Mali-200 Symbian OpenVG DDK package (GX921) **Errata Notice**

This document contains all errata known at the date of issue in supported releases up to and including revision r2p0of Mali-200 Symbian OpenVG 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

16 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	724797	OpenVG	Cat 1	eglMakeCurrent of an OpenVG context may crash in OOM situations
31	New	724845	Base	Cat 1	Driver does not handle reset of the cores while the MMU is in Pagefault mode
32	New	724858	EGL	Cat 1	EGL window surface resize OOM issue
33	New	724861	EGL	Cat 1	EGL pixmap readback OOM issue
34	New	724909	Shared	Cat 1	Potential deadlock between direct and deferred surface writes
36	New	724932	Shared	Cat 1	Potential deadlock between multiple framebuilder flush
17	Updated	669324	Base	Cat 1	Missing lock release in an out-of-memory error handler
18	Updated	669325	Base	Cat 1	Incorrect handling of Mali memory allocation requests larger than 2GB
65	New	724827	OpenVG	Cat 2	vgGetImageSubData from pixmap has red/blue channel swap
66	New	724835	OpenVG	Cat 2	Setting child image as pattern causes segmentation fault
67	New	724865	Base	Cat 2	Wait after core reset in device driver may not be long enough
68	New	724870	Base	Cat 2	Mali memory allocation might fail if a specified OS memory region is not a multiple of 256KB
69	New	724910	Base	Cat 2	The Mali job dump system will program the incorrect MMU for pixel processor jobs
70	New	724923	Base	Cat 2	Heap growth not supported by Mali job dump system
93	New	724047	Base	Cat 3	Crash when handling spurious MMU page fault interrupts
94	New	724048	EGL	Cat 3	bind-to-texture flags wrong on configs
95	New	724496	EGL	Cat 3	eglBindTexImage does not support pbuffer pitch below 64 on HW revisions prior to r0p5
96	New	724808	OpenVG	Cat 3	vgGet[fi]v does not always set error when used incorrectly
97	New	733952	Base	Cat 3	The EGL Blitting counter is never updated
72	Updated	596469	OpenVG	Cat 3	Repeat color ramps sometimes sample from wrong edge texel
73	Updated	602377	OpenVG	Cat 3	A VGImage can be used as both paint pattern and render target
75	Updated	602722	EGL	Cat 3	EGLImage: Missing check for whether pbuffer is bound through eglBindTexImage
77	Updated	668418	EGL	Cat 3	Pbuffer surface flush may fail
80	Updated	669319	Base	Cat 3	Architecture specific job start functions not allowed to fail

86	Updated	716557	Base	Cat 3	Too many jobs are dumped when dumping is combined with instrumentation
89	Updated	716734	OpenVG	Cat 3	Stroked curve discontinuities
90	Updated	716737	OpenVG	Cat 3	Image operations are always performed in sRGB
91	Updated	716777	OpenVG	Cat 3	Mask not reset on surface change
92	Updated	716779	OpenVG	Cat 3	Creation of both EGLImage and Pbuffer from a single VGImage doesn't fail

15 Jun 2009: Changes in Document v1

Page	Status	ID		Cat	Summary
16	New	669220	OpenVG	Cat 1	Memory leak in internal path cache
17	New	669324	Base	Cat 1	Missing lock release in an out-of-memory error handler
18	New	669325	Base	Cat 1	Incorrect handling of Mali memory allocation requests larger than 2GB669324
19	New	707868	Base	Cat 1	mali_mmu_get_table_page doesn't unlock the spinlock in all error exits
20	New	708168	Base	Cat 1	Instrumented driver is unable to handle all out of memory situations gracefully
21	New	716595	EGL	Cat 1	Resizing surfaces will fail when triggering a switching between direct and non-direct rendering
22	New	716756	OpenVG	Cat 1	Complex scenes can trigger infinite loop in stroke cache
23	New	716764	OpenVG	Cat 1	Stale entries in path cache
24	New	716783	OpenVG	Cat 1	Missing workaround for Hardware issue 716163 (GP2 BRESP)
25	New	716922	Base	Cat 1	Instrumented frame pointer not reference counted
38	New	594533	OpenVG	Cat 2	EGLImage created from VGImage is always multiple of 16x16
39	New	601818	OpenVG	Cat 2	Modifying a VGImage created from a pixmap can give unexpected results
40	New	602530	OpenVG	Cat 2	Creating a pbuffer from a VGImage is not thread-safe
41	New	604820	OpenVG	Cat 2	eglCreatePbufferFromClientBuffer does not reference count VGImages
42	New	609419	EGL	Cat 2	__egl_get_main_context is not thread safe
43	New	609766	Shared	Cat 2	Missing color buffer synchronization
44	New	611119	OpenVG	Cat 2	vgChildImage is not thread safe
45	New	668224	OpenVG	Cat 2	vgPointAlongPath may return incorrect tangents for zero-length path segments
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	668917	Shared	Cat 2	Subpixel Specifier workaround error
49	New	669068	OpenVG	Cat 2	Precision issues with radial gradients
50	New	669267	OpenVG	Cat 2	Invalid VG_MATRIX_PATH_USER_TO_SURFACE may cause hang

