My Project

Generated by Doxygen 1.8.11

Contents

Chapter 1

to free to the pool

kbase_mem_pool_free - Free a page to memory pool : Memory pool where page should be freed

: Whether some of the page may be dirty in the cache.

Pages are freed to the pool as follows:

1. If is not full, add

Chapter 2

@pool.

1. Otherwise, if is not NULL and not full, add
   .
2. Finally, free

Chapter 3

MMU hardware interface

3.1 Introduction

This module provides an abstraction for accessing the functionality provided by the midgard MMU and thus allows all MMU HW access to be contained within one common place and allows for different backends (implementations) to be provided.

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

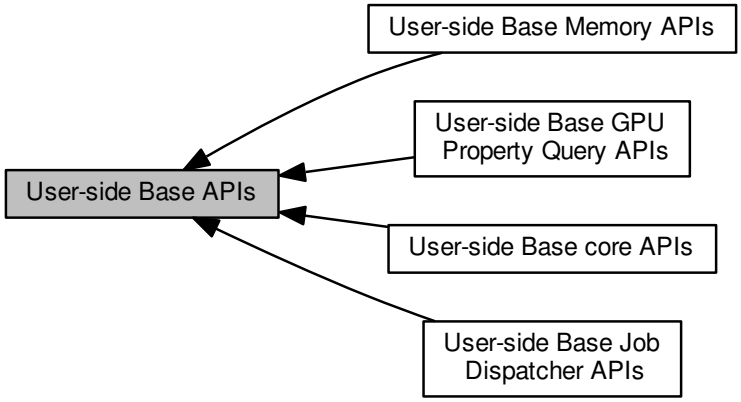
Here is a list of all documented files with brief descriptions:

Chapter 7

Module Documentation

7.1 User-side Base APIs

Collaboration diagram for User-side Base APIs:



Modules

- User-side Base Memory APIs
- User-side Base Job Dispatcher APIs
- User-side Base GPU Property Query APIs
- User-side Base core APIs

Macros

- #define **GPU_MAX_JOB_SLOTS** 16

7.1.1 Detailed Description

7.1.1.1 User-side Base GPU Property Query API

The User-side Base GPU Property Query API encapsulates two sub-modules:

- Dynamic GPU Properties
- Base Platform Config GPU Properties

There is a related third module outside of Base, which is owned by the MIDG module:

- Midgard Compile-time GPU Properties

Base only deals with properties that vary between different Midgard implementations - the Dynamic GPU properties and the Platform Config properties.

For properties that are constant for the Midgard Architecture, refer to the MIDG module. However, we will discuss their relevance here **just to provide background information.**

7.1.2 About the GPU Properties in Base and MIDG modules

The compile-time properties (Platform Config, Midgard Compile-time properties) are exposed as pre-processor macros.

Complementing the compile-time properties are the Dynamic GPU Properties, which act as a conduit for the Midgard Configuration Discovery.

In general, the dynamic properties are present to verify that the platform has been configured correctly with the right set of Platform Config Compile-time Properties.

As a consistent guide across the entire DDK, the choice for dynamic or compile-time should consider the following, in order:

1. Can the code be written so that it doesn't need to know the implementation limits at all?
2. If you need the limits, get the information from the Dynamic Property lookup. This should be done once as you fetch the context, and then cached as part of the context data structure, so it's cheap to access.
3. If there's a clear and arguable inefficiency in using Dynamic Properties, then use a Compile-Time Property (Platform Config, or Midgard Compile-time property). Examples of where this might be sensible follow:
    - Part of a critical inner-loop
    - Frequent re-use throughout the driver, causing significant extra load instructions or control flow that would be worthwhile optimizing out.

We cannot provide an exhaustive set of examples, neither can we provide a rule for every possible situation. Use common sense, and think about: what the rest of the driver will be doing; how the compiler might represent the value if it is a compile-time constant; whether an OEM shipping multiple devices would benefit much more from a single DDK binary, instead of insignificant micro-optimizations.

7.1.3 Dynamic GPU Properties

Dynamic GPU properties are presented in two sets:

1. the commonly used properties in base_gpu_props, which have been unpacked from GPU register bitfields.
2. The full set of raw, unprocessed properties in gpu_raw_gpu_props (also a member of base_gpu_props). All of these are presented in the packed form, as presented by the GPU registers themselves.

The raw properties in gpu_raw_gpu_props are necessary to allow a user of the Mali Tools (e.g. PAT) to determine "Why is this device behaving differently?". In this case, all information about the configuration is potentially useful, but it **does not need to be processed by the driver**. Instead, the raw registers can be processed by the Mali Tools software on the host PC.

The properties returned extend the Midgard Configuration Discovery registers. For example, GPU clock speed is not specified in the Midgard Architecture, but is **necessary for OpenCL's clGetDeviceInfo() function**.

The GPU properties are obtained by a call to base_get_gpu_props(). This simply returns a pointer to a const base_gpu_props structure. It is constant for the life of a base context. Multiple calls to base_get_gpu_props() to a base context return the same pointer to a constant structure. This avoids cache pollution of the common data.

This pointer must not be freed, because it does not point to the start of a region allocated by the memory allocator; instead, just close the base_context.

7.1.4 Kernel Operation

During Base Context Create time, user-side makes a single kernel call:

- A call to fill user memory with GPU information structures

The kernel-side will fill the provided the entire processed base_gpu_props structure, because this information is required in both user and kernel side; it does not make sense to decode it twice.

Coherency groups must be derived from the bitmasks, but this can be done kernel side, and just once at kernel startup: Coherency groups must already be known kernel-side, to support chains that specify a 'Only Coherent Group' SW requirement, or 'Only Coherent Group with Tiler' SW requirement.

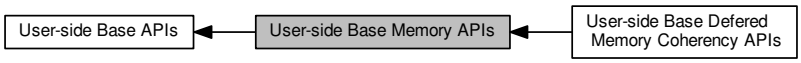
7.1.5 Coherency Group calculation

Creation of the coherent group data is done at device-driver startup, and so is one-time. This will most likely involve a loop with CLZ, shifting, and bit clearing on the L2_PRESENT mask, depending on whether the system is L2 Coherent. The number of shader cores is done by a population count, since faulty cores may be disabled during production, producing a non-contiguous mask.

The memory requirements for this algorithm can be determined either by a u64 population count on the L2_PRE↩SENT mask (a LUT helper already is required for the above), or simple assumption that there can be no more than 16 coherent groups, since core groups are typically 4 cores.

7.2 User-side Base Memory APIs

Collaboration diagram for User-side Base Memory APIs:



Modules

- User-side Base Defered Memory Coherency APIs

Classes

- struct base_mem_import_user_buffer
- struct base_import_handle

Macros

- #define **BASE_MEM_PROT_CPU_RD** ((base_mem_alloc_flags)1 << 0)
- #define **BASE_MEM_PROT_CPU_WR** ((base_mem_alloc_flags)1 << 1)
- #define **BASE_MEM_PROT_GPU_RD** ((base_mem_alloc_flags)1 << 2)
- #define **BASE_MEM_PROT_GPU_WR** ((base_mem_alloc_flags)1 << 3)
- #define **BASE_MEM_PROT_GPU_EX** ((base_mem_alloc_flags)1 << 4)
- #define **BASE_MEM_RESERVED_BIT_5** ((base_mem_alloc_flags)1 << 5)
- #define **BASE_MEM_RESERVED_BIT_6** ((base_mem_alloc_flags)1 << 6)
- #define **BASE_MEM_RESERVED_BIT_7** ((base_mem_alloc_flags)1 << 7)
- #define **BASE_MEM_RESERVED_BIT_8** ((base_mem_alloc_flags)1 << 8)
- #define **BASE_MEM_GROW_ON_GPF** ((base_mem_alloc_flags)1 << 9)
- #define **BASE_MEM_COHERENT_SYSTEM** ((base_mem_alloc_flags)1 << 10)
- #define **BASE_MEM_COHERENT_LOCAL** ((base_mem_alloc_flags)1 << 11)
- #define **BASE_MEM_CACHED_CPU** ((base_mem_alloc_flags)1 << 12)
- #define **BASE_MEM_SAME_VA** ((base_mem_alloc_flags)1 << 13)
- #define **BASE_MEM_NEED_MMAP** ((base_mem_alloc_flags)1 << 14)
- #define **BASE_MEM_COHERENT_SYSTEM_REQUIRED** ((base_mem_alloc_flags)1 << 15)
- #define **BASE_MEM_SECURE** ((base_mem_alloc_flags)1 << 16)
- #define **BASE_MEM_DONT_NEED** ((base_mem_alloc_flags)1 << 17)
- #define **BASE_MEM_IMPORT_SHARED** ((base_mem_alloc_flags)1 << 18)
- #define BASE_MEM_RESERVED_BIT_19 ((base_mem_alloc_flags)1 << 19)
- #define BASE_MEM_TILER_ALIGN_TOP ((base_mem_alloc_flags)1 << 20)
- #define **BASE_MEM_FLAGS_NR_BITS** 21
- #define **BASE_MEM_FLAGS_OUTPUT_MASK** BASE_MEM_NEED_MMAP
- #define **BASE_MEM_FLAGS_INPUT_MASK** (((1 << BASE_MEM_FLAGS_NR_BITS) - 1) & ~BASE_M↩ EM_FLAGS_OUTPUT_MASK)
- #define **BASE_MEM_FLAGS_MODIFIABLE**
- #define **BASE_MEM_FLAGS_RESERVED**
- #define **BASE_MEM_FLAGS_QUERYABLE**

- #define BASE_MEM_INVALID_HANDLE ((base_mem_handle) { {BASEP_MEM_INVALID_HANDLE} })

  Invalid memory handle.
- #define BASE_MEM_WRITE_ALLOC_PAGES_HANDLE ((base_mem_handle) { {BASEP_MEM_WRITE_↩ ALLOC_PAGES_HANDLE} })

  Special write-alloc memory handle.
- #define **BASEP_MEM_INVALID_HANDLE** (0ull << 12)
- #define **BASE_MEM_MMU_DUMP_HANDLE** (1ull << 12)
- #define **BASE_MEM_TRACE_BUFFER_HANDLE** (2ull << 12)
- #define **BASE_MEM_MAP_TRACKING_HANDLE** (3ull << 12)
- #define **BASEP_MEM_WRITE_ALLOC_PAGES_HANDLE** (4ull << 12)
- #define **BASE_MEM_COOKIE_BASE** (64ul << 12)
- #define **BASE_MEM_FIRST_FREE_ADDRESS**
- #define **BASE_MEM_MASK_4GB** 0xfffff000UL
- #define BASE_MEM_TILER_ALIGN_TOP_EXTENT_MAX_PAGES ((2ull * 1024ull * 1024ull) >> (LOCA↩ L_PAGE_SHIFT))
- #define **KBASE_COOKIE_MASK** ~1UL /* bit 0 is reserved */

Typedefs

- typedef u32 base_mem_alloc_flags
- typedef enum base_mem_import_type base_mem_import_type
- typedef enum base_backing_threshold_status base_backing_threshold_status

  Result codes of changing the size of the backing store allocated to a tmem region.
- typedef struct base_import_handle base_import_handle

Enumerations

- enum base_mem_import_type { **BASE_MEM_IMPORT_TYPE_INVALID** = 0, **BASE_MEM_IMPORT_TY**↩ **PE_UMP** = 1, **BASE_MEM_IMPORT_TYPE_UMM** = 2, **BASE_MEM_IMPORT_TYPE_USER_BUFFER** = 3 }
- enum base_backing_threshold_status { BASE_BACKING_THRESHOLD_OK = 0, BASE_BACKING_THR↩ ESHOLD_ERROR_OOM = -2, BASE_BACKING_THRESHOLD_ERROR_INVALID_ARGUMENTS = -4 }

  Result codes of changing the size of the backing store allocated to a tmem region.

7.2.1 Detailed Description

7.2.2 Macro Definition Documentation

7.2.2.1 #define BASE_MEM_FIRST_FREE_ADDRESS

Value:

```
((BITS_PER_LONG << 12) + \
                        BASE_MEM_COOKIE_BASE)
```

7.2.2.2 #define BASE_MEM_FLAGS_MODIFIABLE

Value:

```
(BASE_MEM_DONT_NEED | BASE_MEM_COHERENT_SYSTEM | \
     BASE_MEM_COHERENT_LOCAL)
```

7.2.2.3 #define BASE_MEM_FLAGS_QUERYABLE

Value:

```
(BASE_MEM_FLAGS_INPUT_MASK & ~(BASE_MEM_SAME_VA | \
        BASE_MEM_COHERENT_SYSTEM_REQUIRED | BASE_MEM_DONT_NEED | \
        BASE_MEM_IMPORT_SHARED | BASE_MEM_FLAGS_RESERVED))
```

7.2.2.4 #define BASE_MEM_FLAGS_RESERVED

Value:

```
(BASE_MEM_RESERVED_BIT_5 | BASE_MEM_RESERVED_BIT_6 | \
        BASE_MEM_RESERVED_BIT_7 | BASE_MEM_RESERVED_BIT_8 | \
        BASE_MEM_RESERVED_BIT_19)
```

7.2.2.5 #define BASE_MEM_INVALID_HANDLE ((base_mem_handle) { {BASEP_MEM_INVALID_HANDLE} })

Invalid memory handle.

Return value from functions returning base_mem_handle on error.

Warning

base_mem_handle_new_invalid must be used instead of this macro in C++ code or other situations where compound literals cannot be used.

7.2.2.6 #define BASE_MEM_RESERVED_BIT_19 ((base_mem_alloc_flags)1 << 19)

Bit 19 is reserved.

Do not remove, use the next unreserved bit for new flags

7.2.2.7 #define BASE_MEM_TILER_ALIGN_TOP ((base_mem_alloc_flags)1 << 20)

Memory starting from the end of the initial commit is aligned to 'extent' pages, where 'extent' must be a power of 2 and no more than BASE_MEM_TILER_ALIGN_TOP_EXTENT_MAX_PAGES

7.2.2.8 #define BASE_MEM_TILER_ALIGN_TOP_EXTENT_MAX_PAGES ((2ull * 1024ull * 1024ull) >> (LOCAL_PAGE_SHIFT))

Limit on the 'extent' parameter for an allocation with the BASE_MEM_TILER_ALIGN_TOP flag set

This is the same as the maximum limit for a Buffer Descriptor's chunk size

7.2.2.9 #define BASE_MEM_WRITE_ALLOC_PAGES_HANDLE ((base_mem_handle) { {BASEP_MEM_WRITE_ALLOC_PA↩ GES_HANDLE} })

Special write-alloc memory handle.

A special handle is used to represent a region where a special page is mapped with a write-alloc cache setup, typically used when the write result of the GPU isn't needed, but the GPU must write anyway.

Warning

base_mem_handle_new_write_alloc must be used instead of this macro in C++ code or other situations where compound literals cannot be used.

7.2.3 Typedef Documentation

7.2.3.1 typedef struct base_import_handle base_import_handle

Handle to represent imported memory object. Simple opague handle to imported memory, can't be used with anything but base_external_resource_init to bind to an atom.

7.2.3.2 typedef u32 base_mem_alloc_flags

typedef base_mem_alloc_flags - Memory allocation, access/hint flags.

A combination of MEM_PROT/MEM_HINT flags must be passed to each allocator in order to determine the best cache policy. Some combinations are of course invalid (e.g. MEM_PROT_CPU_WR | MEM_HINT_CPU_RD), which defines a write-only region on the CPU side, which is heavily read by the CPU... Other flags are only meaningful to a particular allocator. More flags can be added to this list, as long as they don't clash (see BASE_↩ MEM_FLAGS_NR_BITS for the number of the first free bit).

7.2.3.3 typedef enum base_mem_import_type base_mem_import_type

enum base_mem_import_type - Memory types supported by *base_mem_import*

: Invalid type : UMP import. Handle type is ump_secure_id. : UMM import. Handle type is a file descriptor (int) : User buffer import. Handle is a base_mem_import_user_buffer

Each type defines what the supported handle type is.

If any new type is added here ARM must be contacted to allocate a numeric value for it. Do not just add a new type without synchronizing with ARM as future releases from ARM might include other new types which could clash with your custom types.

7.2.4 Enumeration Type Documentation

7.2.4.1 enum base_backing_threshold_status

Result codes of changing the size of the backing store allocated to a tmem region.

Enumerator

BASE_BACKING_THRESHOLD_OK Resize successful

BASE_BACKING_THRESHOLD_ERROR_OOM Increase failed due to an out-of-memory condition

BASE_BACKING_THRESHOLD_ERROR_INVALID_ARGUMENTS Invalid arguments (not tmem, illegal size request, etc.)

7.2.4.2 enum base_mem_import_type

enum base_mem_import_type - Memory types supported by *base_mem_import*

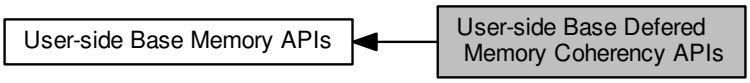
: Invalid type : UMP import. Handle type is ump_secure_id. : UMM import. Handle type is a file descriptor (int) : User buffer import. Handle is a base_mem_import_user_buffer

Each type defines what the supported handle type is.

If any new type is added here ARM must be contacted to allocate a numeric value for it. Do not just add a new type without synchronizing with ARM as future releases from ARM might include other new types which could clash with your custom types.

7.3 User-side Base Defered Memory Coherency APIs

Collaboration diagram for User-side Base Defered Memory Coherency APIs:



Classes

- struct base_syncset

  a basic memory operation (sync-set).

Typedefs

- typedef struct base_syncset base_syncset

  a basic memory operation (sync-set).

7.3.1 Detailed Description

7.3.2 Typedef Documentation

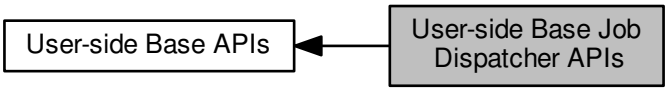
7.3.2.1 typedef struct base_syncset base_syncset

a basic memory operation (sync-set).

The content of this structure is private, and should only be used by the accessors.

7.4 User-side Base Job Dispatcher APIs

Collaboration diagram for User-side Base Job Dispatcher APIs:



Classes

- struct base_stream
- struct base_fence
- struct base_jd_udata

  Per-job data.
- struct base_mem_aliasing_info

  Memory aliasing info.
- struct base_jit_alloc_info
- struct base_dependency
- struct base_jd_atom_v2
- struct base_external_resource
- struct base_external_resource_list
- struct base_jd_debug_copy_buffer
- struct base_jd_event_v2

  Event reporting structure.
- struct base_dump_cpu_gpu_counters

  Structure for BASE_JD_REQ_SOFT_DUMP_CPU_GPU_COUNTERS jobs.

Macros

- #define **INVALID_PLATFORM_FENCE** ((platform_fence_type)-1)
- #define BASE_JD_DEP_TYPE_INVALID (0)
- #define BASE_JD_DEP_TYPE_DATA (1U << 0)
- #define BASE_JD_DEP_TYPE_ORDER (1U << 1)
- #define BASE_JD_REQ_DEP ((base_jd_core_req)0)
- #define BASE_JD_REQ_FS ((base_jd_core_req)1 << 0)
- #define BASE_JD_REQ_CS ((base_jd_core_req)1 << 1)
- #define BASE_JD_REQ_T ((base_jd_core_req)1 << 2)
- #define BASE_JD_REQ_CF ((base_jd_core_req)1 << 3)
- #define BASE_JD_REQ_V ((base_jd_core_req)1 << 4)
- #define **BASE_JD_REQ_FS_AFBC** ((base_jd_core_req)1 << 13)
- #define BASE_JD_REQ_EVENT_COALESCE ((base_jd_core_req)1 << 5)
- #define BASE_JD_REQ_COHERENT_GROUP ((base_jd_core_req)1 << 6)
- #define BASE_JD_REQ_PERMON ((base_jd_core_req)1 << 7)
- #define BASE_JD_REQ_EXTERNAL_RESOURCES ((base_jd_core_req)1 << 8)
- #define BASE_JD_REQ_SOFT_JOB ((base_jd_core_req)1 << 9)

- #define **BASE_JD_REQ_SOFT_DUMP_CPU_GPU_TIME** (BASE_JD_REQ_SOFT_JOB | 0x1)
- #define **BASE_JD_REQ_SOFT_FENCE_TRIGGER** (BASE_JD_REQ_SOFT_JOB | 0x2)
- #define **BASE_JD_REQ_SOFT_FENCE_WAIT** (BASE_JD_REQ_SOFT_JOB | 0x3)
- #define BASE_JD_REQ_SOFT_REPLAY (BASE_JD_REQ_SOFT_JOB | 0x4)
- #define BASE_JD_REQ_SOFT_EVENT_WAIT (BASE_JD_REQ_SOFT_JOB | 0x5)
- #define **BASE_JD_REQ_SOFT_EVENT_SET** (BASE_JD_REQ_SOFT_JOB | 0x6)
- #define **BASE_JD_REQ_SOFT_EVENT_RESET** (BASE_JD_REQ_SOFT_JOB | 0x7)
- #define **BASE_JD_REQ_SOFT_DEBUG_COPY** (BASE_JD_REQ_SOFT_JOB | 0x8)
- #define BASE_JD_REQ_SOFT_JIT_ALLOC (BASE_JD_REQ_SOFT_JOB | 0x9)
- #define BASE_JD_REQ_SOFT_JIT_FREE (BASE_JD_REQ_SOFT_JOB | 0xa)
- #define BASE_JD_REQ_SOFT_EXT_RES_MAP (BASE_JD_REQ_SOFT_JOB | 0xb)
- #define BASE_JD_REQ_SOFT_EXT_RES_UNMAP (BASE_JD_REQ_SOFT_JOB | 0xc)
- #define BASE_JD_REQ_ONLY_COMPUTE ((base_jd_core_req)1 << 10)
- #define BASE_JD_REQ_SPECIFIC_COHERENT_GROUP ((base_jd_core_req)1 << 11)
- #define BASE_JD_REQ_EVENT_ONLY_ON_FAILURE ((base_jd_core_req)1 << 12)
- #define BASEP_JD_REQ_EVENT_NEVER ((base_jd_core_req)1 << 14)
- #define BASE_JD_REQ_SKIP_CACHE_START ((base_jd_core_req)1 << 15)
- #define BASE_JD_REQ_SKIP_CACHE_END ((base_jd_core_req)1 << 16)
- #define BASEP_JD_REQ_RESERVED
- #define BASE_JD_REQ_ATOM_TYPE
- #define BASE_JD_REQ_SOFT_JOB_TYPE (BASE_JD_REQ_SOFT_JOB | 0x1f)
- #define **BASE_JD_REQ_SOFT_JOB_OR_DEP**(core_req)
- #define **BASE_JD_PRIO_MEDIUM** ((base_jd_prio)0)
- #define **BASE_JD_PRIO_HIGH** ((base_jd_prio)1)
- #define **BASE_JD_PRIO_LOW** ((base_jd_prio)2)
- #define **BASE_JD_NR_PRIO_LEVELS** 3
- #define BASE_EXT_RES_COUNT_MAX 10

Typedefs

- typedef int **platform_fence_type**
- typedef struct base_stream base_stream
- typedef struct base_fence base_fence
- typedef struct base_jd_udata base_jd_udata

  Per-job data.
- typedef u8 base_jd_dep_type

  Job dependency type.
- typedef u32 base_jd_core_req

  Job chain hardware requirements.
- typedef u8 **base_jd_prio**
- typedef u8 base_atom_id
- typedef struct base_jd_atom_v2 **base_jd_atom_v2**
- typedef enum base_external_resource_access **base_external_resource_access**
- typedef struct base_external_resource **base_external_resource**
- typedef enum base_jd_event_code base_jd_event_code

  Job chain event codes.
- typedef struct base_jd_event_v2 base_jd_event_v2

  Event reporting structure.
- typedef struct base_dump_cpu_gpu_counters base_dump_cpu_gpu_counters

  Structure for BASE_JD_REQ_SOFT_DUMP_CPU_GPU_COUNTERS jobs.

Enumerations

- enum kbase_atom_coreref_state {
  KBASE_ATOM_COREREF_STATE_NO_CORES_REQUESTED, KBASE_ATOM_COREREF_STATE_↩
  WAITING_FOR_REQUESTED_CORES, KBASE_ATOM_COREREF_STATE_RECHECK_AFFINITY, KB↩
  ASE_ATOM_COREREF_STATE_CHECK_AFFINITY_VIOLATIONS,
  KBASE_ATOM_COREREF_STATE_READY }

  States to model state machine processed by kbasep_js_job_check_ref_cores(), which handles retaining cores for power management and affinity management.
- enum kbase_jd_atom_state {
  KBASE_JD_ATOM_STATE_UNUSED, KBASE_JD_ATOM_STATE_QUEUED, KBASE_JD_ATOM_STA↩
  TE_IN_JS, KBASE_JD_ATOM_STATE_HW_COMPLETED,
  KBASE_JD_ATOM_STATE_COMPLETED }
- enum **base_external_resource_access** { **BASE_EXT_RES_ACCESS_SHARED**, **BASE_EXT_RES_AC↩
  CESS_EXCLUSIVE** }
- enum {
  BASE_JD_SW_EVENT_KERNEL = (1u << 15), BASE_JD_SW_EVENT = (1u << 14), BASE_JD_SW_↩
  EVENT_SUCCESS = (1u << 13), BASE_JD_SW_EVENT_JOB = (0u << 11),
  BASE_JD_SW_EVENT_BAG = (1u << 11), BASE_JD_SW_EVENT_INFO = (2u << 11), BASE_JD_SW↩
  _EVENT_RESERVED = (3u << 11), BASE_JD_SW_EVENT_TYPE_MASK = (3u << 11) }

  Job chain event code bits Defines the bits used to create base_jd_event_code.
- enum base_jd_event_code {
  BASE_JD_EVENT_RANGE_HW_NONFAULT_START = 0, BASE_JD_EVENT_NOT_STARTED = 0x00,
  BASE_JD_EVENT_DONE = 0x01, BASE_JD_EVENT_STOPPED = 0x03,
  BASE_JD_EVENT_TERMINATED = 0x04, BASE_JD_EVENT_ACTIVE = 0x08, BASE_JD_EVENT_RAN↩
  GE_HW_NONFAULT_END = 0x40, BASE_JD_EVENT_RANGE_HW_FAULT_OR_SW_ERROR_START =
  0x40,
  BASE_JD_EVENT_JOB_CONFIG_FAULT = 0x40, **BASE_JD_EVENT_JOB_POWER_FAULT** = 0x41, **B↩
  ASE_JD_EVENT_JOB_READ_FAULT** = 0x42, **BASE_JD_EVENT_JOB_WRITE_FAULT** = 0x43,
  BASE_JD_EVENT_JOB_AFFINITY_FAULT = 0x44, **BASE_JD_EVENT_JOB_BUS_FAULT** = 0x48, **BA↩
  SE_JD_EVENT_INSTR_INVALID_PC** = 0x50, **BASE_JD_EVENT_INSTR_INVALID_ENC** = 0x51,
  BASE_JD_EVENT_INSTR_TYPE_MISMATCH = 0x52, **BASE_JD_EVENT_INSTR_OPERAND_FAULT** =
  0x53, **BASE_JD_EVENT_INSTR_TLS_FAULT** = 0x54, **BASE_JD_EVENT_INSTR_BARRIER_FAULT** =
  0x55,
  BASE_JD_EVENT_INSTR_ALIGN_FAULT = 0x56, **BASE_JD_EVENT_DATA_INVALID_FAULT** = 0x58,
  BASE_JD_EVENT_TILE_RANGE_FAULT = 0x59, **BASE_JD_EVENT_STATE_FAULT** = 0x5A,
  BASE_JD_EVENT_OUT_OF_MEMORY = 0x60, **BASE_JD_EVENT_UNKNOWN** = 0x7F, **BASE_JD_EV↩
  ENT_DELAYED_BUS_FAULT** = 0x80, **BASE_JD_EVENT_SHAREABILITY_FAULT** = 0x88,
  BASE_JD_EVENT_TRANSLATION_FAULT_LEVEL1 = 0xC1, **BASE_JD_EVENT_TRANSLATION_FA↩
  ULT_LEVEL2** = 0xC2, **BASE_JD_EVENT_TRANSLATION_FAULT_LEVEL3** = 0xC3, **BASE_JD_EVEN↩
  T_TRANSLATION_FAULT_LEVEL4** = 0xC4,
  BASE_JD_EVENT_PERMISSION_FAULT = 0xC8, **BASE_JD_EVENT_TRANSTAB_BUS_FAULT_LEV↩
  EL1** = 0xD1, **BASE_JD_EVENT_TRANSTAB_BUS_FAULT_LEVEL2** = 0xD2, **BASE_JD_EVENT_TRA↩
  NSTAB_BUS_FAULT_LEVEL3** = 0xD3,
  BASE_JD_EVENT_TRANSTAB_BUS_FAULT_LEVEL4 = 0xD4, **BASE_JD_EVENT_ACCESS_FLAG** =
  0xD8, **BASE_JD_EVENT_MEM_GROWTH_FAILED** = BASE_JD_SW_EVENT | BASE_JD_SW_EVENT↩
  _JOB | 0x000, **BASE_JD_EVENT_TIMED_OUT** = BASE_JD_SW_EVENT | BASE_JD_SW_EVENT_JOB |
  0x001,
  BASE_JD_EVENT_JOB_CANCELLED = BASE_JD_SW_EVENT | BASE_JD_SW_EVENT_JOB | 0x002,
  BASE_JD_EVENT_JOB_INVALID = BASE_JD_SW_EVENT | BASE_JD_SW_EVENT_JOB | 0x003, **BA↩
  SE_JD_EVENT_PM_EVENT** = BASE_JD_SW_EVENT | BASE_JD_SW_EVENT_JOB | 0x004, **BASE_J↩
  D_EVENT_FORCE_REPLAY** = BASE_JD_SW_EVENT | BASE_JD_SW_EVENT_JOB | 0x005,
  BASE_JD_EVENT_BAG_INVALID = BASE_JD_SW_EVENT | BASE_JD_SW_EVENT_BAG | 0x003, B↩
  ASE_JD_EVENT_RANGE_HW_FAULT_OR_SW_ERROR_END = BASE_JD_SW_EVENT | BASE_JD_S↩
  W_EVENT_RESERVED | 0x3FF, BASE_JD_EVENT_RANGE_SW_SUCCESS_START = BASE_JD_SW↩
  _EVENT | BASE_JD_SW_EVENT_SUCCESS | 0x000, **BASE_JD_EVENT_PROGRESS_REPORT** = BA↩
  SE_JD_SW_EVENT | BASE_JD_SW_EVENT_SUCCESS | BASE_JD_SW_EVENT_JOB | 0x000,
  BASE_JD_EVENT_BAG_DONE = BASE_JD_SW_EVENT | BASE_JD_SW_EVENT_SUCCESS | BASE↩

_JD_SW_EVENT_BAG | 0x000, **BASE_JD_EVENT_DRV_TERMINATED** = BASE_JD_SW_EVENT | BA↩
SE_JD_SW_EVENT_SUCCESS | BASE_JD_SW_EVENT_INFO | 0x000, BASE_JD_EVENT_RANGE_S↩
W_SUCCESS_END = BASE_JD_SW_EVENT | BASE_JD_SW_EVENT_SUCCESS | BASE_JD_SW_EV↩
ENT_RESERVED | 0x3FF, BASE_JD_EVENT_RANGE_KERNEL_ONLY_START = BASE_JD_SW_EVENT
| BASE_JD_SW_EVENT_KERNEL | 0x000,
BASE_JD_EVENT_REMOVED_FROM_NEXT = BASE_JD_SW_EVENT | BASE_JD_SW_EVENT_KER↩
NEL | BASE_JD_SW_EVENT_JOB | 0x000, BASE_JD_EVENT_RANGE_KERNEL_ONLY_END = BASE↩
_JD_SW_EVENT | BASE_JD_SW_EVENT_KERNEL | BASE_JD_SW_EVENT_RESERVED | 0x3FF }

Job chain event codes.

