data pixel packing cannot be used. The 24 bit Z pixel value must be right justified in the 32 bit word.

Frame Buffer Port Pixel Formats

The enhanced adapter has a frame buffer port which allows the host to access the frame buffer, PUP and UAUX bitplanes at the same time. The frame buffer pixel value is placed in bits 0 to 23, the PUP data in bits 24 and 25 and the UAUX data in bits 26 and 27 of the host buffer. If the NOPUP register is set then bits 24 to 27 are all UAUX bits. The RE2 does not perform the same pixel format manipulations as described above for the frame buffer bitplanes. The host must perform the equivalent pixel formatting as described above for RWMODE equal to 0. For example the host would have to duplicate the upper nibble of each RGB color component into the lower nibble if it were writing 12 bit RGB pixels into the frame buffer bitplanes using the frame buffer port. The value in the PIXMASK would then determine which buffer received the pixel data. The bitplanes written through the Frame Buffer port can be WID checked. This means that the Frame Buffer, PUP and UAUX pixels can be WID checked on the enhanced adapter.

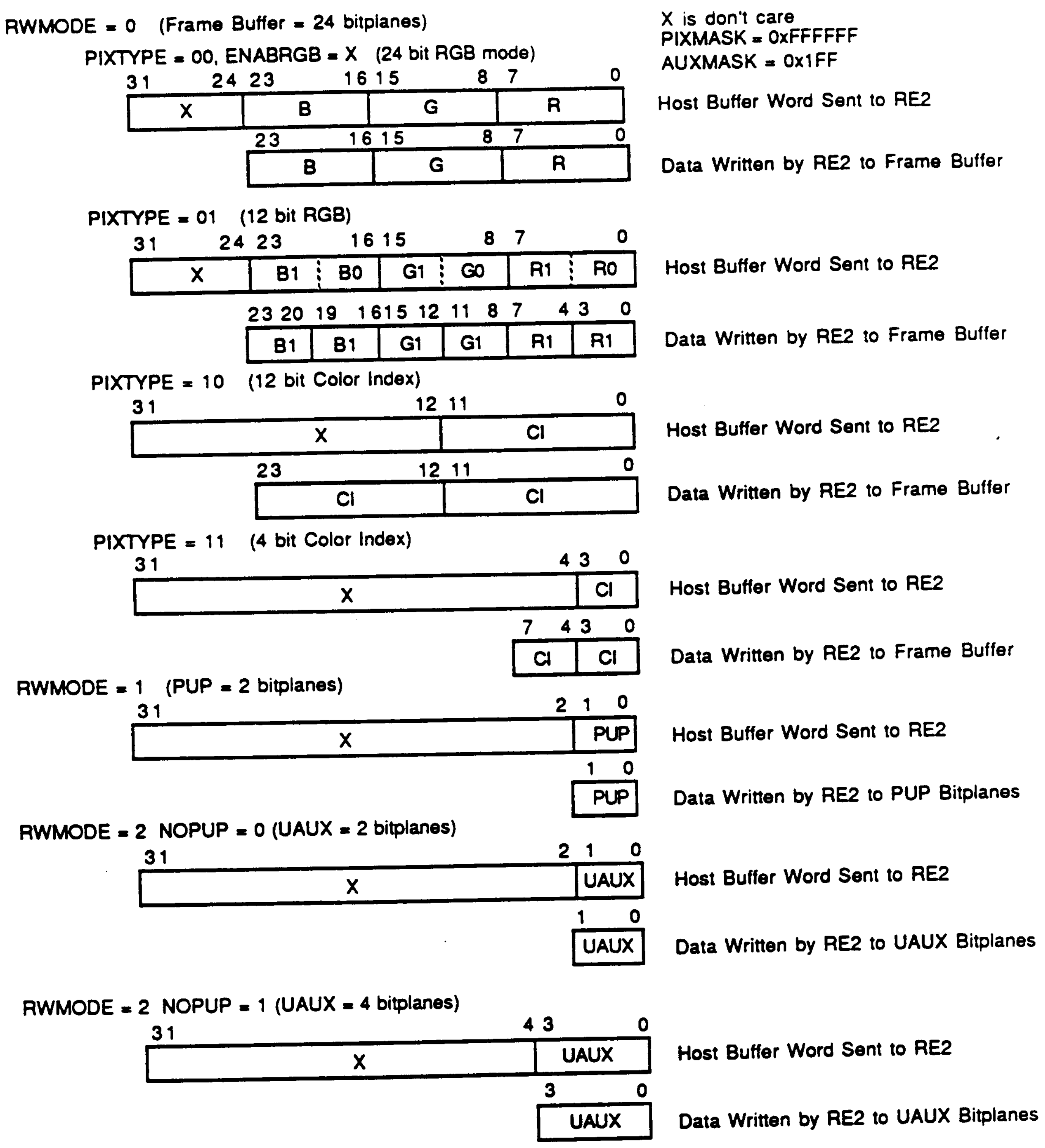

RWMODE = 0 (Frame Buffer = 24 bitplanes)
X is don't care
PIXMASK = 0xFFFFFF
AUXMASK = 0x1FF
PIXTYPE = 00, ENABRGB = X (24 bit RGB mode)
31 24 23 16 15 8 7 0
X B G R
Host Buffer Word Sent to RE2
23 16 15 8 7 0
B G R
Data Written by RE2 to Frame Buffer
PIXTYPE = 01 (12 bit RGB)
31 24 23 16 15 8 7 0
X B1 B0 G1 G0 R1 R0
Host Buffer Word Sent to RE2
23 20 19 16 15 12 11 8 7 4 3 0
B1 B1 G1 G1 R1 R1
Data Written by RE2 to Frame Buffer
PIXTYPE = 10 (12 bit Color Index)
31 12 11 0
X CI
Host Buffer Word Sent to RE2
23 12 11 0
CI CI
Data Written by RE2 to Frame Buffer
PIXTYPE = 11 (4 bit Color Index)
31 4 3 0
X CI
Host Buffer Word Sent to RE2
7 4 3 0
CI CI
Data Written by RE2 to Frame Buffer
RWMODE = 1 (PUP = 2 bitplanes)
31 2 1 0
X PUP
Host Buffer Word Sent to RE2
1 0
PUP
Data Written by RE2 to PUP Bitplanes
RWMODE = 2 NOPUP = 0 (UAUX = 2 bitplanes)
31 2 1 0
X UAUX
Host Buffer Word Sent to RE2
1 0
UAUX
Data Written by RE2 to UAUX Bitplanes
RWMODE = 2 NOPUP = 1 (UAUX = 4 bitplanes)
31 4 3 0
X UAUX
Host Buffer Word Sent to RE2
3 0
UAUX
Data Written by RE2 to UAUX Bitplanes