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	716556	Base	Cat 2	Incremental rendering gives incorrect output
57	New	716749	OpenVG	Cat 2	vgCopyImage does not pad edges
58	New	716751	OpenVG	Cat 2	VG_FORMAT_FILTER_LINEAR/PREMULTIPLIED not properly implemented
59	New	716752	OpenVG	Cat 2	Input rectangle parameters with extreme values are not clipped correctly
60	New	716753	OpenVG	Cat 2	copy-on-write not implemented for filtering
61	New	716759	OpenVG	Cat 2	Erroneus handling of premultiplied images during filtering operations
62	New	716760	OpenVG	Cat 2	Lookup filters don't account for filter channel mask
63	New	716762	OpenVG	Cat 2	Erroneous wrap behaviour in vgPaintPattern
64	New	716763	OpenVG	Cat 2	vgLookupSingle uses wrong default source channel for filtering luminance formats
72	New	596469	OpenVG	Cat 3	Repeat color ramps sometimes sample from wrong edge texel
73	New	602377	OpenVG	Cat 3	A VGImage can be used as both paint pattern and render target
74	New	602721	EGL	Cat 3	Zero-sized window surfaces not supported
75	New	602722	EGL	Cat 3	EGLImage: Missing check for whether pbuffer is bound through eglBindTexImage
76	New	668167	OpenVG	Cat 3	Dash pattern with odd number of segments will disable dashing
77	New	668418	EGL	Cat 3	Pbuffer surface flush may fail
78	New	668471	EGL	Cat 3	eglQueryString does not return error for name -1
79	New	668473	EGL	Cat 3	eglChooseConfig doesn't report EGL_BAD_ATTRIBUTE when attribute values are out of range
80	New	669319	Base	Cat 3	Architecture specific job start functions not allowed to fail
81	New	669323	Base	Cat 3	Incorrect memory location written to during client API negotiation sequence
82	New	669333	Base	Cat 3	Invalid XML produced when large amount of counters is sampled
83	New	708022	Base	Cat 3	Leaked external memory is not cleaned up during process shut down
84	New	708074	Base	Cat 3	Incorrect IRQ mask used during startup and shutdown
85	New	708118	Base	Cat 3	Global mutex not freed

86	New	716557	Base	Cat 3	Too many jobs are dumped when dumping is combined with instrumentation
87	New	716559	Base	Cat 3	Same fragment shader stack used by two or more pixel processors during instrumentation
88	New	716593	EGL	Cat 3	Pre-existing background in pixmap not copied during surface creation
89	New	716734	OpenVG	Cat 3	Stroked arc discontinuities
90	New	716737	OpenVG	Cat 3	Image operations are always performed in sRGB
91	New	716777	OpenVG	Cat 3	Mask not reset on surface change
92	New	716779	OpenVG	Cat 3	Creation of both EGLImage and Pbuffer from a single VGImage doesn't fail

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	
724797	OpenVG	Cat 1	eglMakeCurrent of an OpenVG context may crash in OOM situations	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		
716783	OpenVG	Cat 1	Missing workaround for Hardware issue 716163 (GP2 BRESP)	X		
716764	OpenVG	Cat 1	Stale entries in path cache	X		
716756	OpenVG	Cat 1	Complex scenes can trigger infinite loop in stroke cache	X		
716595	EGL	Cat 1	Resizing surfaces will fail when triggering a switching between direct and non-direct rendering	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		
669220	OpenVG	Cat 1	Memory leak in internal path cache	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	