7.4.1 Detailed Description

7.4.2 Macro Definition Documentation

7.4.2.1 #define BASE_EXT_RES_COUNT_MAX 10

The maximum number of external resources which can be mapped/unmapped in a single request.

7.4.2.2 #define BASE_JD_DEP_TYPE_DATA (1U << 0)

Data dependency

7.4.2.3 #define BASE_JD_DEP_TYPE_INVALID (0)

Invalid dependency

7.4.2.4 #define BASE_JD_DEP_TYPE_ORDER (1U << 1)

Order dependency

7.4.2.5 #define BASE_JD_REQ_ATOM_TYPE

Value:

```
(BASE_JD_REQ_FS | BASE_JD_REQ_CS | BASE_JD_REQ_T |
      BASE_JD_REQ_CF | \
    BASE_JD_REQ_V | BASE_JD_REQ_SOFT_JOB |
      BASE_JD_REQ_ONLY_COMPUTE)
```

Mask of all bits in base_jd_core_req that control the type of the atom.

This allows dependency only atoms to have flags set

7.4.2.6 #define BASE_JD_REQ_CF ((base_jd_core_req)1 << 3)

Requires cache flushes

7.4.2.7 #define BASE_JD_REQ_COHERENT_GROUP ((base_jd_core_req)1 << 6)

SW Only requirement: the job chain requires a coherent core group. We don't mind which coherent core group is used.

7.4.2.8 #define BASE_JD_REQ_CS ((base_jd_core_req)1 << 1)

Requires compute shaders This covers any of the following Midgard Job types:

- Vertex Shader Job
- Geometry Shader Job
- An actual Compute Shader Job

Compare this with BASE_JD_REQ_ONLY_COMPUTE, which specifies that the job is specifically just the "Compute Shader" job type, and not the "Vertex Shader" nor the "Geometry Shader" job type.

7.4.2.9 #define BASE_JD_REQ_DEP ((base_jd_core_req)0)

No requirement, dependency only

7.4.2.10 #define BASE_JD_REQ_EVENT_COALESCE ((base_jd_core_req)1 << 5)

SW-only requirement: coalesce completion events. If this bit is set then completion of this atom will not cause an event to be sent to userspace, whether successful or not; completion events will be deferred until an atom completes which does not have this bit set.

This bit may not be used in combination with BASE_JD_REQ_EXTERNAL_RESOURCES.

7.4.2.11 #define BASE_JD_REQ_EVENT_ONLY_ON_FAILURE ((base_jd_core_req)1 << 12)

SW Flag: If this bit is set then the successful completion of this atom will not cause an event to be sent to userspace

7.4.2.12 #define BASE_JD_REQ_EXTERNAL_RESOURCES ((base_jd_core_req)1 << 8)

SW Only requirement: External resources are referenced by this atom. When external resources are referenced no syncsets can be bundled with the atom but should instead be part of a NULL jobs inserted into the dependency tree. The first pre_dep object must be configured for the external resouces to use, the second pre_dep object can be used to create other dependencies.

This bit may not be used in combination with BASE_JD_REQ_EVENT_COALESCE and BASE_JD_REQ_SOFT↩ _EVENT_WAIT.

7.4.2.13 #define BASE_JD_REQ_FS ((base_jd_core_req)1 << 0)

Requires fragment shaders

7.4.2.14 #define BASE_JD_REQ_ONLY_COMPUTE ((base_jd_core_req)1 << 10)

HW Requirement: Requires Compute shaders (but not Vertex or Geometry Shaders)

This indicates that the Job Chain contains Midgard Jobs of the 'Compute Shaders' type.

In contrast to BASE_JD_REQ_CS, this does **not** indicate that the Job Chain contains 'Geometry Shader' or 'Vertex Shader' jobs.

7.4.2.15 #define BASE_JD_REQ_PERMON ((base_jd_core_req)1 << 7)

SW Only requirement: The performance counters should be enabled only when they are needed, to reduce power consumption.

7.4.2.16 #define BASE_JD_REQ_SKIP_CACHE_END ((base_jd_core_req)1 << 16)

SW Flag: Skip GPU cache clean and invalidation after a GPU job completes.

If this bit is set then the GPU's cache will not be cleaned and invalidated until a GPU job completes which does not have this bit set or a job starts which does not have the BASE_JD_REQ_SKIP_CACHE_START bti set. Do not use if the CPU may read from or partially overwrite memory addressed by the job before the next job without this bit set completes.

7.4.2.17 #define BASE_JD_REQ_SKIP_CACHE_START ((base_jd_core_req)1 << 15)

SW Flag: Skip GPU cache clean and invalidation before starting a GPU job.

If this bit is set then the GPU's cache will not be cleaned and invalidated until a GPU job starts which does not have this bit set or a job completes which does not have the BASE_JD_REQ_SKIP_CACHE_END bit set. Do not use if the CPU may have written to memory addressed by the job since the last job without this bit set was submitted.

7.4.2.18 #define BASE_JD_REQ_SOFT_EVENT_WAIT (BASE_JD_REQ_SOFT_JOB | 0x5)

SW only requirement: event wait/trigger job.

- BASE_JD_REQ_SOFT_EVENT_WAIT: this job will block until the event is set.
- BASE_JD_REQ_SOFT_EVENT_SET: this job sets the event, thus unblocks the other waiting jobs. It completes immediately.
- BASE_JD_REQ_SOFT_EVENT_RESET: this job resets the event, making it possible for other jobs to wait upon. It completes immediately.

7.4.2.19 #define BASE_JD_REQ_SOFT_EXT_RES_MAP (BASE_JD_REQ_SOFT_JOB | 0xb)

SW only requirement: Map external resource

This job requests external resource(s) are mapped once the dependencies of the job have been satisfied. The list of external resources are passed via the jc element of the atom which is a pointer to a .

7.4.2.20 #define BASE_JD_REQ_SOFT_EXT_RES_UNMAP (BASE_JD_REQ_SOFT_JOB | 0xc)

SW only requirement: Unmap external resource

This job requests external resource(s) are unmapped once the dependencies of the job has been satisfied. The list of external resources are passed via the jc element of the atom which is a pointer to a .

7.4.2.21 #define BASE_JD_REQ_SOFT_JIT_ALLOC (BASE_JD_REQ_SOFT_JOB | 0x9)

SW only requirement: Just In Time allocation

This job requests a JIT allocation based on the request in the structure which is passed via the jc element of the atom.

It should be noted that the id entry in must not be reused until it has been released via .

Should this soft job fail it is expected that a soft job to free the JIT allocation is still made.

The job will complete immediately.

7.4.2.22 #define BASE_JD_REQ_SOFT_JIT_FREE (BASE_JD_REQ_SOFT_JOB | 0xa)

SW only requirement: Just In Time free

This job requests a JIT allocation created by to be freed. The ID of the JIT allocation is passed via the jc element of the atom.

The job will complete immediately.

7.4.2.23 #define BASE_JD_REQ_SOFT_JOB ((base_jd_core_req)1 << 9)

SW Only requirement: Software defined job. Jobs with this bit set will not be submitted to the hardware but will cause some action to happen within the driver

7.4.2.24 #define BASE_JD_REQ_SOFT_JOB_OR_DEP(*core_req*)

Value:

```
((core_req & BASE_JD_REQ_SOFT_JOB) || \
    (core_req & BASE_JD_REQ_ATOM_TYPE) == BASE_JD_REQ_DEP)
```

7.4.2.25 #define BASE_JD_REQ_SOFT_JOB_TYPE (BASE_JD_REQ_SOFT_JOB | 0x1f)

Mask of all bits in base_jd_core_req that control the type of a soft job.

7.4.2.26 #define BASE_JD_REQ_SOFT_REPLAY (BASE_JD_REQ_SOFT_JOB | 0x4)

SW Only requirement : Replay job.

If the preceding job fails, the replay job will cause the jobs specified in the list of base_jd_replay_payload pointed to by the jc pointer to be replayed.

A replay job will only cause jobs to be replayed up to BASEP_JD_REPLAY_LIMIT times. If a job fails more than BASEP_JD_REPLAY_LIMIT times then the replay job is failed, as well as any following dependencies.

The replayed jobs will require a number of atom IDs. If there are not enough free atom IDs then the replay job will fail.

If the preceding job does not fail, then the replay job is returned as completed.

The replayed jobs will never be returned to userspace. The preceding failed job will be returned to userspace as failed; the status of this job should be ignored. Completion should be determined by the status of the replay soft job.

In order for the jobs to be replayed, the job headers will have to be modified. The Status field will be reset to NOT_STARTED. If the Job Type field indicates a Vertex Shader Job then it will be changed to Null Job.

The replayed jobs have the following assumptions :

- No external resources. Any required external resources will be held by the replay atom.
- Pre-dependencies are created based on job order.
- Atom numbers are automatically assigned.
- device_nr is set to 0. This is not relevant as BASE_JD_REQ_SPECIFIC_COHERENT_GROUP should not be set.
- Priority is inherited from the replay job.

7.4.2.27 #define BASE_JD_REQ_SPECIFIC_COHERENT_GROUP ((base_jd_core_req)1 << 11)

HW Requirement: Use the base_jd_atom::device_nr field to specify a particular core group

If both BASE_JD_REQ_COHERENT_GROUP and this flag are set, this flag takes priority

This is only guaranteed to work for BASE_JD_REQ_ONLY_COMPUTE atoms.

If the core availability policy is keeping the required core group turned off, then the job will fail with a BASE_JD_↩ EVENT_PM_EVENT error code.

7.4.2.28 #define BASE_JD_REQ_T ((base_jd_core_req)1 << 2)

Requires tiling

7.4.2.29 #define BASE_JD_REQ_V ((base_jd_core_req)1 << 4)

Requires value writeback

7.4.2.30 #define BASEP_JD_REQ_EVENT_NEVER ((base_jd_core_req)1 << 14)

SW Flag: If this bit is set then completion of this atom will not cause an event to be sent to userspace, whether successful or not.

7.4.2.31 #define BASEP_JD_REQ_RESERVED

Value:

```
(~(BASE_JD_REQ_ATOM_TYPE | BASE_JD_REQ_EXTERNAL_RESOURCES
       | \
    BASE_JD_REQ_EVENT_ONLY_ON_FAILURE |
      BASEP_JD_REQ_EVENT_NEVER | \
    BASE_JD_REQ_EVENT_COALESCE | \
    BASE_JD_REQ_COHERENT_GROUP |
      BASE_JD_REQ_SPECIFIC_COHERENT_GROUP | \
    BASE_JD_REQ_FS_AFBC | BASE_JD_REQ_PERMON | \
    BASE_JD_REQ_SKIP_CACHE_START |
      BASE_JD_REQ_SKIP_CACHE_END))
```

These requirement bits are currently unused in base_jd_core_req

7.4.3 Typedef Documentation

7.4.3.1 typedef u8 base_atom_id

Type big enough to store an atom number in

7.4.3.2 typedef struct base_dump_cpu_gpu_counters base_dump_cpu_gpu_counters

Structure for BASE_JD_REQ_SOFT_DUMP_CPU_GPU_COUNTERS jobs.

This structure is stored into the memory pointed to by the `jc` field of base_jd_atom.

It must not occupy the same CPU cache line(s) as any neighboring data. This is to avoid cases where access to pages containing the structure is shared between cached and un-cached memory regions, which would cause memory corruption.

7.4.3.3 typedef struct base_fence base_fence

Base fence handle.

References an underlying base fence object.

7.4.3.4 typedef u32 base_jd_core_req

Job chain hardware requirements.

A job chain must specify what GPU features it needs to allow the driver to schedule the job correctly. By not specifying the correct settings can/will cause an early job termination. Multiple values can be ORed together to specify multiple requirements. Special case is BASE_JD_REQ_DEP, which is used to express complex dependencies, and that doesn't execute anything on the hardware.

7.4.3.5 typedef u8 base_jd_dep_type

Job dependency type.

A flags field will be inserted into the atom structure to specify whether a dependency is a data or ordering dependency (by putting it before/after 'core_req' in the structure it should be possible to add without changing the structure size). When the flag is set for a particular dependency to signal that it is an ordering only dependency then errors will not be propagated.

7.4.3.6 typedef enum base_jd_event_code base_jd_event_code

Job chain event codes.

HW and low-level SW events are represented by event codes. The status of jobs which succeeded are also represented by an event code (see ::BASE_JD_EVENT_DONE). Events are usually reported as part of a ::base_jd_↩ event.

The event codes are encoded in the following way:

- 10:0 - subtype
- 12:11 - type
- 13 - SW success (only valid if the SW bit is set)
- 14 - SW event (HW event if not set)
- 15 - Kernel event (should never be seen in userspace)

Events are split up into ranges as follows:

- BASE_JD_EVENT_RANGE_<description>_START
- BASE_JD_EVENT_RANGE_<description>_END

code is in <description>'s range when:

- `BASE_JD_EVENT_RANGE_<description>_START <= code < BASE_JD_EVENT_RANGE↩ _<description>_END`

Ranges can be asserted for adjacency by testing that the END of the previous is equal to the START of the next. This is useful for optimizing some tests for range.

A limitation is that the last member of this enum must explicitly be handled (with an assert-unreachable statement) in switch statements that use variables of this type. Otherwise, the compiler warns that we have not handled that enum value.

7.4.3.7 typedef struct base_jd_event_v2 base_jd_event_v2

Event reporting structure.

This structure is used by the kernel driver to report information about GPU events. The can either be HW-specific events or low-level SW events, such as job-chain completion.

The event code contains an event type field which can be extracted by ANDing with BASE_JD_SW_EVENT_TY↩PE_MASK.

Based on the event type base_jd_event::data holds:

- BASE_JD_SW_EVENT_JOB : the offset in the ring-buffer for the completed job-chain
- BASE_JD_SW_EVENT_BAG : The address of the ::base_jd_bag that has been completed (ie all contained job-chains have been completed).
- BASE_JD_SW_EVENT_INFO : base_jd_event::data not used

7.4.3.8 typedef struct base_jd_udata base_jd_udata

Per-job data.

This structure is used to store per-job data, and is completely unused by the Base driver. It can be used to store things such as callback function pointer, data to handle job completion. It is guaranteed to be untouched by the Base driver.

7.4.3.9 typedef struct base_stream base_stream

Base stream handle.

References an underlying base stream object.

7.4.4 Enumeration Type Documentation

7.4.4.1 anonymous enum

Job chain event code bits Defines the bits used to create base_jd_event_code.

Enumerator

BASE_JD_SW_EVENT_KERNEL Kernel side event

BASE_JD_SW_EVENT SW defined event

BASE_JD_SW_EVENT_SUCCESS Event idicates success (SW events only)

BASE_JD_SW_EVENT_JOB Job related event

BASE_JD_SW_EVENT_BAG Bag related event

BASE_JD_SW_EVENT_INFO Misc/info event

BASE_JD_SW_EVENT_RESERVED Reserved event type

BASE_JD_SW_EVENT_TYPE_MASK Mask to extract the type from an event code

7.4.4.2 enum base_jd_event_code

Job chain event codes.

HW and low-level SW events are represented by event codes. The status of jobs which succeeded are also represented by an event code (see ::BASE_JD_EVENT_DONE). Events are usually reported as part of a ::base_jd_event.

The event codes are encoded in the following way:

- 10:0 - subtype
- 12:11 - type
- 13 - SW success (only valid if the SW bit is set)
- 14 - SW event (HW event if not set)
- 15 - Kernel event (should never be seen in userspace)

Events are split up into ranges as follows:

- BASE_JD_EVENT_RANGE_<description>_START
- BASE_JD_EVENT_RANGE_<description>_END

code is in <description>'s range when:

- `BASE_JD_EVENT_RANGE_<description>_START <= code < BASE_JD_EVENT_RANGE_<description>_END`

Ranges can be asserted for adjacency by testing that the END of the previous is equal to the START of the next. This is useful for optimizing some tests for range.

A limitation is that the last member of this enum must explicitly be handled (with an assert-unreachable statement) in switch statements that use variables of this type. Otherwise, the compiler warns that we have not handled that enum value.

Enumerator

BASE_JD_EVENT_RANGE_HW_NONFAULT_START Start of HW Non-fault status codes

Note

Obscurely, BASE_JD_EVENT_TERMINATED indicates a real fault, because the job was hard-stopped

BASE_JD_EVENT_NOT_STARTED Can't be seen by userspace, treated as 'previous job done'

BASE_JD_EVENT_STOPPED Can't be seen by userspace, becomes TERMINATED, DONE or JOB_CANCELLED

BASE_JD_EVENT_TERMINATED This is actually a fault status code - the job was hard stopped

BASE_JD_EVENT_ACTIVE Can't be seen by userspace, jobs only returned on complete/fail/cancel

BASE_JD_EVENT_RANGE_HW_NONFAULT_END End of HW Non-fault status codes

Note

Obscurely, BASE_JD_EVENT_TERMINATED indicates a real fault, because the job was hard-stopped

BASE_JD_EVENT_RANGE_HW_FAULT_OR_SW_ERROR_START Start of HW fault and SW Error status codes

BASE_JD_EVENT_RANGE_HW_FAULT_OR_SW_ERROR_END End of HW fault and SW Error status codes

BASE_JD_EVENT_RANGE_SW_SUCCESS_START Start of SW Success status codes

BASE_JD_EVENT_RANGE_SW_SUCCESS_END End of SW Success status codes

BASE_JD_EVENT_RANGE_KERNEL_ONLY_START Start of Kernel-only status codes. Such codes are never returned to user-space

BASE_JD_EVENT_RANGE_KERNEL_ONLY_END End of Kernel-only status codes.

7.4.4.3 enum kbase_atom_coreref_state

States to model state machine processed by kbasep_js_job_check_ref_cores(), which handles retaining cores for power management and affinity management.

The state KBASE_ATOM_COREREF_STATE_RECHECK_AFFINITY prevents an attack where lots of atoms could be submitted before powerup, and each has an affinity chosen that causes other atoms to have an affinity violation. Whilst the affinity was not causing violations at the time it was chosen, it could cause violations thereafter. For example, 1000 jobs could have had their affinity chosen during the powerup time, so any of those 1000 jobs could cause an affinity violation later on.

The attack would otherwise occur because other atoms/contexts have to wait for:

1. the currently running atoms (which are causing the violation) to finish
2. and, the atoms that had their affinity chosen during powerup to finish. These are run preferentially because they don't cause a violation, but instead continue to cause the violation in others.
3. or, the attacker is scheduled out (which might not happen for just 2 contexts)

By re-choosing the affinity (which is designed to avoid violations at the time it's chosen), we break condition (2) of the wait, which minimizes the problem to just waiting for current jobs to finish (which can be bounded if the Job Scheduling Policy has a timer).

Enumerator

KBASE_ATOM_COREREF_STATE_NO_CORES_REQUESTED Starting state: No affinity chosen, and cores must be requested. kbase_jd_atom::affinity==0

KBASE_ATOM_COREREF_STATE_WAITING_FOR_REQUESTED_CORES Cores requested, but waiting for them to be powered. Requested cores given by kbase_jd_atom::affinity

KBASE_ATOM_COREREF_STATE_RECHECK_AFFINITY Cores given by kbase_jd_atom::affinity are powered, but affinity might be out-of-date, so must recheck

KBASE_ATOM_COREREF_STATE_CHECK_AFFINITY_VIOLATIONS Cores given by kbase_jd_atom↩ ::affinity are powered, and affinity is up-to-date, but must check for violations

KBASE_ATOM_COREREF_STATE_READY Cores are powered, kbase_jd_atom::affinity up-to-date, no affinity violations: atom can be submitted to HW

7.4.4.4 enum kbase_jd_atom_state

Enumerator

KBASE_JD_ATOM_STATE_UNUSED Atom is not used

KBASE_JD_ATOM_STATE_QUEUED Atom is queued in JD

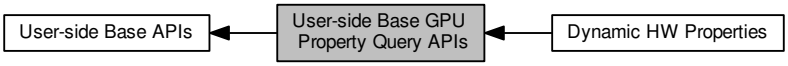
KBASE_JD_ATOM_STATE_IN_JS Atom has been given to JS (is runnable/running)

KBASE_JD_ATOM_STATE_HW_COMPLETED Atom has been completed, but not yet handed back to job dispatcher for dependency resolution

KBASE_JD_ATOM_STATE_COMPLETED Atom has been completed, but not yet handed back to userspace

7.5 User-side Base GPU Property Query APIs

Collaboration diagram for User-side Base GPU Property Query APIs:



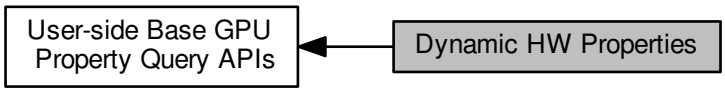
Modules

- Dynamic HW Properties

7.5.1 Detailed Description

7.6 Dynamic HW Properties

Collaboration diagram for Dynamic HW Properties:



Classes

- struct mali_base_gpu_core_props
- struct mali_base_gpu_l2_cache_props
- struct mali_base_gpu_tiler_props
- struct mali_base_gpu_thread_props
- struct mali_base_gpu_coherent_group

  descriptor for a coherent group
- struct mali_base_gpu_coherent_group_info

  Coherency group information.
- struct gpu_raw_gpu_props
- struct base_gpu_props

Macros

- #define **BASE_GPU_NUM_TEXTURE_FEATURES_REGISTERS** 4
- #define **BASE_GPU_NUM_TEXTURE_FEATURES_REGISTERS** 4
- #define **BASE_MAX_COHERENT_GROUPS** 16
- #define **BASE_MAX_COHERENT_GROUPS** 16

Typedefs

- typedef struct base_gpu_props base_gpu_props

7.6.1 Detailed Description

7.6.2 Typedef Documentation

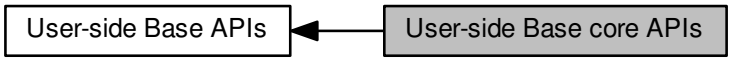
7.6.2.1 typedef struct base_gpu_props base_gpu_props

Return structure for base_get_gpu_props().

NOTE: the raw_props member in this data structure contains the register values from which the value of the other members are derived. The derived members exist to allow for efficient access and/or shielding the details of the layout of the registers.

7.7 User-side Base core APIs

Collaboration diagram for User-side Base core APIs:



Macros

- #define BASE_CONTEXT_CREATE_ALLOWED_FLAGS
- #define BASE_CONTEXT_CREATE_KERNEL_FLAGS ((u32)BASE_CONTEXT_SYSTEM_MONITOR_S↩ UBMIT_DISABLED)
- #define BASEP_CONTEXT_FLAG_JOB_DUMP_DISABLED ((u32)(1 << 31))

Enumerations

- enum base_context_create_flags { BASE_CONTEXT_CREATE_FLAG_NONE = 0, BASE_CONTEXT_C↩ CTX_EMBEDDED = (1u << 0), BASE_CONTEXT_SYSTEM_MONITOR_SUBMIT_DISABLED = (1u << 1) }

7.7.1 Detailed Description

7.7.2 Macro Definition Documentation

7.7.2.1 #define BASE_CONTEXT_CREATE_ALLOWED_FLAGS

Value:

```
(((u32)BASE_CONTEXT_CCTX_EMBEDDED) | \
      ((u32)BASE_CONTEXT_SYSTEM_MONITOR_SUBMIT_DISABLED))
```

Bitpattern describing the base_context_create_flags that can be passed to base_context_init()

7.7.2.2 #define BASE_CONTEXT_CREATE_KERNEL_FLAGS ((u32)BASE_CONTEXT_SYSTEM_MONITOR_SUBMIT↩ _DISABLED)

Bitpattern describing the base_context_create_flags that can be passed to the kernel

7.7.2.3 #define BASEP_CONTEXT_FLAG_JOB_DUMP_DISABLED ((u32)(1 << 31))

Private flag tracking whether job descriptor dumping is disabled

7.7.3 Enumeration Type Documentation

7.7.3.1 enum base_context_create_flags

Flags to pass to ::base_context_init. Flags can be ORed together to enable multiple things.

These share the same space as BASEP_CONTEXT_FLAG_*, and so must not collide with them.

Enumerator

BASE_CONTEXT_CREATE_FLAG_NONE No flags set

BASE_CONTEXT_CCTX_EMBEDDED Base context is embedded in a cctx object (flag used for CINSTR software counter macros)

BASE_CONTEXT_SYSTEM_MONITOR_SUBMIT_DISABLED Base context is a 'System Monitor' context for Hardware counters.

One important side effect of this is that job submission is disabled.

7.8 Base Platform Config GPU Properties

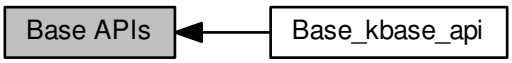
C Pre-processor macros are exposed here to do with Platform Config.

These include:

- GPU Properties that are constant on a particular Midgard Family Implementation e.g. Maximum samples per pixel on Mali-T600.
- General platform config for the GPU, such as the GPU major and minor revison.

7.9 Base APIs

Collaboration diagram for Base APIs:



Modules

- Base_kbase_api

Classes

- struct base_jd_replay_payload

  The payload for a replay job. This must be in GPU memory.
- struct base_jd_replay_jc

  An entry in the linked list of job chains to be replayed. This must be in GPU memory.

Macros

- #define **BASE_JD_REPLAY_F_CHAIN_JOB_LIMIT** 256

Typedefs

- typedef struct base_jd_replay_payload base_jd_replay_payload

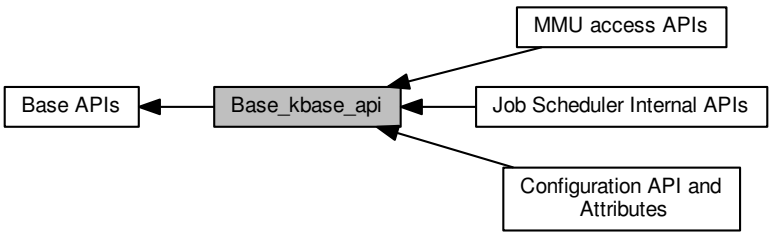
  The payload for a replay job. This must be in GPU memory.
- typedef struct base_jd_replay_jc base_jd_replay_jc

  An entry in the linked list of job chains to be replayed. This must be in GPU memory.

7.9.1 Detailed Description

7.10 Base_kbase_api

Collaboration diagram for Base_kbase_api:



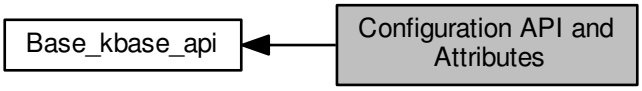
Modules

- Configuration API and Attributes
- Job Scheduler Internal APIs
- MMU access APIs

7.10.1 Detailed Description

7.11 Configuration API and Attributes

Collaboration diagram for Configuration API and Attributes:



Classes

- struct kbase_platform_funcs_conf
- struct kbase_pm_callback_conf
- struct kbase_io_memory_region
- struct kbase_io_resources
- struct kbase_platform_config

Functions

- struct kbase_platform_config * kbase_get_platform_config (void)

  Gets the pointer to platform config.
- int kbasep_platform_device_init (struct kbase_device *kbdev)
- void kbasep_platform_device_term (struct kbase_device *kbdev)
- int kbase_platform_register (void)
- void kbase_platform_unregister (void)

7.11.1 Detailed Description

7.11.2 Function Documentation

7.11.2.1 struct kbase_platform_config* kbase_get_platform_config (void)

Gets the pointer to platform config.

Returns

Pointer to the platform config

7.11.2.2 int kbase_platform_register (void)

kbase_platform_register - Register a platform device for the GPU

This can be used to register a platform device on systems where device tree is not enabled and the platform initialisation code in the kernel doesn't create the GPU device. Where possible device tree should be used instead.

Return: 0 for success, any other fail causes module initialisation to fail

7.11.2.3 void kbase_platform_unregister (void)

kbase_platform_unregister - Unregister a fake platform device

Unregister the platform device created with kbase_platform_register()

7.11.2.4 int kbasep_platform_device_init (struct kbase_device * kbdev)

kbasep_platform_device_init: - Platform specific call to initialize hardware : kbase device pointer

Function calls a platform defined routine if specified in the configuration attributes. The routine can initialize any hardware and context state that is required for the GPU block to function.

Return: 0 if no errors have been found in the config. Negative error code otherwise.

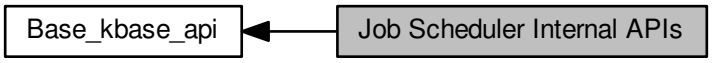
7.11.2.5 void kbasep_platform_device_term (struct kbase_device * kbdev)

kbasep_platform_device_term - Platform specific call to terminate hardware : Kbase device pointer

Function calls a platform defined routine if specified in the configuration attributes. The routine can destroy any platform specific context state and shut down any hardware functionality that are outside of the Power Management callbacks.

