Mali-200 Symbian OpenGL ES DDK package (GX920)

Errata Notice

This document contains all errata known at the date of issue in supported releases up to and including revision r2p0 of Mali-200 Symbian OpenGL ES DDK package

Proprietary notice

Document confidentiality status

This document is Non Confidential.

Web address

http://www.arm.com/

Feedback on the product

If you have any comments or suggestions about this product, contact your supplier giving:

- The product name
- A concise explanation of your comments.

Feedback on this document

If you have any comments on about this document, please send email to mailto:errata@arm.com giving:

- The document title
- The documents number
- The page number(s) to which your comments refer
- A concise explanation of your comments

General suggestion for additions and improvements are also welcome.

Contents

Introduction

Scope

This document describes errata categorised by level of severity. Each description includes:

- a unique defect tracking identifier
- the current status of the defect
- where the implementation deviates from the specification and the conditions under which erroneous behavior occurs
- the implications of the erratum with respect to typical applications
- the application and limitations of a 'work-around' where possible

Categorisation of Errata

Errata recorded in this document are split into three levels of severity:

Category 1	Behavior that is impossible to work around and that severely restricts the use of the product in all, or the majority of applications, rendering the device unusable.
Category 2	Behavior that contravenes the specified behavior and that might limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.
Category 3	Behavior that was not the originally intended behavior but should not cause any problems in applications.

Change Control

14 Apr 2010: Changes in Document v2

Page	Status	ID		Cat	Summary
26	New	718599	Shared	Cat 1	Possible deadlock on eglTerminate
27	New	718966	EGL	Cat 1	Rendering to EGL Surfaces larger than the screen cause memory corruption
28	New	720839	EGL	Cat 1	EGL_BUFFER_PRESERVED causing semaphore lock-up
29	New	724497	EGL	Cat 1	Mapped framebuffer is not completely unmapped
30	New	724845	Base	Cat 1	Driver does not handle reset of the cores while the MMU is in Pagefault mode
31	New	724858	EGL	Cat 1	EGL window surface resize OOM issue
32	New	724861	EGL	Cat 1	EGL pixmap readback OOM issue
33	New	724909	Shared	Cat 1	Potential deadlock between direct and deferred surface writes
35	New	724932	Shared	Cat 1	Potential deadlock between multiple framebuilder flush
19	Updated	669324	Base	Cat 1	Missing lock release in an out-of-memory error handler
20	Updated	669325	Base	Cat 1	Incorrect handling of Mali memory allocation requests larger than 2GB
60	New	724507	OpenGLES	Cat 2	glCopyTexSubImage does not support 4444 and 5551 formats
61	New	724512	OpenGLES	Cat 2	GLES 1: Compressed textures do not disable texture unit channels
62	New	724522	ESSL	Cat 2	GLES 2: Internal compiler error on certain combinations of vector constructors and type conversions
63	New	724865	Base	Cat 2	Wait after core reset in device driver may not be long enough
64	New	724870	Base	Cat 2	Mali memory allocation might fail if a specified OS memory region is not a multiple of 256KB
65	New	724910	Base	Cat 2	The Mali job dump system will program the incorrect MMU for pixel processor jobs
66	New	724923	Base	Cat 2	Heap growth not supported by Mali job dump system
49	Updated	668817	OpenGLES	Cat 2	Point spirite sizes corrupted from vertex shader
59	Updated	716744	OpenGLES	Cat 2	Missing SW workaround for HW issue 719236
108	New	724047	Base	Cat 3	Crash when handling spurious MMU page fault interrupts
109	New	724048	EGL	Cat 3	bind-to-texture flags wrong on configs
110	New	724050	EGL	Cat 3	eglGetProcAddress can return wrong client API function pointer
111	New	724496	EGL	Cat 3	eglBindTexImage does not support pbuffer pitch below 64 on HW revisions prior to r0p5
112	New	724528	OpenGLES	Cat 3	Vertex attribute stride queries return actual stride, not specified stride
113	New	724905	OpenGLES	Cat 3	Rebinding of texture objects not handled properly
114	New	724934	OpenGLES	Cat 3	glBindFramebuffer should not early out

115	New	724941	OpenGLES	Cat 3	glBindBuffer should not early out
116	New	733952	Base	Cat 3	The EGL Blitting counter is never updated
68	Updated	589267	OpenGLES	Cat 3	GLES2: Shader linker does not do invariance checks for built-in varyings
78	Updated	602220	ESSL	Cat 3	GLES 2: Shader compilation may fail when the client application sets the locale to non-US
83	Updated	602722	EGL	Cat 3	EGLImage: Missing check for whether pbuffer is bound through eglBindTexImage
84	Updated	611269	OpenGLES	Cat 3	GLES 2: GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING state getters work incorrectly
85	Updated	667718	OpenGLES	Cat 3	Creation of EGL Images from Mipmap level 10 and above disallowed
90	Updated	668418	EGL	Cat 3	Pbuffer surface flush may fail
93	Updated	669319	Base	Cat 3	Architecture specific job start functions not allowed to fail
103	Updated	716557	Base	Cat 3	Too many jobs are dumped when dumping is combined with instrumentation

15 Jun 2009: Changes in Document v1

Page	Status	ID		Cat	Summary
19	New	669324	Base	Cat 1	Missing lock release in an out-of-memory error handler
20	New	669325	Base	Cat 1	Incorrect handling of Mali memory allocation requests larger than 2GB669324
21	New	707868	Base	Cat 1	mali_mmu_get_table_page doesn't unlock the spinlock in all error exits
22	New	708168	Base	Cat 1	Instrumented driver is unable to handle all out of memory situations gracefully
23	New	716164	OpenGLES	Cat 1	Missing workaround for Hardware issue 716163 (GP2 BRESP issue)
24	New	716595	EGL	Cat 1	Resizing surfaces will fail when triggering a switching between direct and non-direct rendering
25	New	716922	Base	Cat 1	Instrumented frame pointer not reference counted
37	New	580918	OpenGLES	Cat 2	GLES 2: Mixing CPU and GPU-based texture updates gives incorrect results
38	New	589867	OpenGLES	Cat 2	GLES 2: Missing SW workaround for HW issue: Unaligned input data to vertex shader may corrupt previous input stream
39	New	596020	OpenGLES	Cat 2	GLES 2: Rendering to FBOs and using the image as a texture can fail on multi-core Mali setups
40	New	609419	EGL	Cat 2	__egl_get_main_context is not thread safe
41	New	609766	Shared	Cat 2	Missing color buffer synchronization
42	New	667667	OpenGLES	Cat 2	GLES2: Mixing deferred and direct writes to a texture causes loss of data
43	New	668069	OpenGLES	Cat 2	GLES2: vec2 varying arrays corrupts every odd element
44	New	668226	OpenGLES	Cat 2	Depth readback for framebuffer objects uses unsafe varying precision mix

45	New	668369	OpenGLES	Cat 2	gl_FragCoord flipped around the y axis inside FBO drawcalls
46	New	668469	EGL	Cat 2	Missing synchronization of color buffers
47	New	668472	EGL	Cat 2	__egl_main_power_event is not thread safe
48	New	668717	OpenGLES	Cat 2	glViewport affects glClear
49	New	668817	OpenGLES	Cat 2	Point spirite sizes corrupted from vertex shader
50	New	668917	Shared	Cat 2	Subpixel Specifier workaround error
51	New	669321	Base	Cat 2	Low memory condition could block applications from initializing the base driver
52	New	669322	Base	Cat 2	Missing file close during base driver shutdown
53	New	707674	Base	Cat 2	Initiate memory bank prevents base driver from starting if the bank can't provide memory
54	New	707718	Base	Cat 2	Timer in arch backend not cleaned up during base core shutdown
55	New	708169	Base	Cat 2	Incorrect frame might be dumped
56	New	716283	OpenGLES	Cat 2	Respecifying renderbuffers attached to a non-current FBO cause dangling pointers
57	New	716300	OpenGLES	Cat 2	Default point size not clamped by glPointParameter
58	New	716556	Base	Cat 2	Incremental rendering gives incorrect output
59	New	716744	OpenGLES	Cat 2	Missing hardware workaround: Points drawn after a zero-size triangle may disappear
68	New	589267	OpenGLES	Cat 3	GLES2: Shader linker does not do invariance checks for built-in varyings
69	New	589268	OpenGLES	Cat 3	glDrawElements is limited to 24bit index count
70	New	589269	OpenGLES	Cat 3	Multisample information may be lost
71	New	589270	OpenGLES	Cat 3	Previously issued draw-calls may be lost when running out of memory
72	New	589868	OpenGLES	Cat 3	Missing SW workaround for HW issue: MaliGP2 does not support normalized inputs
73	New	596018	OpenGLES	Cat 3	GLES 2: Incorrect return values from glGetIntegerv after glBindFramebuffer
117	New	596019	OpenGLES	Cat 3	Creating EGLImages from texture mip level 10 or above results in EGL_BAD_MATCH
74	New	596068	OpenGLES	Cat 3	Non-integer sized points are sometimes rendered non-square
75	New	596069	OpenGLES	Cat 3	Modifying an EGLimage created from a pixmap within a frame can give wrong result
76	New	602218	ESSL	Cat 3	GLES 2: Compilation of a vertex shader may fail with the error "Register allocation failed for vertex shader"
77	New	602219	ESSL	Cat 3	GLES 2: Shader compiler preprocessor accepts some illegal combinations of preprocessor directives
78	New	602220	ESSL	Cat 3	GLES 2: Shader compilation may fail when the client application sets the locale to non-US

79	New	602370	ESSL	Cat 3	GLES 2: Failing to write gl_Position is reported as a shader compilation error
80	New	602374	ESSL	Cat 3	GLES 2: Shader compiler extension macros only defined for enabled extensions
81	New	602375	ESSL	Cat 3	GLES 2: Function overloading based on array sizes not supported
82	New	602721	EGL	Cat 3	Zero-sized window surfaces not supported
83	New	602722	EGL	Cat 3	EGLImage: Missing check for whether pbuffer is bound through eglBindTexImage
84	New	611269	OpenGLES	Cat 3	GLES 2: GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING state getters work incorrectly
85	New	667718	OpenGLES	Cat 3	Creation of EGL Images from Mipmap level 10 and above disallowed
86	New	667868	OpenGLES	Cat 3	glStencilMask and glColorMask values differing from TRUE or FALSE will crash the driver
87	New	667968	OpenGLES	Cat 3	GLES2: Drawing to FBOs with bad vertex pointers may cause the driver to hang
88	New	668218	OpenGLES	Cat 3	Instrumented GLES driver frame dumping off-by-one frame
89	New	668221	OpenGLES	Cat 3	glCompressedTexImage2D Out of Memory handling segfaults
90	New	668418	EGL	Cat 3	Pbuffer surface flush may fail
91	New	668471	EGL	Cat 3	eglQueryString does not return error for name -1
92	New	668473	EGL	Cat 3	eglChooseConfig doesn't report EGL_BAD_ATTRIBUTE when attribute values are out of range
93	New	669319	Base	Cat 3	Architecture specific job start functions not allowed to fail
94	New	669323	Base	Cat 3	Incorrect memory location written to during client API negotiation sequence
95	New	669333	Base	Cat 3	Invalid XML produced when large amount of counters is sampled
96	New	708022	Base	Cat 3	Leaked external memory is not cleaned up during process shut down
97	New	708074	Base	Cat 3	Incorrect IRQ mask used during startup and shutdown
98	New	708118	Base	Cat 3	Global mutex not freed
99	New	716139	OpenGLES	Cat 3	Deleting framebuffer only reports last error
100	New	716160	OpenGLES	Cat 3	Successful relinking of current program does not update current rendering state
101	New	716281	OpenGLES	Cat 3	Readback state leaks resulting in lost color channels
102	New	716302	OpenGLES	Cat 3	flushing with max_pp_split_count > 1 may lead to assert in debug mode or memleak in release mode
103	New	716557	Base	Cat 3	Too many jobs are dumped when dumping is combined with instrumentation
104	New	716559	Base	Cat 3	Same fragment shader stack used by two or more pixel processors during instrumentation

105	New	716593	EGL	Cat 3	Pre-existing background in pixmap not copied during surface creation
106	New	716748	OpenGLES	Cat 3	Retrieving large integer uniform values may lack precision
107	New	716972	OpenGLES	Cat 3	Missing Workaround: EGL Pixmap pitch requirements

Errata Summary Table

The errata associated with this product affect product versions as below.

