

Errata Sheet MB86297 Carmine

© Fujitsu Microelectronics Europe GmbH

History

Date	Author	Version	Comment
10.10.2005	AG	1.0	First release
12.11.2005	H.Nishi	1.4	Updated same to Japanese errata Rev.1.4
2.3.2006	H.Nishi	1.45	E9 was changed,E18-E25 were added
2.8.2006	H.Nishi	1.52	E26-E31 were added
17.02.2006	AG	1.53	E32-E37 and Carmine versions added,
2.03.2006	H.Nishi	1.71	E38-E42 and which is repaired on ES2 were added. E21 and E24 were modified.
14.04.2006	H.Nishi	1.9	E43-E48 were added. E19,E22,26,27,39 were modified.
30.05.2006	H.Nishi	2.0	E49-E54 added. E9,E13 and E45 modified.
15.06.2006	H.Nishi	2.1	E55 and E56 were added.
11.07.2006	H.Nishi	2.2	E11 and E12 were modified.
03.10.2006	H.Nishi	2.3	E57-E64 were added.
13.11.2006	AG	2.4	E65 added
09.01.2007	H.Nishi	2.5	E66-E68 added
03.09.2007	H.Nishi	2.6	E26 and E65 were modified.
21.09.2007	AvT	2.7	E4 was extended (new workarounds)
28.10.2008	H.Nishi	2.8	E69 added
26.08.2009	AvT	2.81	E23 was extended (new condition)

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH restricts its warranties and its liability for **all products** (e.g. software include or header files, application examples, application Notes, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment.**

1. Fujitsu Microelectronics Europe GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Microelectronics Europe GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.
2. Should a Product turn out to be defect, Fujitsu Microelectronics Europe GmbH's entire liability and the customer's exclusive remedy shall be, at Fujitsu Microelectronics Europe GmbH's sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Microelectronics Europe GmbH in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Microelectronics Europe GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Microelectronics Europe GmbH.
3. To the maximum extent permitted by applicable law Fujitsu Microelectronics Europe GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.
4. To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH's and its suppliers' liability is restricted to intention and gross negligence.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES

To the maximum extent permitted by applicable law, in no event shall Fujitsu Microelectronics Europe GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.

Errata List:

September 3rd, 2007

No	Item	Effectuated samples		
		ES1 (MB86297) DC: 0545	ES2 (MB86297) DC: 0619	ES3 (MB86297A) DC: 0702
		DC : available beginning with date code xxxx X : effected --- : not effected		
E1	Decrementing/incrementing the stencil buffer	X	X	X
E2	texturing with point primitive	X	X	X
E3	texture size limitation	X	X	X
E4	BltCopy: BottomRight	X	X	X
E5	BltCopy.PixelAlpha	X	X	X
E6	Blt.compressedDataTransfer	X	X	X
E7	Blt:CompressedData with clipping	X	X	X
E8	Displaylist setting	X	X	X
E9	problem with Polygon Primitive	X	X	X
E10	DrawBitmap, limitation of RSIZEX	X	X	X
E11	MipMapTextureSize limitation	X	X	X
E12	Polygon drawing and DrawRectP:ClearPolyFlag generates freeze or wrong drawing	X	X	X
E13	Polygon drawing after clearing polygon buffer	X	X	X
E14	BT bit of BC regsiter affects all BitBlit function	X	X	X
E15	Transparent mode affects DrawRectP	X	X	X
E16	DrawBitmapP:DrawBinary with Y clipping under XRR=4096 does not work	X	X	X
E17	Missing triangle that depends on conditions of slope and size and the timing of processing	X	---	---
E18	A setting of broken line does influence a triangle	X	---	---
E19	Writing the pixel two times or wrong start point occurs in broken line	X	X	X
E20	Width of primitive become wrong if primitive's slope value is larger than 4095	X	---	---
E21	Carmine freeze by less than 5 length line after broken line drawing	X	---	---
E22	Missing primitive or lock-up that occurs after the line with either a thick line or an anti-alias line	X	---	---
E23	BLPO register setting ignored	X	X	X
E24	BitBlit with memory reading hanging up	X	---	---
E25	Wrong blending result of anti-alias line	X	---	---
E26	Wrong color at vertex of side change when both Fore and Back color are enabled	X	X	X
E27	Front side material is ignored when G_Material is used between G_Begin and G_End	X	X	X
E28	Lock-up occurs when it uses geometry clipping or coral compatible functions	X	---	---
E29	Wrong pixel of anti-alias line	X	---	---
E30	DrawBitmapP with a rendering clipping	X	X	X