7.12 Job Scheduler Internal APIs

Collaboration diagram for Job Scheduler Internal APIs:



Classes

- struct kbasep_atom_req
- struct kbasep_js_device_data

  KBase Device Data Job Scheduler sub-structure.
- struct kbasep_js_kctx_info

  KBase Context Job Scheduling information structure.
- struct kbasep_js_atom_retained_state

Macros

- #define KBASE_JS_MAX_JOB_SUBMIT_PER_SLOT_PER_IRQ 2

  Maximum number of jobs that can be submitted to a job slot whilst inside the IRQ handler.
- #define KBASEP_JS_RETRY_SUBMIT_SLOT_INVALID (-1)
- #define KBASEP_JS_ATOM_RETAINED_STATE_CORE_REQ_INVALID BASE_JD_REQ_DEP
- #define KBASEP_JS_TICK_RESOLUTION_US 1

  The JS timer resolution, in microseconds.
- #define **KBASE_JS_ATOM_SCHED_PRIO_INVALID** -1
- #define **KBASE_JS_ATOM_SCHED_PRIO_DEFAULT** KBASE_JS_ATOM_SCHED_PRIO_MED

Typedefs

- typedef u32 **kbase_context_flags**
- typedef void(∗ kbasep_js_ctx_job_cb) (struct kbase_device ∗kbdev, struct kbase_jd_atom ∗katom)
- typedef u32 kbasep_js_atom_done_code

Enumerations

- enum kbasep_js_ctx_attr { KBASEP_JS_CTX_ATTR_COMPUTE, KBASEP_JS_CTX_ATTR_NON_COM↩PUTE, KBASEP_JS_CTX_ATTR_COMPUTE_ALL_CORES, KBASEP_JS_CTX_ATTR_COUNT }

  Context attributes.
- enum { KBASE_JS_ATOM_DONE_START_NEW_ATOMS = (1u << 0), KBASE_JS_ATOM_DONE_EVI↩CTED_FROM_NEXT = (1u << 1) }
- enum { **KBASE_JS_ATOM_SCHED_PRIO_HIGH** = 0, **KBASE_JS_ATOM_SCHED_PRIO_MED**, **KBAS↩E_JS_ATOM_SCHED_PRIO_LOW**, **KBASE_JS_ATOM_SCHED_PRIO_COUNT** }

Functions

- int kbasep_js_devdata_init (struct kbase_device ∗const kbdev)

  Initialize the Job Scheduler.
- void kbasep_js_devdata_halt (struct kbase_device ∗kbdev)

  Halt the Job Scheduler.
- void kbasep_js_devdata_term (struct kbase_device ∗kbdev)

  Terminate the Job Scheduler.
- int kbasep_js_kctx_init (struct kbase_context ∗const kctx)

  Initialize the Scheduling Component of a struct kbase_context on the Job Scheduler.
- void kbasep_js_kctx_term (struct kbase_context ∗kctx)

  Terminate the Scheduling Component of a struct kbase_context on the Job Scheduler.
- bool kbasep_js_add_job (struct kbase_context ∗kctx, struct kbase_jd_atom ∗atom)

  Add a job chain to the Job Scheduler, and take necessary actions to schedule the context/run the job.
- void kbasep_js_remove_job (struct kbase_device ∗kbdev, struct kbase_context ∗kctx, struct kbase_jd_atom ∗atom)

  Remove a job chain from the Job Scheduler, except for its 'retained state'.
- bool kbasep_js_remove_cancelled_job (struct kbase_device ∗kbdev, struct kbase_context ∗kctx, struct kbase_jd_atom ∗katom)

  Completely remove a job chain from the Job Scheduler, in the case where the job chain was cancelled.
- bool kbasep_js_runpool_retain_ctx (struct kbase_device ∗kbdev, struct kbase_context ∗kctx)

  Refcount a context as being busy, preventing it from being scheduled out.
- bool kbasep_js_runpool_retain_ctx_nolock (struct kbase_device ∗kbdev, struct kbase_context ∗kctx)

  Refcount a context as being busy, preventing it from being scheduled out.
- struct kbase_context ∗ kbasep_js_runpool_lookup_ctx (struct kbase_device ∗kbdev, int as_nr)

  Lookup a context in the Run Pool based upon its current address space and ensure that is stays scheduled in.
- void kbasep_js_runpool_requeue_or_kill_ctx (struct kbase_device ∗kbdev, struct kbase_context ∗kctx, bool has_pm_ref)

  Handling the requeuing/killing of a context that was evicted from the policy queue or runpool.
- void kbasep_js_runpool_release_ctx (struct kbase_device ∗kbdev, struct kbase_context ∗kctx)

  Release a refcount of a context being busy, allowing it to be scheduled out.
- void kbasep_js_runpool_release_ctx_and_katom_retained_state (struct kbase_device ∗kbdev, struct kbase_context ∗kctx, struct kbasep_js_atom_retained_state ∗katom_retained_state)

  Variant of kbasep_js_runpool_release_ctx() that handles additional actions from completing an atom.
- void kbasep_js_runpool_release_ctx_nolock (struct kbase_device ∗kbdev, struct kbase_context ∗kctx)

  Variant of kbase_js_runpool_release_ctx() that assumes that kbasep_js_device_data::runpool_mutex and kbasep_js_kctx_info::ctx::jsctx_mutex are held by the caller, and does not attempt to schedule new contexts.
- void kbasep_js_schedule_privileged_ctx (struct kbase_device ∗kbdev, struct kbase_context ∗kctx)

  Schedule in a privileged context.
- void kbasep_js_release_privileged_ctx (struct kbase_device ∗kbdev, struct kbase_context ∗kctx)

  Release a privileged context, allowing it to be scheduled out.
- void kbase_js_try_run_jobs (struct kbase_device ∗kbdev)

  Try to submit the next job on each slot.
- void kbasep_js_suspend (struct kbase_device ∗kbdev)

  Suspend the job scheduler during a Power Management Suspend event.
- void kbasep_js_resume (struct kbase_device ∗kbdev)

  Resume the Job Scheduler after a Power Management Resume event.
- bool kbase_js_dep_resolved_submit (struct kbase_context ∗kctx, struct kbase_jd_atom ∗katom)

  Submit an atom to the job scheduler.
- void jsctx_ll_flush_to_rb (struct kbase_context ∗kctx, int prio, int js)
- struct kbase_jd_atom ∗ kbase_js_pull (struct kbase_context ∗kctx, int js)

  Pull an atom from a context in the job scheduler for execution.

- void kbase_js_unpull (struct kbase_context *kctx, struct kbase_jd_atom *katom)

  Return an atom to the job scheduler ringbuffer.
- bool kbase_js_complete_atom_wq (struct kbase_context *kctx, struct kbase_jd_atom *katom)

  Complete an atom from jd_done_worker(), removing it from the job scheduler ringbuffer.
- struct kbase_jd_atom * kbase_js_complete_atom (struct kbase_jd_atom *katom, ktime_t *end_timestamp)

  Complete an atom.
- void kbase_js_sched (struct kbase_device *kbdev, int js_mask)

  Submit atoms from all available contexts.
- void kbase_js_zap_context (struct kbase_context *kctx)
- bool kbase_js_is_atom_valid (struct kbase_device *kbdev, struct kbase_jd_atom *katom)

  Validate an atom.
- void kbase_js_set_timeouts (struct kbase_device *kbdev)
- void kbasep_js_ctx_attr_set_initial_attrs (struct kbase_device *kbdev, struct kbase_context *kctx)
- void kbasep_js_ctx_attr_runpool_retain_ctx (struct kbase_device *kbdev, struct kbase_context *kctx)
- bool kbasep_js_ctx_attr_runpool_release_ctx (struct kbase_device *kbdev, struct kbase_context *kctx)
- void kbasep_js_ctx_attr_ctx_retain_atom (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbase_jd_atom *katom)
- bool kbasep_js_ctx_attr_ctx_release_atom (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbasep_js_atom_retained_state *katom_retained_state)

Variables

- const int **kbasep_js_atom_priority_to_relative** [BASE_JD_NR_PRIO_LEVELS]
- const base_jd_prio **kbasep_js_relative_priority_to_atom** [KBASE_JS_ATOM_SCHED_PRIO_COUNT]

7.12.1 Detailed Description

These APIs are Internal to KBase.

7.12.2 Macro Definition Documentation

7.12.2.1 #define KBASE_JS_MAX_JOB_SUBMIT_PER_SLOT_PER_IRQ 2

Maximum number of jobs that can be submitted to a job slot whilst inside the IRQ handler.

This is important because GPU NULL jobs can complete whilst the IRQ handler is running. Otherwise, it potentially allows an unlimited number of GPU NULL jobs to be submitted inside the IRQ handler, which increases IRQ latency.

7.12.2.2 #define KBASEP_JS_ATOM_RETAINED_STATE_CORE_REQ_INVALID BASE_JD_REQ_DEP

base_jd_core_req value signifying 'invalid' for a kbase_jd_atom_retained_state.

See also

kbase_atom_retained_state_is_valid()

7.12.2.3 #define KBASEP_JS_RETRY_SUBMIT_SLOT_INVALID (-1)

Value signifying 'no retry on a slot required' for:

- kbase_js_atom_retained_state::retry_submit_on_slot
- kbase_jd_atom::retry_submit_on_slot

7.12.2.4 #define KBASEP_JS_TICK_RESOLUTION_US 1

The JS timer resolution, in microseconds.

Any non-zero difference in time will be at least this size.

7.12.3 Typedef Documentation

7.12.3.1 typedef u32 kbasep_js_atom_done_code

Combination of KBASE_JS_ATOM_DONE_<...> bits

7.12.3.2 typedef void(* kbasep_js_ctx_job_cb) (struct kbase_device *kbdev, struct kbase_jd_atom *katom)

Callback function run on all of a context's jobs registered with the Job Scheduler

7.12.4 Enumeration Type Documentation

7.12.4.1 anonymous enum

Enumerator

KBASE_JS_ATOM_DONE_START_NEW_ATOMS Bit indicating that new atom should be started because this atom completed

KBASE_JS_ATOM_DONE_EVICTED_FROM_NEXT Bit indicating that the atom was evicted from the J↩ S_NEXT registers

7.12.4.2 enum kbasep_js_ctx_attr

Context attributes.

Each context attribute can be thought of as a boolean value that caches some state information about either the runpool, or the context:

- In the case of the runpool, it is a cache of "Do any contexts owned by the runpool have attribute X?"
- In the case of a context, it is a cache of "Do any atoms owned by the context have attribute X?"

The boolean value of the context attributes often affect scheduling decisions, such as affinities to use and job slots to use.

To accomodate changes of state in the context, each attribute is refcounted in the context, and in the runpool for all running contexts. Specifically:

- The runpool holds a refcount of how many contexts in the runpool have this attribute.
- The context holds a refcount of how many atoms have this attribute.

Enumerator

KBASEP_JS_CTX_ATTR_COMPUTE Attribute indicating a context that contains Compute jobs. That is, the context has jobs of type BASE_JD_REQ_ONLY_COMPUTE

Note

A context can be both 'Compute' and 'Non Compute' if it contains both types of jobs.

KBASEP_JS_CTX_ATTR_NON_COMPUTE Attribute indicating a context that contains Non-Compute jobs. That is, the context has some jobs that are **not** of type BASE_JD_REQ_ONLY_COMPUTE.

Note

A context can be both 'Compute' and 'Non Compute' if it contains both types of jobs.

KBASEP_JS_CTX_ATTR_COMPUTE_ALL_CORES Attribute indicating that a context contains compute-job atoms that aren't restricted to a coherent group, and can run on all cores.

Specifically, this is when the atom's *core_req* satisfy:

- (*core_req* & (BASE_JD_REQ_CS | BASE_JD_REQ_ONLY_COMPUTE | BASE_JD_REQ_T) // uses slot 1 or slot 2
- && !(*core_req* & BASE_JD_REQ_COHERENT_GROUP) // not restricted to coherent groups

Such atoms could be blocked from running if one of the coherent groups is being used by another job slot, so tracking this context attribute allows us to prevent such situations.

Note

This doesn't take into account the 1-coregroup case, where all compute atoms would effectively be able to run on 'all cores', but contexts will still not always get marked with this attribute. Instead, it is the caller's responsibility to take into account the number of coregroups when interpreting this attribute.
Whilst Tiler atoms are normally combined with BASE_JD_REQ_COHERENT_GROUP, it is possible to send such atoms without BASE_JD_REQ_COHERENT_GROUP set. This is an unlikely case, but it's easy enough to handle anyway.

KBASEP_JS_CTX_ATTR_COUNT Must be the last in the enum

7.12.5 Function Documentation

7.12.5.1 void jsctx_ll_flush_to_rb (struct kbase_context * *kctx,* int *prio,* int *js*)

jsctx_ll_flush_to_rb() - Pushes atoms from the linked list to ringbuffer. : Context Pointer : Priority (specifies the queue together with js). : Job slot (specifies the queue together with prio).

Pushes all possible atoms from the linked list to the ringbuffer. Number of atoms are limited to free space in the ringbuffer and number of available atoms in the linked list.

7.12.5.2 struct kbase_jd_atom* kbase_js_complete_atom (struct kbase_jd_atom * *katom,* ktime_t * *end_timestamp*)

Complete an atom.

Most of the work required to complete an atom will be performed by jd_done_worker().

The HW access lock must be held when calling this function.

Parameters

in	*katom*	Pointer to the atom to complete
in	*end_timestamp*	The time that the atom completed (may be NULL)

Return: Atom that has now been unblocked and can now be run, or NULL if none

7.12.5.3 bool kbase_js_complete_atom_wq (struct kbase_context * *kctx,* struct kbase_jd_atom * *katom*)

Complete an atom from jd_done_worker(), removing it from the job scheduler ringbuffer.

If the atom failed then all dependee atoms marked for failure propagation will also fail.

Parameters

in	*kctx*	Context pointer
in	*katom*	Pointer to the atom to complete

Returns

true if the context is now idle (no jobs pulled) false otherwise

7.12.5.4 bool kbase_js_dep_resolved_submit (struct kbase_context * *kctx,* struct kbase_jd_atom * *katom*)

Submit an atom to the job scheduler.

The atom is enqueued on the context's ringbuffer. The caller must have ensured that all dependencies can be represented in the ringbuffer.

Caller must hold jctx->lock

Parameters

in	*kctx*	Context pointer
in	*atom*	Pointer to the atom to submit

Returns

Whether the context requires to be enqueued.

7.12.5.5 bool kbase_js_is_atom_valid (struct kbase_device * *kbdev*, struct kbase_jd_atom * *katom*)

Validate an atom.

This will determine whether the atom can be scheduled onto the GPU. Atoms with invalid combinations of core requirements will be rejected.

Parameters

in	*kbdev*	Device pointer
in	*katom*	Atom to validate

Returns

true if atom is valid false otherwise

7.12.5.6 struct kbase_jd_atom* kbase_js_pull (struct kbase_context * *kctx*, int *js*)

Pull an atom from a context in the job scheduler for execution.

The atom will not be removed from the ringbuffer at this stage.

The HW access lock must be held when calling this function.

Parameters

in	*kctx*	Context to pull from
in	*js*	Job slot to pull from

Returns

Pointer to an atom, or NULL if there are no atoms for this slot that can be currently run.

7.12.5.7 void kbase_js_sched (struct kbase_device * *kbdev*, int *js_mask*)

Submit atoms from all available contexts.

This will attempt to submit as many jobs as possible to the provided job slots. It will exit when either all job slots are full, or all contexts have been used.

Parameters

in	*kbdev*	Device pointer
in	*js_mask*	Mask of job slots to submit to

7.12.5.8 void kbase_js_set_timeouts (struct kbase_device * kbdev)

kbase_js_set_timeouts - update all JS timeouts with user specified data : Device pointer

Timeouts are specified through the 'js_timeouts' sysfs file. If a timeout is set to a positive number then that becomes the new value used, if a timeout is negative then the default is set.

7.12.5.9 void kbase_js_try_run_jobs (struct kbase_device * kbdev)

Try to submit the next job on each slot.

The following locks may be used:

- kbasep_js_device_data::runpool_mutex
- hwaccess_lock

7.12.5.10 void kbase_js_unpull (struct kbase_context * kctx, struct kbase_jd_atom * katom)

Return an atom to the job scheduler ringbuffer.

An atom is 'unpulled' if execution is stopped but intended to be returned to later. The most common reason for this is that the atom has been soft-stopped.

Note that if multiple atoms are to be 'unpulled', they must be returned in the reverse order to which they were originally pulled. It is a programming error to return atoms in any other order.

The HW access lock must be held when calling this function.

Parameters

in	*kctx*	Context pointer
in	*atom*	Pointer to the atom to unpull

7.12.5.11 void kbase_js_zap_context (struct kbase_context * kctx)

kbase_jd_zap_context - Attempt to deschedule a context that is being destroyed : Context pointer

This will attempt to remove a context from any internal job scheduler queues and perform any other actions to ensure a context will not be submitted from.

If the context is currently scheduled, then the caller must wait for all pending jobs to complete before taking any further action.

7.12.5.12 bool kbasep_js_add_job (struct kbase_context * *kctx,* struct kbase_jd_atom * *atom*)

Add a job chain to the Job Scheduler, and take necessary actions to schedule the context/run the job.

This atomically does the following:

- Update the numbers of jobs information
- Add the job to the run pool if necessary (part of init_job)

Once this is done, then an appropriate action is taken:

- If the ctx is scheduled, it attempts to start the next job (which might be this added job)
- Otherwise, and if this is the first job on the context, it enqueues it on the Policy Queue

The Policy's Queue can be updated by this in the following ways:

- In the above case that this is the first job on the context
- If the context is high priority and the context is not scheduled, then it could cause the Policy to schedule out a low-priority context, allowing this context to be scheduled in.

If the context is already scheduled on the RunPool, then adding a job to it is guarenteed not to update the Policy Queue. And so, the caller is guarenteed to not need to try scheduling a context from the Run Pool - it can safely assert that the result is false.

It is a programming error to have more than U32_MAX jobs in flight at a time.

The following locking conditions are made on the caller:

- it must *not* hold kbasep_js_kctx_info::ctx::jsctx_mutex.
- it must *not* hold hwaccess_lock (as this will be obtained internally)
- it must *not* hold kbasep_js_device_data::runpool_mutex (as this will be obtained internally)
- it must *not* hold kbasep_jd_device_data::queue_mutex (again, it's used internally).

Returns

true indicates that the Policy Queue was updated, and so the caller will need to try scheduling a context onto the Run Pool.
false indicates that no updates were made to the Policy Queue, so no further action is required from the caller. This is **always** returned when the context is currently scheduled.

7.12.5.13 bool kbasep_js_ctx_attr_ctx_release_atom (struct kbase_device * *kbdev,* struct kbase_context * *kctx,* struct kbasep_js_atom_retained_state * *katom_retained_state*)

Release all attributes of an atom, given its retained state.

This occurs after (permanently) removing an atom from a context

Requires:

- jsctx mutex
- If the context is scheduled, then runpool_irq spinlock must also be held

This is a no-op when *katom_retained_state* is invalid.

Returns

true indicates a change in ctx attributes state of the runpool. In this state, the scheduler might be able to submit more jobs than previously, and so the caller should ensure kbasep_js_try_run_next_job_nolock() or similar is called sometime later.
false indicates no change in ctx attributes state of the runpool.

7.12.5.14 void kbasep_js_ctx_attr_ctx_retain_atom (struct kbase_device * *kbdev,* struct kbase_context * *kctx,* struct kbase_jd_atom * *katom*)

Retain all attributes of an atom

This occurs on adding an atom to a context

Requires:

- jsctx mutex
- If the context is scheduled, then runpool_irq spinlock must also be held

7.12.5.15 bool kbasep_js_ctx_attr_runpool_release_ctx (struct kbase_device * *kbdev,* struct kbase_context * *kctx*)

Release all attributes of a context

This occurs on scheduling out the context from the runpool (but before is_scheduled is cleared)

Requires:

- jsctx mutex
- runpool_irq spinlock
- ctx->is_scheduled is true

Returns

true indicates a change in ctx attributes state of the runpool. In this state, the scheduler might be able to submit more jobs than previously, and so the caller should ensure kbasep_js_try_run_next_job_nolock() or similar is called sometime later.
false indicates no change in ctx attributes state of the runpool.

7.12.5.16 void kbasep_js_ctx_attr_runpool_retain_ctx (struct kbase_device * *kbdev*, struct kbase_context * *kctx*)

Retain all attributes of a context

This occurs on scheduling in the context on the runpool (but after is_scheduled is set)

Requires:

- jsctx mutex
- runpool_irq spinlock
- ctx->is_scheduled is true

7.12.5.17 void kbasep_js_ctx_attr_set_initial_attrs (struct kbase_device * *kbdev*, struct kbase_context * *kctx*)

Set the initial attributes of a context (when context create flags are set)

Requires:

- Hold the jsctx_mutex

7.12.5.18 void kbasep_js_devdata_halt (struct kbase_device * *kbdev*)

Halt the Job Scheduler.

It is safe to call this on *kbdev* even if it the kbasep_js_device_data sub-structure was never initialized/failed initialization, to give efficient error-path code.

For this to work, the struct kbasep_js_device_data sub-structure of *kbdev* must be zero initialized before passing to the kbasep_js_devdata_init() function. This is to give efficient error path code.

It is a Programming Error to call this whilst there are still kbase_context structures registered with this scheduler.

7.12.5.19 int kbasep_js_devdata_init (struct kbase_device *const *kbdev*)

Initialize the Job Scheduler.

The struct kbasep_js_device_data sub-structure of *kbdev* must be zero initialized before passing to the kbasep_↩ js_devdata_init() function. This is to give efficient error path code.

7.12.5.20 void kbasep_js_devdata_term (struct kbase_device * *kbdev*)

Terminate the Job Scheduler.

It is safe to call this on *kbdev* even if it the kbasep_js_device_data sub-structure was never initialized/failed initialization, to give efficient error-path code.

For this to work, the struct kbasep_js_device_data sub-structure of *kbdev* must be zero initialized before passing to the kbasep_js_devdata_init() function. This is to give efficient error path code.

It is a Programming Error to call this whilst there are still kbase_context structures registered with this scheduler.

7.12.5.21 int kbasep_js_kctx_init (struct kbase_context *const *kctx*)

Initialize the Scheduling Component of a struct kbase_context on the Job Scheduler.

This effectively registers a struct kbase_context with a Job Scheduler.

It does not register any jobs owned by the struct kbase_context with the scheduler. Those must be separately registered by kbasep_js_add_job().

The struct kbase_context must be zero intitialized before passing to the kbase_js_init() function. This is to give efficient error path code.

7.12.5.22 void kbasep_js_kctx_term (struct kbase_context * *kctx*)

Terminate the Scheduling Component of a struct kbase_context on the Job Scheduler.

This effectively de-registers a struct kbase_context from its Job Scheduler

It is safe to call this on a struct kbase_context that has never had or failed initialization of its jctx.sched_info member, to give efficient error-path code.

For this to work, the struct kbase_context must be zero intitialized before passing to the kbase_js_init() function.

It is a Programming Error to call this whilst there are still jobs registered with this context.

7.12.5.23 void kbasep_js_release_privileged_ctx (struct kbase_device * *kbdev*, struct kbase_context * *kctx*)

Release a privileged context, allowing it to be scheduled out.

See kbasep_js_runpool_release_ctx for potential side effects.

The following locking conditions are made on the caller:

- it must *not* hold the hwaccess_lock, because it will be used internally.
- it must *not* hold kbasep_js_kctx_info::ctx::jsctx_mutex.
- it must *not* hold kbasep_js_device_data::runpool_mutex (as this will be obtained internally)
- it must *not* hold the kbase_device::mmu_hw_mutex (as this will be obtained internally)

7.12.5.24 bool kbasep_js_remove_cancelled_job (struct kbase_device * *kbdev,* struct kbase_context * *kctx,* struct kbase_jd_atom * *katom*)

Completely remove a job chain from the Job Scheduler, in the case where the job chain was cancelled.

This is a variant of kbasep_js_remove_job() that takes care of removing all of the retained state too. This is generally useful for cancelled atoms, which need not be handled in an optimal way.

It is a programming error to call this when:

- *atom* is not a job belonging to kctx.
- *atom* has already been removed from the Job Scheduler.
- *atom* is still in the runpool:
  - it is not being killed with kbasep_jd_cancel()

The following locking conditions are made on the caller:

- it must hold kbasep_js_kctx_info::ctx::jsctx_mutex.
- it must *not* hold the hwaccess_lock, (as this will be obtained internally)
- it must *not* hold kbasep_js_device_data::runpool_mutex (as this could be obtained internally)

Returns

true indicates that ctx attributes have changed and the caller should call kbase_js_sched_all() to try to run more jobs
false otherwise

7.12.5.25 void kbasep_js_remove_job (struct kbase_device * *kbdev,* struct kbase_context * *kctx,* struct kbase_jd_atom * *atom*)

Remove a job chain from the Job Scheduler, except for its 'retained state'.

Completely removing a job requires several calls:

- kbasep_js_copy_atom_retained_state(), to capture the 'retained state' of the atom
- kbasep_js_remove_job(), to partially remove the atom from the Job Scheduler
- kbasep_js_runpool_release_ctx_and_katom_retained_state(), to release the remaining state held as part of the job having been run.

In the common case of atoms completing normally, this set of actions is more optimal for spinlock purposes than having kbasep_js_remove_job() handle all of the actions.

In the case of cancelling atoms, it is easier to call kbasep_js_remove_cancelled_job(), which handles all the necessary actions.

It is a programming error to call this when:

- *atom* is not a job belonging to kctx.
- *atom* has already been removed from the Job Scheduler.
- *atom* is still in the runpool

Do not use this for removing jobs being killed by kbase_jd_cancel() - use kbasep_js_remove_cancelled_job() instead.

The following locking conditions are made on the caller:

- it must hold kbasep_js_kctx_info::ctx::jsctx_mutex.

7.12.5.26 void kbasep_js_resume (struct kbase_device * *kbdev*)

Resume the Job Scheduler after a Power Management Resume event.

This restores the actions from kbasep_js_suspend():

- Schedules contexts back into the runpool
- Resumes running atoms on the GPU

7.12.5.27 struct kbase_context* kbasep_js_runpool_lookup_ctx (struct kbase_device * *kbdev*, int *as_nr*)

Lookup a context in the Run Pool based upon its current address space and ensure that is stays scheduled in.

The context is refcounted as being busy to prevent it from scheduling out. It must be released with kbasep_js_↩ runpool_release_ctx() when it is no longer required to stay scheduled in.

Note

This function can safely be called from IRQ context.

The following locking conditions are made on the caller:

- it must *not* hold the hwaccess_lock, because it will be used internally. If the hwaccess_lock is already held, then the caller should use kbasep_js_runpool_lookup_ctx_nolock() instead.

Returns

a valid struct kbase_context on success, which has been refcounted as being busy.
NULL on failure, indicating that no context was found in *as_nr*

7.12.5.28 void kbasep_js_runpool_release_ctx (struct kbase_device * *kbdev*, struct kbase_context * *kctx*)

Release a refcount of a context being busy, allowing it to be scheduled out.

When the refcount reaches zero and the context *might* be scheduled out (depending on whether the Scheudling Policy has deemed it so, or if it has run out of jobs).

If the context does get scheduled out, then The following actions will be taken as part of deschduling a context:

- For the context being descheduled:
  - If the context is in the processing of dying (all the jobs are being removed from it), then descheduling also kills off any jobs remaining in the context.
  - If the context is not dying, and any jobs remain after descheduling the context then it is re-enqueued to the Policy's Queue.
  - Otherwise, the context is still known to the scheduler, but remains absent from the Policy Queue until a job is next added to it.
  - In all descheduling cases, the Power Manager active reference (obtained during kbasep_js_try_↩ schedule_head_ctx()) is released (kbase_pm_context_idle()).

Whilst the context is being descheduled, this also handles actions that cause more atoms to be run:

- Attempt submitting atoms when the Context Attributes on the Runpool have changed. This is because the context being scheduled out could mean that there are more opportunities to run atoms.
- Attempt submitting to a slot that was previously blocked due to affinity restrictions. This is usually only necessary when releasing a context happens as part of completing a previous job, but is harmless nonetheless.
- Attempt scheduling in a new context (if one is available), and if necessary, running a job from that new context.

Unlike retaining a context in the runpool, this function **cannot** be called from IRQ context.

It is a programming error to call this on a *kctx* that is not currently scheduled, or that already has a zero refcount.

The following locking conditions are made on the caller:

- it must *not* hold the hwaccess_lock, because it will be used internally.
- it must *not* hold kbasep_js_kctx_info::ctx::jsctx_mutex.
- it must *not* hold kbasep_js_device_data::runpool_mutex (as this will be obtained internally)
- it must *not* hold the kbase_device::mmu_hw_mutex (as this will be obtained internally)
- it must *not* hold kbasep_jd_device_data::queue_mutex (as this will be obtained internally)

7.12.5.29 void kbasep_js_runpool_release_ctx_and_katom_retained_state (struct kbase_device * *kbdev,* struct kbase_context * *kctx,* struct kbasep_js_atom_retained_state * *katom_retained_state*)

Variant of kbasep_js_runpool_release_ctx() that handles additional actions from completing an atom.

This is usually called as part of completing an atom and releasing the refcount on the context held by the atom.

Therefore, the extra actions carried out are part of handling actions queued on a completed atom, namely:

- Releasing the atom's context attributes
- Retrying the submission on a particular slot, because we couldn't submit on that slot from an IRQ handler.

The locking conditions of this function are the same as those for kbasep_js_runpool_release_ctx()

7.12.5.30 void kbasep_js_runpool_requeue_or_kill_ctx (struct kbase_device * *kbdev,* struct kbase_context * *kctx,* bool *has_pm_ref*)

Handling the requeuing/killing of a context that was evicted from the policy queue or runpool.

This should be used whenever handing off a context that has been evicted from the policy queue or the runpool:

- If the context is not dying and has jobs, it gets re-added to the policy queue
- Otherwise, it is not added

In addition, if the context is dying the jobs are killed asynchronously.

In all cases, the Power Manager active reference is released (kbase_pm_context_idle()) whenever the has_pm_ref parameter is true. *has_pm_ref* must be set to false whenever the context was not previously in the runpool and does not hold a Power Manager active refcount. Note that contexts in a rollback of kbasep_js_try_schedule_head_ctx() might have an active refcount even though they weren't in the runpool.

The following locking conditions are made on the caller:

- it must hold kbasep_js_kctx_info::ctx::jsctx_mutex.
- it must *not* hold kbasep_jd_device_data::queue_mutex (as this will be obtained internally)

7.12.5.31 bool kbasep_js_runpool_retain_ctx (struct kbase_device * *kbdev,* struct kbase_context * *kctx*)

Refcount a context as being busy, preventing it from being scheduled out.

Note

This function can safely be called from IRQ context.

The following locking conditions are made on the caller:

- it must *not* hold mmu_hw_mutex and hwaccess_lock, because they will be used internally.

Returns

value != false if the retain succeeded, and the context will not be scheduled out.
false if the retain failed (because the context is being/has been scheduled out).

7.12.5.32 bool kbasep_js_runpool_retain_ctx_nolock (struct kbase_device * *kbdev,* struct kbase_context * *kctx*)

Refcount a context as being busy, preventing it from being scheduled out.

Note

This function can safely be called from IRQ context.

The following locks must be held by the caller:

- mmu_hw_mutex, hwaccess_lock

Returns

value != false if the retain succeeded, and the context will not be scheduled out.
false if the retain failed (because the context is being/has been scheduled out).

7.12.5.33 void kbasep_js_schedule_privileged_ctx (struct kbase_device * *kbdev,* struct kbase_context * *kctx*)

Schedule in a privileged context.

This schedules a context in regardless of the context priority. If the runpool is full, a context will be forced out of the runpool and the function will wait for the new context to be scheduled in. The context will be kept scheduled in (and the corresponding address space reserved) until kbasep_js_release_privileged_ctx is called).