A cell shown thus **X** indicates that the defect affects the revision shown at the top of that column.

ID		Cat	Summary of Erratum	r0p2	r1p0	r1p1
724932	Shared	Cat 1	Potential deadlock between multiple framebuilder flush		X	
724909	Shared	Cat 1	Potential deadlock between direct and deferred surface writes		X	
724861	EGL	Cat 1	EGL pixmap readback OOM issue		X	
724858	EGL	Cat 1	EGL window surface resize OOM issue	X	X	
724845	Base	Cat 1	Driver does not handle reset of the cores while the MMU is in Pagefault mode	X	X	
724497	EGL	Cat 1	Mapped framebuffer is not completely unmapped	X	X	
720839	EGL	Cat 1	EGL_BUFFER_PRESERVED causing semaphore lock-up		X	
718966	EGL	Cat 1	Rendering to EGL Surfaces larger than the screen cause memory corruption	X		
718599	Shared	Cat 1	Possible deadlock on eglTerminate	X		
716922	Base	Cat 1	Instrumented frame pointer not reference counted	X		
716595	EGL	Cat 1	Resizing surfaces will fail when triggering a switching between direct and non-direct rendering	X		
716164	OpenGLES	Cat 1	Missing workaround for Hardware issue 716163 (GP2 BRESP issue)	X		
708168	Base	Cat 1	Instrumented driver is unable to handle all out of memory situations gracefully	X		
707868	Base	Cat 1	mali_mmu_get_table_page doesn't unlock the spinlock in all error exits	X		
669325	Base	Cat 1	Incorrect handling of Mali memory allocation requests larger than 2GB	X		
669324	Base	Cat 1	Missing lock release in an out-of-memory error handler	X		
724923	Base	Cat 2	Heap growth not supported by Mali job dump system		X	X
724910	Base	Cat 2	The Mali job dump system will program the incorrect MMU for pixel processor jobs		X	
724870	Base	Cat 2	Mali memory allocation might fail if a specified OS memory region is not a multiple of 256KB	X	X	
724865	Base	Cat 2	Wait after core reset in device driver may not be long enough	X	X	
724522	ESSL	Cat 2	GLES 2: Internal compiler error on certain combinations of vector constructors and type conversions		X	

ID		Cat	Summary of Erratum	r0p2	r1p0	r1p1
724512	OpenGLES	Cat 2	GLES 1: Compressed textures do not disable texture unit channels	X	X	
724507	OpenGLES	Cat 2	glCopyTexSubImage does not support 4444 and 5551 formats	X	X	
716744	OpenGLES	Cat 2	Missing SW workaround for HW issue 719236	X	X	
716556	Base	Cat 2	Incremental rendering gives incorrect output	X		
716300	OpenGLES	Cat 2	Default point size not clamped by glPointParameter	X		
716283	OpenGLES	Cat 2	Respecifying renderbuffers attached to a non-current FBO cause dangling pointers	X		
708169	Base	Cat 2	Incorrect frame might be dumped	X		
707718	Base	Cat 2	Timer in arch backend not cleaned up during base core shutdown	X		
707674	Base	Cat 2	Initiate memory bank prevents base driver from starting if the bank can't provide memory	X		
669322	Base	Cat 2	Missing file close during base driver shutdown	X		
669321	Base	Cat 2	Low memory condition could block applications from initializing the base driver	X		
668917	Shared	Cat 2	Subpixel Specifier workaround error	X		
668817	OpenGLES	Cat 2	Point spirite sizes corrupted from vertex shader	X		
668717	OpenGLES	Cat 2	glViewport affects glClear	X		
668472	EGL	Cat 2	__egl_main_power_event is not thread safe	X		
668469	EGL	Cat 2	Missing synchronization of color buffers	X		
668369	OpenGLES	Cat 2	gl_FragCoord flipped around the y axis inside FBO drawcalls	X		
668226	OpenGLES	Cat 2	Depth readback for framebuffer objects uses unsafe varying precision mix	X		
668069	OpenGLES	Cat 2	GLES2: vec2 varying arrays corrupts every odd element	X		
667667	OpenGLES	Cat 2	GLES2: Mixing deferred and direct writes to a texture causes loss of data	X		
609766	Shared	Cat 2	Missing color buffer synchronization	X		
609419	EGL	Cat 2	__egl_get_main_context is not thread safe	X		
596020	OpenGLES	Cat 2	GLES 2: Rendering to FBOs and using the image as a texture can fail on multi-core Mali setups	X		
589867	OpenGLES	Cat 2	GLES 2: Missing SW workaround for HW issue: Unaligned input data to vertex shader may corrupt previous input stream	X	X	X
580918	OpenGLES	Cat 2	GLES 2: Mixing CPU and GPU-based texture updates gives incorrect results	X		
733952	Base	Cat 3	The EGL Blitting counter is never updated	X		

ID		Cat	Summary of Erratum	r0p2	r1p0	r1p1
724941	OpenGLES	Cat 3	glBindBuffer should not early out	X	X	X
724934	OpenGLES	Cat 3	glBindFramebuffer should not early out	X	X	X
724905	OpenGLES	Cat 3	Rebinding of texture objects not handled properly	X	X	
724528	OpenGLES	Cat 3	Vertex attribute stride queries return actual stride, not specified stride	X	X	X
724496	EGL	Cat 3	eglBindTexImage does not support pbuffer pitch below 64 on HW revisions prior to r0p5			X
724050	EGL	Cat 3	eglGetProcAddress can return wrong client API function pointer	X	X	
724048	EGL	Cat 3	bind-to-texture flags wrong on configs	X	X	
724047	Base	Cat 3	Crash when handling spurious MMU page fault interrupts	X	X	
716972	OpenGLES	Cat 3	Missing Workaround: EGL Pixmap pitch requirements	X	X	X
716748	OpenGLES	Cat 3	Retrieving large integer uniform values may lack precision	X	X	X
716593	EGL	Cat 3	Pre-existing background in pixmap not copied during surface creation	X		
716559	Base	Cat 3	Same fragment shader stack used by two or more pixel processors during instrumentation	X		
716557	Base	Cat 3	Too many jobs are dumped when dumping is combined with instrumentation		X	
716302	OpenGLES	Cat 3	flushing with max_pp_split_count > 1 may lead to assert in debug mode or memleak in release mode	X		
716281	OpenGLES	Cat 3	Readback state leaks resulting in lost color channels	X		
716160	OpenGLES	Cat 3	Successful relinking of current program does not update current rendering state	X		
716139	OpenGLES	Cat 3	Deleting framebuffer objects only reports last error	X		
708118	Base	Cat 3	Global mutex not freed	X		
708074	Base	Cat 3	Incorrect IRQ mask used during startup and shutdown	X		
708022	Base	Cat 3	Leaked external memory is not cleaned up during process shut down	X		
669333	Base	Cat 3	Invalid XML produced when large amount of counters is sampled	X		
669323	Base	Cat 3	Incorrect memory location written to during client API negotiation sequence	X		
669319	Base	Cat 3	Architecture specific job start functions not allowed to fail	X		
668473	EGL	Cat 3	eglChooseConfig doesn't report EGL_BAD_ATTRIBUTE when attribute values are out of range	X		
668471	EGL	Cat 3	eglQueryString does not return error for name -1	X		
668418	EGL	Cat 3	Pbuffer surface flush may fail	X		

ID		Cat	Summary of Erratum	r0p2	r1p0	r1p1
668221	OpenGLES	Cat 3	glCompressedTexImage2D Out of Memory handling segfaults	X		
668218	OpenGLES	Cat 3	Instrumented GLES driver frame dumping off-by-one frame	X		
667968	OpenGLES	Cat 3	GLES2: Drawing to FBOs with bad vertex pointers may cause the driver to hang	X		
667868	OpenGLES	Cat 3	glStencilMask and glColorMask values differing from TRUE or FALSE will crash the driver	X		
667718	OpenGLES	Cat 3	Creation of EGL Images from Mipmap level 10 and above disallowed	X	X	X
611269	OpenGLES	Cat 3	GLES 2: GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING state getters work incorrectly	X		
602722	EGL	Cat 3	EGLImage: Missing check for whether pbuffer is bound through eglBindTexImage	X	X	
602721	EGL	Cat 3	Zero-sized window surfaces not supported	X		
602375	ESSL	Cat 3	GLES 2: Function overloading based on array sizes not supported	X		
602374	ESSL	Cat 3	GLES 2: Shader compiler extension macros only defined for enabled extensions	X		
602370	ESSL	Cat 3	GLES 2: Failing to write gl_Position is reported as a shader compilation error	X		
602220	ESSL	Cat 3	GLES 2: Shader compilation may fail when the client application sets the locale to non-US	X	X	X
602219	ESSL	Cat 3	GLES 2: Shader compiler preprocessor accepts some illegal combinations of preprocessor directives	X	X	X
602218	ESSL	Cat 3	GLES 2: Compilation of a vertex shader may fail with the error "Register allocation failed for vertex shader"	X	X	X
596069	OpenGLES	Cat 3	Modifying an EGLimage created from a pixmap within a frame can give wrong result	X	X	X
596068	OpenGLES	Cat 3	Non-integer sized points are sometimes rendered non-square	X	X	X
596018	OpenGLES	Cat 3	GLES 2: Incorrect return values from glGetIntegerv after glBindFramebuffer	X		
589868	OpenGLES	Cat 3	Missing SW workaround for HW issue: MaliGP2 does not support normalized inputs	X	X	X
589270	OpenGLES	Cat 3	Previously issued draw-calls may be lost when running out of memory	X	X	X
589269	OpenGLES	Cat 3	Multisample information may be lost	X		
589268	OpenGLES	Cat 3	glDrawElements is limited to 24bit index count	X	X	X
589267	OpenGLES	Cat 3	GLES2: Shader linker does not do invariance checks for built-in varyings	X	X	X

Errata - Category 1

669324: Missing lock release in an out-of-memory error handler

Status

Affects:	Base
Fault status:	Cat 1, Present in: r0p2, Fixed in r1p0.