	makes lock-up when minus value is set to RYs			
E31	Lock-up occurs when it uses DrawBitmapP and it depends on internal timing of hardware	X	---	---
E32	Command SaveRestoreReg	X	---	---
E33	Command DrawBitmap, DrawBitmapP	X	X	X
E34	Sequence of SaveRestoreReg	X	---	---
E35	Pixel blender calculates wrong color	X	---	---
E36	Display list index referenced mode	X	---	---
E37	SRCA is used for A factor of FUNCRGB calculation instead of SRCRGB	X	---	---
E38	Fog blending affects A factor	X	---	---
E39	One pixel length broken line with either a clipping or an anti-alias makes lock-up	X	---	---
E40	Incrementing a pointer of broken line in the case of not broken line.	X	---	---
E41	2 times pixel drawing occurs after enabling broken line	X	---	---
E42	Point,DrawLine,DrawTrap missing	X	X	X
E43	Setting Z position of light source ID2 is ignored	X	X	X
E44	SaveRestoreReg with simultaneous register access does wrong saving/restoring	X	X	X
E45	Wrong drawing or lock-up by displaylist that needs waiting internal idle state	X	Partly repaired	Partly repaired
E46	Extra pixel and shortage bottom line of polygon	X	X	X
E47	Polygon flag clearing to outside of rendering clipping area	X	---	---
E48	DrawBitmapP:DrawBitmap make wrong drawing with vertical scaling	X	---	---
E49	DrawBitmap scaling affects BitDraw and it makes lock-up or illegal drawing	X	X	X
E50	In polygon, stencil test makes lock-up and alpha test makes low performance	X	X	X
E51	Write access missing about display and capture register	X	---	---
E52	Frame synchronization error when capture clip is used	X	---	---
E53	Clip error of write-back	X	---	---
E54	Alpha layer selection error	X	---	---
E55	Texture wrapping CLAMP and BORDER wrong result	X	X	X
E56	G_VertexIndex:DrawElement with "ub" format or "us" format makes lock-up	X	X	X
E57	An error about texture size and texture base address by using TXS,TIS and TBR register	X	X	X
E58	Wrong texture mapping of bilinear high speed mode with MIRRORED_REPEAT	X	X	X
E59	Wrong fog blending with over 4096 hight primitive	X	X	X
E60	Problem of backside drawing	X	X	X
E61	Missing part of QUADS with geometry clipping	X	X	X
E62	Lockup or wrong drawing occur when DrawBitmapP:DrawBitmap is executed	X	X	X

	after BltDraw			
E63	G_VertexIndex:DrawArray generates lock-up that depends on internal hardware timing	X	X	X
E64	G_Vertex after G_VertexIndex generates wrong parameter use of G_Vertex	X	X	X
E65	Memory controller, prolonged latency during arbitration	X	X	---
E66	Slit or extra pixel appears when using polygon	X	X	X
E67	Lost line with length less than 1	X	X	X
E68	Wrong center position of a broken line with LINEXT:BPM=1	X	X	X
E69	Point and line performance degradation due to register GMDR2E:TL and SP bits	X	X	X

E1:
Decrementing/incrementing the stencil buffer

[back to top](#)

Detail

Function for INCR/DECR(decrementing/incrementing without wrapping) the contents of the stencil buffer doesn't work correctly.

INCR: FF -> 0, must be FF ->> FF

DECR: 0 -> FF, must be 0 -> 0

E2:
Texturing with point primitive

[back to top](#)

Detail

Different colors can be defined for the front and back side of a primitive.

This doesn't work in the case of texturing with a point primitive.

Workaround:

Don't use the back color in this case. Since there are no sides in the point primitive, a front color is always referred.

E3:
Texture size limitation

[back to top](#)

Detail

Carmine allows a texture size of 4096x4096.

A bug occurs if mip mapping (only with format RGBA8) and bilinear high speed format is used.

The texture size is limited to 2048 in this case.

Workaround:

Set texture size to 2048.

E4:
BltCopy:BottomRight

[back to top](#)

Detail

Problem occurs with BltCopy:BottomRight.

BltCopyP can not handle overlapping areas in the framebuffer.

Workaround:

There are two possible software workarounds:

A) Copy the source to another completely independent memory space.

Then copy the source back to the final destination using a second BltCopyP.

B) Repeatedly use BltCopyP each usage of which copies a horizontal or vertical line.

If 'SrcY < DstY', copy horizontal lines from the bottom.

If 'SrcX < DstX', copy vertical lines from the right.

In all other cases, use TopLeft. Use only TopLeft for BltCopy

E5:
BltCopy.PixelAlpha

[back to top](#)

Detail

A bug occurs in case of BitBlit with alpha blending - Carmine hangs.

Workaround:

If you set 1 to MDR4's AS, you must always also set MDR4's BM as alpha blend.

E6:
Blt.compressedDataTransfer

[back to top](#)

Detail

Blit with compressed data does not work

Workaround:

Resetting the BltParameter by BltCopy(SRCXS =0, SRCYS = 0, BRSIZE =0, BRSIZEY = 0)
(compressed data don't have SRCX, SRCY , BRSIZE...)
following BltCopyCompressedP or BltCopyCompAlphaMapP is working

E7:
Blt:CompressedData with clipping

[back to top](#)

Detail

Clipping does not work, CXMIN parameter is not checked

Workaround:

No workaround

E8:
Displaylist setting

[back to top](#)

Detail

Details of the problem
Rendering engine hangs if an definite display list is used
DL:
- Displaylist that updates GMDR0,GMDR1,GMDR2,GMDR2E,and IVAOGL
- G_LineSetting
- G_PolygonSetting
- G_PolygonOffset
- G_VertexSetting

Workaround:

Place the command "BindTexture" after DL.

E9:
Problem with Polygon Primitive

[back to top](#)

Detail

Carmine freezes if G_Begin(Polygon or PolygonClip)...G_End

Workaround:

A or B avoids problem.

A) Placing the commands after polygon's G_End to the display list

```
0x09E2 0000 // ClearPolyFlag
yyyy xxxx // (xxxx,yyyy) is inside of a rendering clipping area
0x0001 0001
```

B) Placing the commands after polygon's G_End to the display list

```
0xF1010108 // SetRegister(MDR0)
???? ???? // Set 0 to CX and CY, to disable a rendering clipping
0x09E2 0000 // ClearPolyFlag
0x0000 0000
0x0001 0001
0xF1010108 // SetRegister(MDR0)
???? ???? // Put back MDR0 by original value
```

E10:
DrawBitmap, limitation of RSIZE

[back to top](#)

Detail

Carmine hangs, if RSIZE <=8

Workaround:

Use DrawBitmap only for RSIZE > 8

E11:
MipMapTextureSize limitation

[back to top](#)

Detail

When using mip map textures that have different size, wrong texel will be referred.
(e.g. size of Tex 0 != size of Tex 1)

Workaround

If multitexturing is used, set SizeTex0 = SizeTex1
if SingleTexturing is used, put Flush after changing the TexID
e.g.
BindTexture
Flush
G_Begin....