The following locking conditions are made on the caller:

- it must *not* hold the hwaccess_lock, because it will be used internally.
- it must *not* hold kbasep_js_device_data::runpool_mutex (as this will be obtained internally)
- it must *not* hold the kbase_device::mmu_hw_mutex (as this will be obtained internally)
- it must *not* hold kbasep_jd_device_data::queue_mutex (again, it's used internally).
- it must *not* hold kbasep_js_kctx_info::ctx::jsctx_mutex, because it will be used internally.

7.12.5.34 void kbasep_js_suspend (struct kbase_device * *kbdev*)

Suspend the job scheduler during a Power Management Suspend event.

Causes all contexts to be removed from the runpool, and prevents any contexts from (re)entering the runpool.

This does not handle suspending the one privileged context: the caller must instead do this by by suspending the GPU HW Counter Instrumentation.

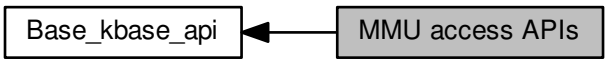
This will eventually cause all Power Management active references held by contexts on the runpool to be released, without running any more atoms.

The caller must then wait for all Power Mangement active refcount to become zero before completing the suspend.

The emptying mechanism may take some time to complete, since it can wait for jobs to complete naturally instead of forcing them to end quickly. However, this is bounded by the Job Scheduler's Job Timeouts. Hence, this function is guaranteed to complete in a finite time.

7.13 MMU access APIs

Collaboration diagram for MMU access APIs:



Enumerations

- enum kbase_mmu_fault_type {
  KBASE_MMU_FAULT_TYPE_UNKNOWN = 0, **KBASE_MMU_FAULT_TYPE_PAGE**, **KBASE_MMU_F↩AULT_TYPE_BUS**, **KBASE_MMU_FAULT_TYPE_PAGE_UNEXPECTED**,
  KBASE_MMU_FAULT_TYPE_BUS_UNEXPECTED }

  MMU fault type descriptor.

Functions

- void kbase_mmu_hw_configure (struct kbase_device ∗kbdev, struct kbase_as ∗as, struct kbase_context ∗kctx)

  Configure an address space for use.
- int kbase_mmu_hw_do_operation (struct kbase_device ∗kbdev, struct kbase_as ∗as, struct kbase_context ∗kctx, u64 vpfn, u32 nr, u32 type, unsigned int handling_irq)

  Issue an operation to the MMU.
- void kbase_mmu_hw_clear_fault (struct kbase_device ∗kbdev, struct kbase_as ∗as, struct kbase_context ∗kctx, enum kbase_mmu_fault_type type)

  Clear a fault that has been previously reported by the MMU.
- void kbase_mmu_hw_enable_fault (struct kbase_device ∗kbdev, struct kbase_as ∗as, struct kbase_context ∗kctx, enum kbase_mmu_fault_type type)

  Enable fault that has been previously reported by the MMU.

7.13.1 Detailed Description

7.13.2 Function Documentation

7.13.2.1 void kbase_mmu_hw_clear_fault (struct kbase_device ∗ *kbdev,* struct kbase_as ∗ *as,* struct kbase_context ∗ *kctx,* enum kbase_mmu_fault_type *type*)

Clear a fault that has been previously reported by the MMU.

Clear a bus error or page fault that has been reported by the MMU.

Parameters

in	*kbdev*	kbase device to clear the fault from.
in	*as*	address space to clear the fault from.
in	*kctx*	kbase context to clear the fault from or NULL.
in	*type*	The type of fault that needs to be cleared.

7.13.2.2 void kbase_mmu_hw_configure (struct kbase_device * *kbdev*, struct kbase_as * *as*, struct kbase_context * *kctx*)

Configure an address space for use.

Configure the MMU using the address space details setup in the kbase_context structure.

Parameters

in	*kbdev*	kbase device to configure.
in	*as*	address space to configure.
in	*kctx*	kbase context to configure.

7.13.2.3 int kbase_mmu_hw_do_operation (struct kbase_device * *kbdev*, struct kbase_as * *as*, struct kbase_context * *kctx*, u64 *vpfn*, u32 *nr*, u32 *type*, unsigned int *handling_irq*)

Issue an operation to the MMU.

Issue an operation (MMU invalidate, MMU flush, etc) on the address space that is associated with the provided kbase_context over the specified range

Parameters

in	*kbdev*	kbase device to issue the MMU operation on.
in	*as*	address space to issue the MMU operation on.
in	*kctx*	kbase context to issue the MMU operation on.
in	*vpfn*	MMU Virtual Page Frame Number to start the operation on.
in	*nr*	Number of pages to work on.
in	*type*	Operation type (written to ASn_COMMAND).
in	*handling_irq*	Is this operation being called during the handling of an interrupt?

Returns

Zero if the operation was successful, non-zero otherwise.

7.13.2.4 void kbase_mmu_hw_enable_fault (struct kbase_device * *kbdev*, struct kbase_as * *as*, struct kbase_context * *kctx*, enum kbase_mmu_fault_type *type*)

Enable fault that has been previously reported by the MMU.

After a page fault or bus error has been reported by the MMU these will be disabled. After these are handled this function needs to be called to enable the page fault or bus error fault again.

Parameters

in	*kbdev*	kbase device to again enable the fault from.
in	*as*	address space to again enable the fault from.
in	*kctx*	kbase context to again enable the fault from.
in	*type*	The type of fault that needs to be enabled again.

Chapter 8

Class Documentation

8.1 base_dependency Struct Reference

Public Attributes

- base_atom_id atom_id
- base_jd_dep_type dependency_type

8.1.1 Member Data Documentation

8.1.1.1 base_atom_id base_dependency::atom_id

An atom number

8.1.1.2 base_jd_dep_type base_dependency::dependency_type

Dependency type

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.2 base_dump_cpu_gpu_counters Struct Reference

Structure for BASE_JD_REQ_SOFT_DUMP_CPU_GPU_COUNTERS jobs.

```
#include <mali_base_kernel.h>
```

Public Attributes

- u64 **system_time**
- u64 **cycle_counter**
- u64 **sec**
- u32 **usec**
- u8 **padding** [36]

8.2.1 Detailed Description

Structure for BASE_JD_REQ_SOFT_DUMP_CPU_GPU_COUNTERS jobs.

This structure is stored into the memory pointed to by the `jc` field of base_jd_atom.

It must not occupy the same CPU cache line(s) as any neighboring data. This is to avoid cases where access to pages containing the structure is shared between cached and un-cached memory regions, which would cause memory corruption.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.3 base_external_resource Struct Reference

Public Attributes

- u64 **ext_resource**

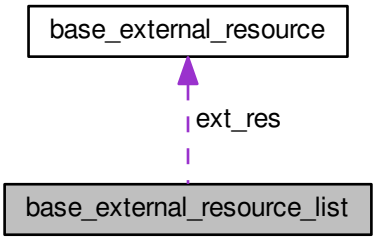
The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.4 base_external_resource_list Struct Reference

`#include <mali_base_kernel.h>`

Collaboration diagram for base_external_resource_list:



Public Attributes

- u64 **count**
- struct base_external_resource **ext_res** [1]

8.4.1 Detailed Description

struct base_external_resource_list - Structure which describes a list of external resources. : The number of resources. : Array of external resources which is sized at allocation time.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.5 base_fence Struct Reference

```
#include <mali_base_kernel.h>
```

Public Attributes

- struct {
  int **fd**
  int **stream_fd**
  } **basep**

8.5.1 Detailed Description

Base fence handle.

References an underlying base fence object.

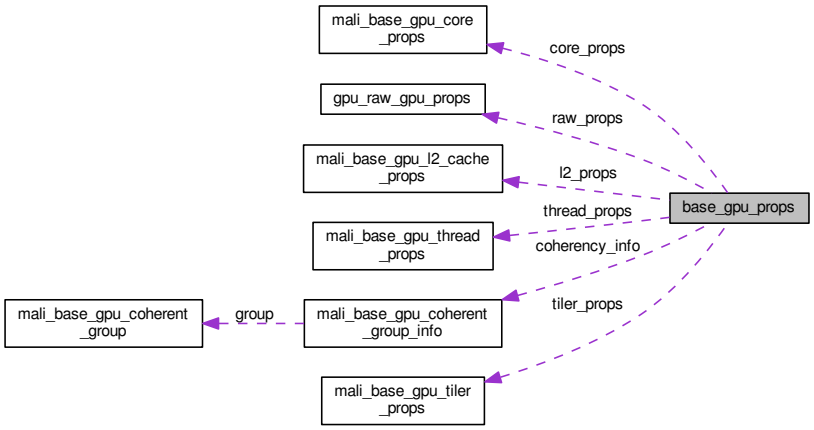
The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.6 base_gpu_props Struct Reference

`#include <mali_base_kernel.h>`

Collaboration diagram for base_gpu_props:



Public Attributes

- struct mali_base_gpu_core_props **core_props**
- struct mali_base_gpu_l2_cache_props **l2_props**
- u64 **unused_1**
- struct mali_base_gpu_tiler_props **tiler_props**
- struct mali_base_gpu_thread_props **thread_props**
- struct gpu_raw_gpu_props raw_props
- struct mali_base_gpu_coherent_group_info coherency_info

8.6.1 Detailed Description

Return structure for base_get_gpu_props().

NOTE: the raw_props member in this data structure contains the register values from which the value of the other members are derived. The derived members exist to allow for efficient access and/or shielding the details of the layout of the registers.

8.6.2 Member Data Documentation

8.6.2.1 struct mali_base_gpu_coherent_group_info base_gpu_props::coherency_info

This must be last member of the structure

8.6.2.2 struct gpu_raw_gpu_props base_gpu_props::raw_props

This member is large, likely to be 128 bytes

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.7 base_import_handle Struct Reference

```
#include <mali_base_kernel.h>
```

Public Attributes

- struct {
  u64 **handle**
  } **basep**

8.7.1 Detailed Description

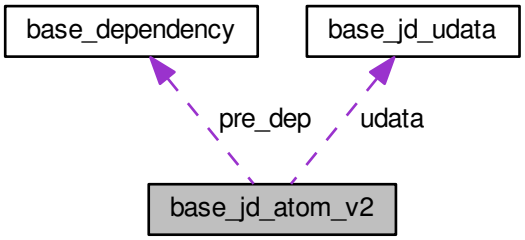
Handle to represent imported memory object. Simple opague handle to imported memory, can't be used with anything but base_external_resource_init to bind to an atom.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.8 base_jd_atom_v2 Struct Reference

Collaboration diagram for base_jd_atom_v2:



Public Attributes

- u64 jc
- struct base_jd_udata udata
- u64 extres_list
- u16 nr_extres
- u16 compat_core_req
- struct base_dependency pre_dep [2]
- base_atom_id atom_number
- base_jd_prio prio
- u8 device_nr
- u8 **padding** [1]
- base_jd_core_req core_req

8.8.1 Member Data Documentation

8.8.1.1 base_atom_id base_jd_atom_v2::atom_number

unique number to identify the atom

8.8.1.2 u16 base_jd_atom_v2::compat_core_req

core requirements which correspond to the legacy support for UK 10.2

8.8.1.3 base_jd_core_req base_jd_atom_v2::core_req

core requirements

8.8.1.4 u8 base_jd_atom_v2::device_nr

coregroup when BASE_JD_REQ_SPECIFIC_COHERENT_GROUP specified

8.8.1.5 u64 base_jd_atom_v2::extres_list

list of external resources

8.8.1.6 u64 base_jd_atom_v2::jc

job-chain GPU address

8.8.1.7 u16 base_jd_atom_v2::nr_extres

nr of external resources

8.8.1.8 struct base_dependency base_jd_atom_v2::pre_dep[2]

pre-dependencies, one need to use SETTER function to assign this field, this is done in order to reduce possibility of improper assigment of a dependency field

8.8.1.9 base_jd_prio base_jd_atom_v2::prio

Atom priority. Refer to base_jd_prio for more details

8.8.1.10 struct base_jd_udata base_jd_atom_v2::udata

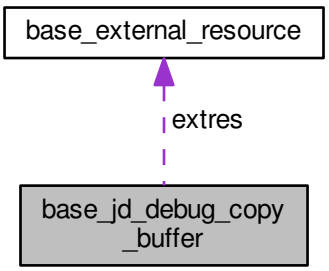
user data

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.9 base_jd_debug_copy_buffer Struct Reference

Collaboration diagram for base_jd_debug_copy_buffer:



Public Attributes

- u64 **address**
- u64 **size**
- struct base_external_resource **extres**

The documentation for this struct was generated from the following file:

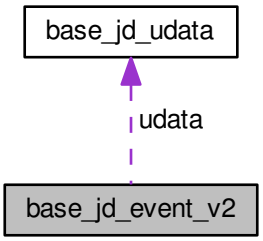
- gpu/arm/midgard/mali_base_kernel.h

8.10 base_jd_event_v2 Struct Reference

Event reporting structure.

`#include <mali_base_kernel.h>`

Collaboration diagram for base_jd_event_v2:



Public Attributes

- base_jd_event_code event_code
- base_atom_id atom_number
- struct base_jd_udata udata

8.10.1 Detailed Description

Event reporting structure.

This structure is used by the kernel driver to report information about GPU events. The can either be HW-specific events or low-level SW events, such as job-chain completion.

The event code contains an event type field which can be extracted by ANDing with BASE_JD_SW_EVENT_TY↩PE_MASK.

Based on the event type base_jd_event::data holds:

- BASE_JD_SW_EVENT_JOB : the offset in the ring-buffer for the completed job-chain
- BASE_JD_SW_EVENT_BAG : The address of the ::base_jd_bag that has been completed (ie all contained job-chains have been completed).
- BASE_JD_SW_EVENT_INFO : base_jd_event::data not used

8.10.2 Member Data Documentation

8.10.2.1 base_atom_id base_jd_event_v2::atom_number

the atom number that has completed

8.10.2.2 base_jd_event_code base_jd_event_v2::event_code

event code

8.10.2.3 struct base_jd_udata base_jd_event_v2::udata

user data

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.11 base_jd_replay_jc Struct Reference

An entry in the linked list of job chains to be replayed. This must be in GPU memory.

```
#include <mali_base_kernel.h>
```

Public Attributes

- u64 next
- u64 jc

8.11.1 Detailed Description

An entry in the linked list of job chains to be replayed. This must be in GPU memory.

8.11.2 Member Data Documentation

8.11.2.1 u64 base_jd_replay_jc::jc

Pointer to the job chain.

8.11.2.2 u64 base_jd_replay_jc::next

Pointer to next entry in the list. A setting of NULL indicates the end of the list.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.12 base_jd_replay_payload Struct Reference

The payload for a replay job. This must be in GPU memory.

`#include <mali_base_kernel.h>`

Public Attributes

- u64 tiler_jc_list
- u64 fragment_jc
- u64 tiler_heap_free
- u16 fragment_hierarchy_mask
- u16 tiler_hierarchy_mask
- u32 hierarchy_default_weight
- base_jd_core_req tiler_core_req
- base_jd_core_req fragment_core_req

8.12.1 Detailed Description

The payload for a replay job. This must be in GPU memory.

8.12.2 Member Data Documentation

8.12.2.1 base_jd_core_req base_jd_replay_payload::fragment_core_req

Core requirements for the fragment job chain

8.12.2.2 u16 base_jd_replay_payload::fragment_hierarchy_mask

Hierarchy mask for the replayed fragment jobs. May be zero.

8.12.2.3 u64 base_jd_replay_payload::fragment_jc

Pointer to the fragment job chain.

8.12.2.4 u32 base_jd_replay_payload::hierarchy_default_weight

Default weight to be used for hierarchy levels not in the original mask.

8.12.2.5 base_jd_core_req base_jd_replay_payload::tiler_core_req

Core requirements for the tiler job chain

8.12.2.6 u64 base_jd_replay_payload::tiler_heap_free

Pointer to the tiler heap free FBD field to be modified.

8.12.2.7 u16 base_jd_replay_payload::tiler_hierarchy_mask

Hierarchy mask for the replayed tiler jobs. May be zero.

8.12.2.8 u64 base_jd_replay_payload::tiler_jc_list

Pointer to the first entry in the base_jd_replay_jc list. These will be replayed in **reverse** order (so that extra ones can be added to the head in future soft jobs without affecting this soft job)

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.13 base_jd_udata Struct Reference

Per-job data.

```
#include <mali_base_kernel.h>
```

Public Attributes

- u64 blob [2]

8.13.1 Detailed Description

Per-job data.

This structure is used to store per-job data, and is completely unused by the Base driver. It can be used to store things such as callback function pointer, data to handle job completion. It is guaranteed to be untouched by the Base driver.

8.13.2 Member Data Documentation

8.13.2.1 u64 base_jd_udata::blob[2]

per-job data array

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.14 base_jit_alloc_info Struct Reference

`#include <mali_base_kernel.h>`

Public Attributes

- u64 **gpu_alloc_addr**
- u64 **va_pages**
- u64 **commit_pages**
- u64 **extent**
- u8 **id**

8.14.1 Detailed Description

struct base_jit_alloc_info - Structure which describes a JIT allocation request. : The GPU virtual address to write the JIT allocated GPU virtual address to. : The minimum number of virtual pages required. : The minimum number of physical pages which should back the allocation. : Granularity of physical pages to grow the allocation by during a fault. : Unique ID provided by the caller, this is used to pair allocation and free requests. Zero is not a valid value.

The documentation for this struct was generated from the following file:

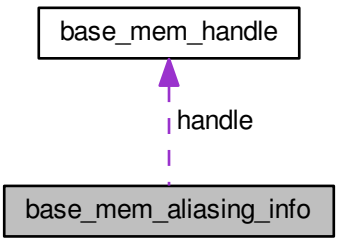
- gpu/arm/midgard/mali_base_kernel.h

8.15 base_mem_aliasing_info Struct Reference

Memory aliasing info.

`#include <mali_base_kernel.h>`

Collaboration diagram for base_mem_aliasing_info:



Public Attributes

- base_mem_handle **handle**
- u64 **offset**
- u64 **length**

8.15.1 Detailed Description

Memory aliasing info.

Describes a memory handle to be aliased. A subset of the handle can be chosen for aliasing, given an offset and a length. A special handle BASE_MEM_WRITE_ALLOC_PAGES_HANDLE is used to represent a region where a special page is mapped with a write-alloc cache setup, typically used when the write result of the GPU isn't needed, but the GPU must write anyway.

Offset and length are specified in pages. Offset must be within the size of the handle. Offset+length must not overrun the size of the handle.

Handle to alias, can be BASE_MEM_WRITE_ALLOC_PAGES_HANDLE Offset within the handle to start aliasing from, in pages. Not used with BASE_MEM_WRITE_ALLOC_PAGES_HANDLE. Length to alias, in pages. For BASE_MEM_WRITE_ALLOC_PAGES_HANDLE specifies the number of times the special page is needed.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.16 base_mem_handle Struct Reference

Public Attributes

- struct {
  u64 **handle**
  } **basep**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.17 base_mem_import_user_buffer Struct Reference

```
#include <mali_base_kernel.h>
```

Public Attributes

- u64 **ptr**
- u64 **length**

8.17.1 Detailed Description

struct base_mem_import_user_buffer - Handle of an imported user buffer

: address of imported user buffer : length of imported user buffer in bytes

This structure is used to represent a handle of an imported user buffer.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.18 base_profiling_controls Struct Reference

Public Attributes

- u32 **profiling_controls** [FBDUMP_CONTROL_MAX]

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.19 base_stream Struct Reference

```
#include <mali_base_kernel.h>
```

Public Attributes

- struct {
    int **fd**
  } **basep**

8.19.1 Detailed Description

Base stream handle.

References an underlying base stream object.

The documentation for this struct was generated from the following file:

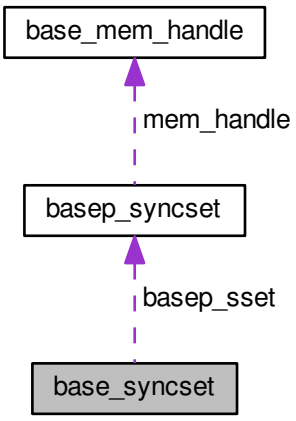
- gpu/arm/midgard/mali_base_kernel.h

8.20 base_syncset Struct Reference

a basic memory operation (sync-set).

```
#include <mali_base_kernel.h>
```

Collaboration diagram for base_syncset:



Public Attributes

- struct basep_syncset **basep_sset**

8.20.1 Detailed Description

a basic memory operation (sync-set).

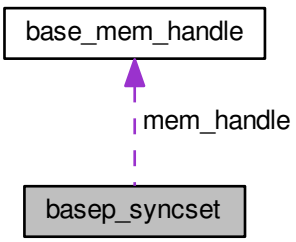
The content of this structure is private, and should only be used by the accessors.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.21 basep_syncset Struct Reference

Collaboration diagram for basep_syncset:



Public Attributes

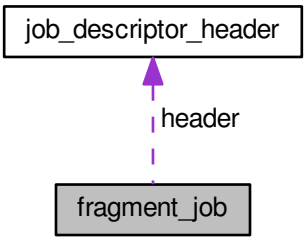
- base_mem_handle **mem_handle**
- u64 **user_addr**
- u64 **size**
- u8 **type**
- u8 **padding** [7]

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_mem_priv.h

8.22 fragment_job Struct Reference

Collaboration diagram for fragment_job:



Public Attributes

- struct job_descriptor_header **header**
- u32 **x** [2]
- union {
  u64 **_64**
  u32 **_32**
  } **fragment_fbd**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_replay.c

8.23 gpu_raw_gpu_props Struct Reference

```
#include <mali_base_kernel.h>
```

Public Attributes

- u64 **shader_present**
- u64 **tiler_present**
- u64 **l2_present**
- u64 **stack_present**
- u32 **l2_features**
- u32 **suspend_size**
- u32 **mem_features**
- u32 **mmu_features**
- u32 **as_present**
- u32 **js_present**
- u32 **js_features** [GPU_MAX_JOB_SLOTS]
- u32 **tiler_features**
- u32 **texture_features** [BASE_GPU_NUM_TEXTURE_FEATURES_REGISTERS]
- u32 **gpu_id**
- u32 **thread_max_threads**
- u32 **thread_max_workgroup_size**
- u32 **thread_max_barrier_size**
- u32 **thread_features**
- u32 **coherency_mode**

8.23.1 Detailed Description

A complete description of the GPU's Hardware Configuration Discovery registers.

The information is presented inefficiently for access. For frequent access, the values should be better expressed in an unpacked form in the base_gpu_props structure.

The raw properties in gpu_raw_gpu_props are necessary to allow a user of the Mali Tools (e.g. PAT) to determine "Why is this device behaving differently?". In this case, all information about the configuration is potentially useful, but it **does not need to be processed by the driver**. Instead, the raw registers can be processed by the Mali Tools software on the host PC.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.24 job_descriptor_header Struct Reference

Public Attributes

- u32 **exception_status**
- u32 **first_incomplete_task**
- u64 **fault_pointer**
- u8 **job_descriptor_size**: 1
- u8 **job_type**: 7
- u8 **job_barrier**: 1
- u8 **_reserved_01**: 1
- u8 **_reserved_1**: 1
- u8 **_reserved_02**: 1

- u8 **_reserved_03**: 1
- u8 **_reserved_2**: 1
- u8 **_reserved_04**: 1
- u8 **_reserved_05**: 1
- u16 **job_index**
- u16 **job_dependency_index_1**
- u16 **job_dependency_index_2**
- union {
  u64 **_64**
  u32 **_32**
  } **next_job**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.25 jsctx_queue Struct Reference

`#include <mali_kbase_defs.h>`

Public Attributes

- struct rb_root **runnable_tree**
- struct list_head **x_dep_head**

8.25.1 Detailed Description

struct jsctx_queue - JS context atom queue : Root of RB-tree containing currently runnable atoms on this job slot. : Head item of the linked list of atoms blocked on cross-slot dependencies. Atoms on this list will be moved to the runnable_tree when the blocking atom completes.

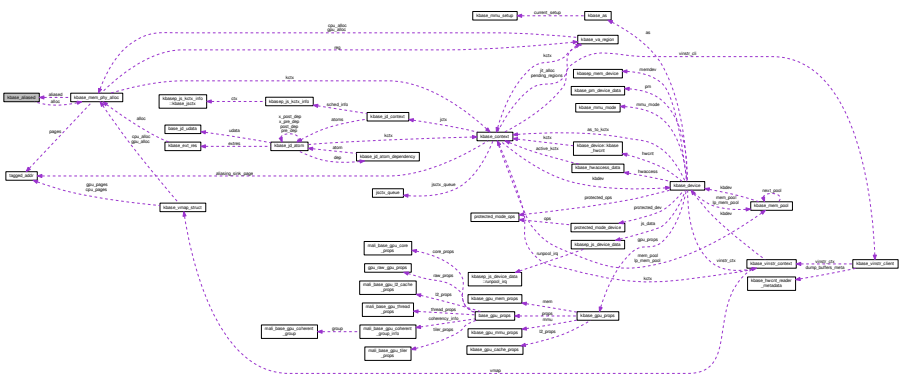
hwaccess_lock must be held when accessing this structure.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.26 kbase_aliased Struct Reference

Collaboration diagram for kbase_aliased:



Public Attributes

- struct kbase_mem_phy_alloc * **alloc**
- u64 **offset**
- u64 **length**

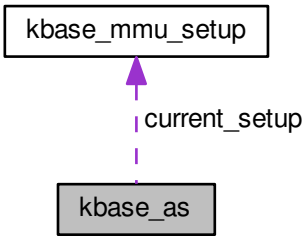
The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_mem.h

8.27 kbase_as Struct Reference

`#include <mali_kbase_defs.h>`

Collaboration diagram for kbase_as:



Public Attributes

- int **number**
- struct workqueue_struct * **pf_wq**
- struct work_struct **work_pagefault**
- struct work_struct **work_busfault**
- enum kbase_mmu_fault_type **fault_type**
- bool **protected_mode**
- u32 **fault_status**
- u64 **fault_addr**
- u64 **fault_extra_addr**
- struct kbase_mmu_setup **current_setup**
- struct workqueue_struct * **poke_wq**
- struct work_struct **poke_work**
- int poke_refcount
- kbase_as_poke_state poke_state
- struct hrtimer **poke_timer**

8.27.1 Detailed Description

Important: Our code makes assumptions that a struct kbase_as structure is always at kbase_device->as[number]. This is used to recover the containing struct kbase_device from a struct kbase_as structure.

Therefore, struct kbase_as structures must not be allocated anywhere else.

8.27.2 Member Data Documentation

8.27.2.1 int kbase_as::poke_refcount

Protected by hwaccess_lock

8.27.2.2 kbase_as_poke_state kbase_as::poke_state

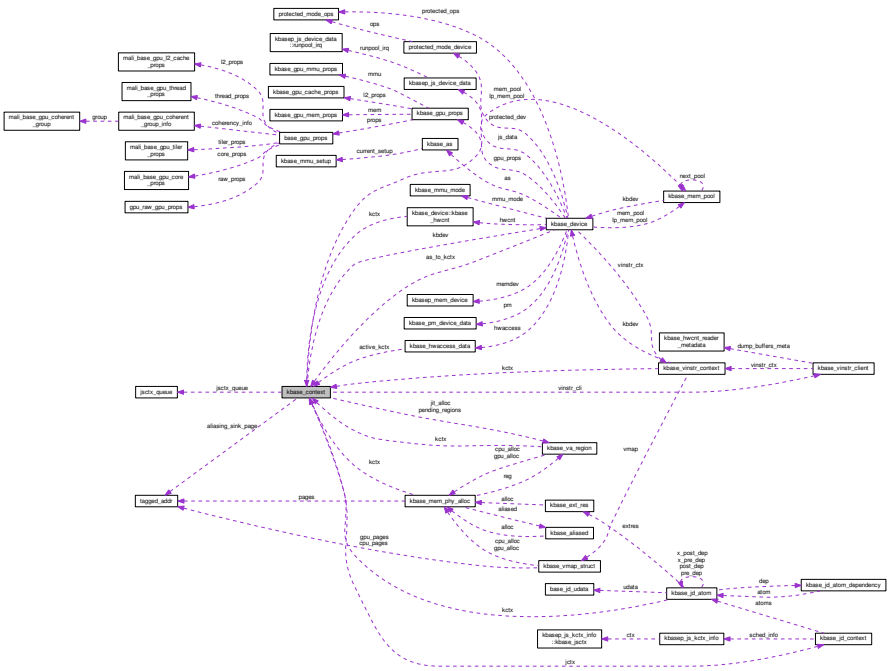
Protected by hwaccess_lock

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.28 kbase_context Struct Reference

Collaboration diagram for kbase_context:



Public Attributes

- struct file * **filp**
- struct kbase_device * **kbdev**
- u32 **id**
- unsigned long **api_version**
- phys_addr_t **pgd**
- struct list_head **event_list**
- struct list_head **event_coalesce_list**
- struct mutex **event_mutex**
- atomic_t **event_closed**
- struct workqueue_struct * **event_workq**
- atomic_t **event_count**
- int **event_coalesce_count**
- atomic_t **flags**
- atomic_t **setup_complete**
- atomic_t **setup_in_progress**
- u64 * **mmu_teardown_pages**
- struct tagged_addr **aliasing_sink_page**
- struct mutex **mem_partials_lock**
- struct list_head **mem_partials**
- struct mutex **mmu_lock**
- struct mutex **reg_lock**
- struct rb_root **reg_rbtree_same**
- struct rb_root **reg_rbtree_exec**
- struct rb_root **reg_rbtree_custom**
- unsigned long **cookies**
- struct kbase_va_region * **pending_regions** [BITS_PER_LONG]
- wait_queue_head_t **event_queue**
- pid_t **tgid**
- pid_t **pid**
- struct kbase_jd_context **jctx**
- atomic_t **used_pages**
- atomic_t **nonmapped_pages**
- struct kbase_mem_pool **mem_pool**
- struct kbase_mem_pool **lp_mem_pool**
- struct shrinker **reclaim**
- struct list_head **evict_list**
- struct list_head **waiting_soft_jobs**
- spinlock_t **waiting_soft_jobs_lock**
- int as_nr
- atomic_t **refcount**
- spinlock_t **mm_update_lock**
- struct mm_struct * **process_mm**
- u64 **same_va_end**
- struct jsctx_queue **jsctx_queue** [KBASE_JS_ATOM_SCHED_PRIO_COUNT][BASE_JM_MAX_NR_SLO↩ TS]
- atomic_t **atoms_pulled**
- atomic_t **atoms_pulled_slot** [BASE_JM_MAX_NR_SLOTS]
- int **atoms_pulled_slot_pri** [BASE_JM_MAX_NR_SLOTS][KBASE_JS_ATOM_SCHED_PRIO_COUNT]
- bool **blocked_js** [BASE_JM_MAX_NR_SLOTS][KBASE_JS_ATOM_SCHED_PRIO_COUNT]
- u32 **slots_pullable**
- struct kbase_context_backend **backend**
- struct work_struct **work**
- struct kbase_vinstr_client * **vinstr_cli**

- struct mutex **vinstr_cli_lock**
- struct list_head **completed_jobs**
- atomic_t **work_count**
- struct timer_list **soft_job_timeout**
- struct kbase_va_region * **jit_alloc** [256]
- struct list_head **jit_active_head**
- struct list_head **jit_pool_head**
- struct list_head **jit_destroy_head**
- struct mutex **jit_evict_lock**
- struct work_struct **jit_work**
- struct list_head **jit_atoms_head**
- struct list_head **jit_pending_alloc**
- struct list_head **ext_res_meta_head**
- atomic_t **drain_pending**
- u32 **age_count**

8.28.1 Member Data Documentation

8.28.1.1 int kbase_context::as_nr

This is effectively part of the Run Pool, because it only has a valid setting (!=KBASEP_AS_NR_INVALID) whilst the context is scheduled in

The hwaccess_lock must be held whilst accessing this.