Description

The MMU page table cache uses a spinlock to protect access to the cache. In one of the cache maintenance operations where the spinlock is held the lock is not released correctly in one out-of-memory error handlers.

Implications

If this specific out of memory error handler is triggered all later page table cache operations will block.

Workaround

None.

669325: Incorrect handling of Mali memory allocation requests larger than 2GB

Status

Affects:	Base
Fault status:	Cat 1, Present in: r0p2, Fixed in r1p0.

Description

The handler for Mali memory allocation requests when the Mali MMU is disabled calculates the next power-of-two from the requested allocation size.

If the requested size is above 2GB the calculation overflows a 32 bit integer. This integer is used to control the calculation loop causing the loop to continue forever.

Implications

If an allocation request larger than 2GB is used the calling process will become stuck in an infinite loop

Workaround

None.

707868: mali_mmu_get_table_page doesn't unlock the spinlock in all error exits

Status

Affects:	Base
Fault status:	Cat 1, Present in: r0p2, Fixed in r1p0.

Description

mali_mmu_get_table_page returns without unlocking a spinlock in some cases where memory allocation fails.

Implications

A deadlock inside kernel space.

Workaround

Don't use the Mali MMU.

708168: Instrumented driver is unable to handle all out of memory situations gracefully

Status

Affects:	Base
Fault status:	Cat 1, Present in: r0p2, Fixed in r1p0.

Description

Instrumented driver is unable to handle all out of memory situations gracefully.

Implications

The application could either crash, hang or leak memory when instrumentation is enabled.

Workaround

Ensure that the platform has enough available system memory when running with instrumented driver.

716164: Missing workaround for Hardware issue 716163 (GP2 BRESP issue)

Status

Affects:	OpenGLES
Fault status:	Cat 1, Present in: r0p2, Fixed in r1p0.

Description

MaliGP2 does not wait for AXI BRESP bus signals after writing to the bus. This may cause subsequent reads from the same destination to read incomplete data. For GLES, this means that vertices may be processed incorrectly on platforms with high latency.

This errata entry notes that there is no driver workaround for this HW issue.

Implications

Visual rendering errors.

Workaround

none

716595: Resizing surfaces will fail when triggering a switching between direct and non-direct rendering

Status

Affects:	EGL
Fault status:	Cat 1, Present in: r0p2, Fixed in r1p0.

Description

Resizing a surface so that its rendering mode switches between direct rendering and non-direct rendering will cause application to terminate.

Implications

If a direct rendered surface is resized to an area which is no longer compatible with direct rendering, the application will fail. This also happens the other way around, ie. resizing from non-compatible direct rendered surface to compatible. A typical scenario where this happen is at the bounds of the display resolution. A direct rendered surface can be no larger than the physical display resolution.

Workaround

No known workaround other than making sure that the render method does not change.

716922: Instrumented frame pointer not reference counted

Status

Affects:	Base
Fault status:	Cat 1, Present in: r0p2, Fixed in r1p0.

Description

The instrumented version of the driver keep track of instrumented values per frame in a structure allocated on the heap. There are situations where multiple rendering jobs will try to use the same structure, for example with frame buffer objects in OpenGL ES or when the driver is trying to recover from an error situation. The problem is that this structure is not reference counted, thus it can be freed to early.

Implications

The application will try to access freed memory. This can cause the application to produce incorrect output, deadlock, crash or behave unexpectedly in other ways.

Workaround

none

718599: Possible deadlock on eglTerminate

Status

Affects:	Shared
Fault status:	Cat 1, Present in: r0p2, Fixed in r1p0.

Description

If Mali GP2 returns an error during rendering the rendering job is considered a failure, and the driver must recover from the error and set an appropriate API error. In normal operations, such errors may happen due to Mali GP2 running out of memory or the hardware or software watchdog timer timing out.

But if a call to eglTerminate occurs after a failing GP job has been started but before it has returned with an error, the error recovery code will try to access memory that may have been deleted. This will result in undefined behavior and quite likely a deadlock. This bug occurs since the lock designed to prevent this from happening is not being held long enough.

Implications

Calling eglTerminate may cause the application to hang.

Workaround

718966: Rendering to EGL Surfaces larger than the screen cause memory corruption

Status

Affects:	EGL
Fault status:	Cat 1, Present in: r0p2, Fixed in r1p0.

Description

When rendering to an EGL Surface larger than the physical display, the driver need to render to an offscreen buffer and copy the relevant subsection of the result to the EGL Surface. If the EGL Surface bitdepth is matching the fbdev setup on the platform, this copy pass happens on a scanline-by-scanline basis.

There is an issue in the code that does this copy scanline-by-scanline pass. The code calculates the target scanline destination wrongly, multiplying the pitch on the output image by two for 16bpp surfaces and four for 32bit surfaces. This means that every other or every three out of four horisontal lines on the output will be undefined, and the copy will write framebuffer data far beyond the memory allocated for the EGL surface, resulting in random memory corruption.

Implications

If the conditions of the bug is met, any output from the hardware will be copied scanline-by-scanline onto the screen, but only at every second or fourth scanline depending on bpp settings. This will result in erroneous output with every other or three out of four scanlines undefined (typically black), and the copypass writing far beyond the allocated memory. This will result in random memory corruption.

Workaround

There is no workaround if you trigger the conditions.

720839: EGL_BUFFER_PRESERVED causing semaphore lock-up

Status

Affects:	EGL
Fault status:	Cat 1, Present in: r1p0, Fixed in r1p1.

Description

Enabling preserved swap buffer behavior can cause a driver lock.

Implications

A deadlock can occur when using preserved swap behavior in combination with direct rendering. The deadlock can occur after calling eglSwapBuffers on such a surface.

Workaround

Use other means of preserving the color buffer. This can be accomplished by client API readback functions, or by rendering to textures.

724497: Mapped framebuffer is not completely unmapped

Status

Affects:	EGL
Fault status:	Cat 1, Present in: r0p2,r1p0, Fixed in r1p1.

Description

The framebuffer device is memory mapped upon initialization. The size of memory unmapped at termination time does not correspond to the memory mapped size.

Implications

You can run out of virtual memory address space when initializing and terminating the framebuffer device a large number of times.

Workaround

Close the display at application termination time.

724845: Driver does not handle reset of the cores while the MMU is in Pagefault mode

Status

Affects:	Base
Fault status:	Cat 1, Present in: r0p2,r1p0, Fixed in r1p1.

Description

The Mali device driver is not able to handle both a Mali MMU page fault and core reset at the same time if that process has a Mali job running on a core. This can typically happen if a process is forcefully terminated by the operating system, like when pressing CTRL-C. The reason for this is that during program termination, the operating system removes the memory from the Mali MMU while a job might still be running, and then reset the cores. The errata will trigger if the Mali MMU page fault is present but not handled before core reset is completed.

Implications

The affected Mali core(s) will not be working, a system reboot is needed to resolve the issue.

Workaround

None.

724858: EGL window surface resize OOM issue

Status

Affects:	EGL
Fault status:	Cat 1, Present in: r0p2,r1p0, Fixed in r1p1.

Description

An out of memory situation when resizing a window surface can leave the driver in a state where a crash is possible.

Implications

When resizing a windowed surface, the old resources should be kept until new resources has been successfully allocated. However, in some situations, the old resources are released before new resources have been successfully allocated.

This can lead to a driver crash later on, when operations are performed on released resources.

Workaround

Other than disabling window surface resize, none.

724861: EGL pixmap readback OOM issue

Status

Affects:	EGL
Fault status:	Cat 1, Present in: r1p0, Fixed in r1p1.

Description

When rendering to a native pixmap, EGL needs to read in the pixmap data. An out of memory situation during this read back process can potentially lead to a memory leak or driver crash.

Implications

Depending on where the actual out of memory situation occurs, you might have a memory leak or a driver crash. The memory leak is caused by missing memory cleanup, while the driver crash is caused by operations on released memory after the read back process has finished.

Workaround

None.

724909: Potential deadlock between direct and deferred surface writes

Status

Affects:	Shared
Fault status:	Cat 1, Present in: r1p0, Fixed in r1p1.

Description

deferred drawcall is a write operation happening through the Mali core. Any draw operation done through glDrawArrays and glDrawElements qualifies as a deferred write to the output surface.

A direct write is any surface draw operation happening immediately (and thus without the aid of Mali). A good example is a call to glTexSubImage2D, but any operation directly modifying the surface pixel values qualifies.

When drawing to a surface, the driver locks the surface - either to modify the pixel data (direct), or to notify the surface that there is an outstanding write in the pipeline (deferred).

A deferred draw will lock down all the surfaces the drawcall in question is rendering to. If you are writing to an FBO with a depth and color buffer, then both the depth and the color surface will be locked in a predetermined order. Attempts to flush the FBO will also lock down the same two buffers in the same predetermined order to ensure that nothing else has written to the surfaces in the meantime.

A direct draw will lock down the surface being written to. If there is an outstanding deferred write to that surface, the deferred write operation will be flushed/finished prior to the direct write taking place.

This is a potential deadlock situation, as direct writes first locks the direct surface, then flush outstanding writes (which may lock other surfaces). The combined set of surface locks in this operation do not happen in the safe predetermined order, and is as such potentially unsafe. The gles driver design prevents most any case from actually taking place, but the EGL Image extension omits many of the safety guards. The following example is still possible:

```
1: Thread/Context A creates a depth renderbuffer (surface 1) 2: Thread/Context A
also creates a color texture/renderbuffer (surface 2)

3: Thread/Context A sets up an FBO to render to these two. 4: The color surface
(2) is shared through an EGL Image

5: Thread/Context B binds up the EGL image (2) as a new texture in its context

6: Thread/Context A adds a deferred write job the FBO (glDraw)

  - Both surfaces are marked with an outstanding write. 7: Thread/Context B
attempts a direct write to the color surface (1).   - This will lock surface (2)

  - Surface 1 has an outstanding write, which will be attempted flushed

  - This will lock the surfaces in the FBO (1, 2), except for 2 which is already
locked.   - The flush will thus only lock surface (1).

  - Locking order of the entire operation is: (2, 1)

8: At the same time, Thread/Context A adds another deferred write job to both
surfaces.   - This will lock both surfaces in a predetermined order (1, 2)
```

Implications

It is possible through the use of the EGL Image extension, to lock a set of surfaces in a different order in two threads at the same time. This will deadlock the driver.