E12:
Polygon drawing and DrawRectP:ClearPolyFlag generates freeze or wrong drawing

[back to top](#)

Detail

When MDR4:BM or MDR4:FE isn't 0, polygon drawing and ClearPolyFlag generates freeze or wrong drawing.

Workaround

Set 0 to MDR4 before polygon drawing and ClearPolyFlag.

E13:

[back to top](#)

Polygon drawing after clearing polygon buffer

Detail

In polygon drawing (SetVertex2i/2iP:PolygonBegin – Draw:PolygonEnd), lock-up occurs when it matches 1) or 2).

- 1) when size of polygon (quad around of polygon) is 0
- 2) A rendering clipping is enabled and all vertices are out of a rendering clipping area.

Workaround

A or B avoids problem.

A) Placing the commands after Draw:PolygonEnd to the display list

```
0x09E2 0000 // ClearPolyFlag
yyyy xxxx // (xxxx,yyyy) is inside of a rendering clipping area
0x0001 0001
```

B) Placing the commands after Draw:PolygonEnd to the display list

```
0xF1010108 // SetRegister(MDR0)
???? ???? // Set 0 to CX and CY, to disable a rendering clipping
0x09E2 0000 // ClearPolyFlag
0x0000 0000
0x0001 0001
0xF1010108 // SetRegister(MDR0)
???? ???? // Put back MDR0 by original value
```

E14:

[back to top](#)

BT bit of BC register affects all BitBlt function

Detail

BT bit of BC register affects not only DrawBitmapP:DrawBinary but also another BitBlt function.

Workaround

Except case that you intend to use BT bit, set 0 to BT bit of BC register.

E15:

[back to top](#)

Transparent mode affects DrawRectP

Detail

TE bit of MDR4 register affects DrawRectP.

Workaround

Set 0 to TE bit of MDR4 register before using DrawRectP.

E16:
DrawBitmapP:DrawBinary with Y clipping under XRR=4096 does not work

[back to top](#)

Detail

If Y clipping occurs under the conditions of XRR=4096, DrawBitmapP:DrawBinary draws pixel to wrong memory address.

Workaround

Set XRR less than 4096, when you use DrawBitmapP:DrawBinary.

E17:
Missing triangle that depends on conditions of slope and size and the timing of processing.

[back to top](#)

Detail

There is a case that a triangle is not drawn. It happens depending on a slope, a triangle size, and the timing of processing. In the case of a big triangle, it becomes easy to occur.

Workaround

In the first place, use "SC v1.2 geometry firmware". Additionally, select a way to avoid bug from two solutions A or B.

- A) In the case of independent triangles (appoint *Triangles to G_Begin)
Insert "Flush" after every three "G_Vertex". If it is difficult to discriminate every three "G_vertex", you can insert "Flush" after each "G_vertex".
Can't apply this method to Triangle_Strip and Triangle_Fan. Carmine freezes when the condition matches any following three conditions.
 - 1. Shadow primitive
 - 2. The top-left-rule non-applied primitive
 - 3. When geometry clipping occurs.
- B) In the case of strip triangles or fan triangles (appoint *Triangle_Strip or *Triangle_Fan to G_Begin)
Use "G_VertexLOG" instead of "G_Vertex".
Though a triangular disappearance is evaded by this method, the performance decreases greatly because it processes writing the log and processes the triangular generating by the firmware. Moreover, The memory space of the amount in which the log of a few minutes of the maximum between G_Begin and G_End can be written is necessary.
If it is possible to divide into an independent triangle, the method A can be used.
It is possible to evade by dealing with the following C when drawing in the triangle by the rendering display list.
- C) In the case of rendering triangle(use SetVertex and DrawVertex)
Insert "Flush" after every triangle. If you use "SetVertex" and "DrawVertex", insert Flush after each "DrawVertex".

Although it is insufficient counter measure, occurring frequency drops provided that you change a scaling value of G_Viewport small in order to draw only a small triangle.

E18:

[back to top](#)

A setting of broken line does influence a triangle.

Detail

A garbage pixel may be drawn when drawing a triangle under the condition that MDR1's BL bit is set as 1.

Workaround

Set 0 to BL bit of MDR1 register when you draw triangle.

E19

[back to top](#)

Writing the pixel two times or wrong start point occurs in broken line.

Detail

Writing the pixel two times occurs at the first broken line drawing after the update of broken line pattern (BLP register).

Workaround

Update BLPO register too, when updating BLP register.

E20:

[back to top](#)

Width of primitive become wrong if primitive's slope value is larger than 4095.

Detail

Slope value become wrong if primitive's slope value is larger than 4095.

Workaround

Do not draw the figure where the slope value is larger than 4095.

E21:

[back to top](#)

Carmine freeze by less than 5 length line after broken line drawing.

Detail

Carmine freeze when drawing less than 5 length line after broken line that is drawn with LINEEXT=1.

Workaround

Draw broken line with LINEEXT=0 after that whenever drawing broken line with LINEEXT=1. Carmine back to normal after that. There are no influences on result if that line(LINEEXT=0) draws into the area that is out of rendering clipping frame in a geometry view volume.

E22:

[back to top](#)

Missing primitive or lock-up that occurs after the line with either a thick line or an anti-alias line.

Detail

The primitive input after a primitive that meets all of the three following requirements might disappear. It depends on the state of an internal module.

- 1) X domain line (X coordinate difference larger than Y coordinate difference between vertices)
- 2) Y direction rendering clipping has been enabled and last 1 span is out of clipping frame.

Workaround

This trouble doesn't happen when neither the thick line nor anti-alias are specified.

Select a way to avoid bug from two solutions A or B.

