FBC Reference Manual

Curtis Priem

SUN Microsystems Inc.

2550 Garcia Avenue

Mountain View, CA 94030

Version 3.0

January 21, 1988

CONTENTS

1 - INTRODUCTION

The FBC (Frame Buffer Controller) chip is the heart of the CG6 architecture. It contains the hardware to do 2D graphic rendering.

The rendering hardware can execute the following commands:

Command	Source	Destination
DRAW	PATTERN Register	Frame Buffer
FONT	CPU	Frame Buffer
BLIT	Frame Buffer	Frame Buffer

2 - REGISTER SUMMARY

There are a total of 76 user registers in the FBC which the CPU can access. The user registers are used to input the actual graphics commands and data. Only 33 of these registers need to be restored after a context switch.

There is one system register which the CPU can access using the HWC chip select. The system register is for hardware configuration and is accessed by the boot proms. They will reside in a different page from the user registers so that they can be protected by the MMU from user accesses. These registers do not need to be saved or restored during context switching.

All accesses to the registers will be 32 bit wide ONLY and are aligned to the 32 bit word boundary. This is the only type of access the 68020, 80386, and RISC processors have in common.

0 = zeros when read
1 = ones when read
s = sign extended when read
fbc = pointer to the FBC data structure
fhc = pointer to the FBC Hardware Configuration data structure (protected)
tec = pointer to the TEC data structure
thc = pointer to the TEC Hardware Configuration data structure (protected)

Address	Name	Description
	31 0	INDEXED ADDRESS REGISTERS
fbc+0x880	ITRIABSX	X for Integer Absolute Tri
fbc+0x884	ITRIABSY	Y for Integer Absolute Tri
fbc+0x888	ITRIABSZ	Z for Integer Absolute Tri
fbc+0x890	ITRIRELX	X for Integer Relative Tri
fbc+0x894	ITRIRELY	Y for Integer Relative Tri
fbc+0x898	ITRIRELZ	Z for Integer Relative Tri
fbc+0x8c0	IQUADABSX	X for Integer Absolute Quad
fbc+0x8c4	IQUADABSY	Y for Integer Absolute Quad
fbc+0x8c8	IQUADABSZ	Z for Integer Absolute Quad
fbc+0x8d0	IQUADRELX	X for Integer Relative Quad
fbc+0x8d4	IQUADRELY	Y for Integer Relative Quad
fbc+0x8d8	IQUADRELZ	Z for Integer Relative Quad
fbc+0x900	IRECTABSX	X for Integer Abs Rectangle
fbc+0x904	IRECTABSY	Y for Integer Abs Rectangle
fbc+0x908	IRECTABSZ	Z for Integer Abs Rectangle
fbc+0x910	IRECTRELX	X for Integer Rel Rectangle
fbc+0x914	IRECTRELY	Y for Integer Rel Rectangle
fbc+0x918	IRECTRELZ	Z for Integer Rel Rectangle
	31 0	RASTER OFFSET REGISTERS
fbc+0x0c0	RASTEROFFX	X offset of raster
fbc+0x0c4	RASTEROFFY	Y offset of raster
	31 0	AUTOINCREMENT REGISTERS
fbc+0x0d0	AUTOINCX	X auto-increment
fbc+0x0d4	AUTOINCY	Y auto-increment

Address	Name	Description
	31 0	WINDOW REGISTERS
fbc+0x0e0	CLIPMINX	Upper Left X of Clip Window
fbc+0x0e4	CLIPMINY	Upper Left Y of Clip Window
fbc+0x0f0	CLIPMAXX	lower Right X of Clip Window
fbc+0x0f4	CLIPMAXY	Lower Right Y of Clip Window
	31 0	CLIP CHECK REGISTER
fbc+0x008	CLIPCHECK	TEC Clip Check Register
	31 0	ATTRIBUTE REGISTERS
fbc+0x100	FCOLOR	Foreground Color Register
fbc+0x104	BCOLOR	Background Color Register
fbc+0x108	RASTEROP	RasterOp Register
fbc+0x10c	PLANEMASK	Plane Mask Register
fbc+0x110	PIXELMASK	Pixel Mask Register
fbc+0x120	PATTERN0	Pattern Register 0
fbc+0x124	PATTERN1	Pattern Register 1
fbc+0x128	PATTERN2	Pattern Register 2
fbc+0x12c	PATTERN3	Pattern Register 3
fbc+0x130	PATTERN4	Pattern Register 4
fbc+0x134	PATTERN5	Pattern Register 5
fbc+0x138	PATTERN6	Pattern Register 6
fbc+0x13c	PATTERN7	Pattern Register 7
fbc+0x11c	PATTALIGN	Pattern Alignment Register

Address	Name	Description
	31 0	STATUS & COMMAND REGISTERS
fbc+0x010	STATUS	Status Register
fbc+0x014	DRAWSTATUS	DRAW Status Register
fbc+0x018	BLITSTATUS	BLIT Status Register
fbc+0x01c	FONT	Font Register
	31 0	MISCELLANEOUS REGISTER
fbc+0x004	MISC	Miscellaneous Register
	31 0	CONFIGURATION REGISTER
fhc+0x000 fbc+0x000	CONFIG	Configuration Register

3.1 - ADDRESS REGISTERS

The address registers have different meanings when doing DRAW, BLIT, or FONT commands. See each of the commands for details. When doing a DRAW or FONT, the address registers can represent either subpixels or pixels depending on if the FBC is in HRMONO mode (high resolution monochrome mode) or COLOR1 (1 plane COLOR mode, the FBC expands the source from 1 plane to 8 planes) respectively. When doing a DRAW, BLIT, or FONT the address registers represent pixel locations if the FBC is in COLOR8 mode (8 plane color mode). The Z registers do not store a Z value, but are used when the Z clip information is autoloaded from the TEC.

All the address registers can be read and written for context switching.

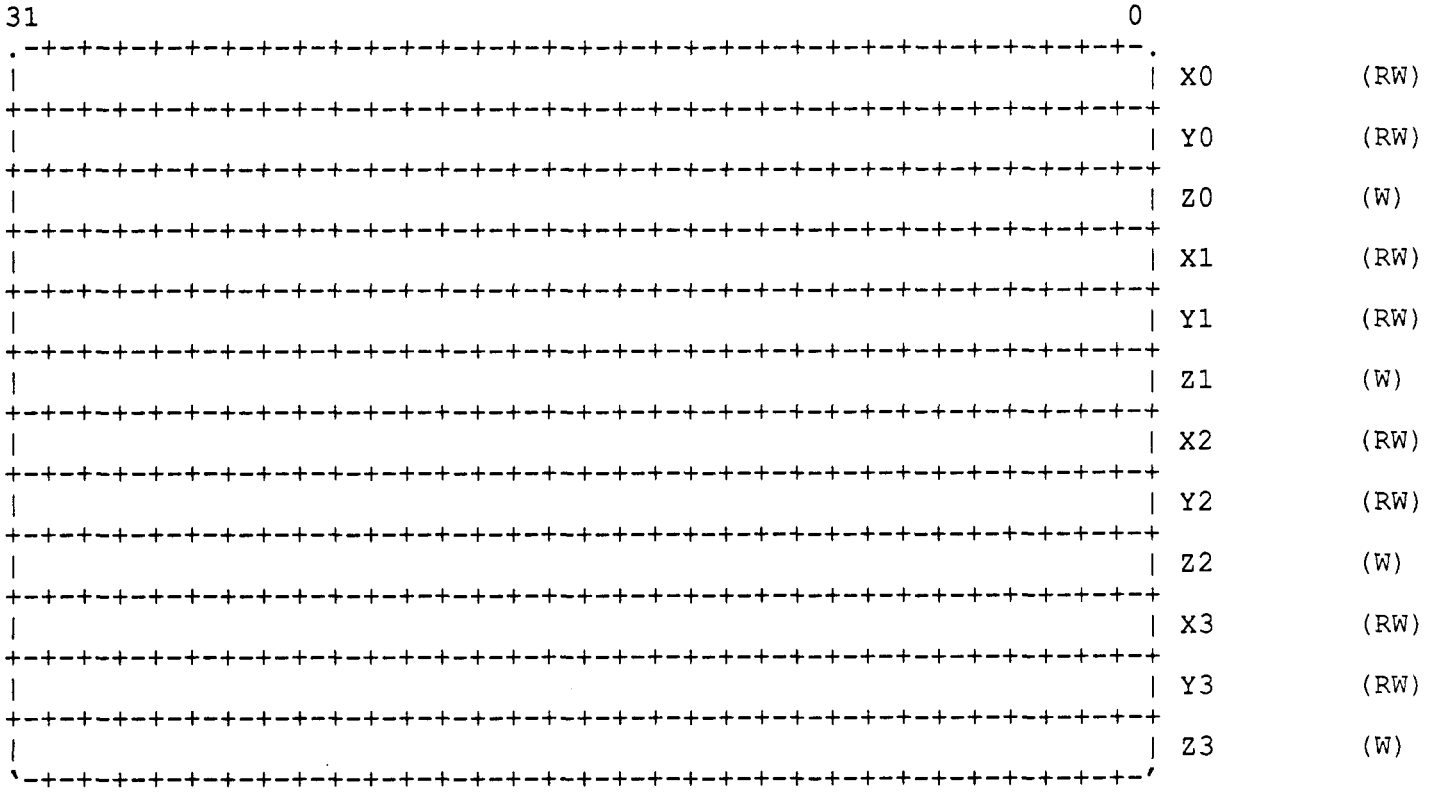


Figure 3-2 Address Registers (Signed Integer)

3.2 INDEXED ADDRESS REGISTERS

There are also a set of indexed address registers which reduces the number of accesses that need to be done to the FBC to load the absolute or relative coordinates for points, lines, triangles, quadrilaterals, and aligned rectangles, especially when these objects are chained. There is a 2 bit address INDEX which points to the next set of address registers that is to be modified. The INDEX is incremented after every X address is written for each of the objects and after Y is written for only the aligned rectangles. This allows the software to write only one x-y vertex to describe a chained point, line, triangle, or quadrilateral, and two vertices to describe a rectangle. The address INDEX can be read or written through the MISC register. The Y coordinate should always be loaded before the X coordinate.

The coordinate type of the indexed address registers can only be integer.

For absolute coordinates, the contents of the RASTEROFFX or RASTEROFFY register is added to the value loaded into the FBC, and the result is placed into the registers pointed to by the address INDEX.

For relative coordinates, the contents of the X[INDEX-1] or Y[INDEX-1] register is added to the value loaded into the FBC, and the result is placed into the registers pointed to by the address INDEX.

Table 3-1 shows which address registers are affected by each of the index registers for every possible INDEX. The table also indicates which commands increment the INDEX.