If the context relating to this as_nr is required, you must use kbasep_js_runpool_retain_ctx() to ensure that the context doesn't disappear whilst you're using it. Alternatively, just hold the hwaccess_lock to ensure the context doesn't disappear (but this has restrictions on what other locks you can take whilst doing this)

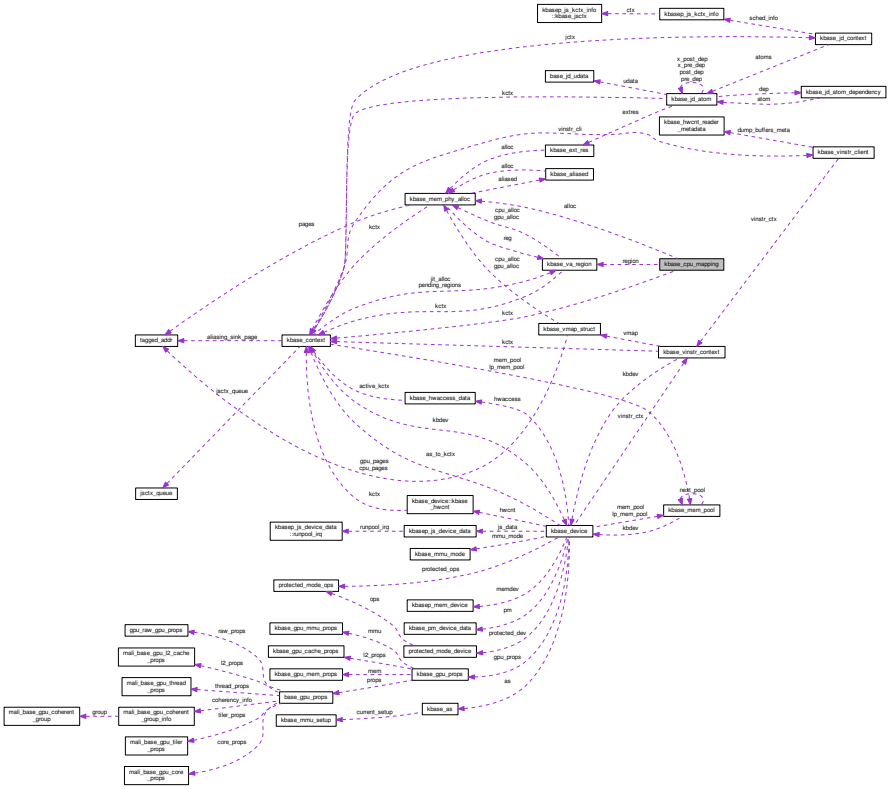
The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.29 kbase_cpu_mapping Struct Reference

```
#include <mali_kbase_mem.h>
```

Collaboration diagram for kbase_cpu_mapping:



Public Attributes

- struct list_head **mappings_list**
- struct kbase_mem_phy_alloc ∗ **alloc**
- struct kbase_context ∗ **kctx**
- struct kbase_va_region ∗ **region**
- int **count**
- int **free_on_close**

8.29.1 Detailed Description

A CPU mapping

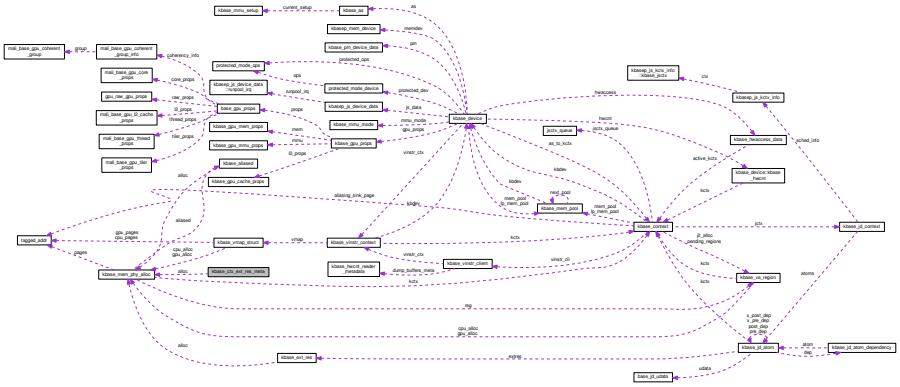
The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_mem.h

8.30 kbase_ctx_ext_res_meta Struct Reference

`#include <mali_kbase_defs.h>`

Collaboration diagram for kbase_ctx_ext_res_meta:



Public Attributes

- struct list_head **ext_res_node**
- struct kbase_mem_phy_alloc * **alloc**
- u64 **gpu_addr**

8.30.1 Detailed Description

struct kbase_ctx_ext_res_meta - Structure which binds an external resource to a . : List head for adding the metadata to a . : The physical memory allocation structure which is mapped. : The GPU virtual address the resource is mapped to.

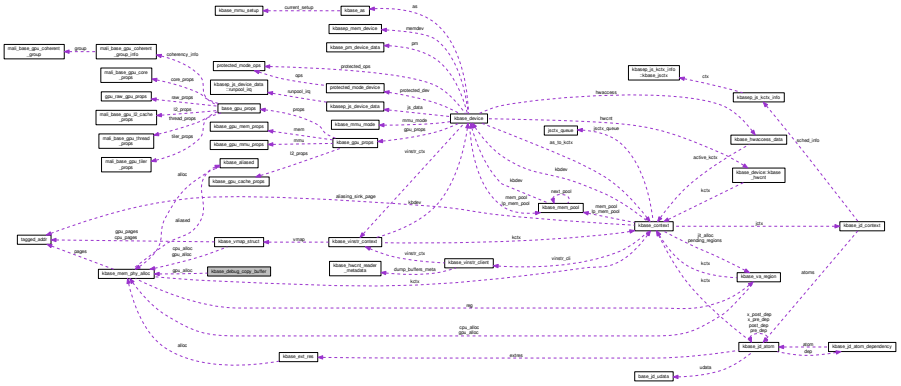
External resources can be mapped into multiple contexts as well as the same context multiple times. As kbase_va_region itself isn't refcounted we can't attach our extra information to it as it could be removed under our feet leaving external resources pinned. This metadata structure binds a single external resource to a single context, ensuring that per context mapping is tracked separately so it can be overridden when needed and abuses by the application (freeing the resource multiple times) don't effect the refcount of the physical allocation.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.31 kbase_debug_copy_buffer Struct Reference

Collaboration diagram for kbase_debug_copy_buffer:



Public Attributes

- size_t **size**
- struct page ** **pages**
- int **nr_pages**
- size_t **offset**
- struct kbase_mem_phy_alloc * **gpu_alloc**
- struct page ** **extres_pages**
- int **nr_extres_pages**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_softjobs.c

8.32 kbase_devfreq_opp Struct Reference

```
#include <mali_kbase_defs.h>
```

Public Attributes

- u64 **opp_freq**
- u64 **real_freq**
- u64 **core_mask**

8.32.1 Detailed Description

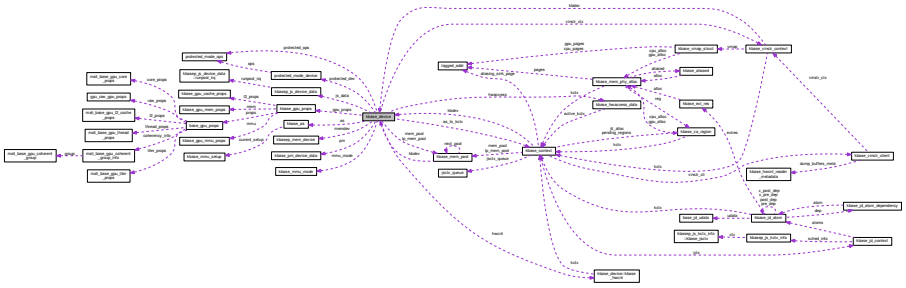
struct kbase_devfreq_opp - Lookup table for converting between nominal OPP frequency, and real frequency and core mask : Nominal OPP frequency : Real GPU frequency : Shader core mask

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.33 kbase_device Struct Reference

Collaboration diagram for kbase_device:



Classes

- struct kbase_hwcnt

Public Attributes

- s8 **slot_submit_count_irq** [BASE_JM_MAX_NR_SLOTS]
- u32 **hw_quirks_sc**
- u32 **hw_quirks_tiler**
- u32 **hw_quirks_mmu**
- u32 **hw_quirks_jm**
- struct list_head **entry**
- struct device * **dev**
- struct miscdevice **mdev**
- u64 **reg_start**
- size_t **reg_size**
- void __iomem * **reg**
- struct {
  int **irq**
  int **flags**
  } **irqs** [3]

- struct clk * **clock**
- char **devname** [DEVNAME_SIZE]
- struct kbase_pm_device_data **pm**
- struct kbasep_js_device_data **js_data**
- struct kbase_mem_pool **mem_pool**
- struct kbase_mem_pool **lp_mem_pool**
- struct kbasep_mem_device **memdev**
- struct kbase_mmu_mode const * **mmu_mode**
- struct kbase_as **as** [BASE_MAX_NR_AS]
- u16 **as_free**
- struct kbase_context * **as_to_kctx** [BASE_MAX_NR_AS]
- spinlock_t **mmu_mask_change**
- struct kbase_gpu_props **gpu_props**
- unsigned long hw_issues_mask [(BASE_HW_ISSUE_END+BITS_PER_LONG-1)/BITS_PER_LONG]
- unsigned long hw_features_mask [(BASE_HW_FEATURE_END+BITS_PER_LONG-1)/BITS_PER_LONG]

- u64 **shader_inuse_bitmap**
- u32 **shader_inuse_cnt** [64]
- u64 **shader_needed_bitmap**
- u32 **shader_needed_cnt** [64]
- u32 **tiler_inuse_cnt**
- u32 **tiler_needed_cnt**
- struct {
  atomic_t **count**
  atomic_t **state**
  } **disjoint_event**

- u32 **l2_users_count**
- u64 **shader_available_bitmap**
- u64 **tiler_available_bitmap**
- u64 **l2_available_bitmap**
- u64 **stack_available_bitmap**
- u64 **shader_ready_bitmap**
- u64 **shader_transitioning_bitmap**
- s8 nr_hw_address_spaces
- s8 nr_user_address_spaces
- struct kbase_device::kbase_hwcnt **hwcnt**
- struct kbase_vinstr_context * **vinstr_ctx**
- u32 **reset_timeout_ms**
- struct mutex **cacheclean_lock**
- void * **platform_context**
- struct list_head **kctx_list**
- struct mutex **kctx_list_lock**
- bool **job_fault_debug**
- u32 **kbase_profiling_controls** [FBDUMP_CONTROL_MAX]
- int **force_replay_limit**
- int **force_replay_count**
- base_jd_core_req **force_replay_core_req**
- bool **force_replay_random**
- atomic_t **ctx_num**
- struct kbase_hwaccess_data **hwaccess**
- atomic_t **faults_pending**
- bool **poweroff_pending**
- bool **infinite_cache_active_default**
- size_t **mem_pool_max_size_default**
- u32 **current_gpu_coherency_mode**
- u32 **system_coherency**
- bool **cci_snoop_enabled**
- u32 **snoop_enable_smc**
- u32 **snoop_disable_smc**
- struct protected_mode_ops * **protected_ops**
- struct protected_mode_device * **protected_dev**
- bool **protected_mode**
- bool **protected_mode_transition**
- bool **protected_mode_support**
- bool **irq_reset_flush**
- u32 **inited_subsys**
- spinlock_t **hwaccess_lock**
- struct mutex **mmu_hw_mutex**
- u8 **serialize_jobs**

8.33.1 Member Data Documentation

8.33.1.1 unsigned long kbase_device::hw_features_mask[(BASE_HW_FEATURE_END+BITS_PER_LONG-1)/BITS_PER_LONG]

List of features available

8.33.1.2 unsigned long kbase_device::hw_issues_mask[(BASE_HW_ISSUE_END+BITS_PER_LONG-1)/BITS_PER_LONG]

List of SW workarounds for HW issues

8.33.1.3 s8 kbase_device::nr_hw_address_spaces

Number of address spaces in the GPU (constant after driver initialisation)

8.33.1.4 s8 kbase_device::nr_user_address_spaces

Number of address spaces available to user contexts

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.34 kbase_device_info Struct Reference

Public Attributes

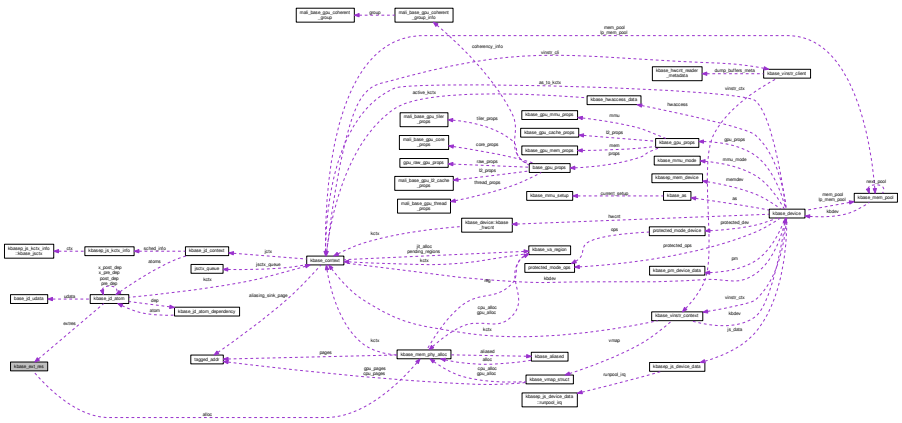
- u32 **features**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.35 kbase_ext_res Struct Reference

Collaboration diagram for kbase_ext_res:



Public Attributes

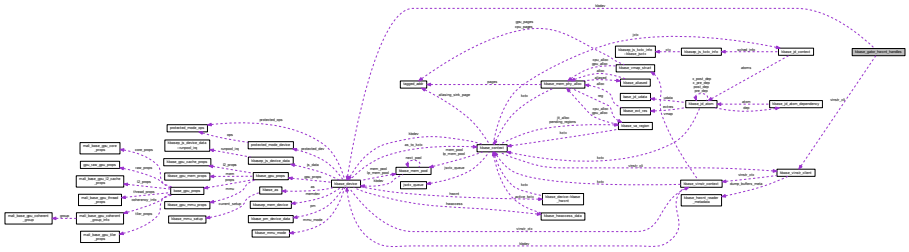
- u64 **gpu_address**
- struct kbase_mem_phy_alloc * **alloc**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.36 kbase_gator_hwcnt_handles Struct Reference

Collaboration diagram for kbase_gator_hwcnt_handles:



Public Attributes

- struct kbase_device * **kbdev**
- struct kbase_vinstr_client * **vinstr_cli**
- void * **vinstr_buffer**
- struct work_struct **dump_work**
- int **dump_complete**
- spinlock_t **dump_lock**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_gator_api.c

8.37 kbase_gator_hwcnt_info Struct Reference

Public Attributes

- uint16_t **bitmask** [4]
- void * **kernel_dump_buffer**
- uint32_t **size**
- uint32_t **gpu_id**
- uint32_t **nr_cores**
- uint32_t **nr_core_groups**
- enum hwc_type * **hwc_layout**
- uint32_t **nr_hwc_blocks**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_gator_api.h

8.38 kbase_gpu_cache_props Struct Reference

Public Attributes

- u8 **associativity**
- u8 **external_bus_width**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_gpuprops_types.h

8.39 kbase_gpu_mem_props Struct Reference

Public Attributes

- u8 **core_group**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_gpuprops_types.h

8.40 kbase_gpu_mmu_props Struct Reference

Public Attributes

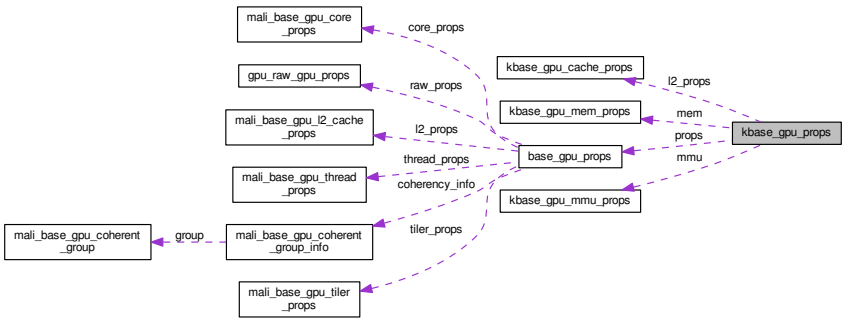
- u8 **va_bits**
- u8 **pa_bits**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_gpuprops_types.h

8.41 kbase_gpu_props Struct Reference

Collaboration diagram for kbase_gpu_props:



Public Attributes

- u8 **num_cores**
- u8 **num_core_groups**
- u8 **num_address_spaces**
- u8 **num_job_slots**
- struct kbase_gpu_cache_props **l2_props**
- struct kbase_gpu_mem_props **mem**
- struct kbase_gpu_mmu_props **mmu**
- base_gpu_props **props**
- u32 **prop_buffer_size**
- void * **prop_buffer**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_gpuprops_types.h

8.42 kbase_gpuprops_regdump Struct Reference

Public Attributes

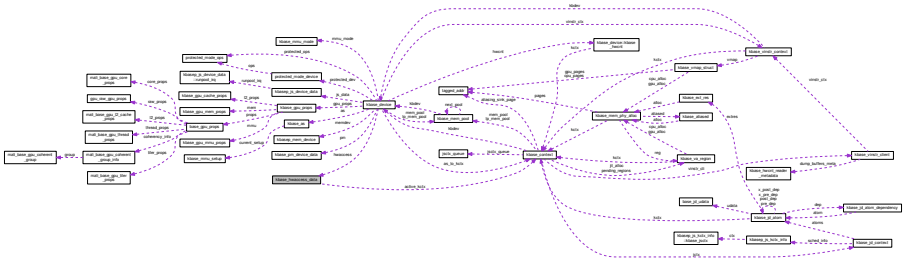
- u32 **gpu_id**
- u32 **l2_features**
- u32 **suspend_size**
- u32 **tiler_features**
- u32 **mem_features**
- u32 **mmu_features**
- u32 **as_present**
- u32 **js_present**
- u32 **thread_max_threads**
- u32 **thread_max_workgroup_size**
- u32 **thread_max_barrier_size**
- u32 **thread_features**
- u32 **texture_features** [BASE_GPU_NUM_TEXTURE_FEATURES_REGISTERS]
- u32 **js_features** [GPU_MAX_JOB_SLOTS]
- u32 **shader_present_lo**
- u32 **shader_present_hi**
- u32 **tiler_present_lo**
- u32 **tiler_present_hi**
- u32 **l2_present_lo**
- u32 **l2_present_hi**
- u32 **stack_present_lo**
- u32 **stack_present_hi**
- u32 **coherency_features**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_gpuprops_types.h

8.43 kbase_hwaccess_data Struct Reference

Collaboration diagram for kbase_hwaccess_data:



Public Attributes

- struct kbase_context * **active_kctx**
- struct kbase_backend_data **backend**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_hwaccess_defs.h

8.44 kbase_hwc_dma_mapping Struct Reference

`#include <mali_kbase_mem_linux.h>`

Public Attributes

- void * **cpu_va**
- dma_addr_t **dma_pa**
- size_t **size**

8.44.1 Detailed Description

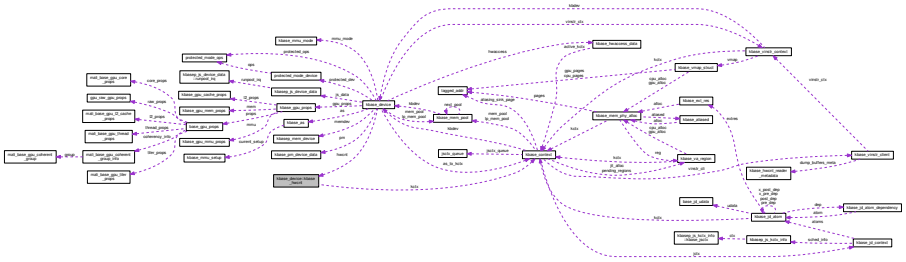
A HWC dump mapping

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_mem_linux.h

8.45 kbase_device::kbase_hwcnt Struct Reference

Collaboration diagram for kbase_device::kbase_hwcnt:



Public Attributes

- spinlock_t **lock**
- struct kbase_context * **kctx**
- u64 **addr**
- struct kbase_instr_backend **backend**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.46 kbase_hwcnt_reader_metadata Struct Reference

`#include <mali_kbase_hwcnt_reader.h>`

Public Attributes

- u64 **timestamp**
- u32 **event_id**
- u32 **buffer_idx**

8.46.1 Detailed Description

struct kbase_hwcnt_reader_metadata - hwcnt reader sample buffer metadata : time when sample was collected : id of an event that triggered sample collection : position in sampling area where sample buffer was stored

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_hwcnt_reader.h

8.47 kbase_io_access Struct Reference

`#include <mali_kbase_defs.h>`

Public Attributes

- uintptr_t **addr**
- u32 **value**

8.47.1 Detailed Description

struct kbase_io_access - holds information about 1 register access

: first bit indicates r/w (r=0, w=1) : value written or read

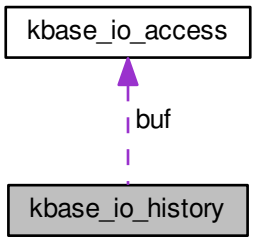
The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.48 kbase_io_history Struct Reference

`#include <mali_kbase_defs.h>`

Collaboration diagram for kbase_io_history:



Public Attributes

- bool **enabled**
- spinlock_t **lock**
- size_t **count**
- u16 **size**
- struct kbase_io_access * **buf**

8.48.1 Detailed Description

struct kbase_io_history - keeps track of all recent register accesses

: true if register accesses are recorded, false otherwise : spinlock protecting kbase_io_access array : number of registers read/written : number of elements in kbase_io_access array : array of kbase_io_access

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.49 kbase_io_memory_region Struct Reference

Public Attributes

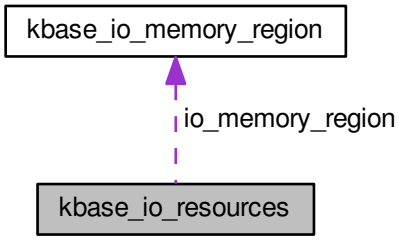
- u64 **start**
- u64 **end**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_config.h

8.50 kbase_io_resources Struct Reference

Collaboration diagram for kbase_io_resources:



Public Attributes

- u32 **job_irq_number**
- u32 **mmu_irq_number**
- u32 **gpu_irq_number**
- struct kbase_io_memory_region **io_memory_region**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_config.h

8.51 kbase_ioctl_cinstr_gwt_dump Union Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- struct {
  __u64 **handle_buffer**
  __u64 **offset_buffer**
  __u64 **size_buffer**
  __u32 **len**
  __u32 **padding**
  } **in**

- struct {
  __u32 **no_of_addr_collected**
  __u8 **more_data_available**
  __u8 **padding** [27]
  } **out**

8.51.1 Detailed Description

union kbase_ioctl_gwt_dump - Used to collect all GPU write fault addresses. : Address of buffer to hold handles of modified areas. : Address of buffer to hold offset size of modified areas (in pages) : Address of buffer to hold size of modified areas (in pages) : Number of addresses the buffers can hold. : Status indicating if more addresses are available. : Number of addresses collected into addr_buffer.

: Input parameters : Output parameters This structure is used when performing a call to dump GPU write fault addresses.

The documentation for this union was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.52 kbase_ioctl_disjoint_query Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u32 **counter**

8.52.1 Detailed Description

struct kbase_ioctl_disjoint_query - Query the disjoint counter : A counter of disjoint events in the kernel

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.53 kbase_ioctl_fence_validate Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- int **fd**

8.53.1 Detailed Description

struct kbase_ioctl_fence_validate - Validate a fd refers to a fence : The file descriptor to validate

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.54 kbase_ioctl_get_context_id Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u32 **id**

8.54.1 Detailed Description

struct kbase_ioctl_get_context_id - Get the kernel context ID

: The kernel context ID

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.55 kbase_ioctl_get_ddk_version Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u64 **version_buffer**
- __u32 **size**
- __u32 **padding**

8.55.1 Detailed Description

struct kbase_ioctl_get_ddk_version - Query the kernel version : Buffer to receive the kernel version string : Size of the buffer : Padding

The ioctl will return the number of bytes written into version_buffer (which includes a NULL byte) or a negative error code

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.56 kbase_ioctl_get_gpuprops Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u64 **buffer**
- __u32 **size**
- __u32 **flags**

8.56.1 Detailed Description

struct kbase_ioctl_get_gpuprops - Read GPU properties from the kernel

: Pointer to the buffer to store properties into : Size of the buffer : Flags - must be zero for now

The ioctl will return the number of bytes stored into or an error on failure (e.g. is too small). If is specified as 0 then no data will be written but the return value will be the number of bytes needed for all the properties.

may be used in the future to request a different format for the buffer. With == 0 the following format is used.

The buffer will be filled with pairs of values, a u32 key identifying the property followed by the value. The size of the value is identified using the bottom bits of the key. The value then immediately followed the key and is tightly packed (there is no padding). All keys and values are little-endian.

00 = u8 01 = u16 10 = u32 11 = u64

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.57 kbase_ioctl_get_profiling_controls Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u64 **buffer**
- __u32 **count**
- __u32 **padding**

8.57.1 Detailed Description

struct kbase_ioctl_get_profiling_controls - Get the profiling controls : The size of in u32 words : The buffer to receive the profiling controls : Padding

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.58 kbase_ioctl_hwcnt_enable Struct Reference

```
#include <mali_kbase_ioctl.h>
```

Public Attributes

- __u64 **dump_buffer**
- __u32 **jm_bm**
- __u32 **shader_bm**
- __u32 **tiler_bm**
- __u32 **mmu_l2_bm**

8.58.1 Detailed Description

struct kbase_ioctl_hwcnt_enable - Enable hardware counter collection : GPU address to write counters to : counters selection bitmask (JM) : counters selection bitmask (Shader) : counters selection bitmask (Tiler) : counters selection bitmask (MMU_L2)

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.59 kbase_ioctl_hwcnt_reader_setup Struct Reference

```
#include <mali_kbase_ioctl.h>
```

Public Attributes

- __u32 **buffer_count**
- __u32 **jm_bm**
- __u32 **shader_bm**
- __u32 **tiler_bm**
- __u32 **mmu_l2_bm**

8.59.1 Detailed Description

struct kbase_ioctl_hwcnt_reader_setup - Setup HWC dumper/reader : requested number of dumping buffers : counters selection bitmask (JM) : counters selection bitmask (Shader) : counters selection bitmask (Tiler) : counters selection bitmask (MMU_L2)

A fd is returned from the ioctl if successful, or a negative value on error

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.60 kbase_ioctl_job_submit Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u64 **addr**
- __u32 **nr_atoms**
- __u32 **stride**

8.60.1 Detailed Description

struct kbase_ioctl_job_submit - Submit jobs/atoms to the kernel

: Memory address of an array of struct base_jd_atom_v2 : Number of entries in the array : sizeof(struct base_jd↩_atom_v2)

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.61 kbase_ioctl_mem_alias Union Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- struct {
  __u64 **flags**
  __u64 **stride**
  __u64 **nents**
  __u64 **aliasing_info**
  } **in**

- struct {
  __u64 **flags**
  __u64 **gpu_va**
  __u64 **va_pages**
  } **out**

8.61.1 Detailed Description

union kbase_ioctl_mem_alias - Create an alias of memory regions : Flags, see BASE_MEM_xxx : Bytes between start of each memory region : The number of regions to pack together into the alias : Pointer to an array of struct base_mem_aliasing_info : Address of the new alias : Size of the new alias

: Input parameters : Output parameters

The documentation for this union was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.62 kbase_ioctl_mem_alloc Union Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- struct {
  __u64 **va_pages**
  __u64 **commit_pages**
  __u64 **extent**
  __u64 **flags**
  } **in**

- struct {
  __u64 **flags**
  __u64 **gpu_va**
  } **out**

8.62.1 Detailed Description

union kbase_ioctl_mem_alloc - Allocate memory on the GPU

: The number of pages of virtual address space to reserve : The number of physical pages to allocate : The number of extra pages to allocate on each GPU fault which grows the region : Flags : The GPU virtual address which is allocated

: Input parameters : Output parameters

The documentation for this union was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.63 kbase_ioctl_mem_commit Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u64 **gpu_addr**
- __u64 **pages**

8.63.1 Detailed Description

struct kbase_ioctl_mem_commit - Change the amount of memory backing a region

: The memory region to modify : The number of physical pages that should be present

The ioctl may return on the following error codes or 0 for success: -ENOMEM: Out of memory -EINVAL: Invalid arguments

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.64 kbase_ioctl_mem_find_cpu_offset Union Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- struct {
  __u64 **gpu_addr**
  __u64 **cpu_addr**
  __u64 **size**
  } **in**

- struct {
  __u64 **offset**
  } **out**

8.64.1 Detailed Description

union kbase_ioctl_mem_find_cpu_offset - Find the offset of a CPU pointer

: The GPU address of the memory region : The CPU address to locate : A size in bytes to validate is contained within the region : The offset from the start of the memory region to

: Input parameters : Output parameters

The documentation for this union was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.65 kbase_ioctl_mem_find_gpu_start_and_offset Union Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- struct {
  __u64 **gpu_addr**
  __u64 **size**
  } **in**

- struct {
  __u64 **start**
  __u64 **offset**
  } **out**

8.65.1 Detailed Description

union kbase_ioctl_mem_find_gpu_start_and_offset - Find the start address of the GPU memory region for the given gpu address and the offset of that address into the region

: GPU virtual address : Size in bytes within the region : Address of the beginning of the memory region enclosing for the length of bytes : The offset from the start of the memory region to

: Input parameters : Output parameters

The documentation for this union was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.66 kbase_ioctl_mem_flags_change Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u64 **gpu_va**
- __u64 **flags**
- __u64 **mask**

8.66.1 Detailed Description

struct kbase_ioctl_mem_flags_change - Change the flags for a memory region : The GPU region to modify : The new flags to set : Mask of the flags to modify

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.67 kbase_ioctl_mem_free Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u64 **gpu_addr**

8.67.1 Detailed Description

struct kbase_ioctl_mem_free - Free a memory region : Handle to the region to free

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.68 kbase_ioctl_mem_import Union Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- struct {
  __u64 **flags**
  __u64 **phandle**
  __u32 **type**
  __u32 **padding**
  } **in**

- struct {
  __u64 **flags**
  __u64 **gpu_va**
  __u64 **va_pages**
  } **out**

8.68.1 Detailed Description

union kbase_ioctl_mem_import - Import memory for use by the GPU : Flags, see BASE_MEM_xxx : Handle to the external memory : Type of external memory, see base_mem_import_type : Amount of extra VA pages to append to the imported buffer : Address of the new alias : Size of the new alias

: Input parameters : Output parameters

The documentation for this union was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.69 kbase_ioctl_mem_jit_init Struct Reference

```
#include <mali_kbase_ioctl.h>
```

Public Attributes

- __u64 **va_pages**

8.69.1 Detailed Description

struct kbase_ioctl_mem_jit_init - Initialise the JIT memory allocator

: Number of VA pages to reserve for JIT

Note that depending on the VA size of the application and GPU, the value specified in may be ignored.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.70 kbase_ioctl_mem_profile_add Struct Reference

```
#include <mali_kbase_ioctl.h>
```

Public Attributes

- __u64 **buffer**
- __u32 **len**
- __u32 **padding**

8.70.1 Detailed Description

struct kbase_ioctl_mem_profile_add - Provide profiling information to kernel : Pointer to the information : Length : Padding

The data provided is accessible through a debugfs file

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.71 kbase_ioctl_mem_query Union Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- struct {
  __u64 **gpu_addr**
  __u64 **query**
  } **in**

- struct {
  __u64 **value**
  } **out**

8.71.1 Detailed Description

struct kbase_ioctl_mem_query - Query properties of a GPU memory region : A GPU address contained within the region : The type of query : The result of the query

Use a KBASE_MEM_QUERY_xxx flag as input for .