A) In the case of independent lines (appoint *Lines to G_Begin)

Insert "Flush" after every three "G_Vertex". If it is difficult to discriminate every three "G_vertex", you can insert "Flush" after each "G_vertex".

Can't apply this method to Line_Strip. Carmine will freeze when geometry clipping occur.

B) In the case of strip lines (appoint *Line_Strip to G_Begin)

Use "G_VertexLOG" instead of "G_Vertex".

Though a primitive disappearance is evaded by this method, the performance decreases greatly because it processes writing the log and processes the triangular generating by the firmware. Moreover, The memory space of the amount in which the log of a few minutes of the maximum between G_Begin and G_End can be written is necessary. If it is possible to divide into an independent line, the method A can be used.

It is possible to evade by dealing with the following C when drawing in the triangle by the rendering display list.

C) In the case of rendering line(use SetVertex and DrawVertex)

Insert "Flush" after every line. If you use "SetVertex" and "DrawVertex", insert Flush after each "DrawVertex".

E23:

[back to top](#)

BLPO register setting ignored

Detail

The BLPO register setting is ignored if either of two conditions are met:

- a) DrawPixel is input after BLPO register update
- b) A primitive is input that isn't drawn after a BLPO register update

"a primitive that isn't drawn" means :

- A primitive is outside the rendering clipping frame or
- A primitive that doesn't pass the ideal pixel area (a small triangle that doesn't include the center of a pixel. A line that doesn't pass in a lozenge area ((diamond rhombus) around the center of a pixel)

Workaround

Input a primitive that draws after a BLPO register update.

E24:

[back to top](#)

BitBlt with memory reading hanging up

Detail

When it uses BitBlt including memory reading, Carmine hangs up or wrong color pixels appear. That depends on the number of processing pixels.

Condition 1:

$Result1 = (Xsize * Ysize * (1 \ll BPP)) \% (128 * 8)$
 $0 < Result1 < 8 \text{ AND } Result1 < (Xsize * (1 \ll BPP))$

Condition 2:

$Result2 = ((BA \% 8) + (Xsize * (1 \ll BPP))) \% 8$; if $(Result2 == 0)$ $Result2 = 8$;
 $Result1 < Result2$

Condition 3: This is for AlphaMap is used

$Result3 = (((BA \% 8) + (Stride * (Ysize - 1))) \% 8) + Xsize \% 8$; if $(Result3 == 0)$ $Result3 = 8$;
 $Result1 < Result3$

BPP : Bit Per Pixel(0=8BPP,1=16BPP,2=32BPP), AlphaMap is BPP=0(8BPP)

BA : BaseAddress(= starting address of drawing) after a rendering clipping.

If you want to use X coordinates instead of BaseAddress,
"starting X << BPP" can be also used as BA.

Xsize : X size after a rendering clipping

Ysize : Y size after a rendering clipping

Stride : B*SizeX

A: 1 AND 2

B: 1 AND 3

- When using Alpha blending/ROP or Forming or AlphaMap and the sizes match condition A
-> Hangs up.
- When it uses BltCopyP or BltCopyAlternateP or BltCopyAltAlphaMapP or BltCopyCompressedP or BltCopyCompAlphaMapP and a destination matches condition A
-> Hangs up.
- When it uses BltCopyP or BltCopyAlternateP or BltCopyAltAlphaMapP or BltCopyCompressedP or BltCopyCompAlphaMapP and a source matches condition A
-> Some wrong color pixels appear.
- When it doesn't match any conditions above and AlphaMap matches condition B
-> Some wrong color pixels appear.

Workaround

1) Change size and position to the value that doesn't match conditions above. This workaround is not realistic when rendering clipping is enabled.

2) Make frame buffer bigger than display area and disable a rendering clipping. If an area that is added around a display area is bigger than biggest figure of BLT, this will work like rendering clipping. In addition, change both size and position to the value that doesn't match conditions.

E25:

[back to top](#)

Wrong blending result of anti-alias line

Detail

Blending result of anti-alias line that matches 1 or 2 will be wrong.

- 1) X domain line and Y coordinate less than 0
- 2) Y domain line and X coordinate less than 0

Workaround

Don't use minus value to coordinates. Or enable a rendering clipping. When rendering clipping is enabled, minus part of a figure isn't drawn.

E26:

[back to top](#)

Wrong color at vertex of side change when both Fore and Back color are enabled

Detail

If condition matches with all of "1)-4)", wrong color triangle appears.

- 1) TriangleFan or TriangleStrip or Quads or Quadstrip
- 2) BC of vertex are enabled
- 3) SM or AS of MDR2 is set as gouraud shading
- 4) Side is different from before

Workaround

It is possible to evade by dealing with either A) or B) or C).

- A) Disable BC and BA of vertex
 - B) Use flat shading
 - C) Specify the same color as FC and BC
-

E27:

[back to top](#)

Front side material is ignored when G_Material is used between G_Begin and G_End

Detail

If the condition matches with all of 1)-3), setting of front side material is ignored.

- 1) G_Material is placed between G_Begin and G_End
- 2) Both side setting (FRT=1,BAK=1) is specified by G_Material
- 3) One side lighting is specified by G_LightSetting

This phenomenon does not occur when it uses G_Material outside of G_Begin-G_End.

Workaround

It is possible to evade by dealing with either A) or B) or C).

- A) Use G_Material outside of G_Begin-G_End
- B) Use one side setting (don't use FRT=1 with BAK=1)
- C) Use both sides lighting whenever it uses material setting of both sides

E28:

[back to top](#)

Lock-up occurs when it uses geometry clipping or coral compatible functions

Detail

When geometry clipping occurs or CORAL compatible special function(GMDR2E) is used, lock-up occurs that depends on input timing of displaylist.

Workaround

The method of evading in each figure type is shown below.

