Errata Sheet MB86298 'Ruby'

Version V1.21

History

Date	**Author**	**Version**	**Comment**
12.11.2008	H.Nishi	1.00	First Version, added E1
13.01.2009	H.Nishi/AvT	1.01	Replaced E1 description, added E2, E3, E4, E5
11.03.2009	AvT	1.02	Extended description of E3
14.04.2009	AvT	1.03	Added E6, E7, E8, E9, E10
17.06.2009	AvT	1.04	Added E11
15.07.2009	AvT	1.05	Added E12
05.10.2009	AvT	1.06	Added E13, E14, E15, E16
30.11.2009	AvT	1.07	Added E17
07.09.2010	AvT	1.08	Modified E6, corrected ES2 fix status in table. Added E18, E19, E20, E21, E22, E23
05.04.2011	AvT	1.09	Added E24, changed company name
27.05.2011	AvT	1.10	Added E25
05.07.2011	AvT	1.20	Added E26
25.3.2012	AvT	1.21	Added E27

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Semiconductor 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 Semiconductor 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 Semiconductor Europe GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days form the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Semiconductor 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 Semiconductor Europe GmbH's entire liability and the customer's exclusive remedy shall be, at Fujitsu Semiconductor 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 Semiconductor 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 Semiconductor Europe GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Semiconductor Europe GmbH.

3. To the maximum extent permitted by applicable law Fujitsu Semiconductor 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 Semiconductor 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 Semiconductor 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:

March 25th, 2012

No	Item	**Effected samples** DC : available beginning with date code xxxx X : effected --- : not effected	
		ES1 (MB86298) DC: 0904	**ES2** (MB86298) DC: TBD
E1	Graphics Core: Wrong attribute reads or hang-up.	X	X
E2	Graphics Core: Stride restriction in framebuffer in 16 BPP mode	X	X
E3	Memory Controller: auto-refresh can not be disabled	X	X
E4	Capture: the 2 rightmost pixels of each line are missing	X	X
E5	Capture: several pixels of the last line are not stored in memory	X	X
E6	Memory Controller: Deadlock in Memory Controller (read and write to same address on same port)	X	-
E7	Graphics Core: Hang-up or wrong drawing result	X	-
E8	PixBLT: PixBlt hang-up when using ShaderMatrix mode with 1*1 Matrix	X	X
E9	PixBLT: PixBlt does not execute second shader operation	X	-
E10	Display: RGBA color format can not be used with upscaling	X	X
E11	CMDSEQ: Access to Command Sequencer registers blocked when CMD-FIFO full	X	X
E12	Graphics Core: Hang-up when processing line primitives whose starting point Y coordinate is zero after ViewPort transformation	X	-
E13	MEMPACK: Changes in cache memory not always correctly flagged	X	-
E14	Graphics Core: Hang-up when rendering and using a scissor	X	X
E15	DISPLAY: Wrong operation of color LUT during blanking period	X	X
E16	Graphics Core: The BLT result is wrong when Graphics Core works in alpha blending or alpha map mode and the blend ratio is 0	X	X
E17	Graphics Core: Depth buffer or stencil buffer updated with incorrect data	X	X
E18	Graphics Core: ARGES may hang-up when the rendering primitive is changed from triangle to line.	X	X
E19	Graphics Core: One pixel may be rendered outside of the scissor frame during line rendering.	X	-
E20	PCIe: Recovery from L1 power savings state as defined by the PCI-Express Base Specification may fail.	X	X
E21	INTERCONNECT BUS Interconnect port #2 hangup	X	-
E22	DISPLAY: Incorrect pixels displayed in blanking when Dithering Unit (DITH) is active	X	X
E23	SSCG: High current in power down	X	X
E24	DISPLAY: Per pixel blending not functional for 16bpp and and indexed color format (1bit alpha)	X	-
E25	Graphics Core: Hang-up or wrong pixel in Bit Block Transfer with AlphaMap	X	X
E26	Graphics Core: Per-pixel blending not functional for 16bpp and indexed color	X	X