Figure 6.104 Enhanced Adapter Pixel Formats for DMA Writes

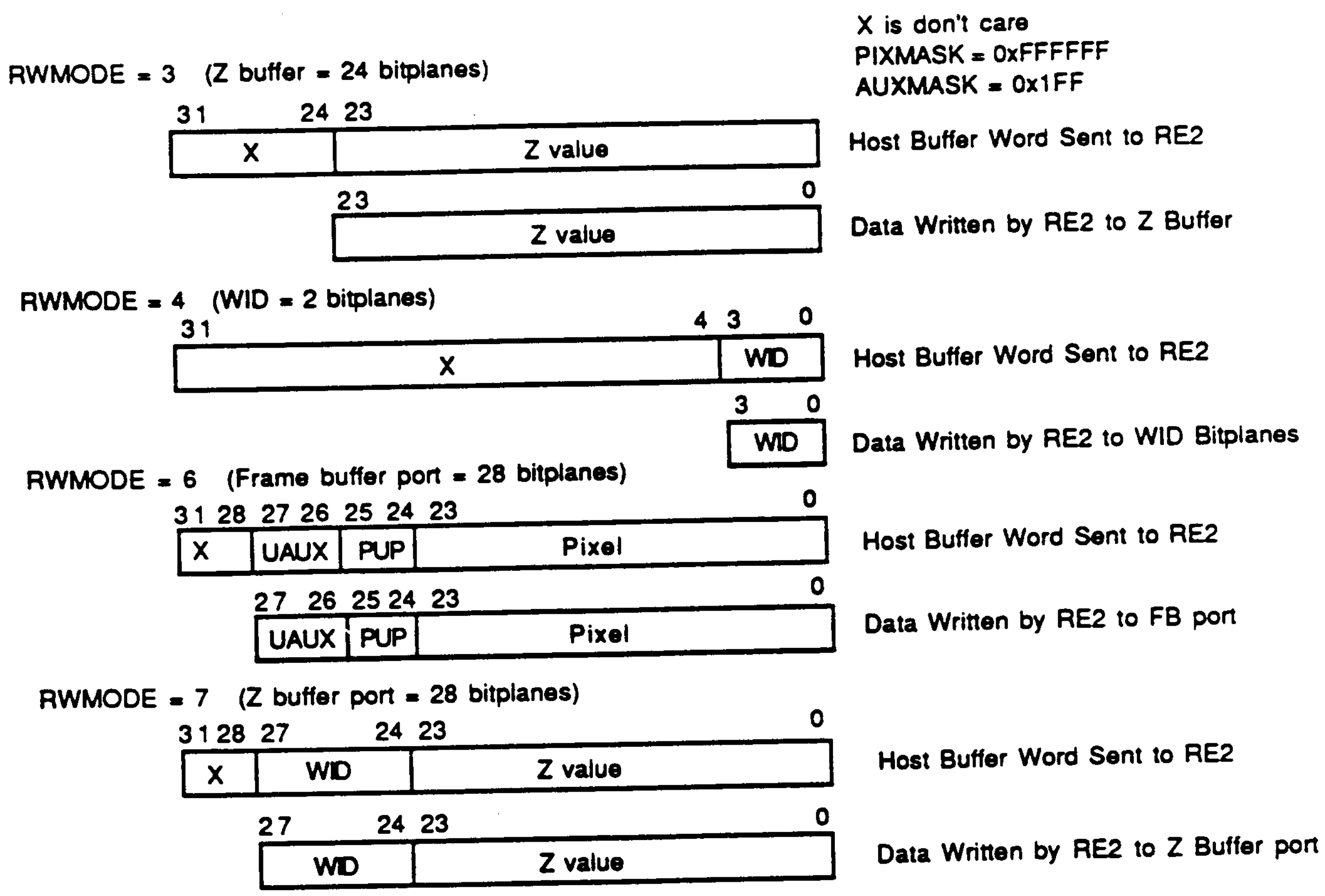

X is don't care
PIXMASK = 0xFFFFFF
AUXMASK = 0x1FF
RWMODE = 3 (Z buffer = 24 bitplanes)
31 24 23
X
Z value
Host Buffer Word Sent to RE2
23 0
Z value
Data Written by RE2 to Z Buffer
RWMODE = 4 (WID = 2 bitplanes)
31 4 3 0
X
WID
Host Buffer Word Sent to RE2
3 0
WID
Data Written by RE2 to WID Bitplanes
RWMODE = 6 (Frame buffer port = 28 bitplanes)
31 28 27 26 25 24 23 0
X
UAUX
PUP
Pixel
Host Buffer Word Sent to RE2
27 26 25 24 23 0
UAUX
PUP
Pixel
Data Written by RE2 to FB port
RWMODE = 7 (Z buffer port = 28 bitplanes)
31 28 27 24 23 0
X
WID
Z value
Host Buffer Word Sent to RE2
27 24 23 0
WID
Z value
Data Written by RE2 to Z Buffer port

Figure 6.104 Enhanced Adapter Pixel Formats for DMA Writes (cont.)

Z Buffer Port Pixel Formats

The enhanced adapter has a Z buffer port which allows the host to access the Z buffer and WID bitplanes at the same time. The 24 bit Z value is placed in bits 0 to 23 and the WID data in bits 24 to 27 of the host buffer. The format of these pixels is the same as described above with the exception that the WID pixel bits must be placed in the indicated bit positions rather than being right justified. The bitplanes written through the Z Buffer port cannot be WID checked. This means that the Z buffer and WID pixels cannot be WID checked.

Pixel Zooming

The host specifies the x and y zoom factors with the GE_ZOOMFACTOR token prior to issuing the GE_RECTCOPY or the GE_WRITEBLOCK tokens. These two tokens use the x and y zoom factors to perform the pixel zooming. When doing a zoom operation the RE2 chip can perform the zoom in the x direction while the GE5 microcode must perform the zoom in the y direction by causing the same pixel lines to be written the y zoom count number of times.