Index Register	INDEX=00	INDEX=01	INDEX=10	INDEX=11	INDEX++
IPOINTABSX,IPOINTRELX	X0,X1,X2,X3	X1,X2,X3,X0	X2,X3,X0,X1	X3,X0,X1,X2	yes
IPOINTABSY,IPOINTRELY	Y0,Y1,Y2,Y3	Y1,Y2,Y3,Y0	Y2,Y3,Y0,Y1	Y3,Y0,Y1,Y2	no
IPOINTABSZ,IPOINTRELZ	Y0,Y1,Y2,Y3	Y1,Y2,Y3,Y0	Y2,Y3,Y0,Y1	Y3,Y0,Y1,Y2	no
ILINEABSX,ILINERELX	X0,X1,X2	X1,X2,X3	X2,X3,X0	X3,X0,X1	yes
ILINEABSY,ILINERELY	Y0,Y1,Y2	Y1,Y2,Y3	Y2,Y3,Y0	Y3,Y0,Y1	no
ILINEABSZ,ILINERELZ	Y0,Y1,Y2	Y1,Y2,Y3	Y2,Y3,Y0	Y3,Y0,Y1	no
ITRIABSX,ITRIRELX	X0,X1	X1,X2	X2,X3	X3,X0	yes
ITRIABSY,ITRIRELY	Y0,Y1	Y1,Y2	Y2,Y3	Y3,Y0	no
ITRIABSZ,ITRIRELZ	Y0,Y1	Y1,Y2	Y2,Y3	Y3,Y0	no
IQUADABSX,IQUADRELX	X0	X1	X2	X3	yes
IQUADABSY,IQUADRELY	Y0	Y1	Y2	Y3	no
IQUADABSZ,IQUADRELZ	Y0	Y1	Y2	Y3	no
IRECTABSX,IRECTRELY	X0,X1	X1,X2	X2,X3	X3,X0	yes
IRECTABSY,IRECTRELY	Y0,Y1	Y1,Y2	Y2,Y3	Y3,Y0	yes
IRECTABSZ,IRECTRELZ	Y0,Y1	Y1,Y2	Y2,Y3	Y3,Y0	no

Table 3-1 Address Registers affected by Index Registers

5.1 DRAW STATUS REGISTER

After the vertices have been entered, the DRAWSTATUS register is checked to see if the object can be drawn by the FBC (DRAW_HARDWARE), or the object must be drawn by the software driver (DRAW_SOFTWARE).

If an object is classified as DRAW_HARDWARE, the FBC will be able to draw the object without any assistance from software. The definition of a DRAW_HARDWARE object is:

A) The object is DRAW_HIDDEN (or)
B) The object surrounds the clip window (or)
C) 1) The object is not DRAW_HIDDEN (and)
   2) The object has only two active edges (and)
   3) All of the vertices are between -2**14 and 2**14-1 (and)
   4) a) All of the vertices are inside the test window (or)
      b) The object is a line and one of the end points is inside the test window (or)
      c) All of the x vertices are inside the test window and the top or bottom vertex is inside the test window (or)
      d) The object is an aligned rectangle (this includes vertical and horizontal lines)

All other objects must be drawn by software. The FBC will only be able to help by doing horizontal lines, clipping, ROPs, etc. The definition of a DRAW_SOFTWARE object is:

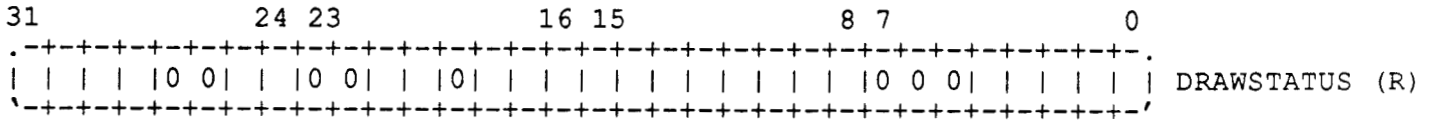
A) The object is not DRAW_HARDWARE

The DRAWSTATUS register is updated whenever a value is written to the address or clip registers. The DRAW_EXCEPTION bit in the register is set if one of the following bits are set:

A) DRAW_SOFTWARE (or)
B) FBC_FULL (or)
C) TEC_EXCEPTION (or)
D) UNSUPPORTED_ATTR (or)
E) ACC_OVERFLOW (or)
F) TEC_INTERSECT

The object is drawn by the FBC if the CPU reads the DRAWSTATUS register and the DRAW_EXCEPTION bit is zero. If the DRAW_EXCEPTION bit is set, the FBC will not render the object.

The DRAWSTATUS register is read only since by reloading the address and clip registers of the FBC during a context switch, the correct status will be regenerated.



Bits	Description	
31	DRAW_EXCEPTION	(1=There is a DRAW exception)
30	TEC_EXCEPTION	(1=TEC has an exception)
29	FULL	(1=FBC internal address registers are full)
28	BUSY	(1=FBC is Busy)
25	UNSUPPORTED_ATTR	(1=Unsupported Attribute)
24	HRMONO	(1=High Resolution Monochrome)
21	ACC_OVERFLOW	(1=Overflow in FBC has occurred)
20	ACC_PICK	(1=Part of the Object Fell Inside PICK Window)
18	TEC_HIDDEN	(1=Object is Hidden)
17	TEC_INTERSECT	(1=Object is Intersecting the TEC CLIP check)
16	TEC_VISIBLE	(1=Object is Totally Inside the TEC CLIP check)
15	BLIT_HARDWARE	(1=FBC can blit the rectangle)
14	BLIT_SOFTWARE	(1=Software must blit the rectangle)
13	BLIT_SRC_HID	(1=Src Rect is Hidden)
12	BLIT_SRC_INT	(1=Src Rect is Intersecting the CLIP Window)
11	BLIT_SRC_VIS	(1=Src Rect is Totally Inside the CLIP Window)
10	BLIT_DST_HID	(1=Dst Rect is Hidden)
9	BLIT_DST_INT	(1=Dst Rect is Intersecting the CLIP Window)
8	BLIT_DST_VIS	(1=Dst Rect is Totally Inside the CLIP Window)
4	DRAW_HARDWARE	(1=FBC can draw the Object)
3	DRAW_SOFTWARE	(1=Software must draw the Object)
2	DRAW_HIDDEN	(1=Object is Hidden)
1	DRAW_INTERSECT	(1=Object is Intersecting the CLIP Window)
0	DRAW_VISIBLE	(1=Object is Totally Inside the CLIP Window)

Figure 5-1 DRAW Status Register

5.2 DRAWING OBJECTS

The polygons can be drawn with bounding lines to form overlapping polygons or without bounding lines to form nonoverlapping polygons. Overlapping polygons are defined as the union of the (balanced) lines that connect the vertices of the polygon and all pixels inside the bounding lines. The points in common between two adjacent overlapping polygons will form a balanced line.

Nonoverlapping polygons are defined as all of the points on or within the ideal lines that connect the vertices except for the right most pixels and bottom line of each (trapezoid) section of the polygon. There are no points in common between two adjacent nonoverlapping polygons. Nonoverlapping lines and points are not drawn since they have no thickness. The attribute that selects between overlapping and nonoverlapping polygons is in the RASTEROP register.

When the edge of a polygon or a line falls half way between the center of two pixels, the FBC rounds to the right for horizontal midpoints or rounds down for vertical midpoints, with respect to the screen (not the x-y addressing).

A quadrilateral can be drawn by entering four unique vertices into the IQUADABS or IQUADREL registers as shown in Figure 5-2 and 5-3. A triangle can be drawn by making two of the vertices the same or by entering only three vertices into the ITRIABS or ITRIREL registers as shown in Figure 5-4 and 5-5. A line can be drawn by making three of the vertices the same, by making two pair of vertices the same or, by entering only two vertices into the ILINEABS or ILINEREL registers as shown in Figure 5-6. A point can be drawn by making all four of the vertices the same or by entering only one vertex into the IPOINTABS or IPOINTREL registers as shown in Figure 5-7.