E27	Graphics Core: Wrong 1/W and varying variables of a fragment	X	X

E1: [back to top]
Graphics Core: Wrong attribute reads or hang-up

Detail

When the ATTRNUM field of the SetShaderInfo:AttrInf0 display list command is 1 or 2, the fault (wrong attribute reads or hang-up) could occur, depending both on internal hardware timing and additional conditions as described below:

- In the case of ATTRNUM=2
  Address of Attribute0[bit29-bit3] != Address of Attribute1[bit29-bit3]
  Address of Attribute1[bit29-bit3] == Address of Attribute2[bit29-bit3]
  Address of Attribute1[bit2-bit0] < Address of Attribute2[bit2-bit0]

- In the case of ATTRNUM =1
  Address of Attribute0[bit29-bit3] == Address of Attribute1[bit29-bit3]
  Address of Attribute0[bit2-bit0] < Address of Attribute1[bit2-bit0]

Workaround

- In the case of ATTRNUM=2
  Writes Base address = 0 and Stride = 0 into Attribute2 by SetIndexPointer display list command.

- In the case of ATTRNUM =1
  Writes Base address = 0 and Stride = 0 into Attribute1 by SetIndexPointer display list command.

E2: [back to top]
Graphics Core: Stride restriction in framebuffer in 16 BPP mode

Detail

In 16 bits-per-pixel color mode, stride values of the framebuffer that are not a multiple of 4 are handled incorrectly.

Workaround

In 16bits/pixel color mode (SetFrameBPP display list command, BPP = 01b), set the stride value of the XRR bitfield of the SetFrame display list command to a multiple of 4.

E3: back to top
Memory Controller: auto-refresh can not be disabled

Detail

If the auto-refresh function of the Memory Controller has been enabled, it is not possible to disable it again. The bug prohibits writing 0x0000 to the TREFI[15:0] bitfield of the SRC register.

Workaround

Terminate all data transmission to memory and do a software reset of the complete Memory Controller by writing a '1' and then a '0' to the DDR2CSFTRST register field in the global control unit. The entire memory configuration has to be repeated before accessing the memory again.

E4: back to top
Capture: the 2 rightmost pixels of each line are missing

Detail

In HD-DDR mode when the HD_SPREAD bitfield of the High Definition Configuration register (HDCFG.HD_SPREAD) is 0x2 or 0x3, the 2 rightmost pixels of each line are dropped, e.g. a frame which is 1280 x 720 in size will only be captured as 1278 x 720.

Workaround

Either avoid using this mode (use HD_SPREAD = 0x0 or HD_SPREAD = 0x1) or display the frame captured without the last 2 pixel columns.

E5: back to top
Capture: several pixels of the last line are not stored in memory

Detail

Capture: In RGB or HD capture mode (configured in the VCM 'Video Capture Mode' register) and when the CHP (Capture Horizontal Pixel) register is set to values that are less than the width of the incoming frame of the external video source several pixels of the last line are not stored in memory.

Workaround

The CHP register must be set to the same value of the width of the external video input.

E6: back to top
Memory Controller: Deadlock in Memory Controller (read and write to same address on same port)

Detail

A memory deadlock can occur under very specific conditions (simultaneous read/write to an address using the same memory controller port and within tight timing constraints). This is very unlikely to occur in a normal application.

Workaround

Avoid accessing the same buffer in DDR memory simultaneously.
Will be fixed in ES2.

E7: back to top
Graphics Core: Hang-up or wrong drawing result

Detail

Triangle type primitives and PointSprite may be incorrectly drawn (pixels, depth buffer, stencil function writing shift). The accumulation of incorrectly drawn pixels may lead finally to a hang-up.

Workaround

This phenomenon can be avoided in a special mode which can be specified by the SetDebugParam display list command. After inserting the following display list command, operation is changed to the special mode:

```
F4200000h  // SetDebugParam with the special sub command 20h
00000001h
```

In this special mode, performance is reduced to 44%-95% of the original rate. The exact performance loss depends on the specific application.