Write Buffer Instruction

To perform a bitplane write DMA transfer the microcode must set up the proper RE2 registers and then issue the Write Buffer instruction to the RE2. This causes the RE2 to process the pixel data which it receives in the RWDATA register. The write DMA transfer is controlled by the HQ1 chip in the Geometry Subsystem. The HQ1 executes a microcode instruction which instructs it to perform the necessary hardware operations to transfer data words from the Host Buffer or the GE5 Data RAM to the RWDATA register in the RE2. The number of word transfers is specified by the count that the microcode had previously loaded into the HQ1 DMA Count register. For each word placed in the RWDATA register the RE2 performs the necessary unpacking of the pixels in the word and then it uses the x zoom count to determine how many times it needs to process each pixel value. For each pixel the RE2 does the screen mask clipping, WID checking and pattern mask checking. If the checks are passed then the RE2 does any necessary pixel formatting of the pixel data and then writes it to the selected bitplanes. These concepts will be explained in greater detail in later paragraphs.

Register Usage

The following list of RE2 registers are the registers which are normally specified and used each time a new Write Buffer instruction is executed.

- X	contains the starting x screen location
- Y	Contains the starting y screen location
- DX = +1	Contains the change in x which is added to the current x location to get the next x location
- DY = 0	Contains the change in y which is added to the current y location to get the next y location
- NUMPIX	Contains a pixel count for the number of pixels to be written
- PIXTYPE	Specifies the pixel type being written if the frame buffer is the destination
- UPACMODE	Contains the number of pixel/32 bit word
- HADDR	Specifies the offset of the first pixel in the first word
- XZOOM	Specifies the number of times each incoming pixel is to be written
- RWMODE	Specifies the destination bitplanes to be written
- RWDATA	The input data port to the RE2 for the Write Buffer instruction

The following registers affect the operation of the Write Buffer instruction but are usually written by the microcode as it executes other tokens and then they remain in effect during the execution of multiple Write Buffer instructions as well as other instructions.

- XMIN	Lower left x coordinate of the hardware screen mask rectangle
- YMIN	Lower left y coordinate of the hardware screen mask rectangle
- XMAX	Upper right x coordinate of the hardware screen mask rectangle
- YMAX	Upper right y coordinate of the hardware screen mask rectangle
- FUNC	Contains the Logical Operation to be applied to each pixel
- PATL, PATH	Contains the 32 bit pattern mask
- ENABPAT	Determines if the pattern mask is enabled
- ENABWID	Determines if the WID checking is performed for each pixel
- FBOPTION	Specifies the adapter type (0 = base, 1 = enhanced)
- ENABRGB	Specifies that 8 bit RGB pixels are to be written on the base adapter
- NOPUP	Specifies if 2 or 4 UAUX bitplanes are available on the enhanced adapter
- DEPTHFN	Bit 3 controls the FastZclear mode and affects WID checking
- CURWID	Specifies the Current WID for the WID checking operations

Once the RE2 registers have been set up and the Write Buffer instruction issued then the RE2 will process the incoming pixel data and determine which pixels are written. During the execution of a Write Buffer instruction only pixels on the current scan line specified by the Y register can be written. The first pixel to be written on the selected scan line is specified by the X register. The following pixel writes then proceed to the right along the scan line. The number of pixels to be written is specified in the NUMPIX register.

Execution Flow

A logical representation of the Write Buffer instruction execution flow is shown in the flow diagram shown in Figure 6.105. The NUMPIX register contains the count for the number of pixels to be processed by the Write Buffer instruction. For each pass through the main loop the NUMPIX count is decremented by the number of pixels processed while going through the loop. The Write Buffer instruction continues to execute while NUMPIX is greater than zero. The main loop represents the processing of each new word received in the RWDATA register. The next inner loop represents the unpacking of the individual pixels from the RWDATA word and finally the inner most loop represents the processing of each individual pixel XZOOM times.

For each pass through the outer most loop the following major operations are performed.

- get the next 32 bit word from the RWDATA register
- unpack the pixels if they are packed
- process each pixel the number of times specified in the XZOOM register
- determine if the pixel value should be written into the current pixel location
- if the pixel can be written do any necessary pixel formatting and write the pixel
- update the pixel location and decrement the NUMPIX register
- when all the pixels in the current 32 bit word have been processed go get the next word

The RWDATA register is the input data port for the Write Buffer instruction. During a host to RE or a GE5 to RE DMA write operation the microcode executes a loop instruction that transfers the number of words specified by the DMA count register to the RE2 RWDATA register one at a time. The RE2 then processes each of the incoming words. The RE2 uses the value in the UPACMODE register to determine if the incoming words contain packed pixel data which must be unpacked. The UPACMODE register contains the number of pixels per 32 bit word minus one. If the value in UPACMODE is greater than zero (meaning 1 pixel/word) then the incoming data must be unpacked and each pixel processed individually. The HADDR register is used to determine the offset to the first pixel in the first word. The first pixel and the others to the right of it in the first word are processed. All the pixels in the following words are processed until the number specified in NUMPIX have been processed. At that time the RE2 has completed the current Write Buffer instruction.

For each incoming pixel the RE2 uses the value in the XZOOM register to determine how many pixel locations the pixel value will be written into. For each pixel location the RE2 must determine whether the pixel can be written or not. For the Write Buffer instruction the RE2 performs a screen mask clipping check, a WID check if it is enabled and a pattern mask check if it is enabled. The screen mask check cannot be disabled and is performed on all pixel writes. If the current pixel location specified by the X and Y registers is outside the screen mask boundary then the pixel is clipped and is not written.