Point: lock-up doesn't occur

Line: Change LineStrip to Lines

Triangle: Change TriangleFan and TriangleStrip to Triangles

Polygon: No evasion plan

This trouble occurs when the displaylist is slowly input. When using indirectDL, the incidence decreases because the speed to which the displaylist is input quickens.

E29:

[back to top](#)

Wrong pixel of anti-alias line

Detail

If the condition matches with "A and (B or C)", wrong pixel appears.

A) Rendering clipping is enabled

B) Width of line is 1 pixel

C) Y domain line (difference of Y is bigger than difference of X)

Workaround

Insert Flush command before using anti-alias line. When using anti-aliased line continuously, only inserting one Flush command at first is needed.

E30:

[back to top](#)

DrawBitmapP with a rendering clipping makes lock-up when minus value is set to RYs

Detail

When it uses minus value to RYs with a rendering clipping enabled, lock-up or wrong drawing occurs. The phenomenon does not occur in 32bit/pixel color.

1) Size

8bit/pixel color : RsizeX is not a multiple of four

16bit/pixel color : RsizeX is not a multiple of two

2) Position of drawing

RYs set to minus.

3) Area that is eliminated by a rendering clipping

8bit/pixel color : Number of eliminated Y pixels is not a multiple of four

16 bit/pixel color : Number of eliminated Y pixels is not a multiple of two

When the condition matches all of 1)-3), lock-up or wrong drawing occurs.

Workaround

It is possible to evade by either of following 1) or 2). This trouble doesn't occur when the rendering clip is invalid.

1) Use RsizeX on the following conditions.

· 8 bit color: Multiple of four.

· 16 bit color: Multiple of two.

2) Draws to another area once with the rendering clip invalidated. After the rendering clip is made effective, the drawing result of DrawBitmapP is copied onto target coordinates by using BitCopyAlternateP.

E31:

[back to top](#)

Lock-up occurs when it uses DrawBitmapP and it depends on internal timing of hardware

Detail

DrawBitmapP makes lock-up that depends on internal hardware timing.

Workaround

There is no complete evasion method.

The incidence of the phenomenon falls when various effects of drawing are invalidated or the drawing size is reduced.

E32:
Command SaveRestoreReg

[back to top](#)

Detail

The command SaveRestoreReg does not work with odd number of data to be saved or restored.
In that case a wrong value is saved or restored.
That phenomenon does not occur if an even number of data is used.

Workaround

Using an even number of data for the SaveRestoreReg command.
or
Placing a dummy save command before a restore command of odd number of data.

E33:
Command DrawBitmap of DrawBitmapP

[back to top](#)

Detail

The commands DrawBitmap of DrawBitmapP do not work if transparent mode is enabled and foreground color is equal to background colour

Workaround

Setting different values to foreground and background colour

E34:
Sequence of SaveRestoreReg

[back to top](#)

Detail

If 2 consecutive SaveRestoreReg commands are executed to restore the registers of different register areas, the 2nd restore command fails.

e.g.
1.step: restore to geometry area
2.step: restore to rendering area

Workaround

Placing a dummy save command to the same area of the 2nd restore command.

1.step: restore to geometry area
2.step: dummy save of rendering area
3.step: restore to rendering area

E35:
Pixel blender calculates wrong color

[back to top](#)

Detail

The result of the pixel blender is wrong. It depends on setting of BLDTU register. If src0 is not PRIMARY_COLOR and src1 or src2 is set to PRIMARY_COLOR and Gouraud shading is enabled.

Workaround:

Using PRIMARY_COLOR to only src0.

E36:
Display list index referenced mode

[back to top](#)

Detail

Carmine locks up if the index referenced display list mode is used.

Workaround

No workaround

E37:
SRCA is used for A factor of FUNCRGB calculation instead of SRCRGB

[back to top](#)

Detail

SRCA is used for FUNCRGB calculation that specified by BLDTU* instead of SRCRGB.

Workaround

Set the same source to both SRCRGB and SRCA.

E38:
Fog blending affects A factor

[back to top](#)

Detail

A factor is changed by Fog blending.

Workaround

Don't use the function with Fog that uses an A factor with Fog. Or don't care an A factor after Fog blending..

E39:

[back to top](#)

One pixel length broken line with either a clipping or an anti-alias makes lock-up

Detail

When all of 1 to 3 matches, lock-up that depends on internal timing occurs.

- 1) Either a clipping or an anti-alias is enabled.
- 2) LINEEXT:BPM is enabled.
- 3) Line pixel length (width or height) is one

Workaround

Avoiding one of the conditions 1-3 avoids the occurring of the phenomenon.

E40:

[back to top](#)

Incrementing a pointer of broken line in the case of not broken line

Detail

When there is fraction part in line length, BLPO is incremented irrespective of MDR1's BL. The line length means length after internal line processing.

Workaround

In case of using geometry engine, don't set 1 to EP of GMDR1.
In case of using a rendering engine, uses only **"NoEnd"** for line.

E41:

[back to top](#)

2 times pixel drawing occurs after enabling broken line

Detail

2 times pixel drawing occurs after enabling broken line (BL of MDR1 = 1).

Workaround

Update BLPO after setting 1 to BL of MDR1. See E23 for updating BLPO.

E42:

[back to top](#)

Point,DrawLine,DrawTrap missing

Detail

1. In the case of G_Begin(*Points*)
This phenomenon occurs only when all factors of vertex are disabled. (GMDR0 or IVAOGL is all 0).
2. In the case of DrawPixel, DrawPixelZ, DrawLine, DrawTrap
This phenomenon occurs when there are no SetRegister after DrawPixel, DrawPixelZ, DrawLine, DrawTrap.