E8: back to top
PixBLT: PixBlt hang-up when using ShaderMatrix mode with 1*1 Matrix

Detail

PixBlt will hang up when using ShaderMatrix with a 1*1 Matrix.

Workaround

Don’t use this mode.

E9: back to top
PixBLT: PixBlt does not execute second shader operation

Detail

There is a bug in PixBlt that prevents executing a second shader operation after the first one finished correctly.

Workaround

Please reset the shader after each shader operation when driven over PixBlt unit.

E10: back to top
Display: RGBA color format can not be used with upscaling

Detail

When upscaling is enabled for the Display Controller and the color format RGBA is switched on, the 'R' component is missing in the displayed picture. The same issue exists in 16 bit RGBA mode.

Workaround

Avoid using the RGBA pixel format with upscaling, in both 32bpp and 16 bpp modes. Use ARGB format instead.

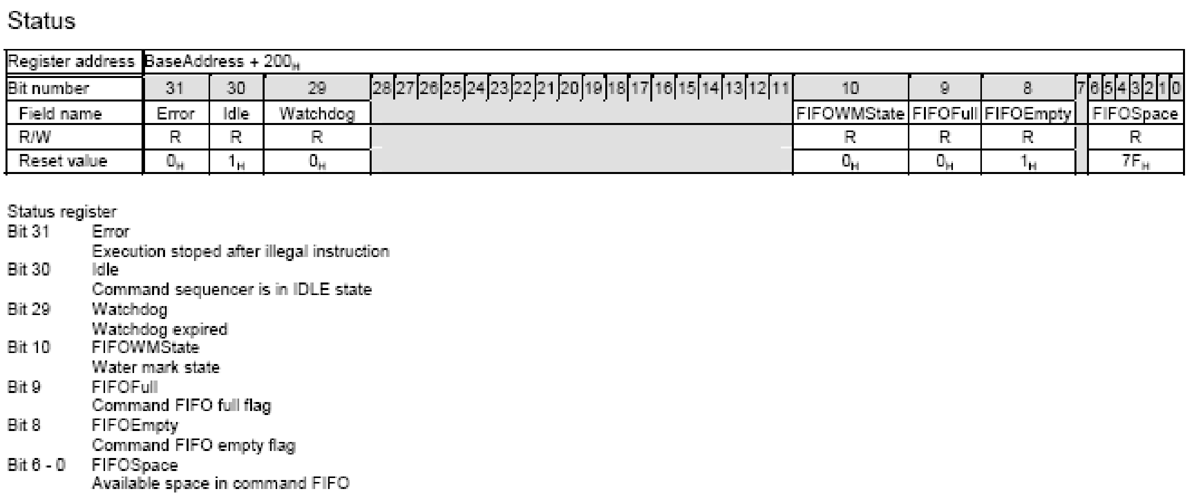
E11: back to top
CMDSEQ: Access to Command Sequencer registers blocked when CMD-FIFO full

Detail

Access to any of the Command Sequencer registers is blocked when the FIFO of the Command Sequencer is completely full.

Workaround

Take particular care not to fill the Command Sequencer completely. To check the available space in the Command Sequencer FIFO, decrement the value read from the Status register at BaseAddress + 0x200 in the bitfield FIFOSpace (Bits 6:0) by 1.



Status

Register address	BaseAddress + 200_H																															
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field name	Error	Idle	Watchdog																			FIFOWMState	FIFOFull	FIFOEmpty		FIFOSpace						
R/W	R	R	R																			R	R	R		R						
Reset value	0_H	1_H	0_H																			0_H	0_H	1_H		$7F_H$						

Status register

Bit 31	Error Execution stoped after illegal instruction
Bit 30	Idle Command sequencer is in IDLE state
Bit 29	Watchdog Watchdog expired
Bit 10	FIFOWMState Water mark state
Bit 9	FIFOFull Command FIFO full flag
Bit 8	FIFOEmpty Command FIFO empty flag
Bit 6 - 0	FIFOSpace Available space in command FIFO

E12: back to top

Graphics Core: Hang-up when processing line primitives whose starting point Y coordinate is zero after ViewPort transformation

Detail

When rendering a line primitive whose starting point Y coordinate is zero after ViewPort transformation, the RASPFO (RASterizations and Post Fragment Operations - a part of the graphics core) module will cause the system to hang.