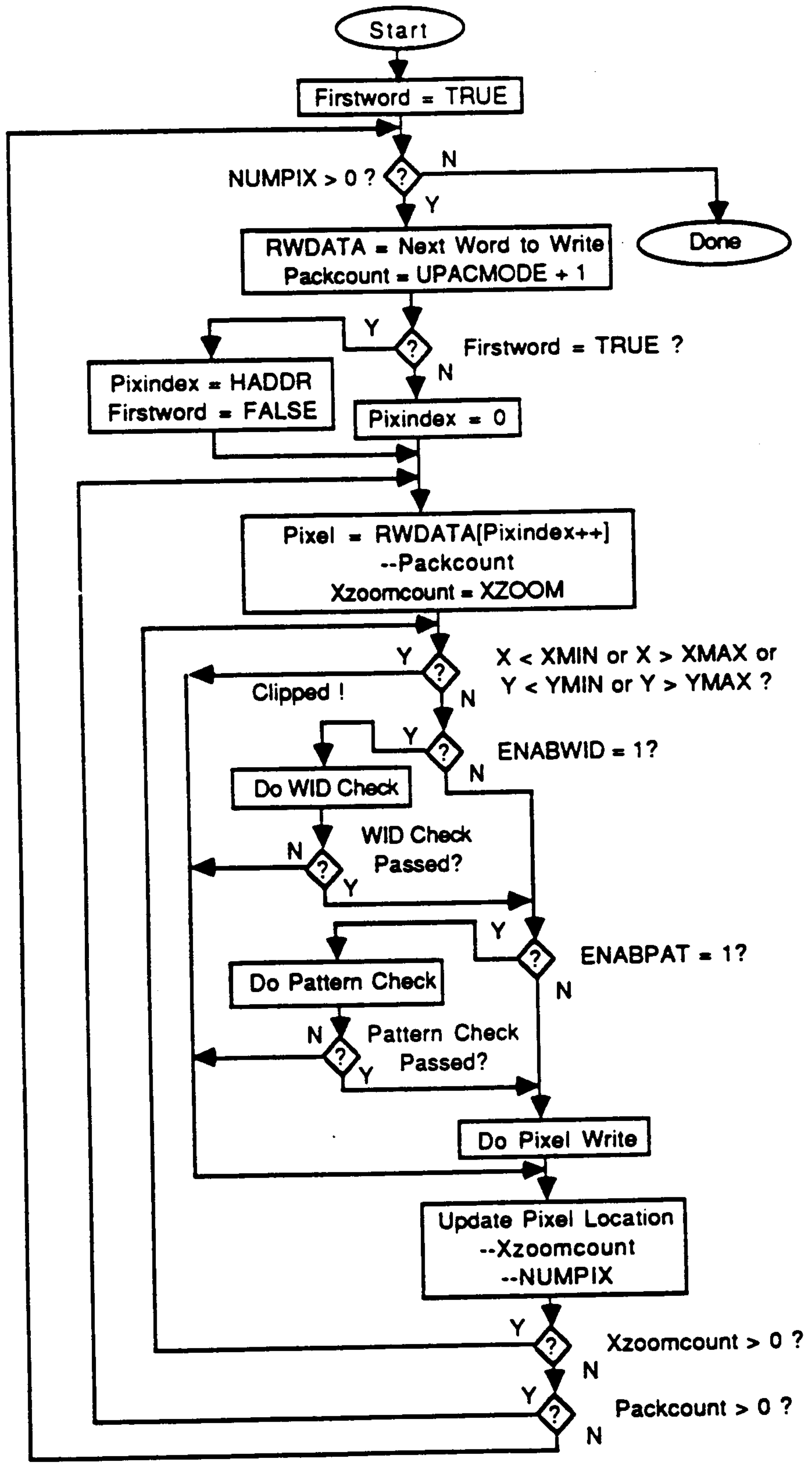


Figure 6.105 Write Buffer Instruction Flow Chart

For pixels which are not clipped, if WID checking has been enabled a WID check is performed and pixels which fail the WID check are not written. Finally for pixels which have passed the clipping and WID checks, a pattern mask check is done if it is enabled. The appropriate bit offset into the pattern mask is calculated based on the current pixel location. If that pattern mask bit is a 1 then the pixel is written. If the pattern mask bit is a 0 then the pixel is not written.

Pixels which have passed all of the above tests will be written. Before the pixel data is written the RE2 does any necessary pixel formatting based on the destination bitplanes and the pixel type. Refer to the pixel format section above for a description of the pixel formatting which is performed. Finally the logical operation specified in the FUNC register is performed on the pixel data. The appropriate write mask is applied to the bits in the data and the unmasked bits in the selected bitplanes are updated with the pixel value.

Once the pixel has been processed and written or bypassed the RE2 moves to the next pixel location. The next pixel location is calculated by adding the DX register to the X register and the DY register to the Y register. Since the DX register is set to +1 and the DY register is set to 0, the next pixel location is to the right of the current location on the same scan line. After the new pixel location has been calculated the NUMPIX register is decremented and the Xzoomcount is decremented so the next pixel location can be processed. Once the same pixel value has been used for Xzoomcount pixel locations the inner most loop is exited and the next pixel value is obtained. If the incoming word contained packed pixel data the next pixel value is obtained from the current RWDATA word. If all the pixels in the current word have been processed then pixel unpacking loop is exited and the next word received in the RWDATA register is processed. The RE2 continues in this fashion until it has repeated the process for NUMPIX pixel locations. The WID checks performed for the Write Buffer instruction are described in the WID checking paragraphs earlier in this chapter. The following paragraphs describe the pixel writing steps in greater detail.

Pixel Writes for the Write Buffer Instruction

For each pixel location that the RE2 writes to it writes both the Frame Buffer port data and the Z buffer port data. This is true for all pixel writes regardless of which instruction the RE2 is currently executing. The individual instructions determine what data is placed in the Frame Buffer and Z buffer ports to be written. For the Write Buffer instruction the RWMODE register specifies where the incoming pixel data is placed in the two ports. For other instructions the pixel data is computed internally by the RE2.

If the RWMODE register is zero then the pixel data in the RWDATA register is processed and placed in the Frame Buffer part of the Frame Buffer port. The processing consists of using the current PIXTYPE value and doing the formatting described in the Pixel Formats section. Once the pixel data has been formatted then the logical operation specified in the FUNC register is done on the data and the result is placed in the Frame Buffer field of the Frame Buffer port. The PUP, UAUX, WID and Z buffer fields of the Frame Buffer and Z Buffer ports will be undefined when the RWMODE is zero. Finally the data in the two ports is written out to the bitplanes. The PIXMASK and the AUXMASK bits which are set to one allow the corresponding bitplanes to be updated while the mask bits which are zero prevent the corresponding bits from being updated.

If the RWMODE register is one then the pixel data in the RWDATA register is processed and placed in the PUP field of the Frame Buffer port on the enhanced adapter and in the Z Buffer port on the base adapter. The processing consists of shifting the data sent by the host the necessary number of bits to the left to align it to the proper location in the Frame Buffer or Z Buffer port. Once the pixel data has been shifted left then the logical operation specified in the FUNC register is done on the data and the result is placed in the PUP field of the Frame Buffer or Z Buffer port. The Frame Buffer, UAUX, WID and Z buffer fields of the Frame Buffer and Z Buffer ports will be undefined when the RWMODE is one. Finally the data in the two ports are written out to the bitplanes. The PIXMASK and the AUXMASK bits which are set to one allow the corresponding bitplanes to be updated while the mask bits which are zero prevent the corresponding bits from being updated.