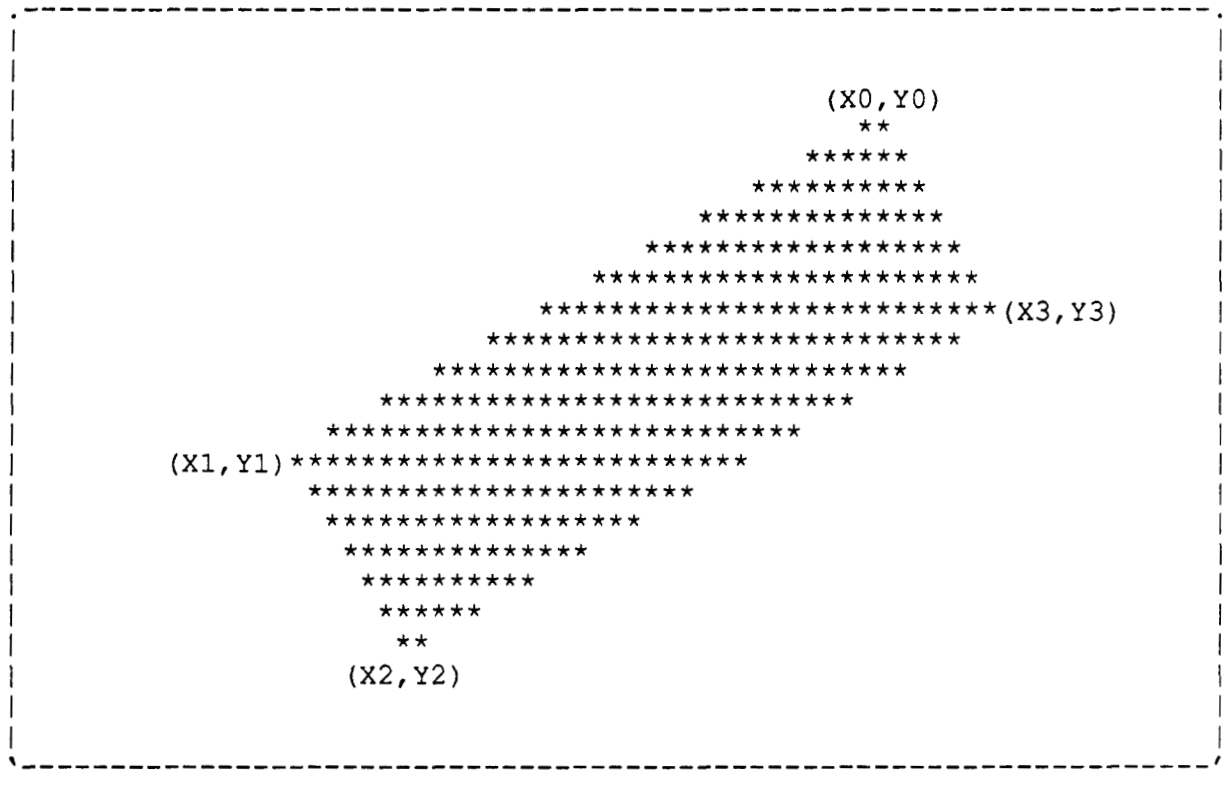


Figure 5-2 Drawing an Overlapping Quadrilateral

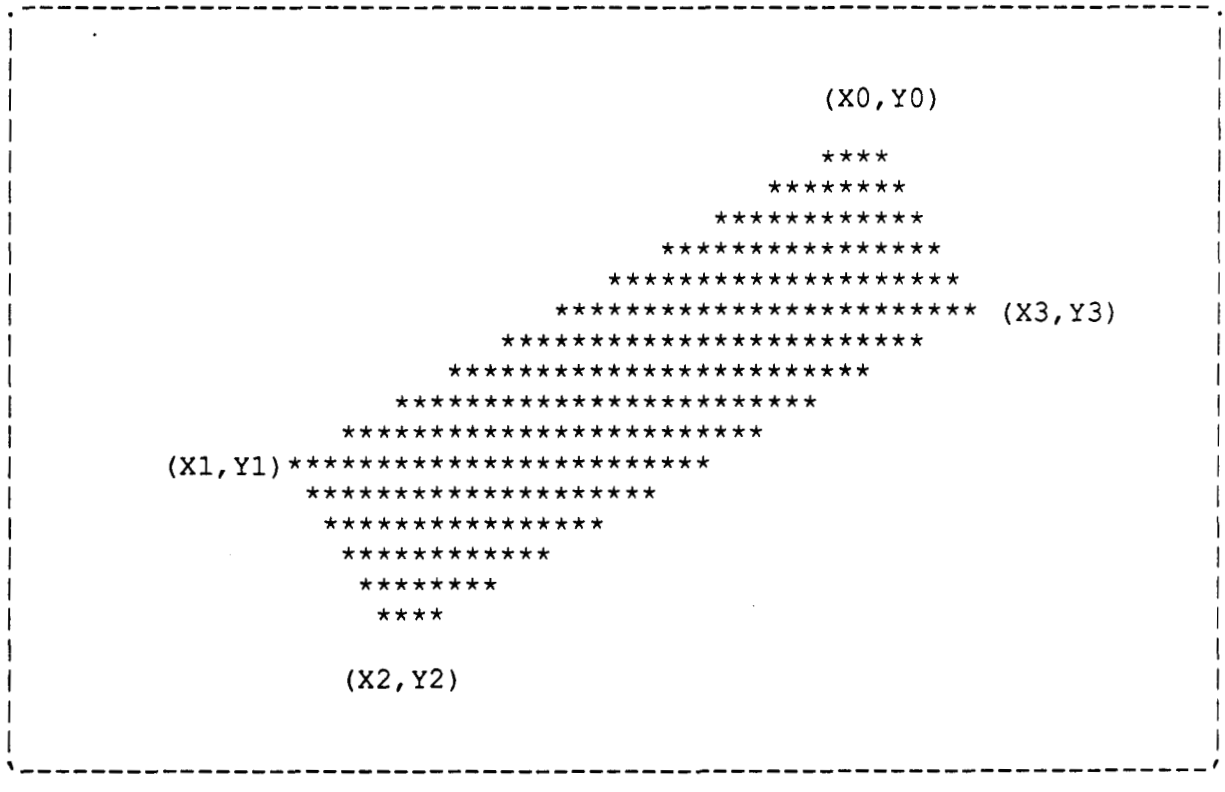


Figure 5-3 Drawing a Nonoverlapping Quadrilateral

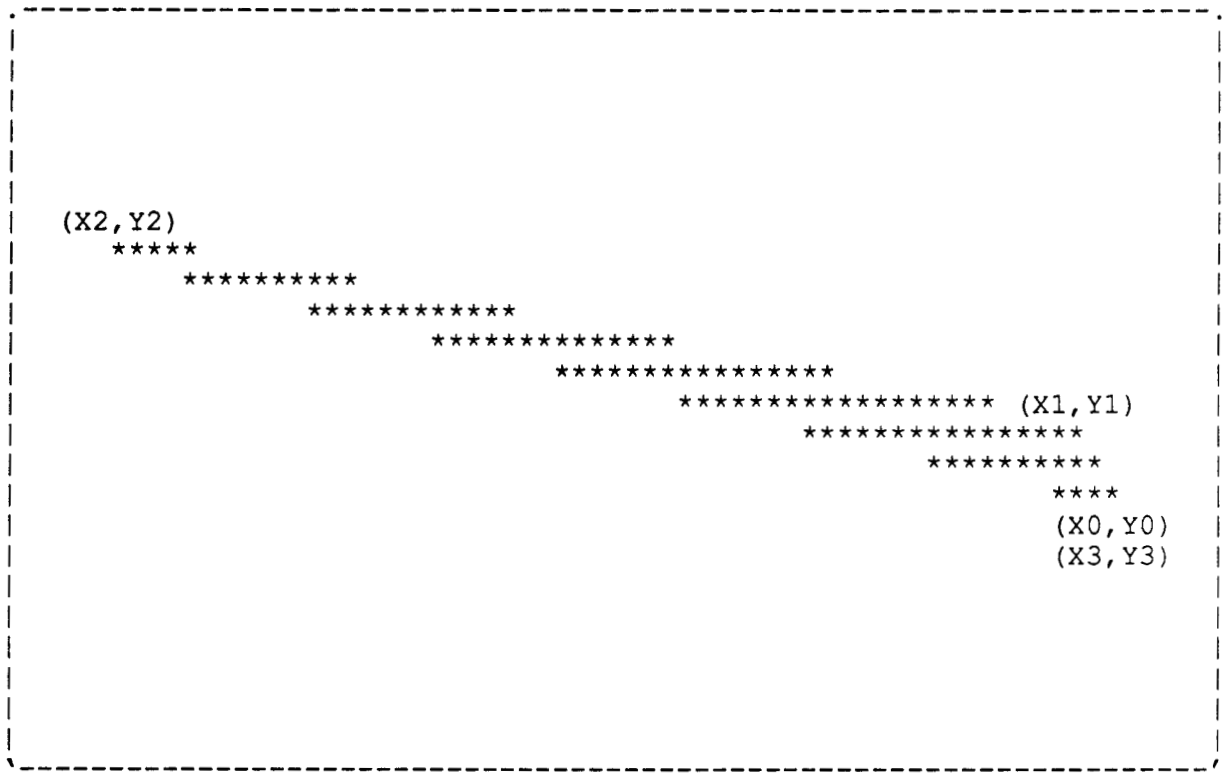


Figure 5-4 Drawing an Overlapping Triangle

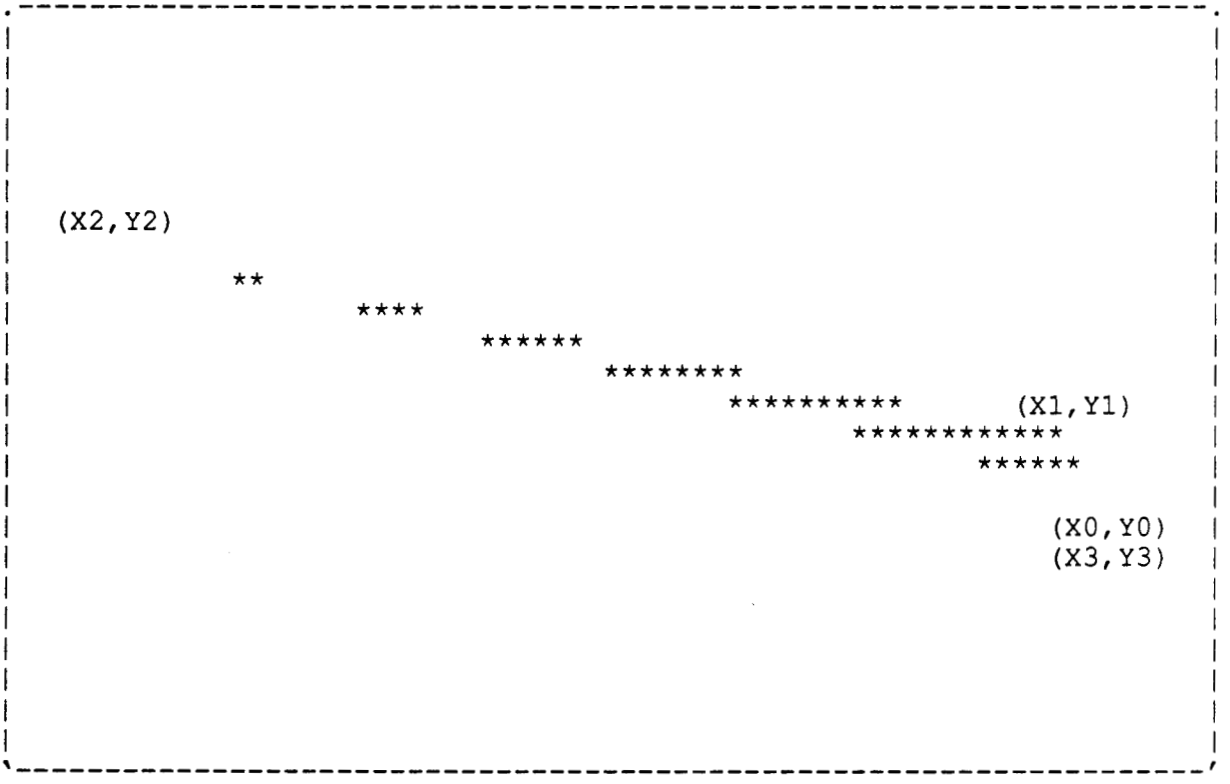


Figure 5-5 Drawing a Nonoverlapping Triangle

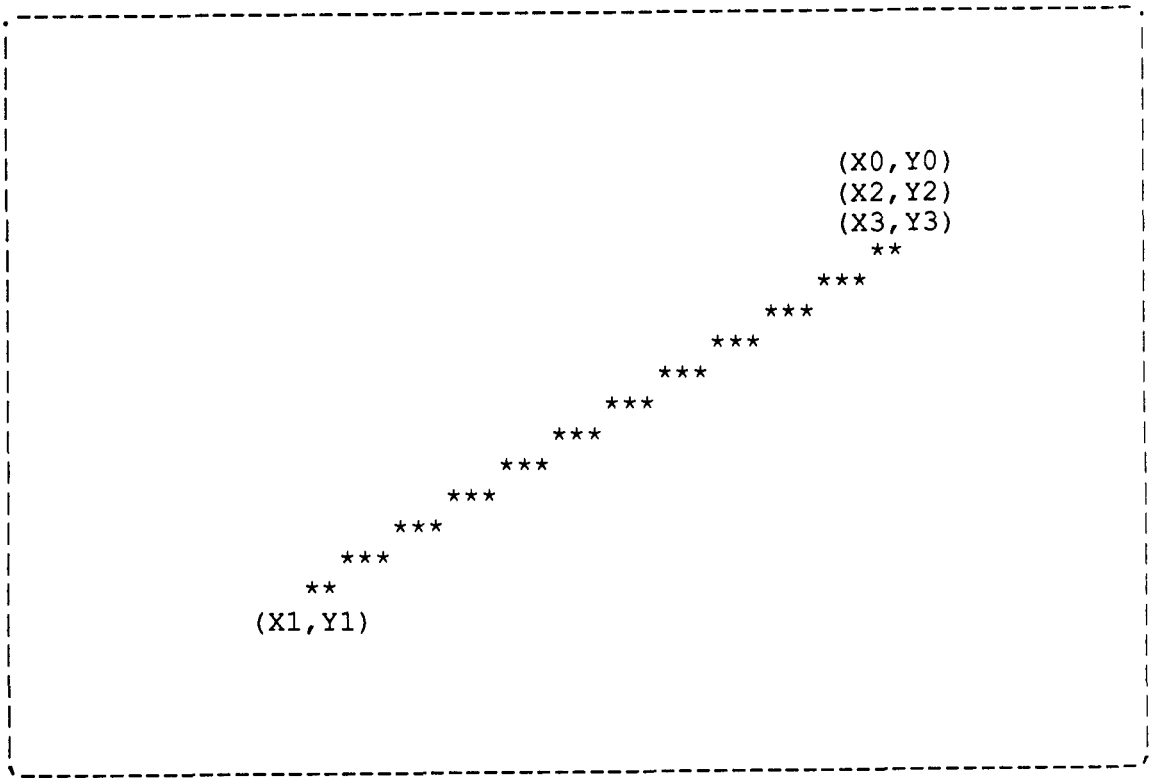


Figure 5-6 Drawing an Overlapping Line

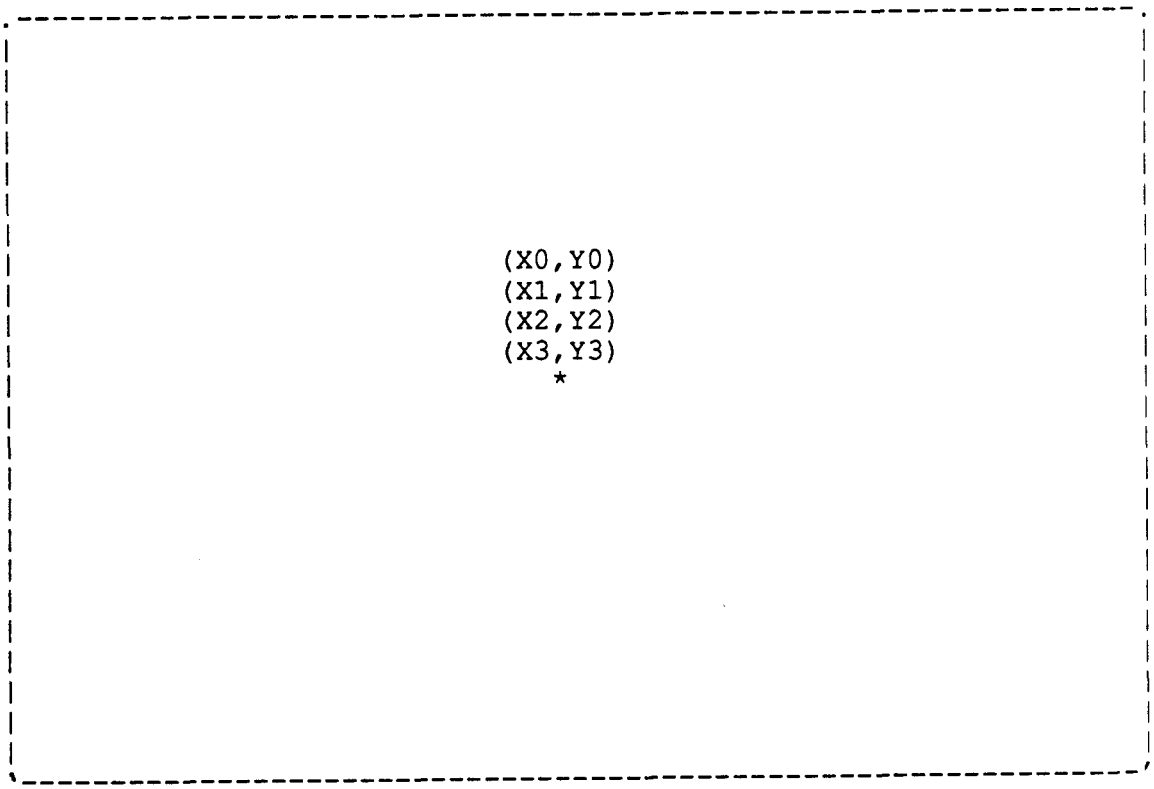


Figure 5-7 Drawing an Overlapping Point

6 - BLIT COMMAND

The FBC has the capability of doing a BLock Image Transfer (BLIT). The software can specify a rectangular area in the frame buffer as the source and another rectangular area in the frame buffer as the destination to which the source will be rasteroped. The source and destination can overlap in any direction and can be non-aligned.

The BLIT command can only be used in pixel coordinates (COLOR1 or COLOR8 mode). The meaning of the address registers for the BLIT command are shown in Table 6-1.

Address Register	Description
X0	X coordinate for upper left corner of source
Y0	Y coordinate for upper left corner of source
X1	X coordinate for lower right corner of source
Y1	Y coordinate for lower right corner of source
X2	X coordinate for upper left corner of destination
Y2	Y coordinate for upper left corner of destination
X3	X coordinate for lower right corner of destination
Y3	Y coordinate for lower right corner of destination

Table 6-1 Address Registers for BLIT Command

The addresses must be in the bounds of the screen and meet the constraints in Table 6-2.

1024x768 Color Mode	1024x1024 Color Mode	1152x900 Color Mode
X2 - X0 = X3 - X1	X2 - X0 = X3 - X1	X2 - X0 = X3 - X1
Y2 - Y0 = Y3 - Y1	Y2 - Y0 = Y3 - Y1	Y2 - Y0 = Y3 - Y1
0 <= X0 <= X1 <= 1023	0 <= X0 <= X1 <= 1023	0 <= X0 <= X1 <= 1151
0 <= X2 <= X3 <= 1023	0 <= X2 <= X3 <= 1023	0 <= X2 <= X3 <= 1151
0 <= Y0 <= Y1 <= 767	0 <= Y0 <= Y1 <= 1023	0 <= Y0 <= Y1 <= 899
0 <= Y2 <= Y3 <= 767	0 <= Y2 <= Y3 <= 1023	0 <= Y2 <= Y3 <= 899

Table 6-2 Legal Address Register Ranges

6.1 BLIT STATUS REGISTER

After the vertices have been entered, the BLITSTATUS register is checked to see if the rectangle can be blitted by the FBC (BLIT_HARDWARE), or it must be done by the software driver (BLIT_SOFTWARE).

If an object is classified as BLIT_HARDWARE, the FBC will be able to blit the rectangle without any assistance from software. The definition of a BLIT_HARDWARE blit is:

```
A) The destination rectangle is BLIT_DST_HID                          (or)
B) 1) The source rectangle is BLIT_SRC_HID                            (and)
   2) Blit source is included (BLIT_SRC_CHK=10)
C) 1) The source rectangle is BLIT_SRC_VIS                            (and)
   2) Blit source is included (BLIT_SRC_CHK=10)                       (and)
   3) The destination rectangle is not BLIT_DST_HID                   (and)
   4) The source rectangle is between -2**14 and 2**14-1              (and)
   5) The destination rectangle is between -2**14 and 2**14-1
D) 1) Blit source is excluded (BLIT_SRC_CHK=01)                       (and)
   2) The destination rectangle is not BLIT_DST_HID                   (and)
   3) The source rectangle is between -2**14 and 2**14-1              (and)