Workaround

Instead of sharing the surface through an EGL image, share the surface through a GLES context sharing. Another option is to stick to only deferred write operations.

724932: Potential deadlock between multiple framebuilder flush

Status

Affects:	Shared
Fault status:	Cat 1, Present in: r1p0, Fixed in r1p1.

Description

A flush may happen as a result of a call to glFlush/glFinish/eglSwapBuffers, or as an implicit action when needing the content of a surface. The flush will lock down the framebuilder, then lock all surfaces the framebuilder is rendering to. In that order.

A surface write will lock down the surfaces, then lock down the framebuilder. Notably, in the opposite order. This is a potential deadlock which can be triggered by the following chain of commands:

```
1: Thread/Context A draws to a texture through an FBO. 2: An EGL Image is made
from the texture.

3: Thread/Context B creates a new texture from the EGL Image. 4: Thread/Context B
does a direct write to the texture.  - This locks the surface (1)

 - The surface has an outstanding draw operation on it

 - The draw operation is flushed by flushing the FBO in Thread/Context A

 - This flush attempts to lock the frame (2)

5. At the same time, Thread/Context A does a glFlush.  - This will attemt to lock
the frame (1) then the surface (2)
```

Implications

Different types of flushes of the same framebuilder at the same time may deadlock the driver.

Workaround

By not sharing surfaces through EGL images (use the gles context sharing) or at least not putting deferred drawcalls on EGL Image surfaces will eliminate the possibility of this deadlock.

Errata - Category 2

580918: GLES 2: Mixing CPU and GPU-based texture updates gives incorrect results

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

Mixing different methods of updating a texture gives incorrect result. The different methods are:

- OpenGL ES texture update functions: glTexImage2D, glCopyTexImage2D, glTexSubImage2D, and glCopyTexSubImage2D.

- Render to texture through Framebuffer Objects (FBOs).

- Render to texture through EGLimage.

- Updating pixmaps that GLES has created textures from that are bound to a framebuffer object. (EGLimage)

- Updating pixmaps that GLES has created renderbuffers from. (EGLimage)

If an application mixes update methods for the same texture, some of the modifications may be lost.

Implications

Some of the texture modifications are lost.

Workaround

Perform all modifications to the contents of a texture using only one method.

589867: GLES 2: Missing SW workaround for HW issue: Unaligned input data to vertex shader may corrupt previous input stream

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2,r1p0,r1p1, Open.

Description

MaliGP2 might output corrupt data if vertex shader input data is crossing 128 bit boundaries. GX525 HW errata, defect 475765: Vertex shader data crossing 128 bit boundaries may corrupt other data, states:

"The number of usable input data stream may be reduced to any number between 16 and 8, depending on the exact data formats and alignments used. Some APIs may query the number of available streams without specifying the stream parameters, in which case only the worst case of 8 streams may be exposed.

Alternatively, if padding the input streams to avoid the line crossing issue, performance will be affected. The padding will increase the bandwidth required to read the data stream, and the software driver may need to copy the data in order to insert padding."

No software workaround for this issue has been implemented.

Implications

Incorrect rendering.

Workaround

The issue will only occur when input data streams are not aligned to 128 bit boundaries. This will only happen for 3-component and packed input streams. A workaround is therefore to use only non-packed 1, 2, or 4 component input data streams.

596020: GLES 2: Rendering to FBOs and using the image as a texture can fail on multi-core Mali setups

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

On multi-core Mali configurations, rendering to an FBO with a texture attachment and subsequently using the same texture in draw-calls can cause an intermediate result to be used for texturing.

Implications

A mixture of old and new texture content is returned from the texture-sampler.

Workaround

Call glFinish before calling glBindFramebuffer when switching from the problematic framebuffer object.

609419: __egl_get_main_context is not thread safe

Status

Affects:	EGL
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

There is a bug in the internal state handling in EGL leading to thread safety issues.

The following conditions has to be met in order to trigger this:

* No EGL API functions has been called after application was started

* Two simultaneous EGL API calls from concurrent threads are executed

The result will be a partly initialized internal state, which can lead to ASSERT or segmentation fault.

Implications

When two threads calls an EGL API function for the first time, simultaneously, you can get a partly initialized internal state in one of the threads, leading to an ASSERT or segmentation fault.

Workaround

Avoid doing the first call to EGL simultaneously in separate threads.

An example of how to avoid this would be to place a

```
eglGetError()
```

at the very start of application, before executing any concurrent threads.

609766: Missing color buffer synchronization

Status

Affects:	Shared
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

A bug in the color buffer state management can cause artifacts when direct rendering is enabled. Rendering can possibly be done into the currently visible buffer because the buffer is considered to be non-visible.

Note that direct rendering is not enabled by default.

Implications

The rendering can be performed to a currently visible buffer.

Workaround

Disable direct rendering (this is default).

667667: GLES2: Mixing deferred and direct writes to a texture causes loss of data

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

There is no safeguard in glTexSubImage2D preventing it from updating a texture currently in use as a rendertarget by the hardware. This can result in direct write operations (like glTexSubImage2D) being applied prematurely compared to an outstanding deferred write operation (like glDrawArrays), resulting in the effective loss of the direct write operation.

Implications

The bug can cause the application to lose calls to glTexSubImage2D. The call will happen, but if there is an outstanding deferred write job writing to the texture this will not necessarily be finished before the call to glTexSubImage2D happens. Effectively, the texture will be updated again by the hardware, losing the results of the call to glTexSubImage2D.

Workaround

Calling glFinish before calling glBindFramebuffer to change away from the FBO with outstanding deferred writes, will cause the FBO in question to be flushed. Since glFinish will not return until this flush is completed, this will ensure that the subsequent calls to glTexSubImage2D is not made on a texture currently being written to by the hardware. The workaround is however not particularly efficient as it disables any chance of making the FBO render work in parallel with subsequent setup code. Like this.

```
glBindFramebuffer(1);
render_something();
glFinish();
glBindFramebuffer(0);
glTexSubImage2D();
```

Also consider manually double-buffering the renderable textures.

668069: GLES2: vec2 varying arrays corrupts every odd element

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

Shaders using varying vec2 arrays will lose every odd-numbered element in the transfer process between the vertex and the fragment shader.

Implications

This issue will cause odd-numbered array elements to contain nothing but zero values on the fragment shader side, regardless of what output the vertex shader produce.

Workaround

Instead of using a vec2 array, use a vec4 arary or simply create a set of standalone variables mimicing the array.

Instead of doing:

```
varying vec2 foo[4];
```

Do:

```
varying vec2 foo0;
varying vec2 foo1;
varying vec2 foo2;
varying vec2 foo3;
```

668226: Depth readback for framebuffer objects uses unsafe varying precision mix

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

When triggering a depth buffer readback, the depth values gets corrupted in the 8 most significant bits. Depth readbacks happen every time the application flush out a framebuffer object and do not clear the buffer afterwards.

Implications

Should a readback occur, the depth buffer read back into the tiles will be corrupted, leading to rendering errors.

Workaround

There is no direct workaround to the problem, but a readback is always avoidable. Ensure that the application clears all buffers it use before rendering begins. Do not modify the depth buffer texture from outside the current rendering job (f.ex through glTexSubImage2D call).

668369: gl_FragCoord flipped around the y axis inside FBO drawcalls

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

When using a fragment shader accessing gl_FragCoord inside a drawcall rendering to an FBO, the y coordinate will be flipped around the y axis. This means that y=0 will be at the top of the screen and y=height will be at the bottom of the screen, unlike what is required in GLES.

Implications

Fragment shaders used in drawcalls rendering to an FBO that also use gl_Fragcoord will get an erroneous y component, leading to rendering errors.

Workaround

Applications will have to manually flip the entries around by recalculating real_y = height - y.

668469: Missing synchronization of color buffers

Status

Affects:	EGL
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

When rendering directly to color buffers visible on screen, you may see parts of the next frame into the currently visible color buffer. This is due to a slight delay between recycling of color buffers and panning of display, resulting into partial rendering into the currently visible buffer, before it is panned.

Note that this only affects direct rendering, not cases where you are utilizing blitting.

Implications

Currently visible color buffer may be rendered into, giving partial output from next frame into current frame.

Workaround

This issue can be worked around by implementing a synchronization method in the platform layer. Locks on color buffers and native vsync are possible options.

668472: __egl_main_power_event is not thread safe

Status

Affects:	EGL
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

__egl_platform_power_event is a platform specific function which can be called at any time. It is routed to _egl_main_power_event, which invalidates EGL contexts. Mutex protection is missing, making it possible to get into a state with undefined behavior.

Implications

__egl_main_power_event is called asynchronously, and it will invalidate all EGL context handles. This is done without any mutex protection, and it is possible that a context might be referenced in another thread at the same time, leading to undefined behavior.

Workaround

Implement locking mechanisms in the platform specific layer, or make sure that no concurrent calls are made to EGL when a power event occurs.

668717: glViewport affects glClear

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

A call to glClear is supposed to clear the specified buffer, only limited by the scissorbox. Your viewport settings is not supposed to limit the area cleared, but it currently does.

Implications

The bug may lead clear calls to clear less than the intended area. This will cause parts of the screen to contain leftover data from previous frames in either buffer, which will easily lead to visual errors.

Workaround

Set the viewport to the full screen dimensions before clearing.

668817: Point spirite sizes corrupted from vertex shader

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

When writing to gl_PointSize in a vertex shader, the data written is overflowing, causing corruption on the subsequent point sizes as well as other streams in the shader. This only happens in glDrawElements calls.

Implications

Pointsize outputs from the vertex shader will be corrupted and corrupt other outputs if used in a glDrawElements call.

Workaround

Instead of using glDrawElements to draw point sprites, use glDrawArrays.

668917: Subpixel Specifier workaround error

Status

Affects:	Shared
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

The subpixel specifier in a hardware setting specifying subpixel precision. This register is unfortunately set wrong, which could lead to missing pixels.

Implications

It is possible in rare cases to lose geometry and pixels. Refer to HW defect 499914 for details.

Workaround

none

669321: Low memory condition could block applications from initializing the base driver

Status

Affects:	Base
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

The base driver preallocates memory during startup. An allocation is performed against each underlying memory provider.

If one of the memory providers have run completely out of memory the reallocation will fail.

The base driver incorrectly handles that as a general out of memory case causing it to cancel the initialization.

Implications

The base driver fails to initialize even if memory is available

Workaround

Don't expose a small and at the same time suggested primary memory resource to the base driver

669322: Missing file close during base driver shutdown

Status

Affects:	Base
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

A file descriptor is opened by the base driver as part of its startup.

During shutdown this file is not closed correctly causing the file descriptor to leak.