Workaround

1. In the case of G_Begin(*Points*)
At least enabling one factor of vertex. (GMDR0 or IVAOGL is not 0).
 2. In the case of DrawPixel, DrawPixelZ, DrawLine, DrawTrap
Input Flush or SetRegister(Don't care which register) after these displaylist.
-

E43:
Setting Z position of light source ID2 is ignored

[back to top](#)

Detail

Z position of light source ID2 is updated by setting light source ID3. Z position of light source ID3 updates Z position of light source ID2 too. There are no way to write Z position of light source ID2.

Workaround

Use same Z position to both light source ID2 and 3. Or, don't use light source ID2.

E44:
SaveRestoreReg with simultaneous access to Kottos register, both accesses fail

[back to top](#)

Detail

When KOTTOS register access occurs in the middle of SaveRestoreReg processing, "register access" and "save/restore" make wrong result.

Workaround

Follow the way described below to avoid KOTTOS register access until SaveRestoreReg finished.

1. Wait IDLE status of KOTTOS (GCTR:ST=0)
2. Input SaveRestoreReg to Displaylist FIFO.
3. Input G_Interrpt with DRAWFIN enabled
4. Wait until interrupt occurs. Must not access KOTTOS register until interrupt occurs.
5. Clears interrupt cause.

E45:

[back to top](#)

Wrong drawing or lock-up by displaylist that needs waiting internal idle state

Detail

Wrong drawing or lock-up occurs depends on internal timing when condition matches below.

Condition)

Displaylist that is included in Group A is input. After that, displaylist that is included in Group B is input.

Group A) The commands of group A define a primitive type. The problem occurs only, if the primitive is changed by such a command.

G_Begin-G_End	(primitive type before was different)
DrawPixel/DrawPixelZ	(primitive type before was not DrawPixel/DrawPixelZ)
DrawLine	(primitive type before was not DrawLine)
DrawLine2i/2iP	(primitive type before was DrawLine or not line)
DrawTrap	(primitive type before was not DrawTrap)
DrawVertex2i/2iP	(primitive type before was not triangle)
DrawVertex:Line	(primitive type before was DrawLine or not line)
DrawVertex:TriangleFan	(primitive type before was not triangle)
Draw:Polygon	(primitive type before was not Draw:Polygon)

Group B)

BLT command(=G_BitBlit, DrawRectP,DrawRectAlphaMapP,DrawBitmapP,DrawBitmapLargeP, BltCopyP,BltCopyAlternateP,BltCopyAltAlphaMapP,BltCopyCompressedP,BltCopyCompAlphaMapP)
LoadFirm
RegTexture
SetFog
Draw:Flush

Workaround

1) or 2) avoids this phenomenon.

- 1) Inputs SetRegister after displaylist that is included in Group A
- 2) Inputs SetRegister before displaylist that is included in Group B

ES2

Only Draw:Flush is repaired in ES2. Draw:Flush can be used for workaround instead of SetRegister in ES2.

E46:

[back to top](#)

Extra pixel and shortage bottom line of polygon

Detail

When polygon viewport coordinates matches condition A, extra pixel(phenomenon A) occurs. When polygon viewport coordinates matches condition B, shortage bottom line(phenomenon B) occurs.

[Condition A]

$\text{Int}(Y_{\max} - Y_{\min}) < \text{int}(Y_{\max+0.5}) - \text{int}(Y_{\min})$

[Condition B]

$\text{int}(Y_{\max} - Y_{\min}) + 1 < \text{int}(Y_{\max+0.5}) - \text{int}(Y_{\min})$

Y_{\min} means most minimum Y in all vertices of polygon. Meaning of $Y_{\max}, X_{\min}, X_{\max}$ is same to Y_{\min}

Workaround

- `G_Begin(*Polygon)-G_End`
Use GeoFirm Rev1.6 and set 1 to SM bit of MDR2 register.
 - `SetVertex2i/2iP:PolygonBegin – Draw:PolygonEnd`
Set the fraction part of Y coordinates to 0.
-

E47:

[back to top](#)

Polygon flag clearing to outside of rendering clipping area

Detail

When it draws polygon that is outside of rendering clipping area and inside of geometry clipping area, outside of flag buffer is cleared.

Geometry clipping area means device coordinates area that is defined by `G_ViewVolumeClipXY` and `G_Viewport`. Rendering clipping area means device coordinates area that is defined by `CXMIN` and `CXMAX` and `CYMIN` and `CYMAX`.

Workaround

- `G_Begin(*Polygon)-G_End`
Set geometry clipping area same as rendering clipping area.
 - `SetVertex2i/2iP:PolygonBegin – Draw:PolygonEnd`
A or B or C avoids this phenomenon.
A) Use `G_Begin(*Polygon)-G_End` with above way. When scale of `G_Viewport` is 1 and offset is 0, these are almost same except command.
B) Reserve non-used memory area at outside of flag buffer by considering maximum size of all polygons.
C) Draw only polygon that is inside of a rendering clipping area.
-

E48:

[back to top](#)

`DrawBitmapP:DrawBitmap` make wrong drawing with vertical scaling

Detail

`DrawBitmapP:DrawBitmap` with vertical scaling(`MDR0:BSV=1`) make wrong drawing. This depends on internal hardware timing. When this phenomenon occurs, wrong pattern word is referred instead of right pattern word.

Workaround

Don't use vertical scaling (`MDR0:BSV=1`).

E49:

[back to top](#)

DrawBitmap scaling affects BltDraw and it makes lock-up or illegal drawing

Detail

Lock-up or illegal drawing occurs when it matches all of 1),2),3).

- 1) DrawBitmapP:BlitDraw or DrawBitmapLargeP
- 2) A rendering clipping is enabled (MDR0's CX or CY is 1)
- 3) MDR0's BSV or BSH is not 0.

Workaround

Set 0 to MDR0's BSV and BSH when it uses DrawBitmapP:BlitDraw or DrawBitmapLargeP.