If the RWMODE register is two then the pixel data in the RWDATA register is processed and placed in the UAUX field of the Frame Buffer port. The processing consists of shifting the data sent by the host the necessary number of bits to the left to align it to the proper location in the Frame Buffer port. Once the pixel data has been shifted left then the logical operation specified in the FUNC register is done on the data and the result is placed in the UAUX field of the Frame Buffer port. The Frame Buffer, PUP, WID and Z buffer fields of the Frame Buffer and Z Buffer ports will be undefined when the RWMODE is two. Finally the data in the two ports are written out to the bitplanes. The PIXMASK and the AUXMASK bits which are set to one allow the corresponding bitplanes to be updated while the mask bits which are zero prevent the corresponding bits from being updated.

If the RWMODE register is three then the pixel data in the RWDATA register is placed in the Z data field of the Z Buffer port. The Frame Buffer, PUP, UAUX and WID fields of the Frame Buffer and Z Buffer ports will be undefined when the RWMODE is three. Finally the data in the two ports are written out to the bitplanes. The PIXMASK and the AUXMASK bits which are set to one allow the corresponding bitplanes to be updated while the mask bits which are zero prevent the corresponding bits from being updated.

If the RWMODE register is four then the pixel data in the RWDATA register is processed and placed in the WID field of the Z Buffer port. The processing consists of shifting the data sent by the host the necessary number of bits to the left to align it to the proper location in the Z Buffer port. Once the pixel data has been shifted left then the logical operation specified in the FUNC register is done on the data and the result is placed in the WID field of the Z Buffer port. The Frame Buffer, PUP, UAUX and Z buffer fields of the Frame Buffer and Z Buffer ports will be undefined when the RWMODE is four. Finally the data in the two ports are written out to the bitplanes. The PIXMASK and the AUXMASK bits which are set to one allow the corresponding bitplanes to be updated while the mask bits which are zero prevent the corresponding bits from being updated.

If the RWMODE register is six then the pixel data in the RWDATA register is placed directly into the Frame Buffer port with no processing or bit shifting. The host must place the properly formatted data in the pixel data sent to the RE2. When writing to the Frame Buffer port directly the host must be aware of whether it is writing to the base or the enhanced adapter so that it places the proper data into the proper locations in the data words it sends. The Z Buffer port data will be undefined when the RWMODE is six. Finally the data in the two ports are written out to the bitplanes. The PIXMASK and the AUXMASK bits which are set to one allow the corresponding bitplanes to be updated while the mask bits which are zero prevent the corresponding bits from being updated.

If the RWMODE register is seven then the pixel data in the RWDATA register is placed directly into the Z Buffer port with no processing or bit shifting. The host must place the properly formatted data in the pixel data sent to the RE2. When writing to the Z Buffer port directly the host must be aware of whether it is writing to the base or the enhanced adapter so that it places the proper data into the proper locations in the data words it sends. The Frame Buffer port data will be undefined when the RWMODE is seven. Finally the data in the two ports are written out to the bitplanes. The PIXMASK and the AUXMASK bits which are set to one allow the corresponding bitplanes to be updated while the mask bits which are zero prevent the corresponding bits from being updated.

This completes the description of the RE2 registers, the following paragraphs describe the clearing of the various bitplanes and the setting of the WID bitplanes.

Clearing the Bitplanes

The host software is responsible for clearing the various bitplanes. After power-on or a reset operation the contents of the bitplanes are undefined. The following paragraphs describe the necessary operations required to clear the Frame Buffer bitplanes, the PUP bitplanes, the UAUX bitplanes, the WID bitplanes and the Z buffer bitplanes.

Clearing the Frame Buffer Bitplanes

The GE_DRAWMODE token must be set to select the Frame Buffer bitplanes and then the GE_COLOR or GE_RGBCOLOR token must be used to specify the color which the bitplanes are to be cleared to. The color token depends on the pixel type selected with the GE_PIXTYPE token. The GE_WRITEPIXMASK token should be used to set the Frame Buffer pixel write mask to all ones so that all the Frame Buffer bitplanes are cleared. The GE_AUXWRITEMASK token should be set to zero so that the PUP, UAUX, WID and Z bitplanes are not affected when the Frame Buffer bitplanes are cleared. Finally the GE_SCREENCLEAR token is used to clear the Frame Buffer bitplanes. The GE_SETPIECES token must have been previously used to set the piece list of the 1 to 4 rectangular pieces that will be cleared. For a full screen clear the piece list should be set to 1 piece that has a single rectangle that covers the entire screen.

Clearing the PUP Bitplanes

The GE_DRAWMODE token must be set to select the PUP bitplanes and then the GE_COLOR token is used to specify the color which the bitplanes are to be cleared to. The color is normally a 0 so that the PUP bitplanes are cleared to transparency. The GE_WRITEPIXMASK token should be used to set the Frame Buffer pixel write mask to all zeroes so that all the Frame Buffer bitplanes are not affected by the PUP bitplane clear operation. The GE_AUXWRITEMASK token should be set to 3 so that the UAUX, WID and Z bitplanes are not affected when the PUP bitplanes are cleared. Finally the GE_SCREENCLEAR token is used to clear the PUP bitplanes.

Clearing the UAUX Bitplanes

The GE_DRAWMODE token must be set to select the UAUX bitplanes and then the GE_COLOR token is used to specify the color which the bitplanes are to be cleared to. The color is normally a 0 so that the UAUX bitplanes are cleared to transparency. The GE_WRITEPIXMASK token should be used to set the Frame Buffer pixel write mask to all zeroes so that all the Frame Buffer bitplanes are not affected by the UAUX bitplane clear operation. The GE_AUXWRITEMASK token should be set to 0x0C so that the PUP, WID and Z bitplanes are not affected when the UAUX bitplanes are cleared. Finally the GE_SCREENCLEAR token is used to clear the UAUX bitplanes.