As systems can be configured to limit the number of open files the limit might be hit because of the leaks.

Implications

If enough base driver reinitialization sequences occur opening of files might fail in the host process depending on system settings.

Workaround

Disable the file descriptor limit on the system

or

Do not load and unload the base driver on demand but instead keep a single instance around for all clients.

707674: Initiate memory bank prevents base driver from starting if the bank can't provide memory

Status

Affects:	Base
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

Initialization of memory banks during start-up will try to preallocate 1MB of RAM from each bank. If the bank is not able to provide this, then the entire base initialization process will fail.

Implications

Initialization of the Mali base layer will fail if there is not at least 1MB of free RAM in each configured memory bank.

Workaround

Make sure there is at least 1MB of free RAM in each memory bank for each running process using Mali. Disable memory banks which are to small.

707718: Timer in arch backend not cleaned up during base core shutdown

Status

Affects:	Base
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

The arch specific timer created in the function arch_initialize is never deleted.

Implications

A resource leak of a arch_timer for every initialization and termination of the base system within a single process. Resource will however be freed when process terminates.

Workaround

Avoid multiple initialization and terminations the base system multiple times in a single application to minimize the memory leak.

708169: Incorrect frame might be dumped

Status

Affects:	Base
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

The frame buffer dumps the previous render target of the color attachment for the frame. This is not always correct, as the next frame might complete before we are able to do the dumping, and thus the render targets will be swapped. This is only possible on a Mali with more than one pixel processor.

Implications

The frame buffer for frame N+1 might be dumped instead of frame buffer for frame N.

Workaround

Disable direct rendering or only use a single pixel processor setup.

716283: Respecifying renderbuffers attached to a non-current FBO cause dangling pointers

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

When respecifying a renderbuffer (though glRenderbufferStorage) the video memory mapped by the renderbuffer is replaced. The old memory is deref'ed and the new memory is addref'ed. When attaching a renderbuffer to an FBO, the FBO addrefs the renderbuffer object, not the memory it is wrapping directly. The currently bound FBO will also addref the memory areas it is rendering to to protect its rendering surfaces from vanishing while rendering to them.

This scheme is not waterproof, and will cause dangling pointers in the FBOs when re-specifying renderbuffers bound to FBOs not currently bound.

A sequence that will trigger this bug:

```
glBindRenderbuffer( mybuffer );
glRenderbufferStorage( ... );
glBindFramebuffer( fbo, mybuffer );
glFramebufferRenderbuffer( fbo, mybuffer);
glBindFramebuffer( fbo, 0 );
glRenderbufferStorage( ... );
glBindFramebuffer( fbo, mybuffer );  // fbo now contains dangling pointers
glDraw( ... ); // this will access the dangling pointer, typically segfaulting.
```

Implications

Respecifying buffers without adhering to the workaround will cause the hardware to access dangling pointers.

Workaround

Only respecify renderbuffers while keeping the FBO they are bound to as current. If the renderbuffer is bound to multiple FBOs, it must be detached prior to respecification.

716300: Default point size not clamped by glPointParameter

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

When drawing GL_POINTS with the point size attribute disabled, the point size is not clamped to the min/max size values specified by glPointParameter.

Implications

Point sprites drawn with given size instead of clamped size, resulting in rendering errors.

Workaround

Clamp point size manually when calling glPointSize.

716556: Incremental rendering gives incorrect output

Status

Affects:	Base
Fault status:	Cat 2, Present in: r0p2, Fixed in r1p0.

Description

When more than 2 pixel processor performance counters are enabled, the driver will execute each pixel processor job several times. The render targets (color buffer, Z-buffer and stencil buffer) is not preserved and later restored between each run. This might cause jobs doing incremental rendering (read back from the write back buffer) to render incorrectly.

Implications

Incorrect rendering.

Workaround

Do not sample more than two hardware performance counters from the pixel processor at the same time.

716744: Missing SW workaround for HW issue 719236

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2,r1p0, Fixed in r1p1.

Description

Hardware issue 719216 identifies a hardware issue where after issuing a drawcall drawing one or more zero-sized triangles, subsequent drawcalls drawing points may disappear.

This errata tracks that there is no automatic workaround for this issue, meaning the applications will need to work around it manually.

Implications

Points drawn after a drawcall which draws zero-sized triangles may disappear, causing rendering errors.

Workaround

Draw a fullscreen transparent quad across the entire screen inbetween drawing triangles at the risk of being zero-sized and points.

724507: glCopyTexSubImage does not support 4444 and 5551 formats

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2,r1p0, Fixed in r1p1.

Description

Doing a glCopyTexSubImage into a texture of format RGBA4444 or RGBA5551 will fail. For debug builds, this will trigger an assert crash. Release build drivers will ignore the glCopyTexSubImage command and return a texture filled with garbage.

Implications

Using glCopyTexSubImage2D on textures with these formats is not possible.

Workaround

Use a different texture format, render to the texture using an FBO (GLES2 only), or call glReadpixels to retrieve the current framebuffer, convert the retrieved buffer to your required format, and re-upload the data with glTexImage2D.

724512: GLES 1: Compressed textures do not disable texture unit channels

Status

Affects:	OpenGLES
Fault status:	Cat 2, Present in: r0p2,r1p0, Fixed in r1p1.

Description

The OpenGLES 1.1 specification states that textures units should only do its operation on channels present in the texture bound to the texture unit in question. Channels in this context refer to "RGB" or "Alpha".

For example, a texture unit set to a GL_REPLACE operation should replace the texture unit input with the texture bound. But if that bound texture doesn't have an alpha channel, the alpha channel is not replaced. Similarly, if the texture doesn't have an RGB channel, the RGB input of the texture unit is not replaced. This is how texture units operate on all uncompressed texture formats.

However, all compressed texture formats are being treated as having all channels present, regardless of texture format. For example, The ETC1_RGB_888 format would for example be treated as a texture with an alpha channel, despite that not being the case. As a result, if this texture was bound to a texture unit, it would also replace the alpha channel.

Implications

OpenGL ES 1.1 applications depending on the texture unit channel disabling working correctly (when using compressed textures) will see erroneous output.

Workaround

Instead of using the default combiner modes, use the GL_COMBINE combiner mode. This mode allows the application developer to directly control the operation of each channel in the texture unit, including the ability to disable them.

Example:

Instead of setting up a texture unit with mode=GL_REPLACE (like this):

<code>

glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);

</code>

Set it up with a mode=GL_COMBINE (like this):

```
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_COMBINE);
glTexEnvi(GL_TEXTURE_ENV, GL_COMBINE_RGB, GL_REPLACE);
glTexEnvi(GL_TEXTURE_ENV, GL_COMBINE_ALPHA, GL_REPLACE);
glTexEnvi(GL_TEXTURE_ENV, GL_OPERAND0_RGB, GL_SRC_COLOR);
glTexEnvi(GL_TEXTURE_ENV, GL_OPERAND0_ALPHA, GL_SRC_ALPHA);
glTexEnvi(GL_TEXTURE_ENV, GL_SRC0_RGB, GL_TEXTURE);
glTexEnvi(GL_TEXTURE_ENV, GL_SRC0_ALPHA, GL_TEXTURE);
```

If it is required to disable a specific channel, do so explicitly by replacing the relevant channel source GL_TEXTURE with GL_PREVIOUS.

724522: GLES 2: Internal compiler error on certain combinations of vector constructors and type conversions

Status

Affects:	ESSL
Fault status:	Cat 2, Present in: r1p0, Fixed in r1p1.

Description

In fragment shaders, certain combinations of vector constructors and type conversions where a value is used more than once and some components of the result are never used, may cause the ESSL compiler to report an internal compiler error.

For instance, the following fragment shader triggers the error:

```
precision mediump float;
varying vec2 v;
uniform sampler2D sam;
void main()
{
        ivec3 a = ivec3(texture2D(sam, v));
        a.xz.y = 2;
        gl_FragColor = vec4(bvec4(true, false, a).b);
}
```

The ESSL compiler will exit gracefully and will not crash or silently generate invalid code.

Implications

Certain rare combinations of ESSL language constructs may cause the compiler to report an error for valid code.

Workaround

A safe way to avoid triggering the internal compiler error is to never construct vectors where only some of the components are ever used. Any shader which violates this principle can easily be changed to adhere to it with no change in functionality.

724865: Wait after core reset in device driver may not be long enough

Status

Affects:	Base
Fault status:	Cat 2, Present in: r0p2,r1p0, Fixed in r1p1.

Description

The device driver does a fixed number of Mali register reads in order to wait for the a Mali core to be ready after a reset. In some systems, this might not be long enough, and the device driver will start using the core before it has completed the reset.

Implications

The setup of the Mali core after reset will not be correctly executed, and unexpected behavior might happen. The core will most likely be rendered unusable, and applications trying to use Mali will hang.

Workaround

This errata can be fixed by replacing the fixed number of dummy register reads with a small for loop after issuing the MALIGP2_REG_VAL_CMD_RESET to the Mali GP management command register and after issuing MALI200_REG_VAL_CTRL_MGMT_FORCE_RESET to the Mali PP management command register.

For the Mali geometry processor, do the following changes:

1) Write the value 0xC0FFE000 to the register MALIGP2_REG_ADDR_MGMT_WRITE_BOUND_LOW right before issuing the MALIGP2_REG_VAL_CMD_RESET command.

2) Replace the dummy register reads after issuing the MALIGP2_REG_VAL_CMD_RESET with a for loop doing a maximum of 15 iterations.

3) In the for loop; write the value 0xC01A0000 to the MALIGP2_REG_ADDR_MGMT_WRITE_BOUND_LOW register, and read the same register afterward. If you get the same value (0xC01A0000), then you can exit the for loop and continue.

4) After the for loop, write the value 0x00000000 to the MALIGP2_REG_ADDR_MGMT_WRITE_BOUND_LOW register.

Do the same for the Mali pixel processor code, except use the MALI200_REG_ADDR_MGMT_WRITE_BOUNDARY_LOW register instead of the MALIGP2_REG_ADDR_MGMT_WRITE_BOUND_LOW register.

724870: Mali memory allocation might fail if a specified OS memory region is not a multiple of 256KB

Status

Affects:	Base
Fault status:	Cat 2, Present in: r0p2,r1p0, Fixed in r1p1.

Description

The documentation fails to mention a limitation for the OS_MEMORY type that can be specified in the config.h file used to build the Mali device driver. The size field must be a multiple of 256KB.

The support for OS_MEMORY has been dropped for the r1p1 release.

Implications

If the OS_MEMORY section size isn't a multiple of 256KB and there is a MEMORY section defined after it and there is an allocation request that cannot be fully fulfilled by the incorrectly configured OS_MEMORY section, then that allocation will fail, even though there are free memory to fulfill the request.

Workaround

Make sure any OS_MEMORY section size is a multiple of 256KB and/or make sure the OS_MEMORY section is defined last in the config.h file.