   4) The destination rectangle is between -2**14 and 2**14-1
```

All other blits must be done by software. The FBC will only be able to help by doing shifting, clipping, ROPs, etc. The definition of a BLIT_SOFTWARE object is:

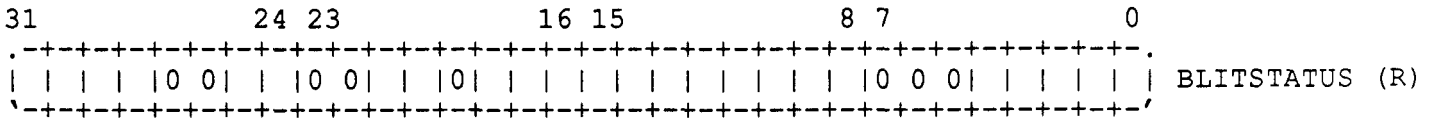
```
A) The object is not BLIT_HARDWARE
```

The BLITSTATUS register is updated whenever a value is written to the address or clip registers. The BLIT_EXCEPTION bit in the register is set if one of the following bits are set:

```
A) BLIT_SOFTWARE          (or)
B) FBC_FULL               (or)
C) UNSUPPORTED_ATTR       (or)
D) ACC_OVERFLOW           (or)
E) HRMONO
```

The rectangle is blitted by the FBC if the CPU reads the BLITSTATUS register and the BLIT_EXCEPTION bit is zero. If the BLIT_EXCEPTION bit is set, the FBC will not blit.

The BLITSTATUS register is read only since by reloading the address and clip registers of the FBC during a context switch, the correct status will be regenerated.



Bits	Description	
31	BLIT_EXCEPTION	(1=There is a BLIT exception)
30	TEC_EXCEPTION	(1=TEC has an exception)
29	FULL	(1=FBC internal address registers are full)
28	BUSY	(1=FBC is Busy)
25	UNSUPPORTED_ATTR	(1=Unsupported Attribute)
24	HRMONO	(1=High Resolution Monochrome)
21	ACC_OVERFLOW	(1=Overflow in FBC has occurred)
20	ACC_PICK	(1=Part of the Object Fell Inside PICK Window)
18	TEC_HIDDEN	(1=Object is Hidden)
17	TEC_INTERSECT	(1=Object is Intersecting the TEC CLIP check)
16	TEC_VISIBLE	(1=Object is Totally Inside the TEC CLIP check)
15	BLIT_HARDWARE	(1=FBC can blit the rectangle)
14	BLIT_SOFTWARE	(1=Software must blit the rectangle)
13	BLIT_SRC_HID	(1=Src Rect is Hidden)
12	BLIT_SRC_INT	(1=Src Rect is Intersecting the CLIP Window)
11	BLIT_SRC_VIS	(1=Src Rect is Totally Inside the CLIP Window)
10	BLIT_DST_HID	(1=Dst Rect is Hidden)
9	BLIT_DST_INT	(1=Dst Rect is Intersecting the CLIP Window)
8	BLIT_DST_VIS	(1=Dst Rect is Totally Inside the CLIP Window)
4	DRAW_HARDWARE	(1=FBC can draw the Object)
3	DRAW_SOFTWARE	(1=Software must draw the Object)
2	DRAW_HIDDEN	(1=Object is Hidden)
1	DRAW_INTERSECT	(1=Object is Intersecting the CLIP Window)
0	DRAW_VISIBLE	(1=Object is Totally Inside the CLIP Window)

Figure 6-1 BLIT Status Register

6.2 BLITTING RECTANGLES

Figure 6-2 gives an example of blitting the source to the destination rectangle.

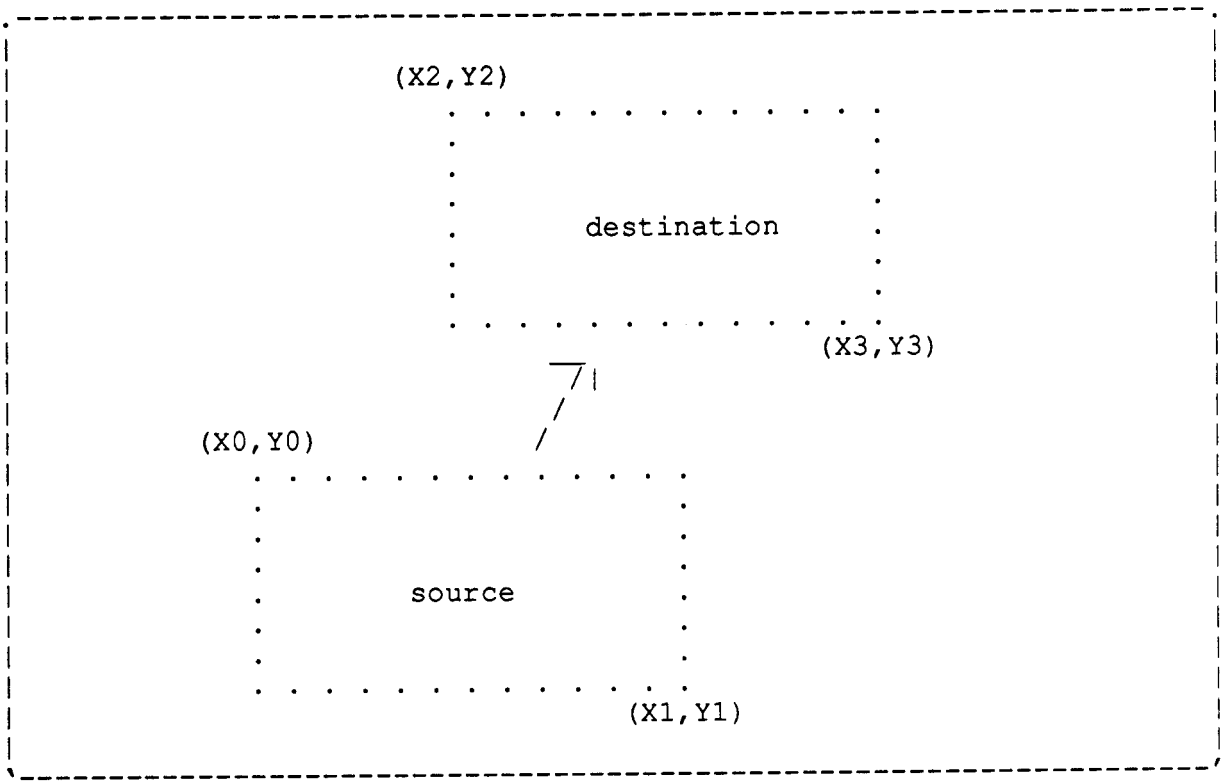


Figure 6-2 Example of a BLIT

7 - FONT COMMAND

The software can read and write to the frame buffer by using the FONT register. There are two ways to access the frame buffer; by plane and by pixel. If a character font is described in a single plane data structure, it can be written to the frame buffer through the FONT register in COLOR1 or HRMONO mode. The FBC will expand it to the eight planes of data (with the correct RASTEROP and COLOR) in COLOR1 mode and expand it to the proper grayscale value in HRMONO mode. The FONT register is write only in these modes since the FBC can not do a reverse conversion (8 planes to one). If a character font or image is described in an eight plane format, it can be written to the the frame buffer (with RASTEROP and COLOR) through the FONT register in COLOR8 mode. The Table 7-1 summarizes the different modes for the FONT register.