Clearing the WID Bitplanes

The GE_DRAWMODE token must be set to select the WID bitplanes and then the GE_COLOR token is used to specify the WID index value which the bitplanes are to be cleared to. The index is normally a 0 so that window 0 is selected. The index could be any of the 16 possible WID values however. The GE_WRITEPIXMASK token should be used to set the Frame Buffer pixel write mask to all zeroes so that all the Frame Buffer bitplanes are not affected by the WID bitplane clear operation. The GE_AUXWRITEMASK token should be set to 0xF0 so that the PUP, UAUX and Z bitplanes are not affected when the WID bitplanes are cleared. Finally the GE_SCREENCLEAR token is used to clear the WID bitplanes.

Clearing the Z Buffer Bitplanes

The Z Buffer can be cleared in two ways one of which is a fast Z clear and the other which takes four times longer to clear the Z Buffer. For the slower method the microcode loads the desired Z value into the Z register and executes Flat 1 Span instructions to write the Z value into the Z Buffer. The Flat 1 Span must be used since the Z Buffer is formed by DRAM chips and the special block write mode provided by the 1 Meg VRAM chips is not available in the DRAM chips. The AUXMASK bit 8 must be set to one for the Z Buffer writes to be allowed of course. The GE_CZCLEAR tokens can be used to clear the Z Buffer bitplanes by the host software.

The fast Z clear is a special RE2 mode provided to allow the Z Buffer clear to be performed four times faster using the Flat 4 Span instruction. This mode allows the Z value to be invalidated so that if Z checking is being performed the Z check automatically passes. The fast Z clear is performed by writing a one into the LSB bits of the WID bitplanes and by enabling the Z invalidate by setting bit 3 in the DEPTHFN register. The WIDDATA should then be set to 0 so that after the Z value has been made invalid the new Z value is written and the LSB of the WID bitplanes is cleared to zero turning off the Z invalidate for this pixel. The GE_DEPTHFN token can be used to set the DEPTHFN register value.

Setting the WID bitplanes

The WID bitplanes are normally written only by the host window manager software. The contents of the WID bitplanes are used to clip pixel writes outside of the current window when WID checking is enabled. The WID values are also used as an index into the mode registers in the XPC1 or XMAP2 chips in the Display Subsystem. When writing the WID bitplanes the host software should disable WID checking, Z Buffer checking and pattern masking. The window manager writes the desired window ID into the WID bitplanes when a window is created. If the window is a single rectangle the screen mask can be set to the size of the rectangle and the GE_SCREENCLEAR token used to clear the screen as described above for the WID bitplane clearing.

The window manager also needs to program the WID bitplanes when a window is moved or is pushed behind other windows or is popped in front of other windows. If the window is 1 to 4 rectangular pieces the GE_SCREENCLEAR token can be used to write the WID bitplanes. This assumes the GE_SETPIECES token has been used to set the piece list to the rectangular areas that comprise the screen. If the window is comprised of more than 4 rectangular pieces the GE_SBOXFI token can be used to write the multiple pieces or multiple piece list can be set and the GE_SCREENCLEAR token used to write the pieces.

Programming Considerations

This section describes the programming considerations for the Raster Subsystem and includes the following topics:

- Cursor Chip Programming Considerations
- Bitplane Programming Considerations
- Raster Subsystem Initialization

Cursor Chip Programming Considerations

The MGR adapter contains two Brooktree Bt431 cursor chips which are used to display either a single multicolor cursor or two single color cursors. The following macros are defined in the file mgr.h. Macros are provided to read and write the address registers, the control registers and the glyph RAM bytes.

```
/* Cursor Chip defines */

#define   CURS_GLYPH_SIZE   512

#define   CURS_CMDMASK      (0x3 << 2)   /* Command Register address offset */
#define   CURS_AREG0        (0x0 << 2)   /* Address low register address offset */
#define   CURS_AREG1        (0x1 << 2)   /* Address high register address offset*/
#define   CURS_GLYPH        (0x2 << 2)   /* Glyph RAM address offset
#define   CURS_CR           (0x3 << 2)   /* Control Registers address offset */

/* Control Register offsets */

#define   CURS_CMD          0
#define   CURS_XLO          1
#define   CURS_XHI          2
#define   CURS_YLO          3
#define   CURS_YHI          4
#define   CURS_WINXLO       5
#define   CURS_WINXHI       6
#define   CURS_WINYLO       7
#define   CURS_WINYHI       8
#define   CURS_WINWLO       9
#define   CURS_WINWHI       10
#define   CURS_WINHLO       11
#define   CURS_WINHHI       12

/* SGI graphics library defines glyph origin as lower left bit.  Hotspot is at the center of the
glyph. */

#define   CURS_XHOTOFF      31           /* x offset from origin to hotspot */
#define   CURS_YHOTOFF      32           /* y offset from origin to hotspot */

/* Cursor offsets for the 4 monitor types */

#define   CURS_XOFF         189          /* 60 Hz monitor x hardware offset */
```

```
#define   CURS_YOFF          6         /* 60 Hz monitor y hardware offset */
#define   CURS_XOFF_30       39        /* 30 Hz monitor x hardware offset */
#define   CURS_YOFF_30       18        /* 30 Hz monitor y hardware offset */
#define   CURS_XOFF_170      -6        /* NTSC monitor x hardware offset */
#define   CURS_YOFF_170      -14       /* NTSC monitor y hardware offset */
#define   CURS_XOFF_PAL      24        /* PAL monitor x hardware offset */
#define   CURS_YOFF_PAL      -9        /* PAL monitor y hardware offset */

/* Cursor initial screen coordinates */

#define   CURS_XINIT         639       /* initial x coordinate for 1280 x 1024 */
#define   CURS_YINIT         511       /* initial y coordinate for 1280 x 1024 */

/* Command register bits */

#define   CURS_BLOCK         0x40      /* block cursor */
#define   CURS_CROSS         0x20      /* cross hair cursor */
#define   CURS_FMT           0x01      /* block and cross hair overlap format */
#define   CURS_5TO1MUX       0x04      /* use 5 to 1 multiplexing */
#define   CURS_1THICK        0x00      /* cross hair 1 pixel thick */
#define   CURS_3THICK        0x01      /* cross hair 3 pixel thick */
#define   CURS_5THICK        0x02      /* cross hair 5 pixel thick */
#define   CURS_7THICK        0x03      /* cross hair 7 pixel thick */

/* Cursor base address offsets */

#define   CURS0_OFF          0x560     /* cursor chip 0 offset */
#define   CURS1_OFF          0x580     /* cursor chip 1 offset */

/* Cursor address registers R/W macros */

#define   CURS_A0RD(curs_off, x) \
    x = *(volatile long *)(GRP | curs_off | CURS_AREG0) & 0xFF

#define   CURS_A1RD(curs_off, x) \
    x = *(volatile long *)(GRP | curs_off | CURS_AREG1) & 0xFF

#define   CURS_A0WR(curs_off, x) \
    *(volatile long *)(GRP | curs_off | CURS_AREG0) = (long) ((x) & 0xFF)

#define   CURS_A1WR(curs_off, x) \
    *(volatile long *)(GRP | curs_off | CURS_AREG1) = (long) ((x) & 0xFF)

/* Cursor glyph RAM R/W macros */

#define   CURS_GLYPHRD(curs_off, x) \
    x = *(volatile long *)(GRP | curs_off | CURS_GLYPH) & 0xFF

#define   CURS_GLYPHWR(curs_off, x) \
    *(volatile long *)(GRP | curs_off | CURS_GLYPH) = (long) ((x) & 0xFF)

/* Cursor control registers R/W macros */

#define   CURS_CRRD(curs_off, x) \
        x = *(volatile long *)(GRP | curs_off | CURS_CR) & 0xFF

    #define    CURS_CRWR(curs_off, x) \
        *(volatile long *)(GRP | curs_off | CURS_CR) = (long) ((x) & 0xFF)
```