ID		Cat	Summary of Erratum	r0p2	r1p0	r1p1
724865	Base	Cat 2	Wait after core reset in device driver may not be long enough	X	X	
724835	OpenVG	Cat 2	Setting child image as pattern causes segmentation fault	X	X	
724827	OpenVG	Cat 2	vgGetImageSubData from pixmap has red/blue channel swap	X		
716763	OpenVG	Cat 2	vgLookupSingle uses wrong default source channel for filtering luminance formats	X		
716762	OpenVG	Cat 2	Erroneous wrap behaviour in vgPaintPattern	X		
716760	OpenVG	Cat 2	Lookup filters don't account for filter channel mask	X		
716759	OpenVG	Cat 2	Erroneus handling of premultiplied images during filtering operations	X		
716753	OpenVG	Cat 2	Copy-on-write not implemented for filtering	X		
716752	OpenVG	Cat 2	Input rectangle parameters with extreme values are not clipped correctly	X		
716751	OpenVG	Cat 2	VG_FORMAT_FILTER_LINEAR/PREMULTIPLIED not properly implemented	X		
716749	OpenVG	Cat 2	vgCopyImage does not pad edges	X		
716556	Base	Cat 2	Incremental rendering gives incorrect output	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		
669267	OpenVG	Cat 2	Invalid VG_MATRIX_PATH_USER_TO_SURFACE may cause hang	X		
669068	OpenVG	Cat 2	Precision issues with radial gradients	X		
668917	Shared	Cat 2	Subpixel Specifier workaround error	X		
668472	EGL	Cat 2	__egl_main_power_event is not thread safe	X		
668469	EGL	Cat 2	Missing synchronization of color buffers	X		
668224	OpenVG	Cat 2	vgPointAlongPath may return incorrect tangents for zero-length path segments	X		
611119	OpenVG	Cat 2	vgChildImage is not thread safe	X		
609766	Shared	Cat 2	Missing color buffer synchronization	X		
609419	EGL	Cat 2	__egl_get_main_context is not thread safe	X		
604820	OpenVG	Cat 2	eglCreatePbufferFromClientBuffer does not reference count VGImages	X		
602530	OpenVG	Cat 2	Creating a pbuffer from a VGImage is not thread-safe	X		

ID		Cat	Summary of Erratum	r0p2	r1p0	r1p1
601818	OpenVG	Cat 2	Modifying a VGImage created from a pixmap can give unexpected results	X	X	X
594533	OpenVG	Cat 2	EGLImage created from VGImage is always multiple of 16x16	X		
733952	Base	Cat 3	The EGL Blitting counter is never updated		X	
724808	OpenVG	Cat 3	vgGet[fi]v does not always set error when used incorrectly	X	X	
724496	EGL	Cat 3	eglBindTexImage does not support pbuffer pitch below 64 on HW revisions prior to r0p5			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	
716779	OpenVG	Cat 3	Creation of both EGLImage and Pbuffer from a single VGImage doesn't fail	X	X	
716777	OpenVG	Cat 3	Mask not reset on surface change	X	X	X
716737	OpenVG	Cat 3	Image operations are always performed in sRGB	X	X	
716734	OpenVG	Cat 3	Stroked curve discontinuities	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	
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		
668167	OpenVG	Cat 3	Dash pattern with odd number of segments will disable dashing	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		
602377	OpenVG	Cat 3	A VGImage can be used as both paint pattern and render target	X		

ID		Cat	Summary of Erratum	r0p2	r1p0	r1p1
596469	OpenVG	Cat 3	Repeat color ramps sometimes sample from wrong edge texel	X	X	X

Errata - Category 1

669220: Memory leak in internal path cache

Status

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

Description

Reorganization of internal path cache nodes may in some rare cases lead to memory being leaked. It's not triggered by simple cases, but given scenes where a large amount of complex paths are being rendered and animated, memory may leak.

Implications

When rendering complex path animation for a period of time, the internal path cache system may leak some memory given certain sequences of events. The amount of memory should be small, but will accumulate over time ultimately leading to memory exhaustion.

Workaround

none

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.

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.

716756: Complex scenes can trigger infinite loop in stroke cache

Status

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

Description

A complex scene triggered a case where a path needs a new cache entry while having existing cache entries. This results in a scenario where the path cache frees one of the paths existing entries. This entry is later reused as the newly allocated cache entry for the same path.