Mode	CPU Format	FB Format	FONT R/W
HRMONO	1 plane	8 planes	W
COLOR1	1 plane	8 planes	W
COLOR8	8 planes	8 planes	W

Table 7-1 FONT Register Modes

Before a character or image is written to the frame buffer, address range and window comparisons must be done. This can be done by setting up the FBC to draw a rectangle around the area that is to be modified. The upper left and lower right points should be entered using the IRECTABSX and IRECTABSY registers. The STATUS register is then checked for any exceptions and to see if the object does not DRAW_INTERSECT the clip window or is DRAW_HIDDEN. If there are no special cases, the software can start downloading the image or font to the frame buffer.

The address registers must then be loaded with the font mask or font width mask (the horizontal line of pixels or subpixels that are to be RASTEROPed). This is done by loading (X0,Y0) with the left point and (X1,Y0) with the right most point of where the data will be placed in the frame buffer. The number of pixels or subpixels (inclusive) between X0 and X1 indicates the width of the data that will be loaded into the FONT register. The difference between X0 and X1 should not be greater than 31 for HRMONO and COLOR1 and should not be greater than 3 for COLOR8. No pixels or subpixels will be modified outside this range. This mask is combined with the clip mask before rendering.

Normally the AUTOINCX and AUTOINCY registers are used in conjunction with the font register.

The meaning of the address registers for a FONT command are given in Table 7-2.

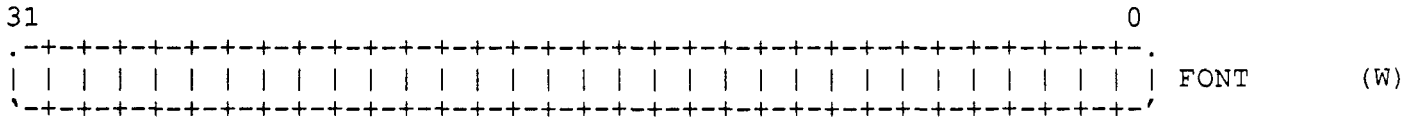
X0	X coordinate for left pixel/subpixel of font
Y0	Y coordinate for left pixel/subpixel of font
X1	X coordinate for right pixel/subpixel of font
Y1	(used only during clip window checking)
X2	(used only during clip window checking)
Y2	(used only during clip window checking)
X3	(used only during clip window checking)
Y3	(used only during clip window checking)

Table 7-2 Address Registers for FONT Command

7.1 FONT REGISTER

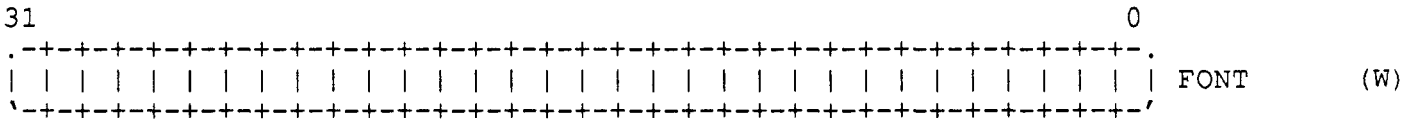
The FONT register can be used in either HRMONO, COLOR1, or COLOR8 mode. The addressing is pixel coordinates for color mode and subpixel coordinates for anti-aliasing mode. Reads from the FONT register while in COLOR8 mode are normally done only for saving the contents of the screen, ie. when a window needs to be saved or moved.

The FONT register does not have to be saved during context switching because the frame buffer data may change before that context is used again.



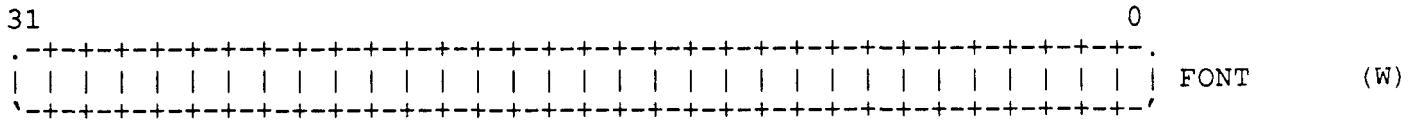
Bits	Description
31	Source Bit for Pixel (X0, Y0)
30	Source Bit for Pixel (X0+1, Y0)
.	. . .
1	Source Bit for Pixel (X0+30,Y0)
0	Source Bit for Pixel (X0+31,Y0)

Figure 7-3 FONT Register Format (COLOR1 mode & 68020/SPARC mode)



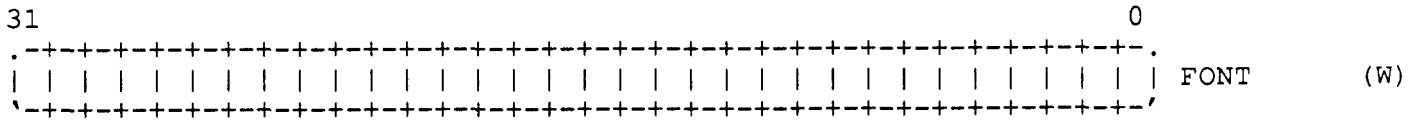
Bits	Description
31	Source Bit for Pixel (X0+31,Y0)
30	Source Bit for Pixel (X0+30,Y0)
.	. . .
1	Source Bit for Pixel (X0+1, Y0)
0	Source Bit for Pixel (X0, Y0)

Figure 7-4 FONT Register Format (COLOR1 mode & 80386 mode)



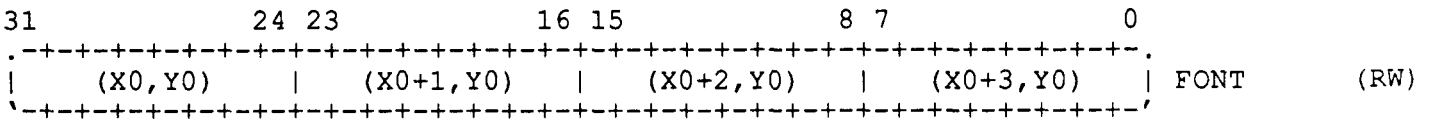
Bits	Description
31	Source Bit for Subpixel (X0, Y0)
30	Source Bit for Subpixel (X0+1, Y0)
.	. . .
1	Source Bit for Subpixel (X0+30,Y0)
0	Source Bit for Subpixel (X0+31,Y0)

Figure 7-5 FONT Register Format (HRMONO mode & 68020/SPARC mode)



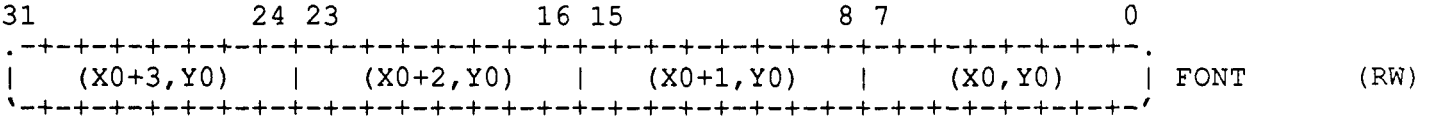
Bits	Description
31	Source Bit for Subpixel (X0+31,Y0)
30	Source Bit for Subpixel (X0+30,Y0)
.	. . .
1	Source Bit for Subpixel (X0+1, Y0)
0	Source Bit for Subpixel (X0, Y0)

Figure 7-6 FONT Register Format (HRMONO mode & 80386 mode)



Bits	Description	Bits	Description
31	Pixel (X0,Y0), Plane 7	15	Pixel (X0+2,Y0), Plane 7
.	. .	.	. .
24	Pixel (X0,Y0), Plane 0	8	Pixel (X0+2,Y0), Plane 0
23	Pixel (X0+1,Y0), Plane 7	7	Pixel (X0+3,Y0), Plane 7
.	. .	.	. .
16	Pixel (X0+1,Y0), Plane 0	0	Pixel (X0+3,Y0), Plane 0

Figure 7-7 FONT Register Format (COLOR8 mode & 68020/SPARC mode)



Bits	Description	Bits	Description
31	Pixel (X0+3,Y0), Plane 7	15	Pixel (X0+1,Y0), Plane 7
.	. .	.	. .
24	Pixel (X0+3,Y0), Plane 0	8	Pixel (X0+1,Y0), Plane 0
23	Pixel (X0+2,Y0), Plane 7	7	Pixel (X0,Y0), Plane 7
.	. .	.	. .
16	Pixel (X0+2,Y0), Plane 0	0	Pixel (X0,Y0), Plane 0

Figure 7-8 FONT Register Format (COLOR8 mode & 80386 mode)

7.2 FONTING CHARACTERS AND IMAGES

Figure 7-9 gives an example of fonting a character in COLOR1 mode with the following initializations: AUTOINCX=0, AUTOINCY=-1, and X1=X0+10. The next value to be written to FONT is 0x04000000. Figure 7-10 gives an example of fonting an image in COLOR8 mode with the following initializations: AUTOINCX=4, AUTOINCY=0, X1=X0+3, mode=68020. The next value to be written to FONT is 0x1f1f001f.