: Input parameters : Output parameters

The documentation for this union was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.72 kbase_ioctl_mem_sync Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u64 **handle**
- __u64 **user_addr**
- __u64 **size**
- __u8 **type**
- __u8 **padding** [7]

8.72.1 Detailed Description

struct kbase_ioctl_mem_sync - Perform cache maintenance on memory

: GPU memory handle (GPU VA) : The address where it is mapped in user space : The number of bytes to synchronise : The direction to synchronise: 0 is sync to memory (clean), 1 is sync from memory (invalidate). Use the BASE_SYNCSET_OP_xxx constants. : Padding to round up to a multiple of 8 bytes, must be zero

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.73 kbase_ioctl_set_flags Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u32 **create_flags**

8.73.1 Detailed Description

struct kbase_ioctl_set_flags - Set kernel context creation flags

: Flags - see base_context_create_flags

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.74 kbase_ioctl_soft_event_update Struct Reference

`#include <mali_kbase_ioctl.h>`

Public Attributes

- __u64 **event**
- __u32 **new_status**
- __u32 **flags**

8.74.1 Detailed Description

struct kbase_ioctl_soft_event_update - Update the status of a soft-event : GPU address of the event which has been updated : The new status to set : Flags for future expansion

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.75 kbase_ioctl_sticky_resource_map Struct Reference

```
#include <mali_kbase_ioctl.h>
```

Public Attributes

- __u64 **count**
- __u64 **address**

8.75.1 Detailed Description

struct kbase_ioctl_sticky_resource_map - Permanently map an external resource : Number of resources : Array of u64 GPU addresses of the external resources to map

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.76 kbase_ioctl_sticky_resource_unmap Struct Reference

```
#include <mali_kbase_ioctl.h>
```

Public Attributes

- __u64 **count**
- __u64 **address**

8.76.1 Detailed Description

struct kbase_ioctl_sticky_resource_map - Unmap a resource mapped which was previously permanently mapped : Number of resources : Array of u64 GPU addresses of the external resources to unmap

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.77 kbase_ioctl_stream_create Struct Reference

Public Attributes

- char **name** [32]

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.78 kbase_ioctl_tlstream_acquire Struct Reference

```
#include <mali_kbase_ioctl.h>
```

Public Attributes

- __u32 **flags**

8.78.1 Detailed Description

struct kbase_ioctl_tlstream_acquire - Acquire a tlstream fd

: Flags

The ioctl returns a file descriptor when successful

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.79 kbase_ioctl_version_check Struct Reference

```
#include <mali_kbase_ioctl.h>
```

Public Attributes

- __u16 **major**
- __u16 **minor**

8.79.1 Detailed Description

struct kbase_ioctl_version_check - Check version compatibility with kernel

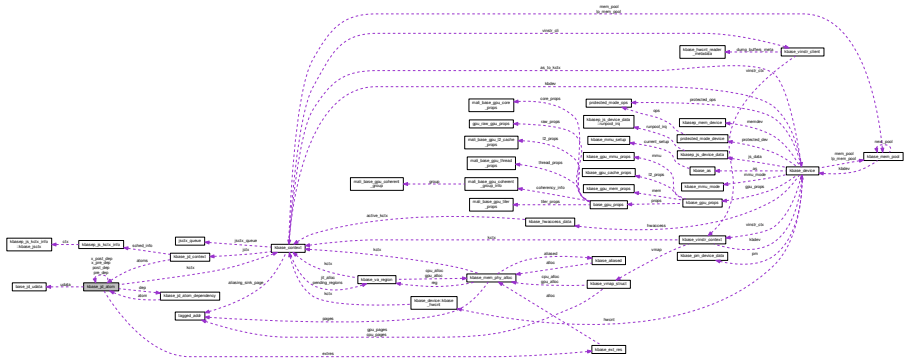
: Major version number : Minor version number

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_ioctl.h

8.80 kbase_jd_atom Struct Reference

Collaboration diagram for kbase_jd_atom:



Public Attributes

- struct work_struct **work**
- ktime_t **start_timestamp**
- struct base_jd_udata **udata**
- struct kbase_context * **kctx**
- struct list_head **dep_head** [2]
- struct list_head **dep_item** [2]
- const struct kbase_jd_atom_dependency **dep** [2]
- struct list_head **jd_item**
- bool **in_jd_list**
- u16 **nr_extres**
- struct kbase_ext_res * **extres**
- u32 **device_nr**
- u64 **affinity**
- u64 **jc**
- void * **softjob_data**
- enum kbase_atom_coreref_state **coreref_state**

- enum base_jd_event_code **event_code**
- base_jd_core_req core_req
- u32 **ticks**
- int **sched_priority**
- int **poking**
- wait_queue_head_t **completed**
- enum kbase_jd_atom_state **status**
- int **slot_nr**
- u32 **atom_flags**
- int **retry_count**
- enum kbase_atom_gpu_rb_state **gpu_rb_state**
- u64 **need_cache_flush_cores_retained**
- atomic_t **blocked**
- struct kbase_jd_atom * **pre_dep**
- struct kbase_jd_atom * **post_dep**
- struct kbase_jd_atom * **x_pre_dep**
- struct kbase_jd_atom * **x_post_dep**
- u32 **flush_id**
- struct kbase_jd_atom_backend **backend**
- struct list_head **queue**
- struct list_head **jit_node**
- bool **jit_blocked**
- enum base_jd_event_code **will_fail_event_code**
- union {
  enum kbase_atom_enter_protected_state **enter**
  enum kbase_atom_exit_protected_state **exit**
  } **protected_state**

- struct rb_node **runnable_tree_node**
- u32 **age**

8.80.1 Member Data Documentation

8.80.1.1 base_jd_core_req kbase_jd_atom::core_req

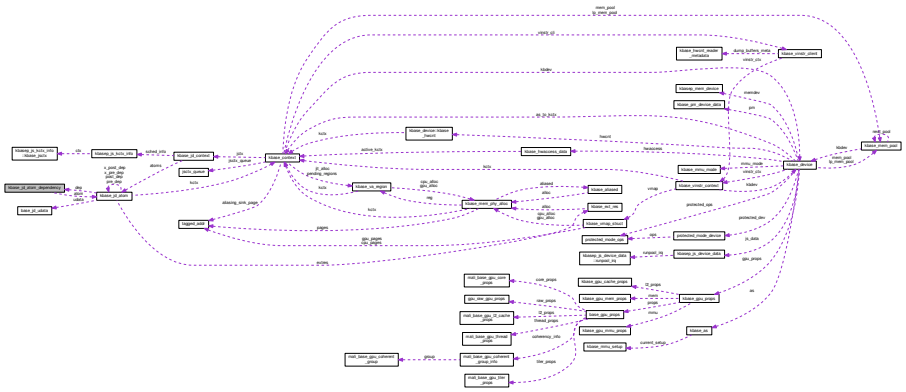
core requirements

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.81 kbase_jd_atom_dependency Struct Reference

Collaboration diagram for kbase_jd_atom_dependency:



Public Attributes

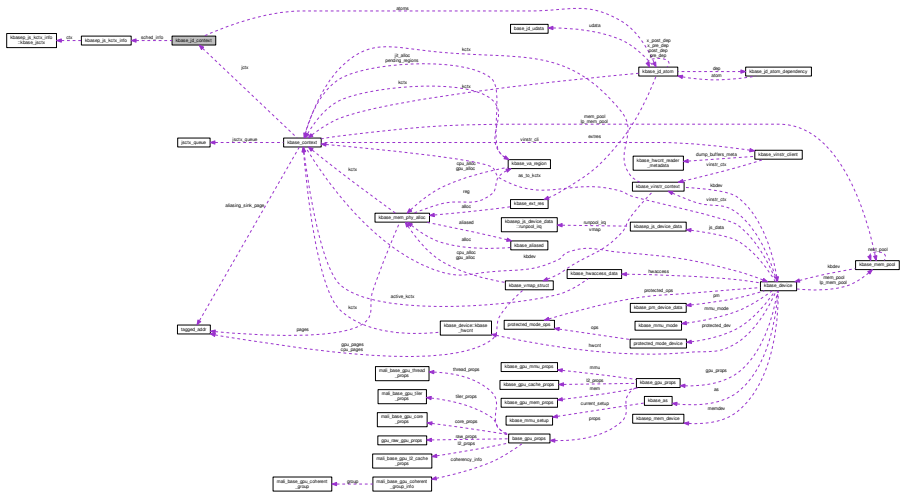
- struct kbase_jd_atom ∗ **atom**
- u8 **dep_type**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.82 kbase_jd_context Struct Reference

Collaboration diagram for kbase_jd_context:



Public Attributes

- struct mutex **lock**
- struct kbasep_js_kctx_info **sched_info**
- struct kbase_jd_atom **atoms** [BASE_JD_ATOM_COUNT]
- u32 job_nr
- wait_queue_head_t zero_jobs_wait
- struct workqueue_struct * job_done_wq
- spinlock_t **tb_lock**
- u32 * **tb**
- size_t **tb_wrap_offset**

8.82.1 Member Data Documentation

8.82.1.1 struct workqueue_struct* kbase_jd_context::job_done_wq

Job Done workqueue.

8.82.1.2 u32 kbase_jd_context::job_nr

Tracks all job-dispatch jobs. This includes those not tracked by the scheduler: 'not ready to run' and 'dependency-only' jobs.

8.82.1.3 wait_queue_head_t kbase_jd_context::zero_jobs_wait

Waitq that reflects whether there are no jobs (including SW-only dependency jobs). This is set when no jobs are present on the ctx, and clear when there are jobs.

Note

: Job Dispatcher knows about more jobs than the Job Scheduler: the Job Scheduler is unaware of jobs that are blocked on dependencies, and SW-only dependency jobs.

This waitq can be waited upon to find out when the context jobs are all done/cancelled (including those that might've been blocked on dependencies) - and so, whether it can be terminated. However, it should only be terminated once it is not present in the run-pool (see kbasep_js_kctx_info::ctx::is_scheduled).

Since the waitq is only set under kbase_jd_context::lock, the waiter should also briefly obtain and drop kbase_jd↩_context::lock to guarentee that the setter has completed its work on the kbase_context

This must be updated atomically with:

- kbase_jd_context::job_nr

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.83 kbasep_js_kctx_info::kbase_jsctx Struct Reference

`#include <mali_kbase_js_defs.h>`

Public Attributes

- struct mutex jsctx_mutex
- u32 nr_jobs
- u32 ctx_attr_ref_count [KBASEP_JS_CTX_ATTR_COUNT]
- wait_queue_head_t is_scheduled_wait
- struct list_head ctx_list_entry [BASE_JM_MAX_NR_SLOTS]

8.83.1 Detailed Description

Job Scheduler Context information sub-structure. These members are accessed regardless of whether the context is:

- In the Policy's Run Pool
- In the Policy's Queue
- Not queued nor in the Run Pool.

You must obtain the jsctx_mutex before accessing any other members of this substructure.

You may not access any of these members from IRQ context.

8.83.2 Member Data Documentation

8.83.2.1 u32 kbasep_js_kctx_info::kbase_jsctx::ctx_attr_ref_count[KBASEP_JS_CTX_ATTR_COUNT]

Context Attributes: Each is large enough to hold a refcount of the number of atoms on the context.

8.83.2.2 struct list_head kbasep_js_kctx_info::kbase_jsctx::ctx_list_entry[BASE_JM_MAX_NR_SLOTS]

Link implementing JS queues. Context can be present on one list per job slot

8.83.2.3 wait_queue_head_t kbasep_js_kctx_info::kbase_jsctx::is_scheduled_wait

Wait queue to wait for KCTX_SHEDULED flag state changes.

8.83.2.4 struct mutex kbasep_js_kctx_info::kbase_jsctx::jsctx_mutex

Job Scheduler Context lock

8.83.2.5 u32 kbasep_js_kctx_info::kbase_jsctx::nr_jobs

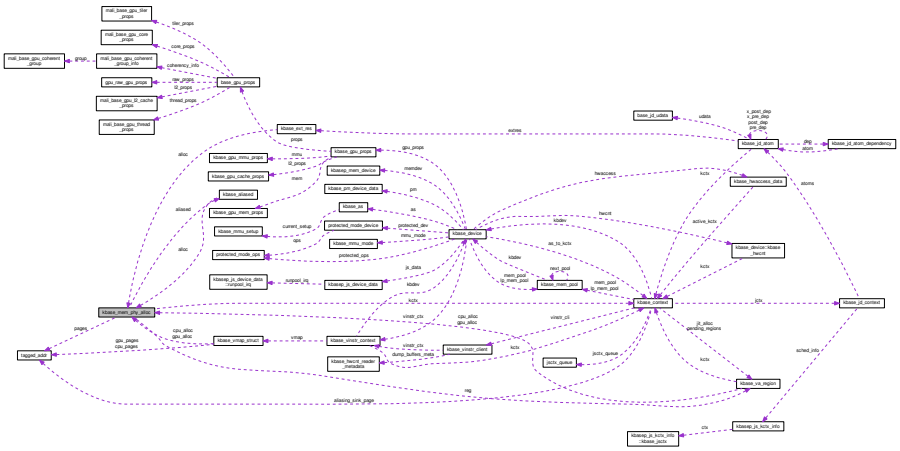
Number of jobs **ready to run** - does *not* include the jobs waiting in the dispatcher, and dependency-only jobs. See kbase_jd_context::job_nr for such jobs

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_js_defs.h

8.84 kbase_mem_phy_alloc Struct Reference

Collaboration diagram for kbase_mem_phy_alloc:



Public Attributes

- struct kref **kref**
- atomic_t **gpu_mappings**
- size_t **nents**
- struct tagged_addr * **pages**
- struct list_head **mappings**
- struct list_head **evict_node**
- size_t **evicted**
- struct kbase_va_region * **reg**
- enum kbase_memory_type **type**
- unsigned long **properties**
- union {
  - struct {
    - u64 **stride**
    - size_t **nents**
    - struct kbase_aliased * **aliased**
  - } **alias**
  - struct kbase_context * **kctx**
  - struct **kbase_alloc_import_user_buf** {
    - unsigned long **address**
    - unsigned long **size**

unsigned long **nr_pages**
struct page ** **pages**
u32 **current_mapping_usage_count**
struct mm_struct * **mm**
dma_addr_t * **dma_addrs**
} **user_buf**
} **imported**

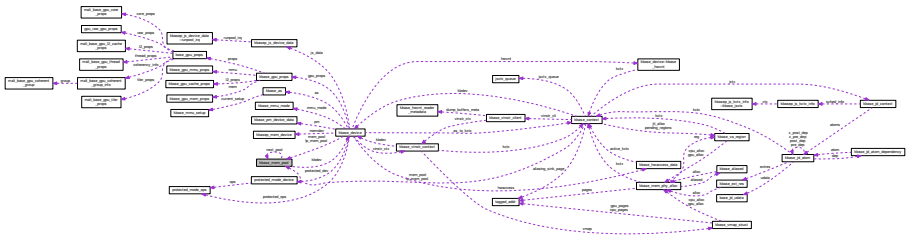
The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_mem.h

8.85 kbase_mem_pool Struct Reference

`#include <mali_kbase_defs.h>`

Collaboration diagram for kbase_mem_pool:



Public Attributes

- struct kbase_device * **kbdev**
- size_t **cur_size**
- size_t **max_size**
- size_t **order**
- spinlock_t **pool_lock**
- struct list_head **page_list**
- struct shrinker **reclaim**
- struct kbase_mem_pool * **next_pool**

8.85.1 Detailed Description

struct kbase_mem_pool - Page based memory pool for kctx/kbdev : Kbase device where memory is used : Number of free pages currently in the pool (may exceed in some corner cases) : Maximum number of free pages in the pool : order = 0 refers to a pool of 4 KB pages order = 9 refers to a pool of 2 MB pages (2^9 * 4KB = 2 MB) : Lock protecting the pool - must be held when modifying and : List of free pages in the pool : Shrinker for kernel reclaim of free pages : Pointer to next pool where pages can be allocated when this pool is empty. Pages will spill over to the next pool when this pool is full. Can be NULL if there is no next pool.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.86 kbase_mmu_mode Struct Reference

Public Attributes

- void(∗ **update**)(struct kbase_context ∗kctx)
- void(∗ **get_as_setup**)(struct kbase_context ∗kctx, struct kbase_mmu_setup ∗const setup)
- void(∗ **disable_as**)(struct kbase_device ∗kbdev, int as_nr)
- phys_addr_t(∗ **pte_to_phy_addr**)(u64 entry)
- int(∗ **ate_is_valid**)(u64 ate, unsigned int level)
- int(∗ **pte_is_valid**)(u64 pte, unsigned int level)
- void(∗ **entry_set_ate**)(u64 ∗entry, struct tagged_addr phy, unsigned long flags, unsigned int level)
- void(∗ **entry_set_pte**)(u64 ∗entry, phys_addr_t phy)
- void(∗ **entry_invalidate**)(u64 ∗entry)

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.87 kbase_mmu_setup Struct Reference

Public Attributes

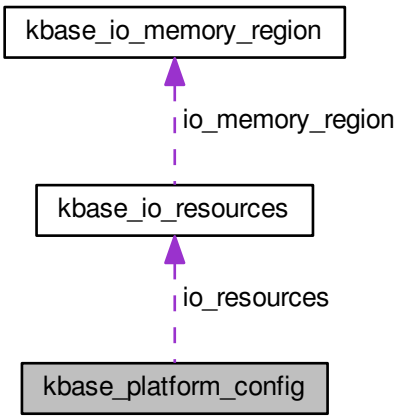
- u64 **transtab**
- u64 **memattr**
- u64 **transcfg**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.88 kbase_platform_config Struct Reference

Collaboration diagram for kbase_platform_config:



Public Attributes

- const struct kbase_io_resources * **io_resources**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_config.h

8.89 kbase_platform_funcs_conf Struct Reference

`#include <mali_kbase_config.h>`

Public Attributes

- int(* platform_init_func)(struct kbase_device *kbdev)
- void(* platform_term_func)(struct kbase_device *kbdev)

8.89.1 Detailed Description

kbase_platform_funcs_conf - Specifies platform init/term function pointers

Specifies the functions pointers for platform specific initialization and termination. By default no functions are required. No additional platform specific control is necessary.

8.89.2 Member Data Documentation

8.89.2.1 int(* kbase_platform_funcs_conf::platform_init_func) (struct kbase_device *kbdev)

platform_init_func - platform specific init function pointer - kbase_device pointer

Returns 0 on success, negative error code otherwise.

Function pointer for platform specific initialization or NULL if no initialization function is required. At the point this the GPU is not active and its power and clocks are in unknown (platform specific state) as kbase doesn't yet have control of power and clocks.

The platform specific private pointer kbase_device::platform_context can be accessed (and possibly initialized) in here.

8.89.2.2 void(* kbase_platform_funcs_conf::platform_term_func) (struct kbase_device *kbdev)

platform_term_func - platform specific termination function pointer - kbase_device pointer

Function pointer for platform specific termination or NULL if no termination function is required. At the point this the GPU will be idle but still powered and clocked.

The platform specific private pointer kbase_device::platform_context can be accessed (and possibly terminated) in here.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_config.h

8.90 kbase_pm_callback_conf Struct Reference

Public Attributes

- void(∗ power_off_callback)(struct kbase_device ∗kbdev)
- int(∗ power_on_callback)(struct kbase_device ∗kbdev)
- void(∗ power_suspend_callback)(struct kbase_device ∗kbdev)
- void(∗ power_resume_callback)(struct kbase_device ∗kbdev)
- int(∗ power_runtime_init_callback)(struct kbase_device ∗kbdev)
- void(∗ power_runtime_term_callback)(struct kbase_device ∗kbdev)
- void(∗ power_runtime_off_callback)(struct kbase_device ∗kbdev)
- int(∗ power_runtime_on_callback)(struct kbase_device ∗kbdev)
- int(∗ **power_runtime_idle_callback**)(struct kbase_device ∗kbdev)

8.90.1 Member Data Documentation

8.90.1.1 void(∗ kbase_pm_callback_conf::power_off_callback) (struct kbase_device ∗kbdev)

Callback for when the GPU is idle and the power to it can be switched off.

The system integrator can decide whether to either do nothing, just switch off the clocks to the GPU, or to completely power down the GPU. The platform specific private pointer kbase_device::platform_context can be accessed and modified in here. It is the platform *callbacks* responsibility to initialize and terminate this pointer if used (see kbase↩ _platform_funcs_conf).

8.90.1.2 int(∗ kbase_pm_callback_conf::power_on_callback) (struct kbase_device ∗kbdev)

Callback for when the GPU is about to become active and power must be supplied.

This function must not return until the GPU is powered and clocked sufficiently for register access to succeed. The return value specifies whether the GPU was powered down since the call to power_off_callback. If the GPU state has been lost then this function must return 1, otherwise it should return 0. The platform specific private pointer kbase_device::platform_context can be accessed and modified in here. It is the platform *callbacks* responsibility to initialize and terminate this pointer if used (see kbase_platform_funcs_conf).

The return value of the first call to this function is ignored.

Returns

1 if the GPU state may have been lost, 0 otherwise.

8.90.1.3 void(∗ kbase_pm_callback_conf::power_resume_callback) (struct kbase_device ∗kbdev)

Callback for when the system is resuming from a suspend and GPU power must be switched on.

Note that if this callback is present, then this may be called without a following call to power_on_callback. Therefore this callback must be able to take any action that might otherwise happen in power_on_callback.

The platform specific private pointer kbase_device::platform_context can be accessed and modified in here. It is the platform *callbacks* responsibility to initialize and terminate this pointer if used (see kbase_platform_funcs_conf).

8.90.1.4 int(∗ kbase_pm_callback_conf::power_runtime_init_callback) (struct kbase_device ∗kbdev)

Callback for handling runtime power management initialization.

The runtime power management callbacks power_runtime_off_callback and power_runtime_on_callback will become active from calls made to the OS from within this function. The runtime calls can be triggered by calls from power_off_callback and power_on_callback. Note: for linux the kernel must have CONFIG_PM_RUNTIME enabled to use this feature.

Returns

0 on success, else int error code.

8.90.1.5 void(∗ kbase_pm_callback_conf::power_runtime_off_callback) (struct kbase_device ∗kbdev)

Callback for runtime power-off power management callback

For linux this callback will be called by the kernel runtime_suspend callback. Note: for linux the kernel must have CONFIG_PM_RUNTIME enabled to use this feature.

Returns

0 on success, else OS error code.

8.90.1.6 int(∗ kbase_pm_callback_conf::power_runtime_on_callback) (struct kbase_device ∗kbdev)

Callback for runtime power-on power management callback

For linux this callback will be called by the kernel runtime_resume callback. Note: for linux the kernel must have CONFIG_PM_RUNTIME enabled to use this feature.

8.90.1.7 void(∗ kbase_pm_callback_conf::power_runtime_term_callback) (struct kbase_device ∗kbdev)

Callback for handling runtime power management termination.

The runtime power management callbacks power_runtime_off_callback and power_runtime_on_callback should no longer be called by the OS on completion of this function. Note: for linux the kernel must have CONFIG_PM_RU↩ NTIME enabled to use this feature.

8.90.1.8 void(∗ kbase_pm_callback_conf::power_suspend_callback) (struct kbase_device ∗kbdev)

Callback for when the system is requesting a suspend and GPU power must be switched off.

Note that if this callback is present, then this may be called without a preceding call to power_off_callback. Therefore this callback must be able to take any action that might otherwise happen in power_off_callback.

The platform specific private pointer kbase_device::platform_context can be accessed and modified in here. It is the platform *callbacks* responsibility to initialize and terminate this pointer if used (see kbase_platform_funcs_conf).

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_config.h

8.91 kbase_pm_device_data Struct Reference

`#include <mali_kbase_defs.h>`

Public Attributes

- struct mutex lock
- int active_count
- bool suspending
- wait_queue_head_t **zero_active_count_wait**
- u64 debug_core_mask [BASE_JM_MAX_NR_SLOTS]
- u64 **debug_core_mask_all**
- int(∗ callback_power_runtime_init)(struct kbase_device ∗kbdev)
- void(∗ callback_power_runtime_term)(struct kbase_device ∗kbdev)
- u32 **dvfs_period**
- ktime_t **gpu_poweroff_time**
- int **poweroff_shader_ticks**
- int **poweroff_gpu_ticks**
- struct kbase_pm_backend_data **backend**

8.91.1 Detailed Description

Data stored per device for power management.

This structure contains data for the power management framework. There is one instance of this structure per device in the system.

8.91.2 Member Data Documentation

8.91.2.1 int kbase_pm_device_data::active_count

The reference count of active contexts on this device.

8.91.2.2 int(∗ kbase_pm_device_data::callback_power_runtime_init) (struct kbase_device ∗kbdev)

Callback for initializing the runtime power management.

Parameters

kbdev	The kbase device

Returns

0 on success, else error code

8.91.2.3 void(∗ kbase_pm_device_data::callback_power_runtime_term) (struct kbase_device ∗kbdev)

Callback for terminating the runtime power management.

Parameters

kbdev	The kbase device

8.91.2.4 u64 kbase_pm_device_data::debug_core_mask[BASE_JM_MAX_NR_SLOTS]

Bit masks identifying the available shader cores that are specified via sysfs. One mask per job slot.

8.91.2.5 struct mutex kbase_pm_device_data::lock

The lock protecting Power Management structures accessed outside of IRQ.

This lock must also be held whenever the GPU is being powered on or off.