Implications

The application may freeze in a scene with stroking.

Workaround

None

716764: Stale entries in path cache

Status

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

Description

Path cache contains stale entries when rendering complex VG scenes.

Implications

The driver asserts with the message "Cache contains stale entries" in debug mode. In release mode the driver can cause a segmentation fault.

Workaround

none

716783: Missing workaround for Hardware issue 716163 (GP2 BRESP)

Status

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

Description

MaliGP2 does not wait for AXI BRESP bus signals after writing anything to the bus. This causes subsequent reads from the same destination to read incomplete data. This means that vertices may be processed erroneously when there is a large number of outstanding writes.

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

Implications

Visual rendering errors. The last triangles of the last drawcall of a frame might get garbage positions or texture coordinates. This bug has never materialized when using VG, but in theory it could happen for systems with high memory latency.

Workaround

A possible workaround could be to render some invisible geometry which has no effect on the frame output as the last draw call of the frame.

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.

724797: eglMakeCurrent of an OpenVG context may crash in OOM situations

Status

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

Description

eglMakeCurrent of OpenVG contexts is not always handling running out of memory gracefully.

Implications

If the system runs out of memory during a call to eglMakeCurrent on an OpenVG context with a pixmap surface, a segmentation fault may be triggered.

Workaround

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

594533: EGLImage created from VGImage is always multiple of 16x16

Status

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

Description

When creating an EGLImage from a VGImage, its width and height will always be increased to the nearest multiple of 16.

Implications

It is impossible to create an EGLImage from a VGImage that is not a multiple of 16x16.

Workaround

Always use images with dimensions that are multiples of 16x16 when creating an EGLImage from a VGImage.

601818: Modifying a VGImage created from a pixmap can give unexpected results

Status

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

Description

If a VGImage is created from an EGLImage made from a pixmap, draw and modify cycles on the VGImage will not produce the expected results. For example the following sequence :

1) Draw VGImage

2) Modify VGImage

3) Draw modified VGImage

4) Show frame

Will only show the modified VGImage in both 1) and 3).

Implications

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

Workaround

Let A be the VGImage made from the EGLImage which originates from a pixmap. Let B be a temporary VGImage used for preserving the changes made to A. Ordinary VGImages have copy on write behavior which preserves image data drawn earlier on in the frame, but VGImages made from pixmaps don't.

For a draw and modify cycle on a VGImage made from a pixmap use a procedure like this:

1) Copy A to B

2) Draw B

3) Modify B

4) Draw B

5) Copy B to A

6) Show frame

602530: Creating a pbuffer from a VGImage is not thread-safe

Status

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

Description

When eglCreatePBufferFromClientBuffer is called with the buftype EGL_OPENVG_IMAGE, the internal implementation is missing locks to protect VGImages from modification.

Implications

The resulting pbuffer may be fully or partially based on a VGImage modified in another thread.

Workaround

Make sure no other threads will access the VGImage used while creating the pbuffer.

604820: eglCreatePbufferFromClientBuffer does not reference count VGImages

Status

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

Description

When an EGL surface is created from a VGImage using eglCreatePbufferFromClientBuffer the image does not get reference counted.

Implications

The lack of reference counting will lead to segmentation fault in the following use-case:

```
pbuf = eglCreatePbufferFromClientBuffer( vg_img );
vgDestroyImage( vg_img );
eglMakeCurrent( pbuf );
```

Workaround

The EGL surface must be deleted before the VGImage that was bound to it.

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).

611119: vgChildImage is not thread safe

Status

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

Description

The internal list of VGImages is not locked while a VGImage pointer is dereferenced.

Implications

Can lead to crash if a VGImage is deleted just before it is accessed in vgChildImage.

Workaround

Add synchronization so that no other function which modifies a VGImage can execute at the same time as vgChildImage.

668224: vgPointAlongPath may return incorrect tangents for zero-length path segments

Status

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

Description

If no valid tangent is found at distance, the OpenVG specification says to look first backwards in the subpath, then forwards, for a valid tangent. The driver will search the entire subpath as limited by MOVE_TO path segments, not limited by the portion of the subpath as specified by startSegment and numSegments.

Implications

vgPointAlongPath may return the last valid tangent before startSegment, instead of returning (1,0) as the OpenVG specification dictates, if no valid tangent is found at distance.