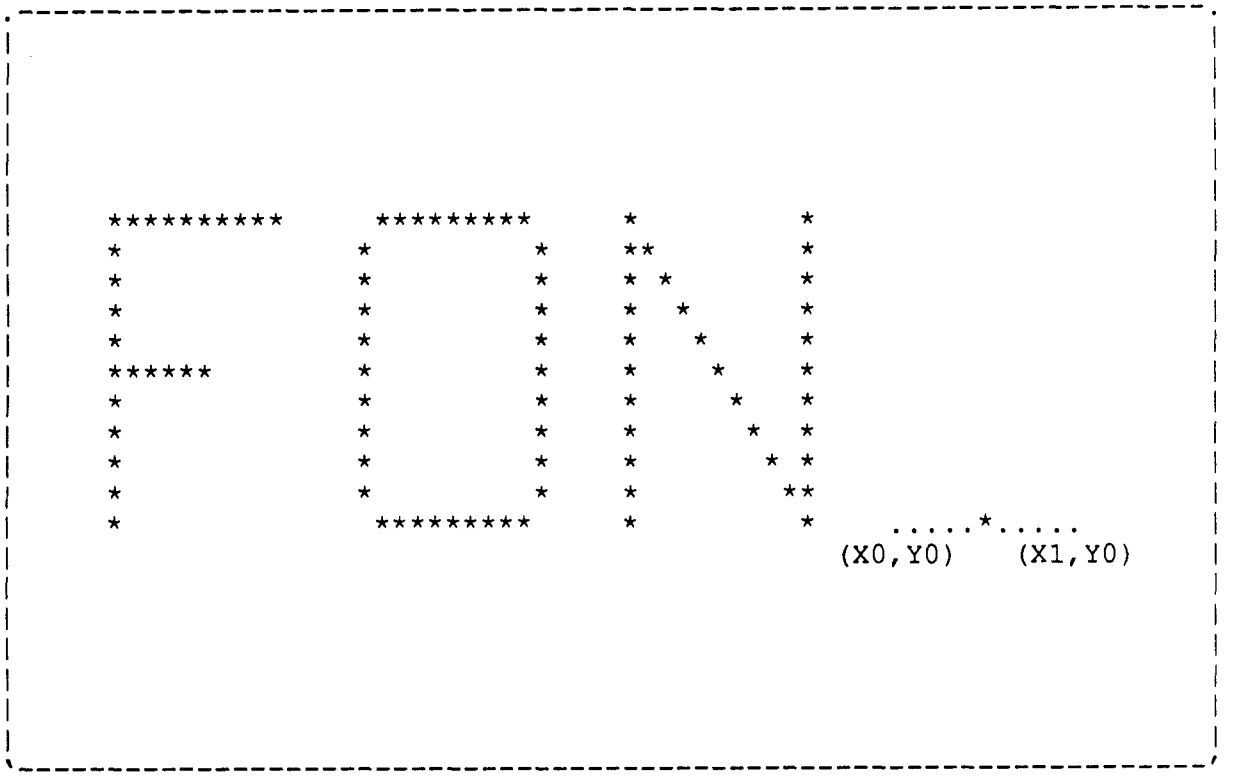


Figure 7-9 Example of a FONTing a Character

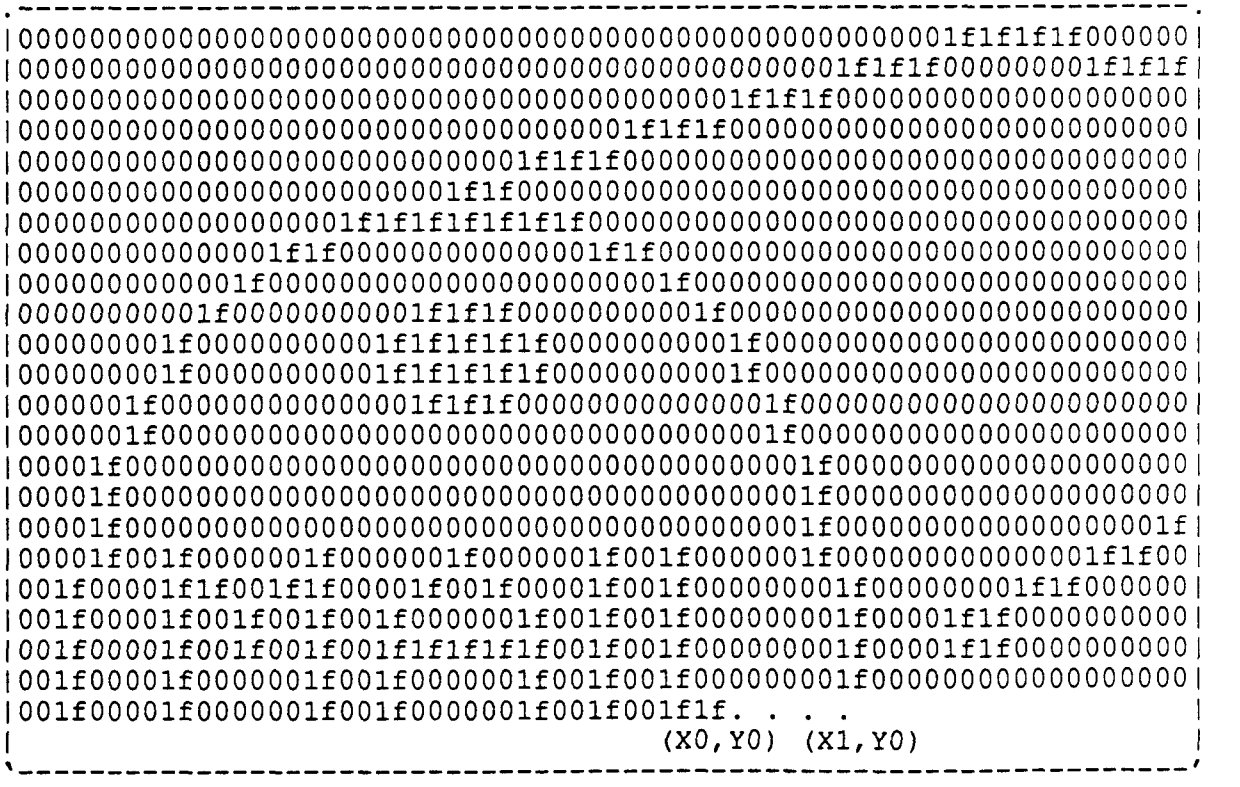


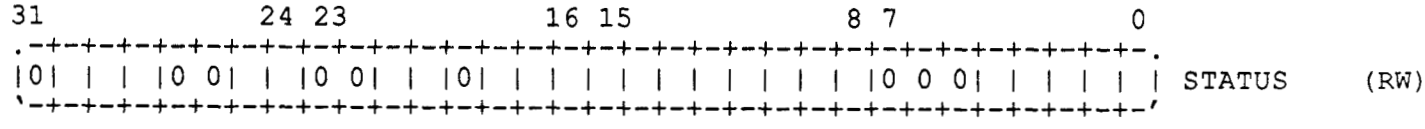
Figure 7-10 Example of a FONTing an Image

8 - STATUS REGISTER

The STATUS register contains the status of DRAWSTATUS, BLITSTATUS, PT_STATUS, and accumulated status. The STATUS register is updated whenever a value is written to the address or clip registers.

The FBC overflow state is accumulated in the STATUS register in the ACC_OVERFLOW bit. The PICK is also accumulated in the STATUS register in the ACC_PICK bit. An accumulated status is cleared by setting the ACC_CLEAR bit and the appropriate bit for the flag that is to be cleared. All flags that do not have their bits set will not be affected. If the ACC_CLEAR bit is not set, the value written will be loaded into the registers.

The status mask register can be read and written for context switching. The ACC_CLEAR bit will always read zero so that when the STATUS is reloaded during a context switch the flags will be set and not cleared. Writing to the STATUS register can only affect the ACC status. The other status will be regenerated during a context switch by reloading the address and clip registers.



Bits	Description	
31	ACC_CLEAR	(1=Clear Flags) (Always reads 0)
30	TEC_EXCEPTION	(1=TEC has an exception)
29	FULL	(1=FBC internal address registers are full)
28	BUSY	(1=FBC is Busy)
25	UNSUPPORTED_ATTR	(1=Unsupported Attribute)
24	HRMONO	(1=High Resolution Monochrome)
21	ACC_OVERFLOW	(1=Overflow in FBC has occurred)
20	ACC_PICK	(1=Part of the Object Fell Inside PICK Window)
18	TEC_HIDDEN	(1=Object is Hidden)
17	TEC_INTERSECT	(1=Object is Intersecting the TEC CLIP check)
16	TEC_VISIBLE	(1=Object is Totally Inside the TEC CLIP check)
15	BLIT_HARDWARE	(1=FBC can blit the rectangle)
14	BLIT_SOFTWARE	(1=Software must blit the rectangle)
13	BLIT_SRC_HID	(1=Src Rect is Hidden)
12	BLIT_SRC_INT	(1=Src Rect is Intersecting the CLIP Window)
11	BLIT_SRC_VIS	(1=Src Rect is Totally Inside the CLIP Window)
10	BLIT_DST_HID	(1=Dst Rect is Hidden)
9	BLIT_DST_INT	(1=Dst Rect is Intersecting the CLIP Window)
8	BLIT_DST_VIS	(1=Dst Rect is Totally Inside the CLIP Window)
4	DRAW_HARDWARE	(1=FBC can draw the Object)
3	DRAW_SOFTWARE	(1=Software must draw the Object)
2	DRAW_HIDDEN	(1=Object is Hidden)
1	DRAW_INTERSECT	(1=Object is Intersecting the CLIP Window)
0	DRAW_VISIBLE	(1=Object is Totally Inside the CLIP Window)

Figure 8-1 STATUS Register

9 - ATTRIBUTE REGISTERS

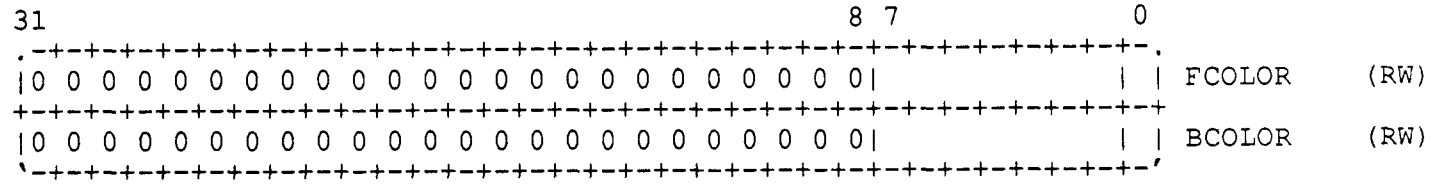
There are five attribute register sets: COLOR, RASTEROP, PLANEMASK, PIXELMASK, and PATTERN. They contain information about the current attributes of the FBC.

All the attribute registers can be read and written for context switching. It is expected that a copy of the attribute registers will be kept in memory so that on a context switch the attribute register contents will not have to be saved. Only the new context's attributes will have to be loaded.

The FBC must be idle before any of these registers can be written to.

9.1 COLOR REGISTERS

The color registers contain the current foreground and background colors. They are used in combination with the RASTEROP register to determine the final graphics operation that is to be done when rendering to the frame buffer. In COLOR1 or COLOR8 mode the eight bits of color registers correspond to the eight planes of the frame buffer. In HRMONO mode only the lowest bit has meaning, the other seven bits are ignored.



Bits	Description
7	Color bit for plane 7 (COLOR Modes Only)
6	Color bit for plane 6 (COLOR Modes Only)
5	Color bit for plane 5 (COLOR Modes Only)
4	Color bit for plane 4 (COLOR Modes Only)
3	Color bit for plane 3 (COLOR Modes Only)
2	Color bit for plane 2 (COLOR Modes Only)
1	Color bit for plane 1 (COLOR Modes Only)
0	Color bit for plane 0 (COLOR or HRMONO Mode)

Figure 9-1 Foreground and Background Color Registers

9.2 RASTEROP REGISTER

The FBC contains two different types of rasterops: BOOLEAN (Pixrect type) and LINEAR (anti-aliasing). The rasterops will be done in parallel across all pixels in the 128 bit (16 pixel) datapath.

Both types of rasterops have 16 different logical operations with two colors and masking. There will be a sixteen bit rasterop register where the software will be able to program any of the 64K combinations of rasterops.

The BOOLEAN rasterops are done between each corresponding source and destination bit of every pixel.

	Code
SOURCE	1100
DESTIN	1010

OP_urop	Code	Description	Pixrect rop	CTX_ROP Values
0	0000	d <- (0)	PIX_OPCLR	CTX_ROP_CLEAR
1	0001	d <- (~((d) | (s)))		CTX_ROP_NOR
2	0010	d <- ((d) & ~(s))		CTX_ROP_ERASE
3	0011	d <- (~(s))		CTX_ROP_DRAW_INVERTED
4	0100	d <- (~(d) & (s))		CTX_ROP_ERASED_REVERSED
5	0101	d <- (~(d))		CTX_ROP_INVERT
6	0110	d <- ((d) ^ (s))		CTX_ROP_XOR
7	0111	d <- (~((d) & (s)))		CTX_ROP_NAND
8	1000	d <- ((d) & (s))		CTX_ROP_AND
9	1001	d <- ((d) ^ ~(s))		CTX_ROP_EQUIVALENT
A	1010	d <- (d)	PIX_OPDST	CTX_ROP_NOP
B	1011	d <- ((d) | ~(s))		CTX_ROP_PAINT_INVERTED
C	1100	d <- (s)	PIX_OPSRC	CTX_ROP_DRAW
D	1101	d <- (~(d) | (s))		CTX_ROP_PAINT_REVERSED
E	1110	d <- ((d) | (s))		CTX_ROP_PAINT
F	1111	d <- (~0)	PIX_OPSET	CTX_ROP_SET

Figure 9-2 16 BOOLEAN Raster Operations

Programming a Raster Operation for the FBC involves selecting the appropriate RasterOp for the four combinations of foreground and background bit values (ie. 00, 01, 10, 11) for any given plane that will result in the desired effect.