E50:

[back to top](#)

In polygon, stencil test makes lock-up and alpha test makes low performance

Detail

Lock-up occurs when it matches all of 1),2),3).

- 1) Stencil test is enabled (MDR6:STCE=1)
- 2) Polygon after triangle
- 3) There isn't setting of MDR5 or MDR6 between triangle and polygon.

Low performance occurs when it matches all of 1),2),3).

- 1) Alpha test is enabled (MDR5:ATE=1)
- 2) Polygon after triangle
- 3) There isn't setting of MDR5 or MDR6 between triangle and polygon.

Workaround

Placing SetRegister of MDR5 or MDR6 before polygon.

E51:

[back to top](#)

Write access missing about display and capture register

Detail

Some write access for display and capture may miss, if the write accesses are done continuously in incremental address order.

Workaround

This phenomenon occurs with PCI burst transaction. To prevent burst transaction, write accesses have to be issued in decremental address order or interleaved with discontinuous address access.

E52:
Frame synchronization error when capture clip is used

[back to top](#)

Detail

Clip rectangle is applied before captured image is written into memory. If the Y-value of the start position (CIVSTR) is not zero, the synchronization between capture and display is blocked and the captured image can not be displayed properly.

If the capture buffer area is allocated close to address 0, unsynchronized capture image is displayed, otherwise any image like un-initialized memory data is displayed.

Workaround

- 1) in case of 656 input
use with CIVSTR=0
use single buffer mode with CIVSTR>0
- 2) in case of RGB input
clip only by input range definition (RGBHEN,RGBVEN)

E53:
Clip error of write-back

[back to top](#)

Detail

Clip rectangle can be applied for write-back by (WIVstr,WIHstr)-(WIVend,WIHend) but it does not work properly.

If WIVstr > 0, starting address of write-back area is displaced by a address proper for WIVstr. If WIVend is valid, last about 16 pixels are not written.

Workaround

Invalidate clip rectangle by defining largest rectangle as follows

(WIVstr,WIHstr)=(0,0)

(WIVend,WIHend)= (4095,4095)

All displayed image is written back with this definition.

With a adjacent BitBLT important data can be separated from un-necessary data.

E54:
Alpha layer selection error

[back to top](#)

Detail

This error occurs for lower three layers after overlap order select is applied.
In other words, this problem is about layers selected by DSL5,6,7.
If alpha dedicated layers are used with these layers, corresponding alpha layer cannot be selected properly or different alpha layer is selected as follow.
layer selected by DLS5 -> alpha selection of layer selected by DLS6 is used
layer selected by DLS6 -> alpha selection of layer selected by DLS7 is used
layer selected by DLS7 -> blend cannot be applied as bottom layer (correct)

Workaround

Cancel exchange of alpha layer selection by LAN values.
If layer overlay order is not changed by DLS register, set LAN values as follow.
Alpha layer for L5 is selected by L6 alpha selection
Alpha layer for L6 is selected by L7 alpha selection

E55:
Texture wrapping CLAMP and BORDER wrong result

[back to top](#)

Detail

Condition "1) and 2)" or "1) and 3)" generate wrong result..
1) filtering is LINEAR or LINEAR_MIPMAP_NEAREST or LINEAR_MIPMAP_LINEAR
2) One of WRAPS and WRAPT is CLAMP and other is BORDER
3) Both WRAPS and WRAPT is CLAMP

Workaround

Use CLAMP_TO_EDGE instead of CLAMP. Difference is only border color blending.

E56:
G_VertexIndex:DrawElement with "ub" format or "us" format makes lock-up

[back to top](#)

Detail

Condition "1) and 2)" makes lock-up.
1) G_VertexIndex:DrawElement
2) IDFOGL:DFIDX is 0(ub) or 1(us)

Workaround

A) or B) avoid lock-up.
A) Use G_VertexIndex:DrawElement in IDFOGL:DFIDX=2(ui).
B) Split DrawElement into limit number of vertices when DFIDX isn't 2. Limit number is calculated in below way.
$$\text{limit number of vertices} = (256 - 8) / \text{limit_addr_num}$$
$$\text{limit_addr_num is calculated below.}$$
$$(X,Y)=2, (X,Y,Z)=2, (X,Y,Z,W)=3,$$
$$(Rf,Gf,Bf)=2, (Rf,Gf,Bf,Af)=3, (Rb,Gb,Bb)=2, (Rb,Gb,Bb,Ab)=3, (Nx,Ny,Nz)=2,$$
$$(S0,T0)=2, (S0,T0,Q0)=3, (S1,T1)=2, (S1,T1,Q1)=3, (F)=1$$

Add all number that is corresponding to enabled element. For example, in case of (X,Y,Z,Nx,Ny,Nz), limit_addr_num is $(X,Y,Z):2 + (Nx,Ny,Nz):2 = 4$.

E57: [back to top](#)
An error about texture size and texture base address by using TXS,TIS and TBR register

Detail

When TBR is used, wrong texture size or tile size is used in hardware.
When TXS or TIS is used, wrong texture base address is used in hardware.

Workaround

- A) or B) avoid this phenomenon.
 - A) Use RegTexture
 - B) Set TXS and TIS after TBR
-

E58: [back to top](#)
Wrong texture mapping of bilinear high speed mode with MIRRORED_REPEAT

Detail

Condition “1) and 2)” makes wrong texture mapping.
1) RegTexture:Base BA=1 (e.q. bilinear high-speed mode)
2) RegTexture:State WRAPS=MIRRORED_REPEAT or WRAPT=MIRRORED_REPEAT

Workaround

Don't use MIRRORED_REPEAT with Bilinear high-speed mode.