Workaround

Specify distances with valid tangents.

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.

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

669068: Precision issues with radial gradients

Status

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

Description

Due to internal limitations in floating point precision, extreme parameter values may lead to visual artifacts when drawing radial gradients.

Implications

There may be artifacts when drawing radial gradients using very small values as parameters.

Workaround

Normalize gradient parameters before passing them on to the driver.

669267: Invalid VG_MATRIX_PATH_USER_TO_SURFACE may cause hang

Status

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

Description

Internal path calculations do not check for invalid input in all cases.

Implications

Uploading a VG_MATRIX_PATH_USER_TO_SURFACE containing invalid float values such as NaN and inf and drawing a path containing curves may lead to the driver getting stuck in an infinite loop.

Workaround

Validate matrix data before uploading them to OpenVG.

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.

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.

716749: vgCopyImage does not pad edges

Status

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

Description

The lack of edge-padding results in edge-garbage when drawing images with sub byte sized texels (VG_BW_1, VG_A_1 and VG_A_4) to a surface using 16x anti-aliasing.

Implications

Following a vgCopyImage(dst, src ...) there could be garbage pixels bleeding into dst when it is drawn. This can happen when these conditions apply :

- the result of vgCopyImage overlaps the edge of the destination image
- The destination image has non-multiple of 16 dimensions.
- The image has a sub-byte format (VG_BW_1 etc).

Workaround

To work around this bug all sub byte image formats must be allocated with sizes that are dividable by 16.

716751: VG_FORMAT_FILTER_LINEAR/PREMULTIPLIED not properly implemented

Status

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

Description

VG_FORMAT_FILTER_LINEAR and VG_FORMAT_FILTER_PREMULTIPLIED flags were not correctly implemented.

Implications

The filter format flags are ignored. No conversion is done which takes them into account.

Workaround

Convert the images to the wanted formats before filtering functions using other VG API functions such as for example vgCopyImage.

716752: Input rectangle parameters with extreme values are not clipped correctly

Status

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

Description

New image tests in the Khronos OpenVG 1.1 conformance triggered arithmetic overflow in _vg_clip_to_region. This resulted in wrong output during rendering.

Implications

_vg_clip_to_region is used in most VG functions which take a rectangle on the form [x,y,width,height] as part of the input. The absolute rectangle used internally will be [x,y,(x+width) & 0xffffffff, (y+height) & 0xffffffff] where there is a chance of overflow in the two last coordinates. This results in a different rectangle being processed in the VG functions.

Workaround

Don't use positions that will over/underflow VGint in the following scenario: position + imagesize

716753: Copy-on-write not implemented for filtering

Status

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

Description

Because of the missing copy-on-write a filter-and-draw loop like:

```
for (i=0; i< 5; ++i) {
  vgConvolve(destImage, .....);
  vgSetPixels(0,32+i, destImage);
}
```

will not render correctly and only the result from the last filtering operation will be drawn.

Implications

If the same destination image is used for filtering operations several times in a single frame, only the last render operation will be visible.

Workaround

To work around this bug you need to trigger a copy-on-write right after the vgSetPixels call. This can be done by calling either vgClearImage or vgImageSubData and update a 1x1 texel.

716759: Erroneus handling of premultiplied images during filtering operations

Status

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

Description

This is seen when filter channel mask is such that it overwrites destination alpha but spares one of the color channels. If destination is in premultiplied format, the alpha for the spared color channel needs to be divided out prior to overwriting alpha.

Implications

A colour channel in a premultiplied destination image which has been protected from writes using a channel mask comes out with an unexpected value after filtering operations.

Workaround

This can be worked around by using non-premultiplied images instead.

716760: Lookup filters don't account for filter channel mask

Status

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

Description

vgLookup and vgLookupSingle writes to all color channels of the destination image regardless of filter channel mask.

Implications

All colour channels are overwritten in vgLookup and vgLookupSingle calls regardless of the channel mask setting.

Workaround

none

716762: Erroneous wrap behaviour in vgPaintPattern

Status

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

Description

vgPaintPattern exhibits erroneous wrap behavior for linear texture addressing and non 16x16 pattern images.

Implications

Garbage pixels extending beyond the edge of the pattern image are drawn just before the image wraps.

Workaround

Only use images that have dimensions divisible by 16 as patterns.