Example Code :

```
    #include  "mgr.h"

    int        i;

    /* assume glyphp points to an array of 512 unsigned char containing glyph definition. */

    unsigned char *glyphp;

    /* assume curs_x and curs_y contain the current cursor x,y position */

    /* set command register for 5 to 1 Multiplexing and enable the block cursor */

    /* set cursor chip 0 */

    CURS_A0WR(CURS0_OFF, CURS_CMD);
    CURS_A1WR(CURS0_OFF, CURS_CMD);
    CURS_CRWR(CURS0_OFF, CURS_BLOCK | CURS_5TO1MUX);

    /* set cursor chip 1 */

    CURS_A0WR(CURS1_OFF, CURS_CMD);
    CURS_A1WR(CURS1_OFF, CURS_CMD);
    CURS_CRWR(CURS1_OFF, CURS_BLOCK | CURS_5TO1MUX);

    /* set the cursor glyph RAM in both chips the same */

    /* set cursor chip 0 */

    CURS_A0WR(CURS0_OFF, 0);  /* set address regs to 0 */
    CURS_A1WR(CURS0_OFF, 0);
    for (i = CURS_GLYPH_SIZE ; i ; --i) {
        CURS_GLYPHWR(CURS0_OFF, *glyphp++);

    /* set cursor chip 1 */

    CURS_A0WR(CURS1_OFF, 0);  /* set address regs to 0 */
    CURS_A1WR(CURS1_OFF, 0);
    for (i = CURS_GLYPH_SIZE ; i ; --i) {
        CURS_GLYPHWR(CURS1_OFF, *glyphp++);

    /* set the cursor x and y position in both chips the same */

    /* set cursor chip 0 */

    CURS_A0WR(CURS0_OFF, CURS_XLO);
    CURS_A1WR(CURS0_OFF, CURS_XLO);
    CURS_CRWR(CURS0_OFF, curs_x & 0xFF);          /* x low */
    CURS_CRWR(CURS0_OFF, (curs_x >> 8) & 0xF);     /* x high */
    CURS_CRWR(CURS0_OFF, curs_y & 0xFF);           /* y low */
    CURS_CRWR(CURS0_OFF, (curs_y >> 8) & 0xF);     /* y high */

    /* set cursor chip 1 */

    CURS_A0WR(CURS1_OFF, CURS_XLO);
    CURS_A1WR(CURS1_OFF, CURS_XLO);
    CURS_CRWR(CURS1_OFF, curs_x & 0xFF);           /* x low */
    CURS_CRWR(CURS1_OFF, (curs_x >> 8) & 0xF);     /* x high */
    CURS_CRWR(CURS1_OFF, curs_y & 0xFF);           /* y low */
    CURS_CRWR(CURS1_OFF, (curs_y >> 8) & 0xF);     /* y high */
```

Bitplane Programming Considerations

The Geometry Subsystem chapter contains a section which describes how to program the various bitplanes so refer to that chapter for details on the programming considerations for the various bitplanes.

Raster Subsystem Initialization

To initialize the Raster Subsystem the host software must initialize the TOPSCAN register and the ENABRGB register. It must also initialize the two cursor chips and clear the various bitplanes. The following example code shows the initialization steps.