The LINEAR rasterops assume that the pixels each contain a linear grayscale value that represent the intensity of that pixel. If the FBC is in COLOR mode, the grayscale value will be operated on directly. If the FBC is in HRMONO mode, a source bit represents one of 16 subpixels for each pixel in the frame buffer. These subpixels can be altered to be s, ~s, 0, or 1 which is then masked againsted the clip window. The resulting subpixels are then applied to a flat antialiasing filter. The flat filter weighting is:

Flat
----
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1

The grayscale value is calculated by adding together the weighting coefficients of the subpixels that are set and are not masked for a given subpixel line. The resulting value is the linear grayscale source which is applied to the linear rasterop.

There are 10 linear rasterops which are supported by the FBC. The 6 rasterops that are not supported must be done in a two step process. 1) The background must be prepared by clearing, setting, or complementing the data in the frame buffer. 2) The image must be merged into the frame buffer through the use of the DRAW, ERASE, etc. rasterops.

The linear rasterops are composed of the add/subtract with/without saturate functions. Figure 9-3 shows the LINEAR functions that corresponds to the BOOLEAN functions. When complementing the destination, the FBC must be set to PLOT for the initial drawing and to UNPLOT mode to erase the object. Bit 7 of the frame buffer is a direction bit that corresponds to the carry out of the add/subtract function. This bit allows the destination to be complemented and uncomplemented correctly. There are two indexes that correspond to each grayscale level.

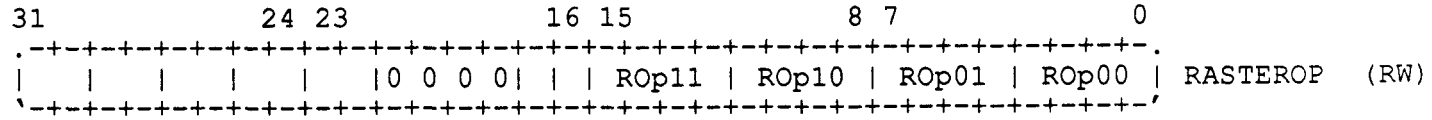
	Code
SOURCE	1100
DESTIN	1010

OP_urop	Code	Description	PLOT	UNPLOT
0	0000	d <- (0)	d = sat(d - 1)	d = sat(d - 1)
1	0001	d <- (~((d) | (s)))	na	na
2	0010	d <- ((d) & ~(s))	d = sat(d - s)	d = sat(d - s)
3	0011	d <- (~(s))	na	na
4	0100	d <- (~(d) & (s))	na	na
5	0101	d <- (~(d))	d = d + 1	d = d - 1
6	0110	d <- ((d) ^ (s))	d = d + s	d = d - s
7	0111	d <- (~((d) & (s)))	na	na
8	1000	d <- ((d) & (s))	d = sat(d -~s)	d = sat(d -~s)
9	1001	d <- ((d) ^ ~(s))	d = d +~s	d = d -~s
A	1010	d <- (d)	d = d	d = d
B	1011	d <- ((d) | ~(s))	d = sat(d +~s)	d = sat(d +~s)
C	1100	d <- (s)	na	na
D	1101	d <- (~(d) | (s))	na	na
E	1110	d <- ((d) | (s))	d = sat(d + s)	d = sat(d + s)
F	1111	d <- (~0)	d = sat(d + 1)	d = sat(d + 1)

Figure 9-3 10 LINEAR Raster Operations

Programming a Raster Operation for the FBC involves selecting the appropriate RasterOp for the four combinations of foreground and background bit values (ie. 00, 01, 10, 11) for only plane 0 that will result in the desired effect.

The RASTEROP register can override the values in the PLANEMASK, PIXELMASK, and PATTERN registers without affecting the values in the registers. The type of polygon is also stored in the RASTEROP register.



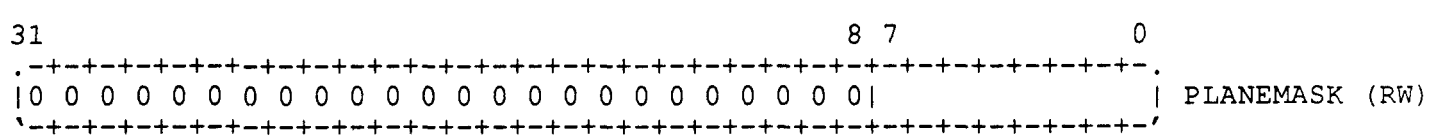
Bits	Description
31-30	Override Plane Mask Select (00=Ignore, 01=Use zeroes, 10=Use ones, 11=Use PLANEMASK)
29-28	Override Pixel Mask Select (00=Ignore, 01=Use zeroes, 10=Use ones, 11=Use PIXELMASK)
27-26	Override Pattern Select (00=Ignore, 01=Use zeroes, 10=Use ones, 11=Use PATTERN)
25-24	Polygon Draw Select (00=Ignore, 01=Overlapping, 10=Nonoverlapping, 11=Illegal)
23-22	UNSUPPORTED_ATTR (00=Ignore, 01=Unsupported, 10=Supported, 11=Illegal)
17	Rasterop Mode (0=BOOLEAN, 1=LINEAR)
16	Plot/Unplot Mode (0=PLOT, 1=UNPLOT)
15-12	Rasterop used when FCOLOR[p]=1 and BCOLOR[p]=1
11-8	Rasterop used when FCOLOR[p]=1 and BCOLOR[p]=0
7-4	Rasterop used when FCOLOR[p]=0 and BCOLOR[p]=1
3-0	Rasterop used when FCOLOR[p]=0 and BCOLOR[p]=0

where: FCOLOR[p] is the foreground color bit for plane p.
BCOLOR[p] is the background color bit for plane p.

Figure 9-4 RasterOp Register

9.3 PLANEMASK REGISTER

The plane mask register determines which planes should be written to and which should not. This allows the software to write only to the specified planes. The plane mask is tied directly to the frame buffer (it actual resides in the video ram) and works the same in color or anti-aliasing modes.

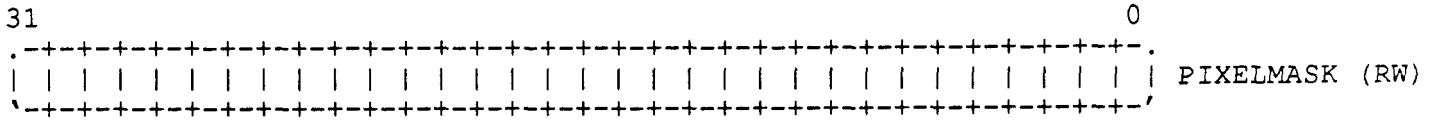


Bits	Description
7	Plane 7 Mask (0=Writes disabled, 1=Writes enabled)
6	Plane 6 Mask (0=Writes disabled, 1=Writes enabled)
.	. . .
1	Plane 1 Mask (0=Writes disabled, 1=Writes enabled)
0	Plane 0 Mask (0=Writes disabled, 1=Writes enabled)

Figure 9-5 Plane Mask Register

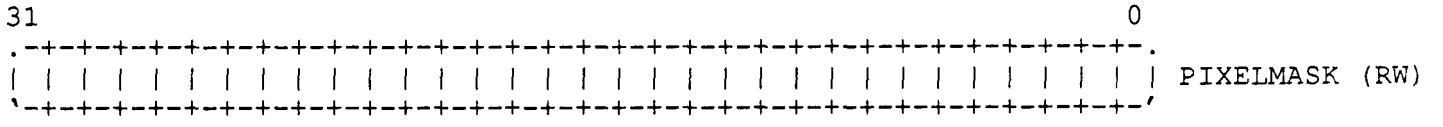
9.4 PIXELMASK REGISTER

The pixel mask register determines which pixels should be written to and which should not. This allows the software to write to specified pixels for operations such as stenciling. The pixel mask is tied directly to the frame buffer and works the same in color or anti-aliasing mode.



Bits	Description		
31	Mask for Pixel (x%32==0)	(left most)	(1=Writes enabled)
30	Mask for Pixel (x%32==1)		(1=Writes enabled)
29	Mask for Pixel (x%32==2)		(1=Writes enabled)
.	.		.
2	Mask for Pixel (x%32==29)		(1=Writes enabled)
1	Mask for Pixel (x%32==30)		(1=Writes enabled)
0	Mask for Pixel (x%32==31)	(right most)	(1=Writes enabled)

Figure 9-6 PIXELMASK Register (68020/SPARC mode)



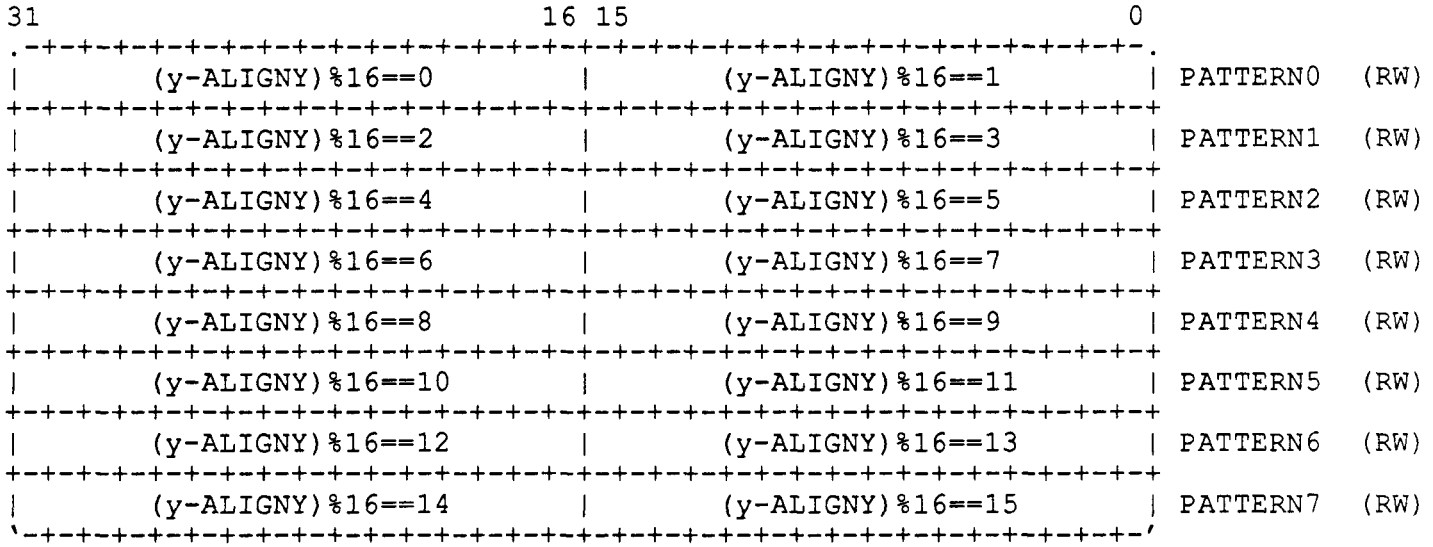
Bits	Description		
31	Mask for Pixel (x%32==31)	(right most)	(1=Writes enabled)
30	Mask for Pixel (x%32==30)		(1=Writes enabled)
29	Mask for Pixel (x%32==29)		(1=Writes enabled)
.	.		.
2	Mask for Pixel (x%32==2)		(1=Writes enabled)
1	Mask for Pixel (x%32==1)		(1=Writes enabled)
0	Mask for Pixel (x%32==0)	(left most)	(1=Writes enabled)

Figure 9-7 PIXELMASK Register (80386 mode)