716763: vgLookupSingle uses wrong default source channel for filtering luminance formats

Status

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

Description

For vgLookupSingle the channel assigned for looking up was wrong for the following image formats: VG_sL8, VG_lL8 and VG_BW_1

Implications

vgLookupSingle produces a uniform color for source images using the VG_sL_8, VG_lL_8 or VG_BW_1 format when it is supposed to produce a variety of color values.

Workaround

Use RGBA source image instead and set RGB=luminance and A=255.

724827: vgGetImageSubData from pixmap has red/blue channel swap

Status

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

Description

vgGetImageSubData swaps red and blue color channels when used with pixmaps.

Implications

Applications using vgGetImageSubData with pixmaps will see wrong colour output.

Workaround

Use BGR formats instead of RGB and vice versa to compensate for wrong output.

724835: Setting child image as pattern causes segmentation fault

Status

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

Description

Child images are not handled correctly in pattern setup.

Implications

When an application attempts to set a child image as a paint pattern, the driver may abruptly terminate with a segmentation fault.

Workaround

Make a copy of the image instead of a child image before using as pattern.

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

596469: Repeat color ramps sometimes sample from wrong edge texel

Status

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

Description

The gradient sampling method causes gradients to wrap around prematurely in repeat mode given certain sets of parameters.

Implications

Repeat gradients may repeat slightly before intended.

Workaround

This effect is mostly a problem when a single repeat of the gradient is expected and the last color in the drawn gradient is not the end color. In this case use the pad mode instead of repeat mode.

602377: A VGImage can be used as both paint pattern and render target

Status

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

Description

Setting a VGImage as both a paint pattern source and a render target simultaneously is not prevented.

The OpenVG 1.0.1 specification says in section 9.4 under vgPatternPaint that this is not allowed:

"While an image is set as the paint pattern for any paint object, it may not be used as a rendering target. Conversely, an image that is currently a rendering target may not be set as a paint pattern."

Implications

No error message will be generated if a single image is used both as pattern paint and render target.

Workaround

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.

668167: Dash pattern with odd number of segments will disable dashing

Status

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

Description

An odd number of dash segments set from the API is not handled correctly in the driver, and will be treated as invalid input instead of silently ignoring the last segment like the OpenVG specification dictates.

Implications

Dashing will be disabled if a pattern with more "on" than "off" segments is set.

Workaround

Make sure you have matching on/off pairs when setting a dash pattern.

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.

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.

716734: Stroked curve discontinuities

Status

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

Description

Rendering stroked curves have discontinuity errors given extreme PATH_USER_TO_SURFACE matrix transformations.

Implications

You can get holes in stroked paths with arcs when zooming in. The artefacts resemble a dash pattern.

Workaround

none

716737: Image operations are always performed in sRGB

Status

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

Description

The OpenVG specification says that filtering of images, as well as the color multiplication in MULTIPLY image mode, are performed "in the color space of the image".

Linear images are handled by converting the data to sRGB, which means that filtering and multiplication are always performed in sRGB, giving incorrect results.

Implications

Linear images will look as though they are converted to sRGB before filtering and when multiplied with a paint.

Workaround

none

716777: Mask not reset on surface change

Status

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

Description

When changing to a new surface, the alpha mask buffer should be reset according to the OpenVG spec. The driver does not reset the mask, but instead keeps the mask from the last surface the VG context was bound to.

Implications

The mask is not reset to all 1s when a new surface is made current.

Workaround

Manually set the mask to all 1s after making a new surface current.

716779: Creation of both EGLImage and Pbuffer from a single VGImage doesn't fail

Status

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

Description

There is nothing in the driver to stop a user from creating both a pbuffer and an EGLImage from the same VGImage, although the extension spec says it's not allowed. The user can do it in the following way: First create an EGLImage from a VGImage, then create a Pbuffer from the same VGImage.

Implications

The specification doesn't allow it, but the drivers allow you to do this anyway. The behaviour is undefined in this case.

Workaround

none

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.

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.

724808: vgGet[fi]v does not always set error when used incorrectly

Status

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

Description

VG_ILLEGAL_ARGUMENT_ERROR is not set in all situations where it should be when calling vgGet[if]v with illegal arguments.

Implications

If an application calls vgGet[fi]v with illegal arguments, vgGetError() will not generate an error message in some cases.

Workaround

None.

733952: The EGL Blitting counter is never updated

Status

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

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.