Example Code :

```
    #include  "mgr.h"
    #include  "gecmds.h"
    #include  "imsetup.h"

    im_GEsetup;

    int       red, green, blue;
    int       color, zero = 0, one = 1, mone = -1;
    int       curs_x,  curs_y;

    /* initialize the TOPSCAN register and the ENABRGB register */

    /* microcode must be already downloaded and the three data parameters sent down FIFO */

    /* set TOPSCAN for 1 Meg VRAM and 60 Hz 1280 x 1024 monitor */

    FIFO_WR(GE_LOADRE, zero);
    tmp = RE_TOPSCAN;
    FIFO_WR(GE_DATA, tmp);
    tmp = 0x3FF;                 /* 1 Meg VRAM 60 Hz monitor value */
    FIFO_WR(GE_DATA, tmp);

    /* set the ENABRGB register to 1 */

    FIFO_WR(GE_LOADRE, zero);
    tmp = RE_ENABRGB;
    FIFO_WR(GE_DATA, tmp);
```

```
tmp = 1;
FIFO_WR(GE_DATA, tmp);

/* Now clear the bitplanes */

/* Clear the Frame Buffer bitplanes */

FIFO_WR(GE_DRAWMODE, zero);
FIFO_WR(GE_DATA, one);
FIFO_WR(GE_DATA, one);
FIFO_WR(GE_DATA, one);

/* set the aux write mask so we don't affect the PUP, UAUX and WID bitplanes */

FIFO_WR(GE_AUXWRITEMASK, zero);
FIFO_WR(GE_DATA, zero);

FIFO_WR(GE_PIXTYPE, zero);
tmp = 2;                        /* 12 bit color index pixtype */
FIFO_WR(GE_DATA, tmp);

FIFO_WR(GE_COLOR, zero);
FIFO_WR(GE_DATA, zero);

/* set the pixel write mask so all 24 bits are cleared */

FIFO_WR(GE_PIXWRITEMASK, zero);
tmp = 0xFFFFFF;
FIFO_WR(GE_DATA, tmp);

FIFO_WR(GE_SCREENCLEAR, zero);

/* set the pixel write mask to 0 so it prevents FB bitplane writes during other clears */

FIFO_WR(GE_PIXWRITEMASK, zero);
FIFO_WR(GE_DATA, zero);

/* clear PUP bitplanes */

FIFO_WR(GE_DRAWMODE, zero);
FIFO_WR(GE_DATA, mone);
FIFO_WR(GE_DATA, mone);
FIFO_WR(GE_DATA, one);

/* set the aux write mask to allow the PUP bitplanes to be written and the UAUX, WID and Z
bitplanes to not be written. */

FIFO_WR(GE_AUXWRITEMASK, zero);
tmp = 0x3;
FIFO_WR(GE_DATA, tmp);

FIFO_WR(GE_COLOR, zero);
FIFO_WR(GE_DATA, zero);
FIFO_WR(GE_SCREENCLEAR, zero);

/* clear UAUX bitplanes */

FIFO_WR(GE_DRAWMODE, zero);
FIFO_WR(GE_DATA, mone);
FIFO_WR(GE_DATA, mone);
FIFO_WR(GE_DATA, mone);

/* set the aux write mask to allow the UAUX bitplanes to be written and the PUP, WID and Z
bitplanes to not be written. */

FIFO_WR(GE_AUXWRITEMASK, zero);
tmp = 0x0C;
FIFO_WR(GE_DATA, tmp);

FIFO_WR(GE_COLOR, zero);
FIFO_WR(GE_DATA, zero);

FIFO_WR(GE_SCREENCLEAR, zero);

/* clear WID bitplanes */

FIFO_WR(GE_DRAWMODE, zero);
FIFO_WR(GE_DATA, mone);
FIFO_WR(GE_DATA, one);
FIFO_WR(GE_DATA, one);

/* set the aux write mask to allow the WID bitplanes to be written and the PUP, UAUX and Z
bitplanes to not be written. */

FIFO_WR(GE_AUXWRITEMASK, zero);
tmp = 0xF0;
FIFO_WR(GE_DATA, tmp);

FIFO_WR(GE_COLOR, zero);
FIFO_WR(GE_DATA, zero);

FIFO_WR(GE_SCREENCLEAR, zero);

/* leave the aux write mask set to 0 */

FIFO_WR(GE_AUXWRITEMASK, zero);
FIFO_WR(GE_DATA, zero);

/* Now initialize the cursor chips */

/* load the cursor 0 glyph with zeros */

CURS_A0WR(CURS0_OFF, 0);
CURS_A1WR(CURS0_OFF, 0);
for (i = CURS_GLYPH_SIZE ; i ; --i)
    CURS_GLYPHWR(CURS0_OFF, 0);
```

```
/* load the cursor 1 glyph with zeroes */

CURS_A0WR(CURS1_OFF, 0);
CURS_A1WR(CURS1_OFF, 0);
for (i = CURS_GLYPH_SIZE ; i ; --i)
    CURS_GLYPHWR(CURS1_OFF, 0);

curs_x = CURS_XINIT + CURS_XOFF;
curs_y = CURS_YINIT + CURS_YOFF;

/* set cursor chip 0 position */

CURS_A0WR(CURS0_OFF, CURS_XLO);
CURS_A1WR(CURS0_OFF, CURS_XLO);
CURS_CRWR(CURS0_OFF, curs_x & 0xFF);            /* x low */
CURS_CRWR(CURS0_OFF, (curs_x >> 8) & 0xF);      /* x high */
CURS_CRWR(CURS0_OFF, curs_y & 0xFF);            /* y low */
CURS_CRWR(CURS0_OFF, (curs_y >> 8) & 0xF);      /* y high */

/* set cross hair cursor window for full screen */

CURS_CRWR(CURS0_OFF, 0);                        /* window x low */
CURS_CRWR(CURS0_OFF, 0);                        /* window x high */
CURS_CRWR(CURS0_OFF, 0);                        /* window y low */
CURS_CRWR(CURS0_OFF, 0);                        /* window y high */

CURS_CRWR(CURS0_OFF, 0xFF);                     /* window width low */
CURS_CRWR(CURS0_OFF, 0xF);                      /* window width high */
CURS_CRWR(CURS0_OFF, 0xFF);                     /* window height low */
CURS_CRWR(CURS0_OFF, 0xF);                      /* window height high */

/* set cursor chip 1 position */

CURS_A0WR(CURS1_OFF, CURS_XLO);
CURS_A1WR(CURS1_OFF, CURS_XLO);
CURS_CRWR(CURS1_OFF, curs_x & 0xFF);            /* x low */
CURS_CRWR(CURS1_OFF, (curs_x >> 8) & 0xF);      /* x high */
CURS_CRWR(CURS1_OFF, curs_y & 0xFF);            /* y low */
CURS_CRWR(CURS1_OFF, (curs_y >> 8) & 0xF);      /* y high */

/* set cross hair cursor window for full screen */

CURS_CRWR(CURS1_OFF, 0);                        /* window x low */
CURS_CRWR(CURS1_OFF, 0);                        /* window x high */
CURS_CRWR(CURS1_OFF, 0);                        /* window y low */
CURS_CRWR(CURS1_OFF, 0);                        /* window y high */

CURS_CRWR(CURS1_OFF, 0xFF);                     /* window width low */
CURS_CRWR(CURS1_OFF, 0xF);                      /* window width high */
CURS_CRWR(CURS1_OFF, 0xFF);                     /* window height low */
CURS_CRWR(CURS1_OFF, 0xF);                      /* window height high */

/* set the cursor 0 command register for block cursor and 5 to 1 multiplexing */
```

```
CURS_A0WR(CURS0_OFF, CURS_CMD);
CURS_A1WR(CURS0_OFF, CURS_CMD);
CURS_CRWR(CURS0_OFF, CURS_BLOCK | CURS_5TO1MUX);

/* set the cursor 1 command register for block cursor and 5 to 1 multiplexing */

CURS_A0WR(CURS1_OFF, CURS_CMD);
CURS_A1WR(CURS1_OFF, CURS_CMD);
CURS_CRWR(CURS1_OFF, CURS_BLOCK | CURS_5TO1MUX);
```