8.91.2.6 bool kbase_pm_device_data::suspending

Flag indicating suspending/suspended

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.92 kbase_sub_alloc Struct Reference

Public Member Functions

- **DECLARE_BITMAP** (sub_pages, SZ_2M/SZ_4K)

Public Attributes

- struct list_head **link**
- struct page ∗ **page**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.93 kbase_sync_fence_info Struct Reference

Public Attributes

- void * **fence**
- char **name** [32]
- int **status**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_sync.h

8.94 kbase_trace Struct Reference

Public Attributes

- struct timespec **timestamp**
- u32 **thread_id**
- u32 **cpu**
- void * **ctx**
- bool **katom**
- int **atom_number**
- u64 **atom_udata** [2]
- u64 **gpu_addr**
- unsigned long **info_val**
- u8 **code**
- u8 **jobslot**
- u8 **refcount**
- u8 **flags**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.95 kbase_uk_hwcnt_reader_setup Struct Reference

Public Attributes

- u32 **buffer_count**
- u32 **jm_bm**
- u32 **shader_bm**
- u32 **tiler_bm**
- u32 **mmu_l2_bm**
- s32 **fd**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_vinstr.h

8.96 kbase_uk_hwcnt_setup Struct Reference

Public Attributes

- u64 **dump_buffer**
- u32 **jm_bm**
- u32 **shader_bm**
- u32 **tiler_bm**
- u32 **unused_1**
- u32 **mmu_l2_bm**
- u32 **padding**

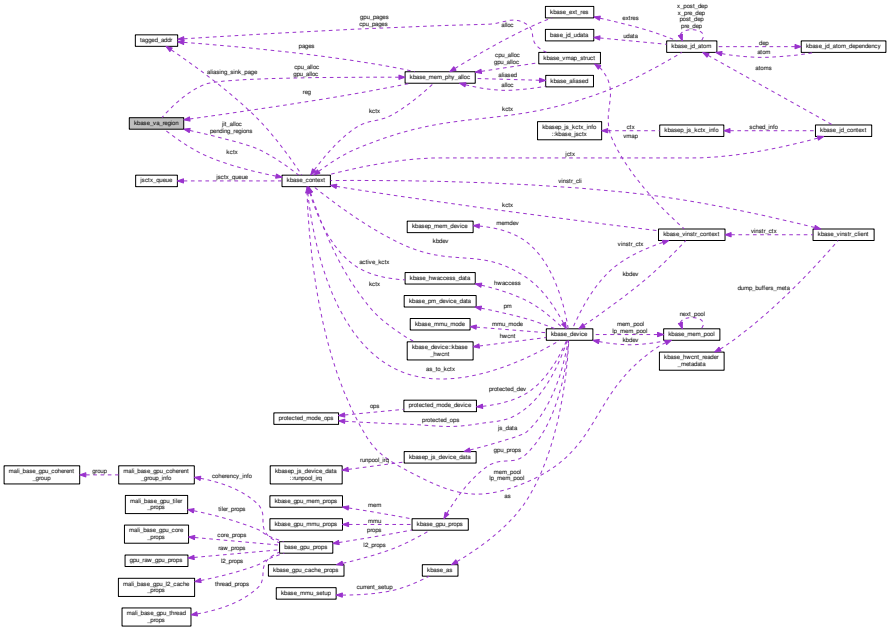
The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_vinstr.h

8.97 kbase_va_region Struct Reference

`#include <mali_kbase_mem.h>`

Collaboration diagram for kbase_va_region:



Public Attributes

- struct rb_node **rblink**
- struct list_head **link**
- struct kbase_context * **kctx**
- u64 **start_pfn**
- size_t **nr_pages**
- size_t **initial_commit**
- unsigned long **flags**
- size_t **extent**
- struct kbase_mem_phy_alloc * **cpu_alloc**
- struct kbase_mem_phy_alloc * **gpu_alloc**
- struct list_head **jit_node**

8.97.1 Detailed Description

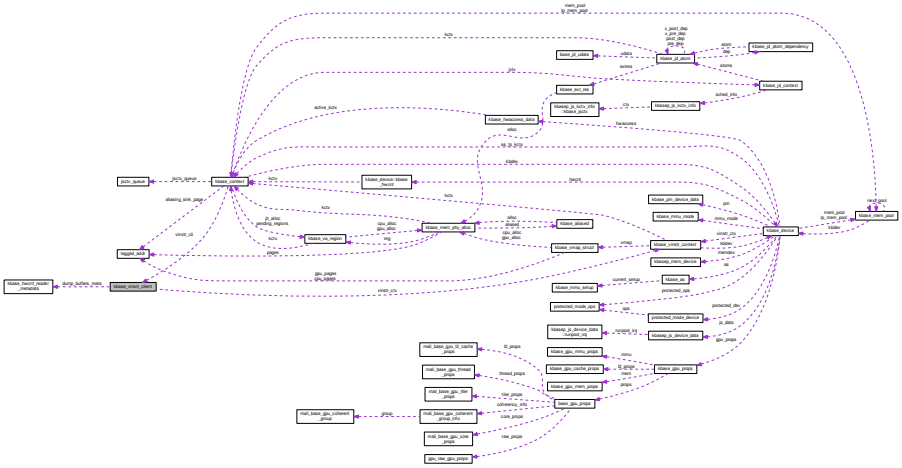
A GPU memory region, and attributes for CPU mappings.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_mem.h

8.98 kbase_vinstr_client Struct Reference

Collaboration diagram for kbase_vinstr_client:



Public Attributes

- struct kbase_vinstr_context * **vinstr_ctx**
- struct list_head **list**
- unsigned int **buffer_count**
- u32 **event_mask**
- size_t **dump_size**
- u32 **bitmap** [4]
- void __user * **legacy_buffer**
- void * **kernel_buffer**
- void * **accum_buffer**
- u64 **dump_time**
- u32 **dump_interval**
- char * **dump_buffers**
- struct kbase_hwcnt_reader_metadata * **dump_buffers_meta**
- atomic_t **meta_idx**
- atomic_t **read_idx**
- atomic_t **write_idx**
- wait_queue_head_t **waitq**
- bool **pending**

8.98.1 Detailed Description

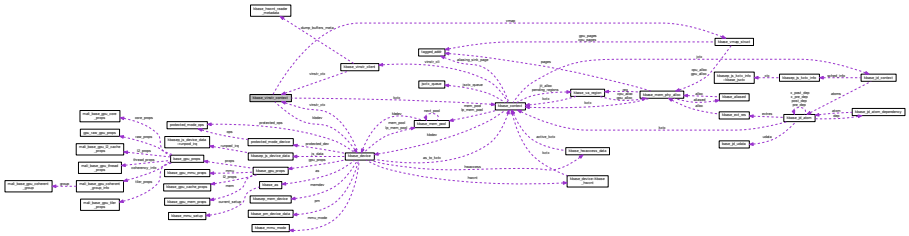
struct kbase_vinstr_client - a vinstr client attached to a vinstr context : vinstr context client is attached to : node used to attach this client to list in vinstr context : number of buffers this client is using : events this client reacts to : size of one dump buffer in bytes : bitmap request for JM, TILER, SHADER and MMU counters : userspace hwcnt dump buffer (legacy interface) : kernel hwcnt dump buffer (kernel client interface) : temporary accumulation buffer for preserving counters : next time this clients shall request hwcnt dump : interval between periodic hwcnt dumps : kernel hwcnt dump buffers allocated by this client : metadata of dump buffers : index of metadata being accessed by userspace : index of buffer read by userspace : index of buffer being written by dumping service : client's notification queue : when true, client has attached but hwcnt not yet updated

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_vinstr.c

8.99 kbase_vinstr_context Struct Reference

Collaboration diagram for kbase_vinstr_context:



Public Attributes

- struct mutex **lock**
- struct kbase_device * **kbdev**
- struct kbase_context * **kctx**
- struct kbase_vmap_struct **vmap**
- u64 **gpu_va**
- void * **cpu_va**
- size_t **dump_size**
- u32 **bitmap** [4]
- bool **reprogram**
- enum vinstr_state **state**
- struct spinlock **state_lock**
- wait_queue_head_t **suspend_waitq**
- unsigned int **suspend_cnt**
- struct work_struct **suspend_work**
- struct work_struct **resume_work**
- u32 **nclients**
- struct list_head **waiting_clients**
- struct list_head **idle_clients**
- struct list_head **suspended_clients**
- struct task_struct * **thread**
- wait_queue_head_t **waitq**
- atomic_t **request_pending**
- bool **clients_present**

8.99.1 Detailed Description

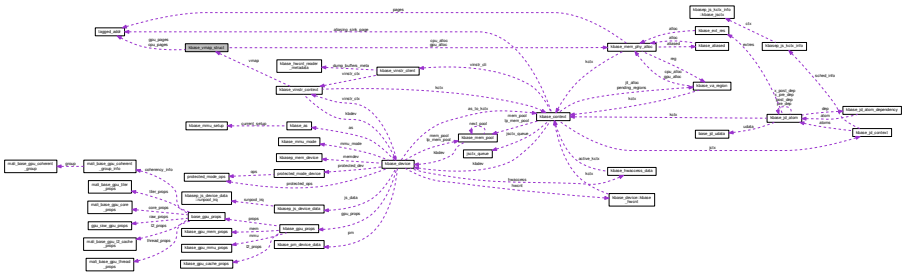
struct kbase_vinstr_context - vinstr context per device : protects the entire vinstr context : pointer to kbase device : pointer to kbase context : vinstr vmap for mapping hwcnt dump buffer : GPU hwcnt dump buffer address : the CPU side mapping of the hwcnt dump buffer : size of the dump buffer in bytes : current set of counters monitored, not always in sync with hardware : when true, reprogram hwcnt block with the new set of counters : vinstr state : protects information about vinstr state : notification queue to trigger state re-validation : reference counter of vinstr's suspend state : worker to execute on entering suspended state : worker to execute on leaving suspended state : number of attached clients, pending or otherwise : head of list of clients being periodically sampled : head of list of clients being idle : head of list of clients being suspended : periodic sampling thread : notification queue of sampling thread : request for action for sampling thread : when true, we have at least one client Note: this variable is in sync. with nclients and is present to preserve simplicity. Protected by state_lock.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_vinstr.c

8.100 kbase_vmap_struct Struct Reference

Collaboration diagram for kbase_vmap_struct:



Public Attributes

- u64 **gpu_addr**
- struct kbase_mem_phy_alloc * **cpu_alloc**
- struct kbase_mem_phy_alloc * **gpu_alloc**
- struct tagged_addr * **cpu_pages**
- struct tagged_addr * **gpu_pages**
- void * **addr**
- size_t **size**
- bool **sync_needed**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_mem_linux.h

8.101 kbasep_atom_req Struct Reference

Public Attributes

- base_jd_core_req **core_req**
- kbase_context_flags **ctx_req**
- u32 **device_nr**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_js_defs.h

8.102 kbasep_debug_assert_cb Struct Reference

Public Attributes

- kbase_debug_assert_hook * **func**
- void * **param**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_debug.h

8.103 kbasep_js_atom_retained_state Struct Reference

`#include <mali_kbase_js_defs.h>`

Public Attributes

- enum base_jd_event_code event_code
- base_jd_core_req core_req
- int **sched_priority**
- u32 **device_nr**

8.103.1 Detailed Description

Subset of atom state that can be available after jd_done_nolock() is called on that atom. A copy must be taken via kbasep_js_atom_retained_state_copy(), because the original atom could disappear.

8.103.2 Member Data Documentation

8.103.2.1 base_jd_core_req kbasep_js_atom_retained_state::core_req

core requirements

8.103.2.2 enum base_jd_event_code kbasep_js_atom_retained_state::event_code

Event code - to determine whether the atom has finished

The documentation for this struct was generated from the following file:

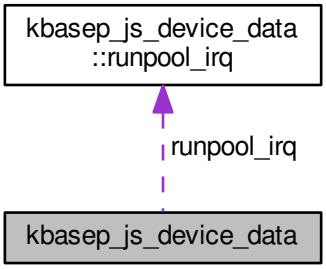
- gpu/arm/midgard/mali_kbase_js_defs.h

8.104 kbasep_js_device_data Struct Reference

KBase Device Data Job Scheduler sub-structure.

`#include <mali_kbase_js_defs.h>`

Collaboration diagram for kbasep_js_device_data:



Classes

- struct runpool_irq

Public Attributes

- struct kbasep_js_device_data::runpool_irq **runpool_irq**
- struct mutex runpool_mutex
- struct mutex queue_mutex
- struct semaphore schedule_sem
- struct list_head ctx_list_pullable [BASE_JM_MAX_NR_SLOTS]
- struct list_head ctx_list_unpullable [BASE_JM_MAX_NR_SLOTS]
- s8 nr_user_contexts_running
- s8 nr_all_contexts_running
- base_jd_core_req js_reqs [BASE_JM_MAX_NR_SLOTS]
- u32 **scheduling_period_ns**
- u32 **soft_stop_ticks**
- u32 **soft_stop_ticks_cl**

- u32 **hard_stop_ticks_ss**
- u32 **hard_stop_ticks_cl**
- u32 **hard_stop_ticks_dumping**
- u32 **gpu_reset_ticks_ss**
- u32 **gpu_reset_ticks_cl**
- u32 **gpu_reset_ticks_dumping**
- u32 ctx_timeslice_ns
- atomic_t **soft_job_timeout_ms**
- struct list_head suspended_soft_jobs_list
- int init_status
- u32 **nr_contexts_pullable**
- atomic_t **nr_contexts_runnable**

8.104.1 Detailed Description

KBase Device Data Job Scheduler sub-structure.

This encapsulates the current context of the Job Scheduler on a particular device. This context is global to the device, and is not tied to any particular struct kbase_context running on the device.

nr_contexts_running and as_free are optimized for packing together (by making them smaller types than u32). The operations on them should rarely involve masking. The use of signed types for arithmetic indicates to the compiler that the value will not rollover (which would be undefined behavior), and so under the Total License model, it is free to make optimizations based on that (i.e. to remove masking).

8.104.2 Member Data Documentation

8.104.2.1 struct list_head kbasep_js_device_data::ctx_list_pullable[BASE_JM_MAX_NR_SLOTS]

List of contexts that can currently be pulled from

8.104.2.2 struct list_head kbasep_js_device_data::ctx_list_unpullable[BASE_JM_MAX_NR_SLOTS]

List of contexts that can not currently be pulled from, but have jobs currently running.

8.104.2.3 u32 kbasep_js_device_data::ctx_timeslice_ns

Value for JS_CTX_TIMESLICE_NS Value for JS_SOFT_JOB_TIMEOUT

8.104.2.4 int kbasep_js_device_data::init_status

The initalized-flag is placed at the end, to avoid cache-pollution (we should only be using this during init/term paths).

Note

This is a write-once member, and so no locking is required to read

8.104.2.5 base_jd_core_req kbasep_js_device_data::js_reqs[BASE_JM_MAX_NR_SLOTS]

Core Requirements to match up with base_js_atom's core_req memeber

Note

This is a write-once member, and so no locking is required to read

8.104.2.6 s8 kbasep_js_device_data::nr_all_contexts_running

Number of currently scheduled contexts (including ones that are not submitting jobs)

8.104.2.7 s8 kbasep_js_device_data::nr_user_contexts_running

Number of currently scheduled user contexts (excluding ones that are not submitting jobs)

8.104.2.8 struct mutex kbasep_js_device_data::queue_mutex

Queue Lock, used to access the Policy's queue of contexts independently of the Run Pool.

Of course, you don't need the Run Pool lock to access this.

8.104.2.9 struct mutex kbasep_js_device_data::runpool_mutex

Run Pool mutex, for managing contexts within the runpool. Unless otherwise specified, you must hold this lock whilst accessing any members that follow

In addition, this is used to access:

- the kbasep_js_kctx_info::runpool substructure

8.104.2.10 struct semaphore kbasep_js_device_data::schedule_sem

Scheduling semaphore. This must be held when calling kbase_jm_kick()

8.104.2.11 struct list_head kbasep_js_device_data::suspended_soft_jobs_list

List of suspended soft jobs

The documentation for this struct was generated from the following file:

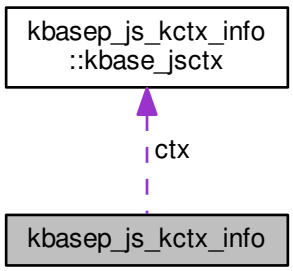
- gpu/arm/midgard/mali_kbase_js_defs.h

8.105 kbasep_js_kctx_info Struct Reference

KBase Context Job Scheduling information structure.

`#include <mali_kbase_js_defs.h>`

Collaboration diagram for kbasep_js_kctx_info:



Classes

- struct kbase_jsctx

Public Attributes

- struct kbasep_js_kctx_info::kbase_jsctx **ctx**
- int **init_status**

8.105.1 Detailed Description

KBase Context Job Scheduling information structure.

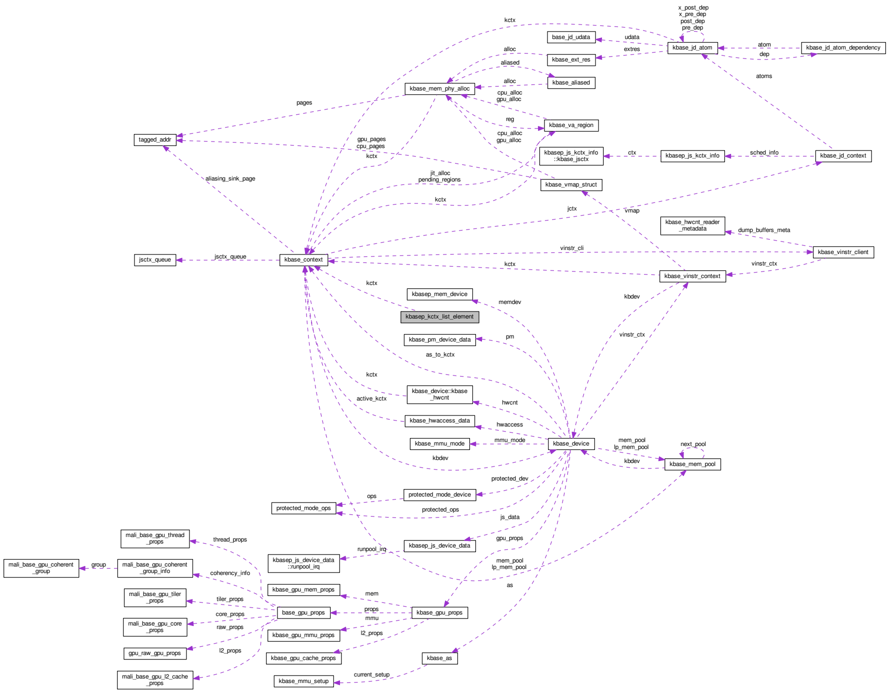
This is a substructure in the struct kbase_context that encapsulates all the scheduling information.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_js_defs.h

8.106 kbasep_kctx_list_element Struct Reference

Collaboration diagram for kbasep_kctx_list_element:



Public Attributes

- struct list_head **link**
- struct kbase_context * **kctx**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.107 kbasep_mem_device Struct Reference

Public Attributes

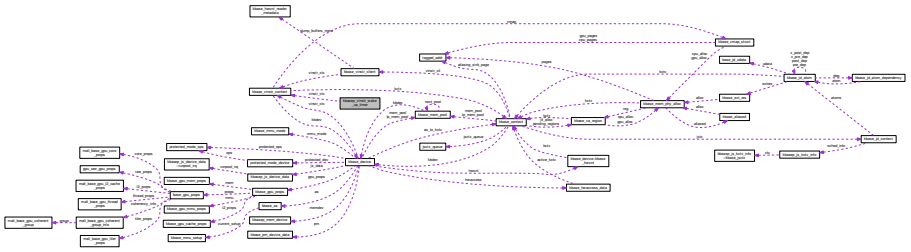
- atomic_t **used_pages**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_defs.h

8.108 kbasep_vinstr_wake_up_timer Struct Reference

Collaboration diagram for kbasep_vinstr_wake_up_timer:



Public Attributes

- struct hrtimer **hrtimer**
- struct kbase_vinstr_context * **vinstr_ctx**

8.108.1 Detailed Description

struct kbasep_vinstr_wake_up_timer - vinstr service thread wake up timer : high resolution timer : vinstr context

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_vinstr.c

8.109 mali_base_gpu_coherent_group Struct Reference

descriptor for a coherent group

`#include <mali_base_kernel.h>`

Public Attributes

- u64 core_mask
- u16 num_cores
- u16 **padding** [3]

8.109.1 Detailed Description

descriptor for a coherent group

`core_mask` exposes all cores in that coherent group, and `num_cores` provides a cached population-count for that mask.

Note

Whilst all cores are exposed in the mask, not all may be available to the application, depending on the Kernel Power policy.
if u64s must be 8-byte aligned, then this structure has 32-bits of wastage.

8.109.2 Member Data Documentation

8.109.2.1 u64 mali_base_gpu_coherent_group::core_mask

Core restriction mask required for the group

8.109.2.2 u16 mali_base_gpu_coherent_group::num_cores

Number of cores in the group

The documentation for this struct was generated from the following file:

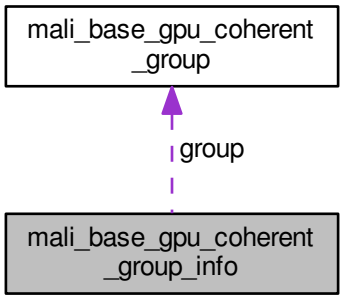
- gpu/arm/midgard/mali_base_kernel.h

8.110 mali_base_gpu_coherent_group_info Struct Reference

Coherency group information.

```
#include <mali_base_kernel.h>
```

Collaboration diagram for mali_base_gpu_coherent_group_info:



Public Attributes

- u32 **num_groups**
- u32 num_core_groups
- u32 coherency
- u32 **padding**
- struct mali_base_gpu_coherent_group group [BASE_MAX_COHERENT_GROUPS]

8.110.1 Detailed Description

Coherency group information.

Note that the sizes of the members could be reduced. However, the `group` member might be 8-byte aligned to ensure the u64 core_mask is 8-byte aligned, thus leading to wastage if the other members sizes were reduced.

The groups are sorted by core mask. The core masks are non-repeating and do not intersect.

8.110.2 Member Data Documentation

8.110.2.1 u32 mali_base_gpu_coherent_group_info::coherency

Coherency features of the memory, accessed by gpu_mem_features methods

8.110.2.2 struct mali_base_gpu_coherent_group mali_base_gpu_coherent_group_info::group[BASE_MAX_COHEREN↩ T_GROUPS]

Descriptors of coherent groups

8.110.2.3 u32 mali_base_gpu_coherent_group_info::num_core_groups

Number of core groups (coherent or not) in the GPU. Equivalent to the number of L2 Caches.

The GPU Counter dumping writes 2048 bytes per core group, regardless of whether the core groups are coherent or not. Hence this member is needed to calculate how much memory is required for dumping.

Note

Do not use it to work out how many valid elements are in the group[] member. Use num_groups instead.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.111 mali_base_gpu_core_props Struct Reference

Public Attributes

- u32 product_id
- u16 version_status
- u16 minor_revision
- u16 major_revision
- u16 **padding**
- u32 **gpu_freq_khz_max**
- u32 log2_program_counter_size
- u32 texture_features [BASE_GPU_NUM_TEXTURE_FEATURES_REGISTERS]
- u64 gpu_available_memory_size

8.111.1 Member Data Documentation

8.111.1.1 u64 mali_base_gpu_core_props::gpu_available_memory_size

Theoretical maximum memory available to the GPU. It is unlikely that a client will be able to allocate all of this memory for their own purposes, but this at least provides an upper bound on the memory available to the GPU.

This is required for OpenCL's clGetDeviceInfo() call when CL_DEVICE_GLOBAL_MEM_SIZE is requested, for OpenCL GPU devices. The client will not be expecting to allocate anywhere near this value.

8.111.1.2 u32 mali_base_gpu_core_props::log2_program_counter_size

Size of the shader program counter, in bits.

8.111.1.3 u16 mali_base_gpu_core_props::major_revision

Major release number of the GPU. "R" part of an "RnPn" release number. 4 bit values (0-15).

8.111.1.4 u16 mali_base_gpu_core_props::minor_revision

Minor release number of the GPU. "P" part of an "RnPn" release number. 8 bit values (0-255).

8.111.1.5 u32 mali_base_gpu_core_props::product_id

Product specific value.

8.111.1.6 u32 mali_base_gpu_core_props::texture_features[BASE_GPU_NUM_TEXTURE_FEATURES_REGISTERS]

TEXTURE_FEATURES_x registers, as exposed by the GPU. This is a bitpattern where a set bit indicates that the format is supported.

Before using a texture format, it is recommended that the corresponding bit be checked.

8.111.1.7 u16 mali_base_gpu_core_props::version_status

Status of the GPU release. No defined values, but starts at 0 and increases by one for each release status (alpha, beta, EAC, etc.). 4 bit values (0-15).

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.112 mali_base_gpu_l2_cache_props Struct Reference

`#include <mali_base_kernel.h>`

Public Attributes

- u8 **log2_line_size**
- u8 **log2_cache_size**
- u8 **num_l2_slices**
- u8 **padding** [5]

8.112.1 Detailed Description

More information is possible - but associativity and bus width are not required by upper-level apis.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.113 mali_base_gpu_thread_props Struct Reference

`#include <mali_base_kernel.h>`

Public Attributes

- u32 **max_threads**
- u32 **max_workgroup_size**
- u32 **max_barrier_size**
- u16 **max_registers**
- u8 **max_task_queue**
- u8 **max_thread_group_split**
- u8 **impl_tech**
- u8 **padding** [7]

8.113.1 Detailed Description

GPU threading system details.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.114 mali_base_gpu_tiler_props Struct Reference

Public Attributes

- u32 **bin_size_bytes**
- u32 **max_active_levels**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_base_kernel.h

8.115 mali_sync_pt Struct Reference

Public Attributes

- struct sync_pt **pt**
- int **order**
- int **result**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_sync_android.c

8.116 mali_sync_timeline Struct Reference

Public Attributes

- struct sync_timeline **timeline**
- atomic_t **counter**
- atomic_t **signaled**

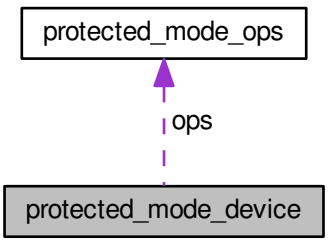
The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_sync_android.c

8.117 protected_mode_device Struct Reference

```
#include <protected_mode_switcher.h>
```

Collaboration diagram for protected_mode_device:



Public Attributes

- struct protected_mode_ops **ops**
- void * **data**

8.117.1 Detailed Description

struct protected_mode_device - Device structure for protected mode devices

- Callbacks associated with this device - Pointer to device private data

This structure should be registered with the platform device using platform_set_drvdata().

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/protected_mode_switcher.h

8.118 protected_mode_ops Struct Reference

`#include <protected_mode_switcher.h>`

Public Attributes

- int(* protected_mode_enable)(struct protected_mode_device *protected_dev)
- int(* protected_mode_disable)(struct protected_mode_device *protected_dev)

8.118.1 Detailed Description

struct protected_mode_ops - Callbacks for protected mode switch operations

: Callback to enable protected mode for device : Callback to disable protected mode for device

8.118.2 Member Data Documentation

8.118.2.1 int(* protected_mode_ops::protected_mode_disable) (struct protected_mode_device *protected_dev)

protected_mode_disable() - Disable protected mode on device, and reset device : The struct device

Return: 0 on success, non-zero on error

8.118.2.2 int(* protected_mode_ops::protected_mode_enable) (struct protected_mode_device *protected_dev)

protected_mode_enable() - Enable protected mode on device : The struct device

Return: 0 on success, non-zero on error

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/protected_mode_switcher.h

8.119 kbasep_js_device_data::runpool_irq Struct Reference

Public Attributes

- u16 submit_allowed
- s8 ctx_attr_ref_count [KBASEP_JS_CTX_ATTR_COUNT]
- u64 slot_affinities [BASE_JM_MAX_NR_SLOTS]
- s8 slot_affinity_refcount [BASE_JM_MAX_NR_SLOTS][64]

8.119.1 Member Data Documentation

8.119.1.1 s8 kbasep_js_device_data::runpool_irq::ctx_attr_ref_count[KBASEP_JS_CTX_ATTR_COUNT]

Context Attributes: Each is large enough to hold a refcount of the number of contexts that can fit into the runpool. This is currently BASE_MAX_NR_AS

Note that when BASE_MAX_NR_AS==16 we need 5 bits (not 4) to store the refcount. Hence, it's not worthwhile reducing this to bit-manipulation on u32s to save space (where in contrast, 4 bit sub-fields would be easy to do and would save space).

Whilst this must not become negative, the sign bit is used for:

- error detection in debug builds
- Optimization: it is undefined for a signed int to overflow, and so the compiler can optimize for that never happening (thus, no masking is required on updating the variable)

8.119.1.2 u64 kbasep_js_device_data::runpool_irq::slot_affinities[BASE_JM_MAX_NR_SLOTS]

Bitvector to aid affinity checking. Element 'n' bit 'i' indicates that slot 'n' is using core i (i.e. slot_affinity_refcount[n][i] > 0)

8.119.1.3 s8 kbasep_js_device_data::runpool_irq::slot_affinity_refcount[BASE_JM_MAX_NR_SLOTS][64]

Refcount for each core owned by each slot. Used to generate the slot_affinities array of bitvectors

The value of the refcount will not exceed BASE_JM_SUBMIT_SLOTS, because it is refcounted only when a job is definitely about to be submitted to a slot, and is de-refcounted immediately after a job finishes

8.119.1.4 u16 kbasep_js_device_data::runpool_irq::submit_allowed

Bitvector indicating whether a currently scheduled context is allowed to submit jobs. When bit 'N' is set in this, it indicates whether the context bound to address space 'N' is allowed to submit jobs.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_js_defs.h

8.120 tagged_addr Struct Reference

Public Attributes

- phys_addr_t **tagged_addr**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_mem_lowlevel.h

8.121 tl_stream Struct Reference

Public Attributes

- spinlock_t **lock**
- struct {
    atomic_t **size**
    char **data** [PACKET_SIZE]
  } **buffer** [PACKET_COUNT]

- atomic_t **wbi**
- atomic_t **rbi**
- int **numbered**
- atomic_t **autoflush_counter**

8.121.1 Detailed Description

struct tl_stream - timeline stream structure : message order lock : array of buffers : write buffer index : read buffer index : if non-zero stream's packets are sequentially numbered : counter tracking stream's autoflush state

This structure holds information needed to construct proper packets in the timeline stream. Each message in sequence must bear timestamp that is greater to one in previous message in the same stream. For this reason lock is held throughout the process of message creation. Each stream contains set of buffers. Each buffer will hold one MIPE packet. In case there is no free space required to store incoming message the oldest buffer is discarded. Each packet in timeline body stream has sequence number embedded (this value must increment monotonically and is used by packets receiver to discover buffer overflows. Autoflush counter is set to negative number when there is no data pending for flush and it is set to zero on every update of the buffer. Autoflush timer will increment the counter by one on every expiry. In case there will be no activity on the buffer during two consecutive timer expiries, stream buffer will be flushed.