724910: The Mali job dump system will program the incorrect MMU for pixel processor jobs

Status

Affects:	Base
Fault status:	Cat 2, Present in: r1p0, Fixed in r1p1.

Description

When the MMU programming steps for a Mali pixel processor job are written into the config.txt file, then the addresses of the MMU for the Mali geometry processor is used, instead of the addresses of the MMU for the Mali pixel processor.

Implications

Playback of dump cannot be done with MMU support.

Workaround

Manually edit the config.txt file and adjust the MMU register addresses to match the one of the MMU for the pixel processor desired, or don't use MMU during playback.

724923: Heap growth not supported by Mali job dump system

Status

Affects:	Base
Fault status:	Cat 2, Present in: r1p0,r1p1, Open.

Description

New MMU tables is not dumped when dumping a Mali geometry processor which is resumed after being given more heap memory.

Implications

The dump might contain an incomplete MMU table, and thus cause page faults during playback.

Workaround

Don't use MMU support during dumping or playback.

Errata - Category 3

589267: GLES2: Shader linker does not do invariance checks for built-in varyings

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

Section 4.6.4 of the OpenGL ES 2.0 specification, Invariance and Linkage, requires invariance checking between the built-in shader varyings gl_FragCoord, gl_Position, gl_PointCoord, gl_PointSize, and gl_FrontFacing. This is not being done by the linker. Instead, all varyings are being treated as if they were invariant

Implications

This defect has no implications for valid programs.

Workaround

Do not declare any of the built-in varyings invariant.

589268: glDrawElements is limited to 24bit index count

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

glDrawElements does not support an index count larger that 2^24-1.

Implications

The index count in glDrawElements will be limited to 2^24-1, meaning that indices will be discarded if the count is above this value.

Workaround

If an index count of 2^24 or more is needed, the call to glDrawElements should be split into several calls with index count less than 2^24.

589269: Multisample information may be lost

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

Calling glDrawElements or glDrawArrays followed by a glReadPixels, glFlush or glFinish more than 100 times, without calling glClear, will cause the driver to lose multisample information.

Implications

Multisample buffers will not be preserved when glDrawElements or glDrawArrays followed by a glReadPixels, glFlush or glFinish are called more than 100 times in succession.

Workaround

none

589270: Previously issued draw-calls may be lost when running out of memory

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

If more than 100 GL calls that flush the pipeline (glReadPixels, glFlush and glFinish) are interleaved with drawing calls (this can be any combination of glDrawElements, glDrawArrays, and glClear without all buffers bits or with scissoring enabled) are issued without any calls to glClear (with all buffer bits set and scissoring disabled) or eglSwapBuffers in between, and the driver runs out of memory, the result of previously issued drawing calls can be lost.

Implications

Previously rendered content is lost.

Workaround

If you accept the loss of depth, stencil and multisample information you can manually trigger a "incremental render" by doing the following:

glCopyTexImage to capture the color buffer content

glClear all buffers

glDraw the captured texture back to the screen

589868: Missing SW workaround for HW issue: MaliGP2 does not support normalized inputs

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

The vertex loader in MaliGP2 (r0p4 and earlier) does not support normalized inputs. GX525 HW errata, defect 509016: MaliGP2 does not support normalized inputs, states:

"The OpenGL specification allows the user to specify some values in normalized format, where the input is interpreted as a value in the range [0, 1] or [-1, 1]. MaliGP2 does not support these formats directly, making it perform worse than otherwise possible for these kinds of inputs."

No software workaround has been implemented for this issue.

Implications

Normalized vertex attribute data is divided by the maximum value of the data-type plus one, and not the maximum value as the standard specifies. This means that for normalized vertex attributes, all values will be slightly smaller than they should, and it is impossible to represent the exact value of 1.0.

Workaround

Do not use GL_UNSIGNED_BYTE on normalized vertex attribute data, or normalize data before use. In OpenGL ES 2.0, data can be normalized in the vertex shader by adding this to the shader:

```
attribute float byte_attrib;
// your code
void main()
{
// your code
float corrected_value;
corrected_value = (byte_attrib * 256)/255;
}
```

596018: GLES 2: Incorrect return values from glGetIntegerv after glBindFramebuffer

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

If glGetInteger is called with the pname parameter GL_STENCIL_BITS, GL_DEPTH_BITS, GL_RED_BITS, GL_GREEN_BITS, GL_BLUE_BITS, or GL_ALPHA_BITS and no draw-calls have been made since the previous call to glBindFramebuffer, the value 0 is incorrectly returned instead of the expected value.

Implications

Wrong color/depth/stencil bit values returned when an FBO is bound.

Workaround

Draw a single, off-screen point between the call to glBindFramebuffer and the call to glGetIntegerv.

596068: Non-integer sized points are sometimes rendered non-square

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

Under some conditions, points with non-integer size can be rasterized non-square by Mali, and this is not worked around by the OpenGL ES driver.

Implications

Points with non-integer size can be rasterized as non-square.

Workaround

Round the point-size to an integer in the vertex-shader in OpenGL ES 2.0.

No workaround is known for OpenGL ES 1.x.

596069: Modifying an EGLimage created from a pixmap within a frame can give wrong result

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

If a texture is created from an EGLimage that was created from a pixmap and that texture is used for rendering and then modified in the same frame, the rendering will use the modified texture instead of the original.

Implications

The modified texture is used for draw-calls that happened before the modification took place.

Workaround

Manually double-buffer the EGL Images.

602218: GLES 2: Compilation of a vertex shader may fail with the error "Register allocation failed for vertex shader"

Status

Affects:	ESSL
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

MaliGP2 has only limited internal bandwidth between its registers and execution units. In some rare cases, the register allocator for MaliGP2 in the shader compiler runs into a situation where there are more operations executed in one cycle than can be fed from the registers simultaneously. The compiler aborts the compilation in this case.

Implications

Some very big or very complex vertex shaders fail to compile.

Workaround

Changing the shader code slightly will often eliminate the problem. Try to rewrite some of the shader to do the calculations differently.

602219: GLES 2: Shader compiler preprocessor accepts some illegal combinations of preprocessor directives

Status

Affects:	ESSL
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

The ESSL compiler may in some cases accept shaders containing illegal combinations of #if / #ifdef / #ifndef / #elif / #else / #endif as valid.

Implications

This defect has no implications for valid programs.

Workaround

No workaround needed.

602220: GLES 2: Shader compilation may fail when the client application sets the locale to non-US

Status

Affects:	ESSL
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

The ESSL compiler uses strtod() for parsing floating point literals. Therefore, if the online compiler is used with an OpenGL ES application and the application either sets the locale explicitly to a locale with different numerical conventions than the US with e.g.

setlocale(LC_NUMERIC, "nb_NO");

or picks up the current locale from environment variables with e.g.

setlocale(LC_NUMERIC, "");

and the environment variables are set in a way that indicates a locale with adifferent numerical convention than the US, the parser will fail with errors such as

0:28: L0001: Error while parsing floating point literal '0.0'

This is because the strtod() function now is looking for literals with a different decimal separator.

Implications

Online compilation of valid shaders using floating-point literals may fail.

Workaround

Either use the off-line compiler and load shader binaries, or surround each call to glCompileShader with setlocale calls, e.g.

char *prev_locale = setlocale(LC_NUMERIC, "C"); /* reset numeric locale to default,

save the old locale */

glCompileShader(...);

setlocale(LC_NUMERIC, prev_locale); /* restore the old locale */

602370: GLES 2: Failing to write gl_Position is reported as a shader compilation error

Status

Affects:	ESSL
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

If the gl_Position built-in variable is not written anywhere in a vertex shader, the compiler reports an error. This error should only be a warning, since this (useless) situation is not explicitly forbidden by the language specification.

Implications

No implications for any useful shader code.

Workaround

No workaround needed.

602374: GLES 2: Shader compiler extension macros only defined for enabled extensions

Status

Affects:	ESSL
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

For each language extension supported by an implementation, a macro with the extension name is automatically defined by the compiler to indicate that the extension is supported. However, the compiler erroneously defines these macros only for extensions that are explicitly enabled.

Implications

Only code that is written to compile on several different implementations supporting different extensions are affected. In such code, it is not possible to explicitly test for support of an extension and conditionally compile different code depending on whether it is supported or not.

Workaround

Unconditionally enable all extensions attempted used in the shader program.

602375: GLES 2: Function overloading based on array sizes not supported

Status

Affects:	ESSL
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

The ESSL language allows functions to be overloaded based on array sizes in their parameter types, i.e. one can have two functions with identical names and identical parameter types except for the sizes of array types. The compiler treats functions with same parameter types except for array sizes as having the same signature.

Implications

Shaders containing functions with identical names and identical parameter names except for array sizes are rejected by the compiler. For instance, compilation of the program

```
vec4 foo(float a[2])
{
    return vec4(0.0, 1.0, 0.0, 0.0);
}
vec4 foo(float a[3])
{
    return vec4(1.0, 0.0, 0.0, 0.0);
}
void main()
{
    float a[2];
    gl_Position = foo(a);
}
```

will fail with the error message

```
0:9: S0023: Function 'foo' redefined
```

Workaround

Change the function signatures to use different names.

602721: Zero-sized window surfaces not supported

Status

Affects:	EGL
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

EGL does not support creating zero sized window surfaces or resizing existing window surfaces to zero size.

Implications

Failure to create or resize a window surface with either height or width equal to zero.

Workaround

Clamp window size to 1x1.

602722: EGLImage: Missing check for whether pbuffer is bound through eglBindTexImage

Status

Affects:	EGL
Fault status:	Cat 3, Present in: r0p2,r1p0, Fixed in r1p1.

Description

eglCreateImageKHR is missing a check for whether a pbuffer is bound through eglBindTexImage, and will therefore succeed even if the pbuffer is bound. The specification states that this should not be allowed.

Implications

Creation of EGLImage will succeed, even if the supplied pbuffer is bound through eglBindTexImage, violating the specification. No other impact.

Workaround

Unbind the pbuffer using eglReleaseTexImage before creating an EGLImage.

611269: GLES 2: GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING state getters work incorrectly

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

Calling glGetVertexAttribiv with index n and pname GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING is supposed to return the buffer binding for vertex attribute n. The OpenGL ES driver generates a GL_INVALID_ENUM instead. Similarly, the GL_INVALID_ENUM error should be generated when calling glGetIntegeriv with pname GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING, but instead the driver returns a constant value of 0.

Implications

No value returned from getter and wrong error set for glGetVertexAttribiv. Wrong value returned and wrong error set for glGetIntegeriv.

Workaround

None.

667718: Creation of EGL Images from Mipmap level 10 and above disallowed

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

Mali requires the driver to pack the data in mipmap level 10+ together into one memory allocation. The driver does this for normal textures as required, but this repacking is currently colliding with the requirement in EGL Images to preserve the texture memory area.