Workaround

In the software application, please modify the ViewPort parameter (Y_Scaling and Y_Offset) or the ViewVolumeXYClip parameter (YMIN or YMAX) to satisfy the following conditions:

YMIN * Y_Scaling + Y_Offset > 0 and Y_Scaling > 0

or

YMAX * Y_Scaling + Y_Offset > 0 and Y_Scaling < 0

E13: back to top

MEMPACK: Changes in cache memory not always correctly flagged

Detail

When internal cache memory has been written to and it is written back to memory by a software or timer triggered flush, the modification flags are not correctly cleared when the cache line is read again. This error causes data not to be read from DDR2 memory, but taken from the cache although it is outdated. The prerequisistes for the bug to occur are:

1) Write access to the same address is from two different mempack ports
2) Cache lines in the first mempack port are not reused after the last write and before read access to the same address again

Workaround

1) Access buffers from the same mempack port when they are smaller than 4 kb or if the last data written might be read back first and it's possible that the data might have been modified in the memory via a different mempack port.

2) Make 16 dummy writes to unused addresses from different cache line addresses (at least 256 bytes apart) after every flush (re-use of cachelines clears the modified byte flags correctly) if the buffer is smaller than 4kb or if the last data written might be read back first and the data could have been modified in the memory via a different mempack port.

E14: back to top
Graphics Core: Hang-up when rendering and using a scissor

Detail

When rendering a line primitive whose starting and end point Y coordinates are the same and both more than 4094, the RASPFO (RASterizations and Post Fragment Operations - a part of the graphics core) module will cause the system to hang.

Workaround

Please modify the ViewPort (X_Scaling and X_Offset) and ViewVolumeXYClip parameters (Xmin and Xmax) to satisfy the following conditions:

(Xmax*X_Scaling+X_Offset) - (Xmin*X_Scaling+X_Offset) <= 4094

Alternatively, please modify the ViewPort (X_Scaling and X_Offset), ViewVolumeXYClip (Xmin) and SetScissorFrame parameters (CXmax) to satisfy the following conditions:

CXmax - (Xmin*X_Scaling+X_Offset) <= 4094

E15: back to top
DISPLAY: Wrong operation of color LUT during blanking period

Detail

The CLUT values effect both the active pixels and the pixels in the blanking period. This is a problem if the blanking-level of the display controller is used in any subsequent device for clamping or gain control.

Workaround

1. Do not change the entry of the CLUT effecting the blanking-level OR
2. The external device must evaluate the data enable signal of the display controller.

E16: back to top
GRAPHICS CORE: The BLT result is wrong when GRAPHICS CORE works in alpha blending or alpha map mode and the blend ratio is 0

Detail

The BLT result is wrong when GRAPHICS CORE uses the following commands and the blend ratio is 0. The framebuffer should be not updated, but the alpha component *is* updated in GRAPHICS CORE as is the source image:

<alpha blending mode>

```
 DrawRectP, DrawBitmapP, DrawBitmapLargeP,
  BltCopyP, BltCopyAlternateP, BltCopyCompressedP
```

<alpha map mode>

```
 DrawRectAlphaMapP, BltCopyAltAlphaMapP, BltCopyCompAlphaMapPxxx
```

Workaround

None

E17: back to top
GRAPHICS CORE: Depth buffer or stencil buffer updated with incorrect data

Detail

When rendering in the sequence described below, if step 3's SetMask command is executed immediately after step 2, then the graphics core (ARGES) updates the depth buffer or stencil buffer with incorrect data.