E59: [back to top](#)
Wrong fog blending with over 4096 height primitive

Detail

Condition “1) or 2)” makes wrong fog blending.
1) Height of primitive(device coordinates) is greater than 4096
2) F coordinates after viewport transformation or F coordinates of SetVertex/DrawVertex is greater than 32767

Workaround

- A) or B) avoid lock-up.
 - A) Using primitive that height is less than 4097(device coordinates)
 - B) Using F coordinates that is less than 32768
-

E60: [back to top](#)
Problem of back side drawing

Detail

An error of backside color occurs in the mode that draws both sides. Lock-up occurs in the mode that draws only backside. Condition “1)” makes wrong color of backside. “2)” makes lock-up.

- 1) IVAOGL:FC=1 and BC=1
- 2) IVAOGL:FC=0 and BC=1

Workaround

- A) or B) or C) avoid this phenomenon.
- A) Don't use backside drawing (Set 1 to G_PolygonSettings CLB)
- B) Don't use backside color (Set 0 to IVAOGL:BC)
- C) Set same color as FC and BC of vertex.

E61:

[back to top](#)

Missing part of QUADS with geometry clipping

Detail

A part of QUADS isn't drawn when geometry clipping occur. This problem doesn't occur in QUADS_Strip.

Workaround

A) or B) avoid this phenomenon.

A) Insert displaylists below after each QUADS(e.q. each 4 vertices)

4D000001h

00000001h

FF000001h

B) Put only one QUADS between G_Begin and G_End

E62:

[back to top](#)

Lockup or wrong drawing occur when DrawBitmapP:DrawBitmap is executed after BitDraw

Detail

Lockup or wrong drawing occur when DrawBitmapP:DrawBitmap is executed after DrawBitmapP:BitDraw or DrawBitmapLargeP.

Workaround

1) or 2) avoid this phenomenon.

1) When rendering clipping is enabled (MDR0:CX=1,CY=1)

Draws dummy DrawBitmapP:DrawBitmap at outside of clipping area, before DrawBitmapP:DrawBitmap that is doubled in vertical.

2) When rendering clipping is disabled (MDR0:CX=0,CY=0)

Draws dummy "DrawBitmapP:DrawBitmap" that satisfies conditions A and B and C before "DrawBitmapP:DrawBitmap" that is doubled in vertical.

A) MDR0:BSV != 1

B) All of pattern data is 0

C) MDR4 register's TE is 1 and TColor register is same as BC register, or BC register's BT is 1.

E63:

[back to top](#)

G_VertexIndex:DrawArray generates lock-up that depends on internal hardware timing

Detail

G_VertexIndex:DrawArray generates lock-up that depends on internal hardware timing.

Workaround

Put one of below displaylists before G_VertexIndex:DrawArray.

DC_LogOutAddr, OverlapXYOfft, OverlapZOfft, SetCoorRegister, SetModeRegister, SetGModeRegister

Attention : E18's workaround is needed in the case of using SetModeRegister or SetGModeRegister.

E64:

[back to top](#)

G_Vertex after G_VertexIndex generates wrong parameter use of G_Vertex

Detail

Wrong status that depends on internal hardware timing is generated after executing G_VertexIndex. In this wrong status, G_Vertex makes an error of parameter. But G_VertexIndex works correctly in this wrong status.

Workaround

- Case of IndirectDL mode
Set Word Count of G_IndirectDL to finish IndirectDL mode before G_Vertex. After return to DirectDL mode, the status returns correct.
- Case of DirectDL mode
Store G_VertexIndex in graphics memory and use IndirectDL mode. Apply workaround of above at that time.

E65:

[back to top](#)

Memory controller, prolonged latency during arbitration

Detail

The prioritisation of memory accesses from different internal modules of Carmine does not work properly with some applications.

In the event of many random accesses to the memory (e.g. drawing vertical lines that are 1 pixel wide), the latency of following read accesses from the display controller may be too long.

It seems that distortions become visible on the display, which is caused by a prolonged latency during the arbitration process.

Workaround

A general workaround is currently not available.

E66:

[back to top](#)

Slit or extra pixel appears when using polygon

Detail

A slit or extra pixel appears when using polygons. This is due to internal hardware calculation precision.

Workaround

Use GeoFirm v1.9 or later and use a new mode of polygon drawing as shown below:

- Case of normal polygon
GMDR2E bit10 = 1
MDR2 bit19 = 1
- Case of non-top left rule polygon
GMDR2E bit10 = 1
MDR2 bit19 = 1, bit18 = 1

[This operation is needed for both cases]:

Add first vertex of polygon again as last vertex.

[Restriction]

Texture and fog don't work correctly in this new mode of polygon drawing. Depth testing can work with Z coordinate restriction which has been described in hardware manual.

E67:

[back to top](#)

Lost line with length less than 1

Detail

If the device co-ordinate has a length value which is less than 1, it is possible that a line is lost.

Workaround

Draw line, setting 1 to the EP bit of the GMDR1 register.

* Note that it often causes the drawing to be done twice at the connection point.

E68:

[back to top](#)

Wrong center position of a broken line with LINEEXT:BPM=1

Detail

1) $|(V1.X - V0.X)| < |(V1.Y - V0.Y)|$

*This means Y domain line

2) BPM bit of LINEEXT register is 1

3) $((V0.Y < V1.Y) \&\& (V0.X > V1.X)) \parallel ((V0.Y > V1.Y) \&\& (V0.X > V1.X))$

*This means slope is a negative value.

Workaround

Draw broken line with setting 0 in the BPM bit of LINEEXT register.

E69:

[back to top](#)

Point and line performance degradation due to register GMDR2E:TL and SP bits

Detail

When the TL (top-left rule mode) and SP (Shadow Primitive) bits of the GMDR2E (Geometry Mode register for Triangle Extension) register are set to 1, unnecessary chip-internal firmware processing for points and lines occurs, degrading the GDC's performance.

Workaround

Set 0 to both TL and SP of GMDR2E before drawing a point or a line.