Implications

There is no way of creating an EGL Image from a GLES texture, mipmap level 10 and above.

Workaround

none

667868: glStencilMask and glColorMask values differing from TRUE or FALSE will crash the driver

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

The getters for setting color mask or depth mask in GLES are GLboolean values and are as such defined to be converted on input to either 0 or 1. This does not happen, and the value is sent unchecked directly to GLES. Debug build versions of the GLES driver will assert that the mask value are either 0 or 1, causing it to crash the application if this is not the case. Drivers built in release mode will simply pass the least significant bit of the input value directly to the hardware as a boolean flag.

Implications

Applications that deliberately pass non-boolean values to the two entry points in question can cause the GLES driver to crash in debug mode, and behave incorrectly in release mode.

Workaround

Do not pass other values than GL_TRUE or GL_FALSE to glColorMask or glDepthMask.

667968: GLES2: Drawing to FBOs with bad vertex pointers may cause the driver to hang

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

If a drawcall that renders to an FBO fails due to your vertex attribute pointers being illegal (NULL or bound to a non-specified VBO), the function will early out with a GL_ILLEGAL_OPERATION. Unfortunately, this may result in the FBO not being unlocked properly, causing the next drawcall to this FBO to hang.

Implications

For an application that does not pass illegal vertex pointers to a draw call, this issue will never occur.

Workaround

Make sure that the vertex attribute pointers are always non-NULL and that they are always bound to specified vertex buffer objects if applicable.

668218: Instrumented GLES driver frame dumping off-by-one frame

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

GLES drivers with instrumented support and frame dumping enabled will dump the previous frame instead of the current frame. This causes off-by-one frame error in the dumped result compared to what appears on the screen.

Implications

When calling eglSwapBuffers on an instrumented driver with framedumping enabled, the dumped frame will be equal to the previous frame shown on the screen, not the new frame produced by eglSwapBuffers.

Workaround

When comparing outputs dumped with the instrumented driver against reference images, use the next output dumped instead of the current outputted frame.

668221: glCompressedTexImage2D Out of Memory handling segfaults

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

If running out of system memory while calling glCompressedTexImage2D with a palette format, the driver may end up using the pointer it failed to allocate. This will lead to segfaults in the driver.

Implications

The issue only happens when palette compressed textures are used.

Applications using palette compressed textures will risk suffering a segfault error if running out of system memory. Typically, the application will run out of platform memory before that, leaving this as a very low risk issue, but in platform setups with a unified memory pool and where the Mali device driver is configured to use a very significant part of the memory pool available, there may still be a theoretical possibility to trigger this bug.

Workaround

Unpack the texture data in software, upload to GLES as a non-compressed texture.

668418: Pbuffer surface flush may fail

Status

Affects:	EGL
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

When switching between APIs and contexts using `eglBindAPI` and `eglMakeCurrent`, certain conditions can lead to failing flush of pbuffer surfaces. Note that this only affects pbuffers created with `eglCreatePbufferFromClientBuffer`

The following scenario will fail:

- bind to vg
- make a pbuffer created from a client buffer current
- bind to gles
- bind to vg
- make another (any kind of) surface current

The last step will implicitly flush the pbuffer surface into the vg image, but will fail in this condition. Workaround is to call `eglMakeCurrent( display, EGL_NO_SURFACE, EGL_NO_SURFACE, EGL_NO_CONTEXT );` before switching to another API.

Implications

A pbuffer surface is flushed when made not current. This flushing may fail to complete.

Workaround

Call `eglMakeCurrent( display, EGL_NO_SURFACE, EGL_NO_SURFACE, EGL_NO_CONTEXT );` when switching between APIs using eglBindAPI while having a pbuffer surface current.

668471: eglQueryString does not return error for name -1

Status

Affects:	EGL
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

If -1 is given as name for the call to eglQueryString you will receive a string with build information. This build information contains settings used during build of the various drivers.

Implications

Internal build info string returned instead of NULL. Error state set to EGL_SUCCESS instead of EGL_BAD_PARAMETER.

Workaround

none

668473: eglChooseConfig doesn't report EGL_BAD_ATTRIBUTE when attribute values are out of range

Status

Affects:	EGL
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

According to EGL specification, an EGL_BAD_ATTRIBUTE error should be generated if the specified attribute value is not within the supported range. This error will not be generated, instead EGL will look for any configurations matching the given attribute values. In the end this will result in zero matched configs along with the EGL_SUCCESS error.

Implications

EGL_BAD_ATTRIBUTE not reported for attribute values out of range. This will not affect the returned list of configs, since the attribute values will not give any matches for any configs if they are out of range.

Workaround

Call eglChooseConfig with valid attribute values.

669319: Architecture specific job start functions not allowed to fail

Status

Affects:	Base
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

The architecture backend functions used to start jobs on the hardware returns a boolean value stating if the job was started or not. The interface specification states that the 'not started' value means that the hardware is busy and that a callback will be given once the hardware becomes idle and the call should be retried. The interface does not allow the backend to return a value indicating internal failure.

Implications

If the architecture implementation encounters internal problems which causes it to return the status 'not started' and it never calls the job completion callback function jobs will become stuck on the job queue.

Workaround

Design the job start routines to not depend on routines which might fail or call the job completion callback routine in case of an internal failure stating that the job has failed.

669323: Incorrect memory location written to during client API negotiation sequence

Status

Affects:	Base
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

When a session towards the Mali device driver is established, an API negotiation sequence is performed. When the version is agreed upon the API version is supposed to be stored in the session object. But instead of writing to the session object a pointer to a different object is used.

Implications

The API version number is incorrectly written into a mutex object. The mutex still works as long as it does not become congested. For it to be congested kernel preemption must be enabled. Kernel preemption is not enabled in the supported kernel configuration. The DDK has not been tested with this kernel preemption and other issues than this might show up if kernel preemption is enabled.

Workaround

Disable kernel preemption.

669333: Invalid XML produced when large amount of counters is sampled

Status

Affects:	Base
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

Data sampled by the instrumented driver is stored in an XML file. The header of the XML file contains information about which counters the file has sample data for. Internally this header is created in a temporary buffer of a fixed size. Writes to this buffer use snprintf which limits the amount of bytes to write to the buffer size.

Implications

If the number of counters exceed what the temporary buffer can hold invalid XML is produced.

Workaround

Sample the counters in two or more separate runs and merge the resulting XML files.

708022: Leaked external memory is not cleaned up during process shut down

Status

Affects:	Base
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

External memory used by the Mali device driver is not freed during process termination.

Implications

External memory allocated through the _mali_mem_add_phys_mem function is not freed in the device driver when the process terminates without explicitly freeing that memory by calling _mali_mem_free.

Workaround

Make sure that external memory allocated with the _mali_mem_add_phys_mem function is freed (by calling _mali_mem_free) before the process terminates.

708074: Incorrect IRQ mask used during startup and shutdown

Status

Affects:	Base
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

All Mali cores are reset at device driver initialization and termination without setting the IRQ correctly. This can lead to unhandled IRQs during loading and/or unloading of the driver if IRQ sharing is used.

Implications

This defect can result in an unhandled IRQ. For the supported reference platforms, this will only result in output into the dmesg log. The implications are unknown for customer setup with their own IRQ handlers installed before and/or after the Mali IRQ handler.

Workaround

Use IRQ probing, if possible.

708118: Global mutex not freed

Status

Affects:	Base
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

A global mutex used to serialize GP jobs when running with instrumentation is not freed when the Mali driver is unloaded.

Implications

For each time the Mali driver is unloaded, a mutex will not be freed. These should however all be freed by the operating system when the application terminates.

Workaround

Only load the Mali driver once per application in order to avoid accumulating leakage.

716139: Deleting framebuffer objects only reports last error

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

When deleting a framebuffer object through a glDeleteFramebuffers call, the framebuffer object will be flushed prior to deletion. This flush stands the risk of running out of memory, but in such cases, this out-of-memory error will be stored in the GL state and is retrievable with glGetError().

However, when deleting multiple framebuffer objects in one go, only the error from the last flushed framebuffer object will be stored in the gles error state. In a call which deletes two framebuffer objects, where the first runs out of memory and the second is deleted without issues, no error code will be set and the application is unable to detect whether all flushes happened as expected.

Implications

The application is unable to detect potential errors while flushing the framebuffer objects being deleted.

Workaround

Delete framebuffer objects one at a time to ensure that errors are set properly.

716160: Successful relinking of current program does not update current rendering state

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

When using a GLES 2.0 program object, subsequent drawcalls will utilize its shaders for vertex and fragment processing (hereby referred to as its "program rendering state"). You can modify the bound program object after linking, with no impact on later drawcalls - the drawcalls will only refer to the "program rendering state", not the status of the program object itself.

But calling glLinkProgram will generate a new "program rendering state". Should the program object you re-link be bound to the current GLES context, one of two things can happen.

If the linking failed, the context's old program rendering state is retained, and you can keep on drawing. But with the old link-result, as if a re-link never took place.

If the linking succeeded, the context's old program rendering state is replaced by the new one, and you can keep on drawing. This time with the new link result.

The latter part unfortunately does not happen in the Mali GLES driver, and a re-binding of the current program must happen manually.

Implications

After relinking the currently bound program, subsequent drawcalls will keep rendering as if the relinking never took place.

Workaround

Manually call glUseProgram to re-bind the current program if you re-link it.

716281: Readback state leaks resulting in lost color channels

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

A readback occurs when Mali reads the output buffer back into the tilelists. This may happen by mixing glDraw and immediate rendertarget updates (like glTexSubImage2D), or enable multiple FBOs to render to the same surface.

Readbacks use the current GL writemask state for the readback call. This means if f.ex the red channel is disabled prior to flushing, the readback will not read back the red channel.

A typical sequence triggering this bug

writemask=RGBA

glDraw

writemask=GBA

glDraw

flush

readback

The readback here will skip the red channel output of the first drawcall.

Implications

Rendering errors, specifically loss of color channels when triggering readbacks.

Workaround

Enable all color channels with glColorMask before flushing.

716302: flushing with max_pp_split_count > 1 may lead to assert in debug mode or memleak in release mode

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

This bug only affects drivers build with max_pp_split_count > 1.

When running out of memory in a flush, the driver allocates memory to handle one job per split count. In case this fails due to out of memory, the flush will be aborted with an out of memory error message. But the abort handler fails to free the jobs, which will cause the driver to leak memory in release builds, and cause the abort handler to assert crash because the job has not been freed in debug builds.

Implications

As a result of running out of memory, release builds may leak memory, debug builds may assertcrash.

Workaround

No workaround available. Don't run out of memory.

716557: Too many jobs are dumped when dumping is combined with instrumentation

Status

Affects:	Base
Fault status:	Cat 3, Present in: r1p0, Fixed in r1p1.

Description

Enabling dumping while more than two pixel processor performance counters are enabled causes the same job to be dumped multiple times (once for every two enabled counters).