1. Set all the buffers (Frame, Depth, Stencil)' writemasks to "ON"
2. Render primitives
3. Change one or two buffers' writemask settings to "OFF"
4. Render primitives

Workaround

Insert a Flush command just before the SetMask command.

E18: back to top
GRAPHICS CORE: ARGES may hang-up when the rendering primitive is changed from triangle to line.

Detail

When the rendering primitive type is changed from triangle to line and specific input timing conditions occur, the ARGES graphics core may hang up.

Workaround

Please insert the following commands when the rendering primitive is changed from triangle to line:

```
0x9a070000
0x00000000
```

E19: back to top
GRAPHICS CORE: One pixel may be rendered outside of the scissor frame during line rendering.

Detail

If the sub axis's co-ordinate is equal to "MAX + Minimum unit in hardware" during line rendering, one pixel may be rendered outside of the scissor frame. MAX is the maximum co-ordinate of the scissor frame in the direction of the line's sub axis. The minimum unit in hardware is equal to 1/65536.

Workaround

There are two workarounds:

1. Set the width and height of the frame to be one pixel larger than the display size.
2. Do a scissor test in the fragment shader when line rendering. If the fragment is outside of the scissor frame, discard it.

E20: back to top

PCIe: Recovery from L1 power savings state as defined by the PCI-Express Base Specification may fail.

Detail

Normally the internal link training and status state machine (LTSSM) transition is 'L1' > 'Recovery' > 'L0' . However, in some cases, the link training operation fails during recovery and the LTSSM transition is then 'L1' > 'Recovery' > 'Detect' instead. The Detect state is the default state and the link is then down, therefore the recognition of the device fails.

Workaround

This problem can be solved by changing the internal configuration of the PCIe macro. The setting extends the CDR lockup time setting and start training sequence during recovery from the L1 power saving state. The new, longer value is 20us.

In PCIe register space write the value:

Write Addr = 0xC84 Data = 0x1010_8a04

E21: back to top

INTERCONNECT BUS: Interconnect port #2 hangup

Detail

If several processing blocks (internal bus masters) have been routed to the memory controller port 2 (by assigning '10' in the Interconnect Config1 ... Config5 registers) and several masters continuously issue frequent READ and WRITE accesses to memory or excessive WRITE accesses (i.e. high bandwidth), it is possible that this memory port will hang.

Workaround

Do not use Port 2 (i.e. program '10' in the Interconnect Config1 ... Config5 registers - bitfields, see below) for the following processing blocks that send WRITE requests to memory:

Register:	**Bitfield:**	**Bitfield Name:**
Config1	[21:20]	PortWb
Config2	[29:28]	PortCap3
	[21:20]	PortCap2
	[13:12]	PortCap1
	[5:4]	PortCap0
Config3	[29:28]	PortPixw
Config4	[31:30]	PortPciew
	[23:22]	PortArgw3
	[15:14]	PortArgw2
	[7:6]	PortArgw1
Config5	[21:20]	PortSPI
	[13:12]	PortCmdw
	[5:4]	PortCmdr

E22: back to top

DISPLAY: Incorrect pixels displayed in blanking when Dithering Unit (DITH) is active

Detail

The DITH values effect both the active pixels and the pixels in the blanking period. They are not set to zero in blanking, instead the last output value is reused. This is a problem if the blanking-level of the display controller is used in any subsequent device for clamping or gain control.

Workaround

1. Set the last pixel of the video lines to black OR
2. The external device must evaluate the data enable signal of the display controller.

E23: back to top

SSCG: High current in power down

Detail

The SSCG current consumption in power down mode can be higher than the specified value and higher than the value in functional mode, which could become a lifetime issue for the device if the power down mode is extensively used.

Workaround:

Avoid using the power down mode. Once the SSCG unit has been activated, keep it running.

E24: back to top

DISPLAY: Per pixel blending not functional for 16bpp and and indexed color format (1bit alpha)

Detail

Blending effect is not observable (alpha_pix is always zero) in all per pixel blending modes (all modes except mode 3) together with 16bpp or indexed color format.