9.5 PATTERN REGISTERS

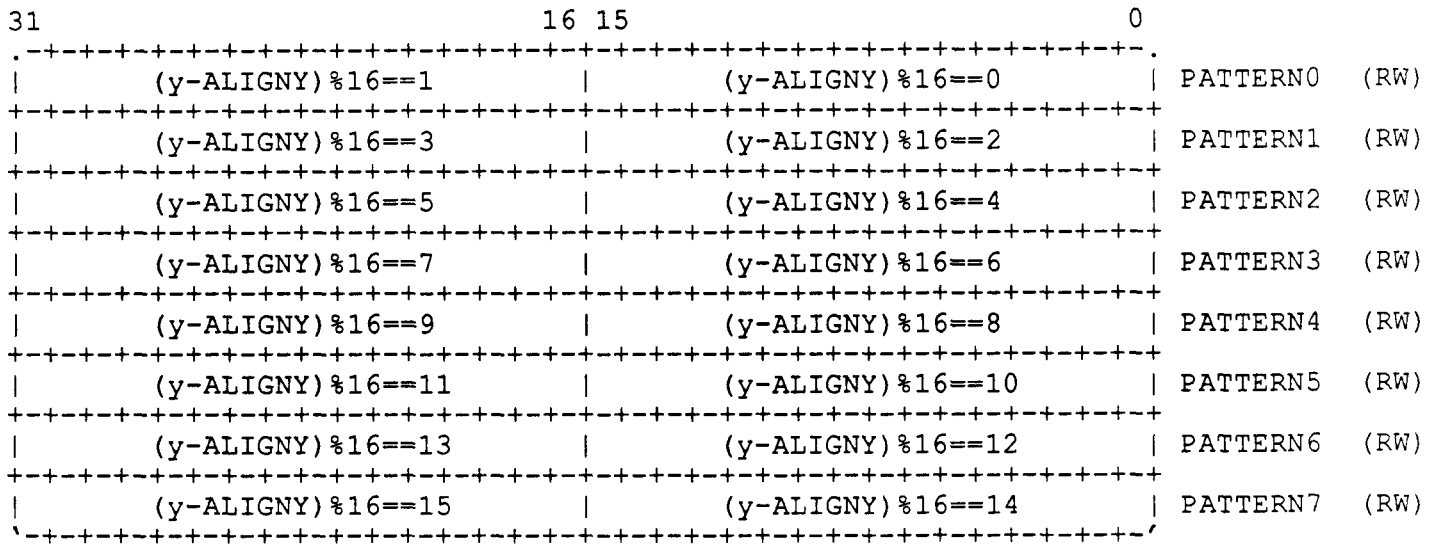
The pattern registers contain a 16 x 16 repeating pattern that is used as the source for the draw function. The pattern can be aligned in the x and y direction with respect to the screen by writing the appropriate offset to the pattern alignment register PATTALIGN. The pattern can repeat in both the x or y direction by a binary number (1,2,4,8,16) up to 16 by calculating the correct 16 x 16 pattern in software. An operation that uses any other pattern (ie. repeats every 3 pixels) must be must done totally in software. The pattern can be disabled by setting the pattern disable bit and pattern source bit in the RASTEROP register. A pattern of all one's or all zero's (as selected) will be substituted as the source pattern and the contents of the pattern registers will not be altered.

In the color modes the pattern is tied to the pixel boundaries and repeats every 16 pixels. In anti-aliasing mode the pattern is tied to the subpixel and repeats every 4 pixels (16 subpixels). In either mode the pattern is mainly used for setting the gray level of the draw command.



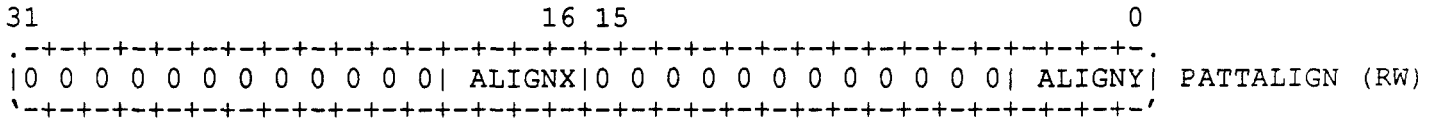
Bits	Description
31,15	Source for Pixel or Subpixel (x-ALIGNX)%16==0 (left most)
30,14	Source for Pixel or Subpixel (x-ALIGNX)%16==1
.	.
1,17	Source for Pixel or Subpixel (x-ALIGNX)%16==14
0,16	Source for Pixel or Subpixel (x-ALIGNX)%16==15 (right most)

Figure 9-8 Pattern Registers (68020/SPARC mode)



Bits	Description
31,15	Source for Pixel or Subpixel (x-ALIGNX)%16==15 (right most)
30,14	Source for Pixel or Subpixel (x-ALIGNX)%16==14
.	.
1,17	Source for Pixel or Subpixel (x-ALIGNX)%16==1
0,16	Source for Pixel or Subpixel (x-ALIGNX)%16==0 (left most)

Figure 9-9 Pattern Registers (80386 mode)



Bits	Description
19-16	Pattern Alignment X Offset
3-0	Pattern Alignment Y Offset

Figure 9-10 Pattern Alignment Register

10 - MISCELLANEOUS REGISTER

The MISC register contains a bunch of miscellaneous leftover stuff.

The FBC's EXCEPTION status for BLITs can be configured to include or not include the information that the source of a blit intersects the clip window (BLIT_SRC_INT). If it is included and the source is hidden (BLIT_SRC_HID), no exception will be given and nothing will be blitted.

VBLANK_OCCURED bit is set every time the display reaches the vertical blanking. Writing a one to the VBLANK_OCCURED bit will clear VBLANK_OCCURED. This bit is used when the user's software wants to wait (by spinning on a bit) for the next vertical blanking to occur instead of generating an interrupt.

The FBC can be in one of three different addressing modes: COLOR8 (8 plane color), COLOR1 (1 plane color), and HRMONO (1 plane high resolution antialiased monochrome). Figure 10-1 shows the different addressing modes for each of the commands.

			Addressing Modes		
Command	Source	Destination	COLOR8	COLOR1	HRMONO
DRAW	Pattern Reg	Frame Buffer	Expand Pixel	Expand Pixel	Filter Subpixel
FONT	CPU	Frame Buffer	Normal Pixel	Expand Pixel	Filter Subpixel
BLIT	Frame Buffer	Frame Buffer	Normal Pixel	Normal Pixel	*

* Illegal Combination, BLITSTATUS will cause an exception
Normal = all data is accessed as 8 planes
Expand = duplicate 1 plane source across all 8 planes
Filter = create grayscale value from 1 plane source
Pixel = addresses are pixel coordinates
Subpixel = addresses are subpixel coordinates

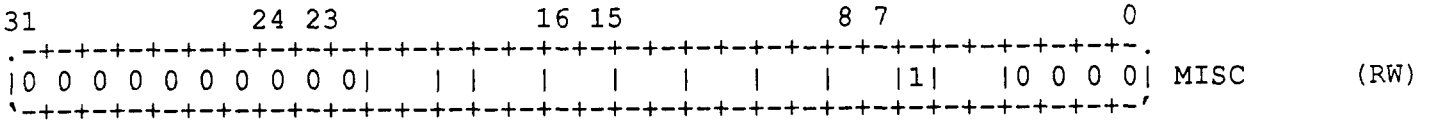
Figure 10-1 Addressing Modes

The FBC can be set to RENDER or PICK mode. If it is in RENDER mode, objects will be clipped to the clip window and rendered to the frame buffer. If the FBC is in PICK mode, objects will be clipped to the clip window to see if it is picked and will not be rendered to the frame buffer.

The FBC also handles double buffering. The user can chose which buffer is to be displayed, which buffer (or both) is to be written to, and which buffer the reads should come from.

The address INDEX bits contain the index for the Index Address Registers. It can be modified by writing the new value to the MISC register with the Modify Address INDEX bit set.

The miscellaneous register can be read and written for context switching.



Bits	Description
21-20	BLIT_SRC_CHK (00=ignore, 01=Exclude Src, 10=Include Src, 11=Illegal)
19	VBLANK_OCCURED (1=VBLANK has occurred)
18-17	Anti-Aliasing/Color Mode Select (00=ignore, 01=COLOR8, 10=COLOR1, 11=HRMONO)
16-15	Render/Pick Mode Select (00=ignore, 01=RENDER, 10=PICK, 11=Illegal)
14-13	Buffer 0 Write Enable (00=ignore, 01=Enable, 10=Disable, 11=Illegal)
12-11	Buffer 1 Write Enable (00=ignore, 01=Enable, 10=Disable, 11=Illegal)
10-9	Buffer Read Enable (00=ignore, 01=Read from Buffer0, 10=Read Buffer1, 11=Illegal)
8-7	Buffer Display Enable (00=ignore, 01=Display Buffer0, 10=Display Buffer1, 11=Illegal)
6	Modify Address INDEX (1=Modify Address Index)
5-4	Address INDEX

Figure 10-2 Miscellaneous Register

11 - CONFIGURATION REGISTER

The CONFIG register contains a bunch of miscellaneous leftover stuff that has to do with the hardware configuration of the frame buffer and system.

When the RESET bit is set all state machines except for the CPU interface are reset.

There are four ID pins (ID3 to ID0) on the FBC which correspond to bits (27 and 24) of the configuration register, respectively. The frame buffer ID number, number of buffers, etc. can be hard coded on each board.

The Chip Version Number is a unique number for every FBC developed.

The Row, Source, and Destination Caches can be individually disabled.

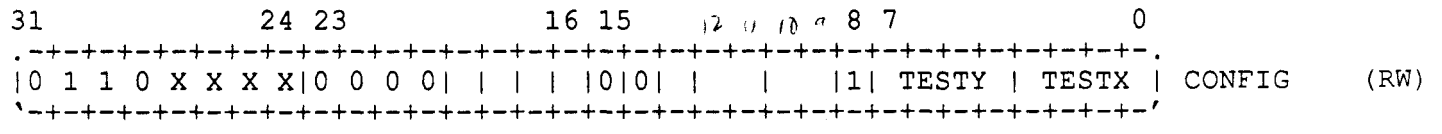
When the RESET bit is set all state machines except for the CPU interface are reset.

The Resolution bits select the width of the screen. It can either be 1024 or 1152 pixels across. Any vertical resolution can be obtained by programming the TEC with the appropriate numbers. The most common resolutions will be be 1024x768, 1024x1024, 1152x900, 1280x1024, or 1600x1280. All VRAMs must be populated for any of these modes.

The SPARC/020/386 Mode selects the byte ordering for 32 bit access for the FONT register during COLOR8 mode. It also selects the correct decoding of the size and address pins during the Dumb Frame Buffer Emulation.

The test window's offset from the CLIPPING window (TESTX,TESTY) are also stored in the CONFIG register. They can be modified by writing the new value to the CONFIG register with the Modify TESTX & TESTY bit set.

The configuration register can be read and written from the FBC hardware configuration address space (fhc). It also can be read from regular FBC address space (fbc), the configuration register will not change if it is written to in the FBC address space (fbc).



Bits	Description		
31-24	Frame Buffer ID Number	(0x60=CG6)	(R)
23-20	Chip Version Number		(R)
19	Disable FROPs	(1=Disable Fast ROPs)	
18	Row Cache Disable	(1=Disable Row Cache)	
17	Source Cache Disable	(1=Disable Source Cache)	
16	Dest Cache Disable	(1=Disable Dest Cache)	
15	RESET	(1=Reset FBC)	
13	Bit/Byte Reversal	(0=68020/SPARC, 1=80386)	
12-11	Resolution	(00=1024, 01=1152, 10=1280, 11=1600)	
10-9	SPARC/020/386 Mode	(00=SPARC, 01=68020, 10=80386)	
8	Modify TESTX & TESTY	(1=Modify Tests)	
7-4	TESTY		
3-0	TESTX		

Figure 11-1 Configuration Register