Implications

The same pixel processor job will be dumped several times.

Workaround

Do not enable dumping and instrumentation of pixel processor jobs at the same time.

716559: Same fragment shader stack used by two or more pixel processors during instrumentation

Status

Affects:	Base
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

When more than two pixel processor performance counters are enabled, the same pixel processor job is executed several times with the same fragment shader stack. On systems with more than one pixel processor (Mali-200 r0p1 test chip), this can cause two jobs to use the same stack at the same time. This can lead to random rendering artifacts.

Implications

Rendering artifacts.

Workaround

Do not enable more than two pixel processor performance counters at the same time or use a single pixel processor setup.

716593: Pre-existing background in pixmap not copied during surface creation

Status

Affects:	EGL
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

When creating a new EGL pixmap surface, the native pixmap content is not copied into the EGL surface.

Implications

Initial EGL surface will be empty, and does not contain the pixmap data.

Workaround

The contents of a native pixmap can be synchronized with the EGL surface at any time by calling `eglWaitNative`. This copies the content of the native pixmap into the EGL surface.

716748: Retrieving large integer uniform values may lack precision

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

The GLES driver stores all uniforms as FP32 floats. This includes integer uniforms, which are transformed to FP32 floating point values when specified by glUniform. This conversion is within legal bounds as far as GLES is concerned, and Mali require all integers to be floating point values anyway. But it does create an issue for the uniform getters as the getters will access a value with less precision than the one originally set.

Implications

Integer uniforms retrieved with glGetUniform will differ from the uniform value set with glUniform.

Workaround

Track integer uniforms on the application side instead of retrieving them with glGetUniform

716972: Missing Workaround: EGL Pixmap pitch requirements

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

EGL Pixmaps are specified with dimensions and bpp per color channel. The pitch is implicitly calculated as width * bpp. Mali has some HW requirements for the pitch if it is to write to your pixmap, and EGL will refuse to create Pixmap Surfaces and EGL Images from EGL pixmaps not adhering to these requirements.

For HW Revision R0P1 the pitch must be divisible by 64. Refer to HW issue 509019. For HW Revision <= R0P4 the width of the surface must be divisible by 16. For all HW Revisions, the pitch must be divisible by 8.

Implications

Can not create EGL Images from pixmaps that do not adhere to the pitch requirements

Workaround

EGL Pixmaps must be created large enough to adhere to the requirements.

724047: Crash when handling spurious MMU page fault interrupts

Status

Affects:	Base
Fault status:	Cat 3, Present in: r0p2,r1p0, Fixed in r1p1.

Description

The Mali MMU page fault handling will try dereference a NULL pointer in order to access a mutex if a spurious page fault interrupt is received. That is, a page fault when there are no active cores behind the Mali MMU. This should not happen unless there is some HW malfunction.

Implications

A NULL pointer will be dereferenced in kernel space, and we will get a system crash.

Workaround

None.

724048: bind-to-texture flags wrong on configs

Status

Affects:	EGL
Fault status:	Cat 3, Present in: r0p2,r1p0, Fixed in r1p1.

Description

In the list of configs in EGL, all the GLES configs have both EGL_BIND_TO_TEXTURE_RGB and EGL_BIND_TO_TEXTURE_RGBA set to EGL_TRUE. However, an RGB config can not support binding to an RGBA texture, because there is no alpha channel available.

Implications

EGL claims support for binding an RGB surface to RGBA texture, which is not allowed.

Workaround

Do not bind an RGB surface to an RGBA texture. Use a surface that supports alpha.

724050: eglGetProcAddress can return wrong client API function pointer

Status

Affects:	EGL
Fault status:	Cat 3, Present in: r0p2,r1p0, Fixed in r1p1.

Description

eglGetProcAddress is used to retrieve client API extension function pointers. In some cases, EGL can return a function pointer which is invalid for the client API version to be used.

The following conditions has to be met in order for this to happen:

* Both OpenGL ES 1.x and OpenGL ES 2.x has to be available from within EGL

* eglGetProcAddress has to be queried for the glEGLImageTargetTexture2DOES extension pointer

If the following two conditions are met, then the returned extension pointer will be not be usable for OpenGL ES 2.x, since the OpenGL ES 1.x extension entrypoint is returned. glEGLImageTargetTexture2DOES is the only extension that is affected by this.

Implications

The returned extension pointer will not be valid for OpenGL ES 2.x. It will result in a NOP, since the two libraries can't be active at the same time from within EGL.

Workaround

One option is to have separate libraries for EGL/OpenGL ES 1.x and EGL/OpenGL ES 2.x. Another is to load the symbol of the extension manually by using for example libdl.

724496: eglBindTexImage does not support pbuffer pitch below 64 on HW revisions prior to r0p5

Status

Affects:	EGL
Fault status:	Cat 3, Present in: r1p1, Open.

Description

Hardware revisions prior to r0p5 does not support linear textures with a pitch below 64. The pbuffer surface used with eglBindTexImage has to have a pitch of at least 64. 16x16 with 32bpp is allowed, while 16x16 with 16bpp is not.

Implications

eglBindTexImage will fail with the error EGL_BAD_MATCH for surfaces with a pitch below 64.

Workaround

Use a larger pbuffer surface as eglBindTexImage source, and scale your texture coordinates accordingly when using the surface as a texture.

724528: Vertex attribute stride queries return actual stride, not specified stride

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

When specifying an attribute stride as 0, the GLES driver will calculate the actual minimum stride based on datatype and component count. Retrieving the stride through glGet should however return the number specified when setting up the stream, not the calculated stride.

The gles driver erroneously returns the actual stride, and not the specified stride.

Implications

The following sequence should set "value" to 0, but is setting it to 16 instead.

glVertexPointer(4, GL_FLOAT, 0, vertices);

glGetIntegerv(GL_VERTEX_ARRAY_STRIDE, &value);

Workaround

Don't query the driver for the attribute stride, unless the actual stride is an acceptable return value for your application. Applications truly needing to retrieve the number 0 in this case should track this value manually.

724905: Rebinding of texture objects not handled properly

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0, Fixed in r1p1.

Description

When binding a texture target to an object already bound, the driver early out of the bind operation as it technically doesn't do anything. This is however not correct. When objects are deleted in a scenario with multiple contexts, the object is only deleted in the context where the delete operation took place. Other contexts may still have the texture object bound, but the object is no longer present in the object list.

A bind operation will need to check the object list, and should there be no object present, create a new one. In the scenario with multiple contexts, "rebinding" an object may thus result in a new object being created, as illustrated by this example:

1. Context A does BindTexture(GL_TEXTURE_2D, 1);
2. Context A does TexImage2D(GL_TEXTURE_2D, ...)
3. Context B does DeleteTextures(1, {1});
4. Context A does BindTexture(GL_TEXTURE_2D, 1);

The GLES spec will require step 4 to check the object list, notice that that the object list is empty and create a new texture object. The driver is earlying out of that operation, resulting in the old deleted texture object still being bound.

Implications

Bound texture objects may have a life-time longer than what is dictated by the GLES specification. Applications relying on the texture being a new empty texture after step 4 in the example above will work on false assumptions, which might in the worst case lead to the wrong (old) texture being used.

Workaround

Bind to texture object 0 before binding to the required texture object if this specific behavior is required. The "early out" will then not take place.

724934: glBindFramebuffer should not early out

Status

Affects: OpenGLES

Fault status: Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

When binding the framebuffer target to an object already bound, the driver early out of the bind operation as it technically doesn't do anything. This is however not correct. When objects are deleted in a scenario with multiple contexts, the object is only deleted in the context where the delete operation took place. Other contexts may still have the framebuffer object bound, but the object is no longer present in the object list.

A bind operation will need to check the object list, and should there be no object present, create a new one. In the scenario with multiple contexts, "rebinding" an object may thus result in a new object being created, as illustrated by this example:

1. Context A does BindFramebuffer(GL_FRAMEBUFFER, 1);
2. Context A does FramebufferTexture2D(GL_FRAMEBUFFER, ..., <sometexture>)
3. Context B does DeleteFramebuffer(1, {1});
4. Context A does BindFramebuffer(GL_FRAMEBUFFER, 1);

The GLES spec will require step 4 to check the object list, notice that that the object list is empty and create a new framebuffer object. The driver is earlying out of that operation, resulting in the old deleted framebuffer object still being bound.

Implications

Bound framebuffer objects may have a life-time longer than what is dictated by the GLES specification. Applications relying on the framebuffer object being a new empty object after step 4 in the example above will work on false assumptions, which might in the worst case lead to the wrong (old) set of attachments being rendered to.

Workaround

Bind to framebuffer object 0 before binding to the required framebuffer object if this specific behavior is required. The "early out" will then not take place.

724941: glBindBuffer should not early out

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0,r1p1, Open.

Description

When binding a vertexbuffer target to an object already bound, the driver early out of the bind operation as it technically doesn't do anything. This is however not correct. When objects are deleted in a scenario with multiple contexts, the object is only deleted in the context where the delete operation took place. Other contexts may still have the framebuffer object bound, but the object is no longer present in the object list.

A bind operation will need to check the object list, and should there be no object present, create a new one. In the scenario with multiple contexts, "rebinding" an object may thus result in a new object being created, as illustrated by this example:

1. Context A does BindBuffer(GL_ARRAY_BUFFER, 1);
2. Context A does BufferData(GL_ARRAY_BUFFER, ...)
3. Context B does DeleteBuffer(1, {1});
4. Context A does BindBuffer(GL_ARRAY_BUFFER, 1);

The GLES spec will require step 4 to check the object list, notice that that the object list is empty and create a new vertexbuffer object. The driver is earlying out of that operation, resulting in the old deleted vertexbuffer object still being bound.

Implications

Bound vertexbuffer objects may have a life-time longer than what is dictated by the GLES specification. Applications relying on the vertexbuffer object being a new empty object after step 4 in the example above will work on false assumptions, which might in the worst case lead to the wrong (old) set of bufferdata being used.

Workaround

Bind to vertexbuffer object 0 before binding to the required vertexbuffer object if this specific behavior is required. The "early out" will then not take place.

733952: The EGL Blitting counter is never updated

Status

Affects:	Base
Fault status:	Cat 3, Present in: r0p2, Fixed in r1p0.

Description

The EGL blitting counter is never updated and will always show the value zero, even if blitting is performed.

Implications

The EGL blitting counter will always show the value zero.

Workaround

None.

596019: Creating EGLImages from texture mip level 10 or above results in EGL_BAD_MATCH

Status

Affects:	OpenGLES
Fault status:	Cat 3, Present in: r0p2,r1p0, Open.

Description

Creating an EGLImage from an OpenGL ES texture with a mip level of 10 or above will give an EGL_BAD_MATCH error instead of working as expected. This is due to a driver-limitation.

Implications

Mip levels 10 or above cannot be shared through EGLimages.

Workaround

none