Workaround:

Hint for blending mode 1 (standard alpha blending): Per pixel transparency is possible by using a transparent color for related pixels.

Hint for blending mode 4 (simultaneous alpha): Use constant alpha blending mode with a transparent color for related pixels.

1. Use the alpha layer to define an alpha value per pixel OR
2. Set the color for the pixel that shall not be drawn to a defined color that is then also programmed to the transparent color register value. Enable transparent color mode

E25: back to top
Graphics Core: Hang-up or wrong pixel in Bit Block Transfer with AlphaMap

Detail

If both of the conditions below are used, a hang-up or a wrong blending ratio occurs depending on a hardware internal state.

1. Displaylist command is BltCopyAltAlphaMapP.
2. After the scissor test, a number of pixels for copy meets the below conditions.
   [In the case of 32bit/pixel color] : (2 * 128 * n) +1 (n>=1)
   [In the case of 16bit/pixel color] : (4 * 128 * n) +m (n>=1, m=1,2,3)
   [In the case of 8bit/pixel color] : (8 * 128 * n) +m (n>=1, m=1,2,3,4,5,6,7)

Workaround

Changing the copy size after the scissor test to the size below. The scissor test has to be considered, because the scissor test changes an actual copy size.
[In the case of 32bit/pixel color] : X size or Y size is a multiple of two.
[In the case of 16bit/pixel color] : X size or Y size is a multiple of four.
[In the case of 8bit/pixel color] : X size or Y size is a multiple of eight.

E26: back to top
Graphics Core: Per-pixel blending not functional for 16bpp and indexed color

Detail

Per-pixel blending is not functional for 16bpp and indexed color (1bit alpha). This effects all per-pixel blending modes (except mode 3) . The bug is recognized by the fact that the blending effect is not observable (the alpha_pix is always zero).

Workaround

1. Use the alpha layer to define an alpha value per pixel
2. Set the color for pixels that shall not be drawn to a defined color that is then also programmed in the transparent color register. Enable transparent color mode.

Hint for blending mode 1 (standard alpha blending): Per pixel transparency is possible by using transparent color for related pixels.

Hint for blending mode 4 (simultaneous alpha): Use constant alpha blending mode with the transparent color for related pixels.

Related registers: ('x' = the number of the layer, e.g. LxTC = L1TC, L2TC etc.)
LxBlend
LxTransparencyControl
LxColorMode

E27: back to top

Graphics Core: Wrong 1/W and varying variables of a fragment

Detail

Occurrence conditions:
The condition is related to 1/W and varying variables in device coordinates which are the result coordinates after vertex shader processing, view volume clipping, and a scissor test.

The bug will occur if an exponent underflow occurs and the exponential part of its result is -128 (non-biased representation) during the 3D graphics engine's parameter calculations for the fragment shader input. It can therefore occur depending on the coordinates and parameters of a vertex, but will not occur sporadically if it has never occurred at software runtime. Unless there is a change which influences to device coordinates (eg. vertex coordinates or parameters change, vertex shader program change, etc.).

Phenomenon:
The 1/W and varying variable parameters are incorrect for fragment shader input.

- The 1/W value calculated is the maximum of absolute value.
- The varying parameter calculated is the maximum of absolute value * W.

* The maximum of absolute value means 7F7FFFFFh or FF7FFFFFh in the hex representation of a 32bit floating point value.

Workaround

There are two workarounds as described below:

- The application or vertex shader program can limit the vertex shader output's value of W to satisfy the conditions below.

  W <= 2^{90}

  * 32bits floating point value itself already has a limit of 2^{128}. This limitation is changed to 2^{90} by this workaround.

- Define the condition of the varying variable below as an error indicator, and change the varying variable accordingly to 0.0 using the fragment shader program. The application must exclude values that lead to the conditions shown below:

  Error indicator in FP32 format: |Varying * max(1.0,1/W)| >= 2^{126}
  Error indicator in FP16 format: |Varying * max(1.0,1/W)| >= 2^{14}

  * |x| means the absolute value of 'x'.