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_tlstream.c

8.122 tp_desc Struct Reference

Public Attributes

- u32 **id**
- const char * **id_str**
- const char * **name**
- const char * **arg_types**
- const char * **arg_names**

The documentation for this struct was generated from the following file:

- gpu/arm/midgard/mali_kbase_tlstream.c

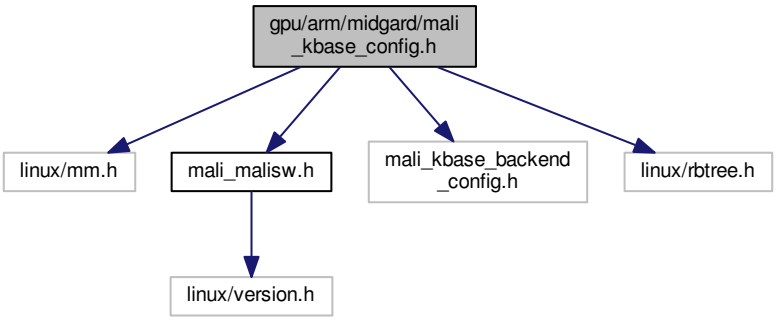
Chapter 9

File Documentation

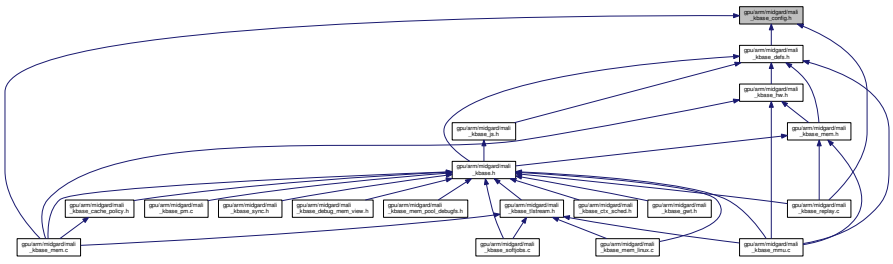
9.1 gpu/arm/midgard/mali_kbase_config.h File Reference

```
#include <linux/mm.h>
#include <mali_malisw.h>
#include <mali_kbase_backend_config.h>
#include <linux/rbtree.h>
```

Include dependency graph for mali_kbase_config.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct kbase_platform_funcs_conf
- struct kbase_pm_callback_conf
- struct kbase_io_memory_region
- struct kbase_io_resources
- struct kbase_platform_config

Functions

- struct kbase_platform_config * kbase_get_platform_config (void)

  Gets the pointer to platform config.
- int kbasep_platform_device_init (struct kbase_device *kbdev)
- void kbasep_platform_device_term (struct kbase_device *kbdev)
- int kbase_platform_register (void)
- void kbase_platform_unregister (void)

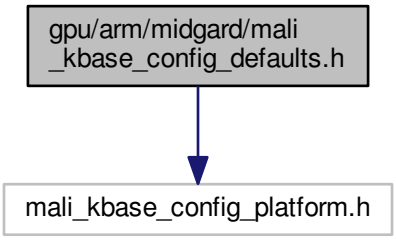
9.1.1 Detailed Description

Configuration API and Attributes for KBase

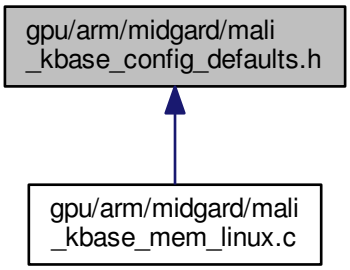
9.2 gpu/arm/midgard/mali_kbase_config_defaults.h File Reference

```
#include <mali_kbase_config_platform.h>
```

Include dependency graph for mali_kbase_config_defaults.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define DEFAULT_SECURE_BUT_LOSS_OF_PERFORMANCE false
- #define DEFAULT_ARID_LIMIT KBASE_AID_32
- #define DEFAULT_AWID_LIMIT KBASE_AID_32
- #define DEFAULT_3BIT_ARID_LIMIT KBASE_3BIT_AID_32
- #define DEFAULT_3BIT_AWID_LIMIT KBASE_3BIT_AID_32
- #define DEFAULT_UMP_GPU_DEVICE_SHIFT UMP_DEVICE_Z_SHIFT
- #define **DEFAULT_PM_DVFS_PERIOD** 100 /∗ 100ms ∗/
- #define **DEFAULT_PM_GPU_POWEROFF_TICK_NS** (400000) /∗ 400us ∗/
- #define **DEFAULT_PM_POWEROFF_TICK_SHADER** (2) /∗ 400-800us ∗/
- #define **DEFAULT_PM_POWEROFF_TICK_GPU** (2) /∗ 400-800us ∗/
- #define **DEFAULT_JS_SCHEDULING_PERIOD_NS** (100000000u) /∗ 100ms ∗/
- #define **DEFAULT_JS_SOFT_STOP_TICKS** (1) /∗ 100ms-200ms ∗/
- #define **DEFAULT_JS_SOFT_STOP_TICKS_CL** (1) /∗ 100ms-200ms ∗/
- #define **DEFAULT_JS_HARD_STOP_TICKS_SS** (50) /∗ 5s ∗/
- #define **DEFAULT_JS_HARD_STOP_TICKS_SS_8408** (300) /∗ 30s ∗/
- #define **DEFAULT_JS_HARD_STOP_TICKS_CL** (50) /∗ 5s ∗/
- #define **DEFAULT_JS_HARD_STOP_TICKS_DUMPING** (15000) /∗ 1500s ∗/
- #define **DEFAULT_JS_SOFT_JOB_TIMEOUT** (3000) /∗ 3s ∗/
- #define **DEFAULT_JS_RESET_TICKS_SS** (55) /∗ 5.5s ∗/
- #define **DEFAULT_JS_RESET_TICKS_SS_8408** (450) /∗ 45s ∗/
- #define **DEFAULT_JS_RESET_TICKS_CL** (55) /∗ 5.5s ∗/
- #define **DEFAULT_JS_RESET_TICKS_DUMPING** (15020) /∗ 1502s ∗/
- #define **DEFAULT_RESET_TIMEOUT_MS** (3000) /∗ 3s ∗/
- #define **DEFAULT_JS_CTX_TIMESLICE_NS** (50000000) /∗ 50ms ∗/
- #define **PLATFORM_POWER_DOWN_ONLY** (0)
- #define **DEFAULT_GPU_FREQ_KHZ_MAX** (5000)

Enumerations

- enum { KBASE_AID_32 = 0x0, KBASE_AID_16 = 0x3, KBASE_AID_8 = 0x2, KBASE_AID_4 = 0x1 }
- enum {
  KBASE_3BIT_AID_32 = 0x0, **KBASE_3BIT_AID_28** = 0x1, **KBASE_3BIT_AID_24** = 0x2, **KBASE_3BIT_↩ AID_20** = 0x3,
  KBASE_3BIT_AID_16 = 0x4, **KBASE_3BIT_AID_12** = 0x5, **KBASE_3BIT_AID_8** = 0x6, **KBASE_3BIT_↩ AID_4** = 0x7 }

9.2.1 Detailed Description

Default values for configuration settings

9.2.2 Macro Definition Documentation

9.2.2.1 #define DEFAULT_3BIT_ARID_LIMIT KBASE_3BIT_AID_32

Default setting for read Address ID limiting on AXI bus.

Default value: KBASE_3BIT_AID_32 (no limit). Note hardware implementation may limit to a lower value.

9.2.2.2 #define DEFAULT_3BIT_AWID_LIMIT KBASE_3BIT_AID_32

Default setting for write Address ID limiting on AXI.

Default value: KBASE_3BIT_AID_32 (no limit). Note hardware implementation may limit to a lower value.

9.2.2.3 #define DEFAULT_ARID_LIMIT KBASE_AID_32

Default setting for read Address ID limiting on AXI bus.

Attached value: u32 register value KBASE_AID_32 - use the full 32 IDs (5 ID bits) KBASE_AID_16 - use 16 IDs (4 ID bits) KBASE_AID_8 - use 8 IDs (3 ID bits) KBASE_AID_4 - use 4 IDs (2 ID bits) Default value: KBASE_AID_32 (no limit). Note hardware implementation may limit to a lower value.

9.2.2.4 #define DEFAULT_AWID_LIMIT KBASE_AID_32

Default setting for write Address ID limiting on AXI.

Attached value: u32 register value KBASE_AID_32 - use the full 32 IDs (5 ID bits) KBASE_AID_16 - use 16 IDs (4 ID bits) KBASE_AID_8 - use 8 IDs (3 ID bits) KBASE_AID_4 - use 4 IDs (2 ID bits) Default value: KBASE_AID_32 (no limit). Note hardware implementation may limit to a lower value.

9.2.2.5 #define DEFAULT_SECURE_BUT_LOSS_OF_PERFORMANCE false

Boolean indicating whether the driver is configured to be secure at a potential loss of performance.

This currently affects only r0p0-15dev0 HW and earlier.

On r0p0-15dev0 HW and earlier, there are tradeoffs between security and performance:

- When this is set to true, the driver remains fully secure, but potentially loses performance compared with setting this to false.
- When set to false, the driver is open to certain security attacks.

From r0p0-00rel0 and onwards, there is no security loss by setting this to false, and no performance loss by setting it to true.

9.2.2.6 #define DEFAULT_UMP_GPU_DEVICE_SHIFT UMP_DEVICE_Z_SHIFT

Default UMP device mapping. A UMP_DEVICE_<device>_SHIFT value which defines which UMP device this GPU should be mapped to.

9.2.3 Enumeration Type Documentation

9.2.3.1 anonymous enum

Enumerator

KBASE_AID_32 Use unrestricted Address ID width on the AXI bus.

KBASE_AID_16 Restrict GPU to a half of maximum Address ID count. This will reduce performance, but reduce bus load due to GPU.

KBASE_AID_8 Restrict GPU to a quarter of maximum Address ID count. This will reduce performance, but reduce bus load due to GPU.

KBASE_AID_4 Restrict GPU to an eighth of maximum Address ID count. This will reduce performance, but reduce bus load due to GPU.

9.2.3.2 anonymous enum

Enumerator

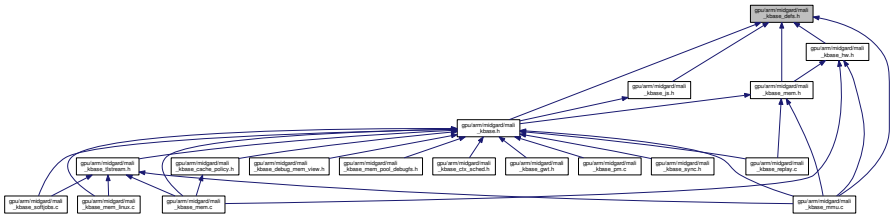
KBASE_3BIT_AID_32 Use unrestricted Address ID width on the AXI bus. Restricting ID width will reduce performance & bus load due to GPU.

9.3 gpu/arm/midgard/mali_kbase_defs.h File Reference

```
#include <mali_kbase_config.h>
#include <mali_base_hwconfig_features.h>
#include <mali_base_hwconfig_issues.h>
#include <mali_kbase_mem_lowlevel.h>
#include <mali_kbase_mmu_hw.h>
#include <mali_kbase_instr_defs.h>
#include <mali_kbase_pm.h>
#include <mali_kbase_gpuprops_types.h>
#include <protected_mode_switcher.h>
#include <linux/atomic.h>
#include <linux/mempool.h>
#include <linux/slab.h>
#include <linux/file.h>
#include <linux/sizes.h>
#include "mali_kbase_fence_defs.h"
#include <linux/clk.h>
#include <linux/regulator/consumer.h>
#include "mali_kbase_js_defs.h"
#include "mali_kbase_hwaccess_defs.h"
#include "mali_kbase_trace_defs.h"
```

Include dependency graph for mali_kbase_defs.h:

This graph shows which files directly or indirectly include this file:



Classes

- struct kbase_jd_atom_dependency
- struct kbase_io_access
- struct kbase_io_history
- struct kbase_ext_res
- struct kbase_jd_atom
- struct kbase_jd_context
- struct kbase_device_info
- struct kbase_mmu_setup
- struct kbase_as
- struct kbasep_mem_device
- struct kbase_trace
- struct kbasep_kctx_list_element
- struct kbase_pm_device_data
- struct kbase_mem_pool
- struct kbase_devfreq_opp
- struct kbase_mmu_mode
- struct kbase_device
- struct kbase_device::kbase_hwcnt
- struct jsctx_queue
- struct kbase_sub_alloc
- struct kbase_context
- struct kbase_ctx_ext_res_meta
- struct job_descriptor_header

Macros

- #define KBASE_TRACE_ENABLE 0
- #define KBASE_TRACE_DUMP_ON_JOB_SLOT_ERROR 1
- #define ZAP_TIMEOUT 1000
- #define RESET_TIMEOUT 500
- #define KBASE_DISABLE_SCHEDULING_SOFT_STOPS 0
- #define KBASE_DISABLE_SCHEDULING_HARD_STOPS 0
- #define BASE_JM_MAX_NR_SLOTS 3
- #define BASE_MAX_NR_AS 16
- #define **MIDGARD_MMU_VA_BITS** 48
- #define **MIDGARD_MMU_LEVEL**(x) (x)
- #define **MIDGARD_MMU_TOPLEVEL** MIDGARD_MMU_LEVEL(0)
- #define **MIDGARD_MMU_BOTTOMLEVEL** MIDGARD_MMU_LEVEL(3)
- #define **GROWABLE_FLAGS_REQUIRED** (KBASE_REG_PF_GROW | KBASE_REG_GPU_WR)

- #define KBASEP_AS_NR_INVALID (-1)
- #define **KBASE_LOCK_REGION_MAX_SIZE** (63)
- #define **KBASE_LOCK_REGION_MIN_SIZE** (11)
- #define **KBASE_TRACE_SIZE_LOG2** 8 /* 256 entries */
- #define **KBASE_TRACE_SIZE** (1 << KBASE_TRACE_SIZE_LOG2)
- #define **KBASE_TRACE_MASK** ((1 << KBASE_TRACE_SIZE_LOG2)-1)
- #define **KBASEP_FORCE_REPLAY_DISABLED** 0
- #define **KBASEP_FORCE_REPLAY_RANDOM_LIMIT** 16
- #define KBASE_KATOM_FLAG_BEEN_SOFT_STOPPPED (1<<1)
- #define KBASE_KATOM_FLAGS_RERUN (1<<2)
- #define **KBASE_KATOM_FLAGS_JOBCHAIN** (1<<3)
- #define KBASE_KATOM_FLAG_BEEN_HARD_STOPPED (1<<4)
- #define KBASE_KATOM_FLAG_IN_DISJOINT (1<<5)
- #define **KBASE_KATOM_FLAG_X_DEP_BLOCKED** (1<<7)
- #define **KBASE_KATOM_FLAG_FAIL_BLOCKER** (1<<8)
- #define **KBASE_KATOM_FLAG_JSCTX_IN_X_DEP_LIST** (1<<9)
- #define **KBASE_KATOM_FLAG_HOLDING_CTX_REF** (1<<10)
- #define **KBASE_KATOM_FLAG_PROTECTED** (1<<11)
- #define **KBASE_KATOM_FLAG_JSCTX_IN_TREE** (1<<12)
- #define JS_COMMAND_SW_CAUSES_DISJOINT 0x100
- #define JS_COMMAND_SW_BITS (JS_COMMAND_SW_CAUSES_DISJOINT)
- #define JS_COMMAND_SOFT_STOP_WITH_SW_DISJOINT (JS_COMMAND_SW_CAUSES_DISJOINT | JS_COMMAND_SOFT_STOP)
- #define **KBASEP_ATOM_ID_INVALID** BASE_JD_ATOM_COUNT
- #define **KBASE_SERIALIZE_INTRA_SLOT** (1 << 0)
- #define **KBASE_SERIALIZE_INTER_SLOT** (1 << 1)
- #define **KBASE_SERIALIZE_RESET** (1 << 2)
- #define **KBASE_JD_DEP_QUEUE_SIZE** 256
- #define **KBASE_TRACE_CODE**(X) KBASE_TRACE_CODE_ ## X
- #define **KBASE_TRACE_CODE_MAKE_CODE**(X) KBASE_TRACE_CODE(X)
- #define **KBASE_TRACE_FLAG_REFCOUNT** (((u8)1) << 0)
- #define **KBASE_TRACE_FLAG_JOBSLOT** (((u8)1) << 1)
- #define **DEVNAME_SIZE** 16
- #define **KBASE_API_VERSION**(major, minor)
- #define **HR_TIMER_DELAY_MSEC**(x) (ns_to_ktime(((u64)(x))*1000000U))
- #define **HR_TIMER_DELAY_NSEC**(x) (ns_to_ktime(x))
- #define **KBASE_CLEAN_CACHE_MAX_LOOPS** 100000
- #define **KBASE_AS_INACTIVE_MAX_LOOPS** 100000
- #define **BASEP_JD_REPLAY_LIMIT** 15

Typedefs

- typedef u32 kbase_as_poke_state

Enumerations

- enum **kbase_atom_gpu_rb_state** {
  KBASE_ATOM_GPU_RB_NOT_IN_SLOT_RB, **KBASE_ATOM_GPU_RB_WAITING_BLOCKED**, **KBASE_ATOM_GPU_RB_WAITING_PROTECTED_MODE_PREV**, **KBASE_ATOM_GPU_RB_WAITING_PROTECTED_MODE_TRANSITION**,
  KBASE_ATOM_GPU_RB_WAITING_FOR_CORE_AVAILABLE, **KBASE_ATOM_GPU_RB_WAITING_AFFINITY**, **KBASE_ATOM_GPU_RB_READY**, **KBASE_ATOM_GPU_RB_SUBMITTED**,
  KBASE_ATOM_GPU_RB_RETURN_TO_JS = -1 }

- enum **kbase_atom_enter_protected_state** { **KBASE_ATOM_ENTER_PROTECTED_CHECK** = 0, **KBA↩ SE_ATOM_ENTER_PROTECTED_VINSTR**, **KBASE_ATOM_ENTER_PROTECTED_IDLE_L2**, **KBASE↩ _ATOM_ENTER_PROTECTED_FINISHED** }
- enum **kbase_atom_exit_protected_state** { **KBASE_ATOM_EXIT_PROTECTED_CHECK** = 0, **KBASE_↩ ATOM_EXIT_PROTECTED_IDLE_L2**, **KBASE_ATOM_EXIT_PROTECTED_RESET**, **KBASE_ATOM_E↩ XIT_PROTECTED_RESET_WAIT** }
- enum { **KBASE_AS_POKE_STATE_IN_FLIGHT** = 1<<0, **KBASE_AS_POKE_STATE_KILLING_POKE** = 1<<1 }
- enum **kbase_trace_code** {
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩ ODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **K↩ BASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
  KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_C↩**

ODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY),
KBASE_TRACE_CODE_MAKE_CODE =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_MAKE_CODE** =(CORE_CTX_DESTROY), **KBASE_TRACE_CODE_COUNT** }

- enum kbase_timeline_pm_event {
  KBASEP_TIMELINE_PM_EVENT_FIRST, KBASE_TIMELINE_PM_EVENT_RESERVED_0 = KBASEP_TIMELINE_PM_EVENT_FIRST, KBASE_TIMELINE_PM_EVENT_GPU_STATE_CHANGED, KBASE_TIMELINE_PM_EVENT_GPU_ACTIVE,
  KBASE_TIMELINE_PM_EVENT_GPU_IDLE, KBASE_TIMELINE_PM_EVENT_RESERVED_4, KBASE_TIMELINE_PM_EVENT_RESERVED_5, KBASE_TIMELINE_PM_EVENT_RESERVED_6,
  KBASE_TIMELINE_PM_EVENT_CHANGE_GPU_STATE, **KBASEP_TIMELINE_PM_EVENT_LAST** = KBASE_TIMELINE_PM_EVENT_CHANGE_GPU_STATE }
- enum kbase_context_flags {
  KCTX_COMPAT = 1U << 0, **KCTX_RUNNABLE_REF** = 1U << 1, **KCTX_ACTIVE** = 1U << 2, **KCTX_PULLED** = 1U << 3,
  KCTX_MEM_PROFILE_INITIALIZED = 1U << 4, **KCTX_INFINITE_CACHE** = 1U << 5, **KCTX_SUBMIT_DISABLED** = 1U << 6, **KCTX_PRIVILEGED** = 1U << 7,
  KCTX_SCHEDULED = 1U << 8, **KCTX_DYING** = 1U << 9, **KCTX_NO_IMPLICIT_SYNC** = 1U << 10, **KCTX_FORCE_SAME_VA** = 1U << 11 }
- enum **kbase_reg_access_type** { **REG_READ**, **REG_WRITE** }
- enum **kbase_share_attr_bits** { **SHARE_BOTH_BITS** = (2ULL << 8), **SHARE_INNER_BITS** = (3ULL << 8) }

Functions

- struct kbase_mmu_mode const ∗ **kbase_mmu_mode_get_lpae** (void)
- struct kbase_mmu_mode const ∗ **kbase_mmu_mode_get_aarch64** (void)

9.3.1 Detailed Description

Defintions (types, defines, etcs) common to Kbase. They are placed here to allow the hierarchy of header files to work.

9.3.2 Macro Definition Documentation

9.3.2.1 #define BASE_JM_MAX_NR_SLOTS 3

The maximum number of Job Slots to support in the Hardware.

You can optimize this down if your target devices will only ever support a small number of job slots.

9.3.2.2 #define BASE_MAX_NR_AS 16

The maximum number of Address Spaces to support in the Hardware.

You can optimize this down if your target devices will only ever support a small number of Address Spaces

9.3.2.3 #define JS_COMMAND_SOFT_STOP_WITH_SW_DISJOINT (JS_COMMAND_SW_CAUSES_DISJOINT | JS_COMMAND_SOFT_STOP)

Soft-stop command that causes a Disjoint event. This of course isn't entirely masked off by JS_COMMAND_MASK

9.3.2.4 #define JS_COMMAND_SW_BITS (JS_COMMAND_SW_CAUSES_DISJOINT)

Bitmask of all SW related flags

9.3.2.5 #define JS_COMMAND_SW_CAUSES_DISJOINT 0x100

This command causes a disjoint event

9.3.2.6 #define KBASE_API_VERSION(*major, minor*)

Value:

```
((((major) & 0xFFF) << 20)  | \
                    (((minor) & 0xFFF) << 8) | \
                    ((0 & 0xFF) << 0))
```

9.3.2.7 #define KBASE_DISABLE_SCHEDULING_HARD_STOPS 0

Prevent hard-stops from occuring in scheduling situations

This is not due to HW issues, but when scheduling is desired to be more predictable.

Note

Hard stop will still be used for non-scheduling purposes e.g. when terminating a context.
if not in use, define this value to 0 instead of #undef'ing it

9.3.2.8 #define KBASE_DISABLE_SCHEDULING_SOFT_STOPS 0

Prevent soft-stops from occuring in scheduling situations

This is not due to HW issues, but when scheduling is desired to be more predictable.

Therefore, soft stop may still be disabled due to HW issues.

Note

Soft stop will still be used for non-scheduling purposes e.g. when terminating a context.
if not in use, define this value to 0 instead of #undef'ing it

9.3.2.9 #define KBASE_KATOM_FLAG_BEEN_HARD_STOPPED (1<<4)

Atom has been previously hard-stopped.

9.3.2.10 #define KBASE_KATOM_FLAG_BEEN_SOFT_STOPPPED (1<<1)

Atom has been previously soft-stoppped

9.3.2.11 #define KBASE_KATOM_FLAG_IN_DISJOINT (1<<5)

Atom has caused us to enter disjoint state

9.3.2.12 #define KBASE_KATOM_FLAGS_RERUN (1<<2)

Atom has been previously retried to execute

9.3.2.13 #define KBASE_TRACE_DUMP_ON_JOB_SLOT_ERROR 1

Dump Job slot trace on error (only active if KBASE_TRACE_ENABLE != 0)

9.3.2.14 #define KBASE_TRACE_ENABLE 0

Enable SW tracing when set

9.3.2.15 #define KBASEP_AS_NR_INVALID (-1)

setting in kbase_context::as_nr that indicates it's invalid

9.3.2.16 #define RESET_TIMEOUT 500

Number of milliseconds before we time out on a GPU soft/hard reset

9.3.2.17 #define ZAP_TIMEOUT 1000

Number of milliseconds before resetting the GPU when a job cannot be "zapped" from the hardware. Note that the time is actually ZAP_TIMEOUT+SOFT_STOP_RESET_TIMEOUT between the context zap starting and the GPU actually being reset to give other contexts time for their jobs to be soft-stopped and removed from the hardware before resetting.

9.3.3 Typedef Documentation

9.3.3.1 typedef u32 kbase_as_poke_state

Poking state for BASE_HW_ISSUE_8316

9.3.4 Enumeration Type Documentation

9.3.4.1 anonymous enum

Poking state for BASE_HW_ISSUE_8316

9.3.4.2 enum kbase_context_flags

enum kbase_context_flags - Flags for kbase contexts

: Set when the context process is a compat process, 32-bit process on a 64-bit kernel.

: Set when context is counted in kbdev->js_data.nr_contexts_runnable. Must hold queue_mutex when accessing.

: Set when the context is active.

: Set when last kick() caused atoms to be pulled from this context.

: Set when the context's memory profile has been initialized.

: Set when infinite cache is to be enabled for new allocations. Existing allocations will not change.

: Set to prevent context from submitting any jobs.

:Set if the context uses an address space and should be kept scheduled in.

: Set when the context is scheduled on the Run Pool. This is only ever updated whilst the jsctx_mutex is held.

: Set when the context process is in the process of being evicted.

: Set when explicit Android fences are in use on this context, to disable use of implicit dma-buf fences. This is used to avoid potential synchronization deadlocks.

: Set when BASE_MEM_SAME_VA should be forced on memory allocations. For 64-bit clients it is enabled by default, and disabled by default on 32-bit clients. Being able to clear this flag is only used for testing purposes of the custom zone allocation on 64-bit user-space builds, where we also require more control than is available through e.g. the JIT allocation mechanism. However, the 64-bit user-space client must still reserve a JIT region using KBASE_IOCTL_MEM_JIT_INIT

All members need to be separate bits. This enum is intended for use in a bitmask where multiple values get OR-ed together.

9.3.4.3 enum kbase_timeline_pm_event

Event IDs for the power management framework.

Any of these events might be missed, so they should not be relied upon to find the precise state of the GPU at a particular time in the trace. Overall, we should get a high percentage of these events for statisical purposes, and so a few missing should not be a problem

Enumerator

KBASE_TIMELINE_PM_EVENT_RESERVED_0 Event reserved for backwards compatibility with 'init' events

KBASE_TIMELINE_PM_EVENT_GPU_STATE_CHANGED The power state of the device has changed.
Specifically, the device has reached a desired or available state.

KBASE_TIMELINE_PM_EVENT_GPU_ACTIVE The GPU is becoming active.
This event is sent when the first context is about to use the GPU.

KBASE_TIMELINE_PM_EVENT_GPU_IDLE The GPU is becoming idle.
This event is sent when the last context has finished using the GPU.

KBASE_TIMELINE_PM_EVENT_RESERVED_4 Event reserved for backwards compatibility with 'policy_change' events

KBASE_TIMELINE_PM_EVENT_RESERVED_5 Event reserved for backwards compatibility with 'system_suspend' events

KBASE_TIMELINE_PM_EVENT_RESERVED_6 Event reserved for backwards compatibility with 'system_resume' events

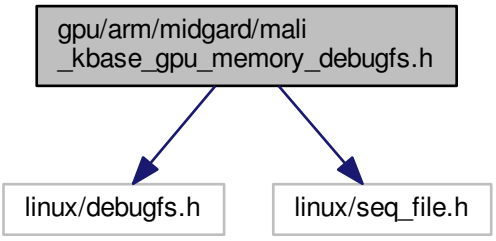
KBASE_TIMELINE_PM_EVENT_CHANGE_GPU_STATE The job scheduler is requesting to power up/down cores.
This event is sent when:

- powered down cores are needed to complete a job
- powered up cores are not needed anymore

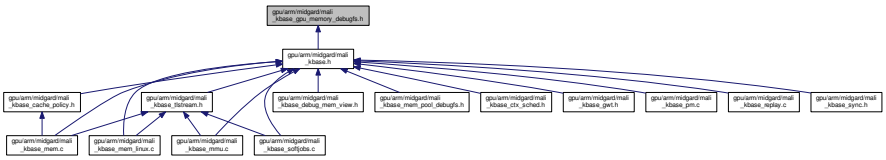
9.4 gpu/arm/midgard/mali_kbase_gpu_memory_debugfs.h File Reference

```
#include <linux/debugfs.h>
#include <linux/seq_file.h>
```

Include dependency graph for mali_kbase_gpu_memory_debugfs.h:



This graph shows which files directly or indirectly include this file:



Functions

- void kbasep_gpu_memory_debugfs_init (struct kbase_device *kbdev)

  Initialize gpu_memory debugfs entry.

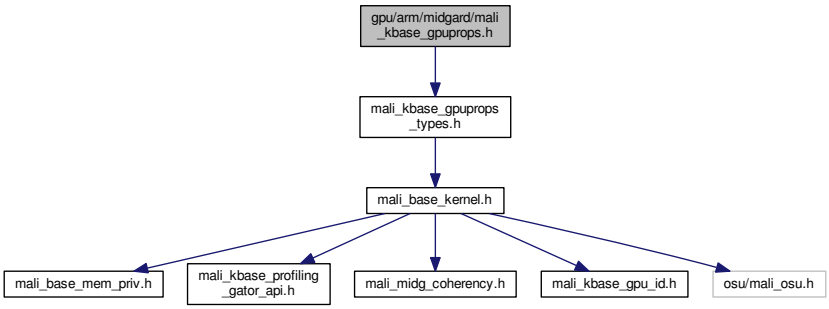
9.4.1 Detailed Description

Header file for gpu_memory entry in debugfs

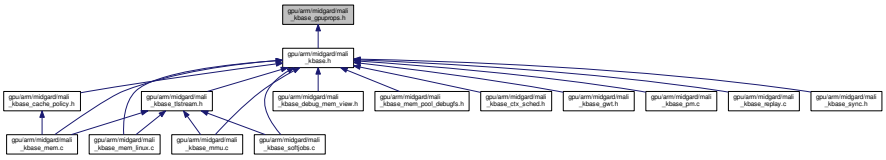
9.5 gpu/arm/midgard/mali_kbase_gpuprops.h File Reference

```
#include "mali_kbase_gpuprops_types.h"
```

Include dependency graph for mali_kbase_gpuprops.h:



This graph shows which files directly or indirectly include this file:



Functions

- void kbase_gpuprops_set (struct kbase_device *kbdev)

  Set up Kbase GPU properties.
- void kbase_gpuprops_set_features (struct kbase_device *kbdev)
- int kbase_gpuprops_populate_user_buffer (struct kbase_device *kbdev)
- void kbase_gpuprops_update_core_props_gpu_id (base_gpu_props *const gpu_props)

9.5.1 Detailed Description

Base kernel property query APIs

9.5.2 Function Documentation

9.5.2.1 int kbase_gpuprops_populate_user_buffer (struct kbase_device * *kbdev*)

kbase_gpuprops_populate_user_buffer - Populate the GPU properties buffer : The kbase device

Fills kbdev->gpu_props->prop_buffer with the GPU properties for user space to read.

9.5.2.2 void kbase_gpuprops_set (struct kbase_device * *kbdev*)

Set up Kbase GPU properties.

Set up Kbase GPU properties with information from the GPU registers

Parameters

kbdev	The struct kbase_device structure for the device

9.5.2.3 void kbase_gpuprops_set_features (struct kbase_device * *kbdev*)

kbase_gpuprops_set_features - Set up Kbase GPU properties : Device pointer

This function sets up GPU properties that are dependent on the hardware features bitmask. This function must be preceeded by a call to kbase_hw_set_features_mask().

9.5.2.4 void kbase_gpuprops_update_core_props_gpu_id (base_gpu_props *const *gpu_props*)

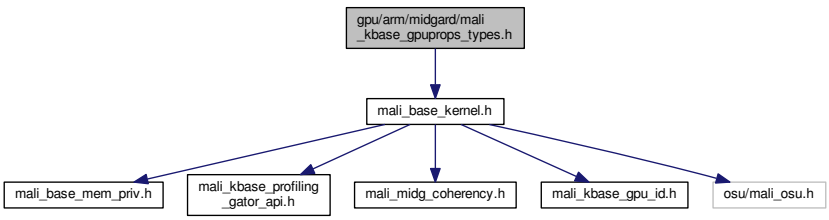
kbase_gpuprops_update_core_props_gpu_id - break down gpu id value : the &base_gpu_props structure

Break down gpu_id value stored in base_gpu_props::raw_props.gpu_id into separate fields (version_status, minor↩ _revision, major_revision, product_id) stored in base_gpu_props::core_props.

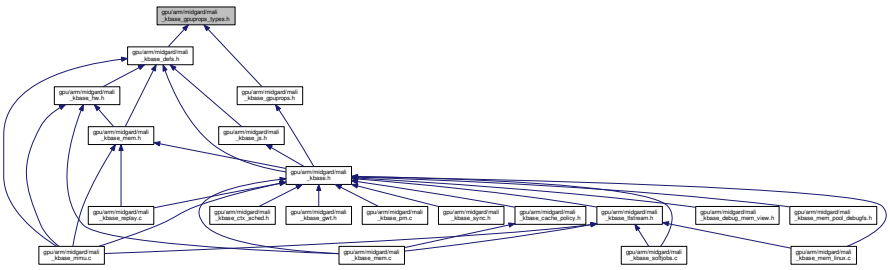
9.6 gpu/arm/midgard/mali_kbase_gpuprops_types.h File Reference

```
#include "mali_base_kernel.h"
```

Include dependency graph for mali_kbase_gpuprops_types.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct kbase_gpuprops_regdump
- struct kbase_gpu_cache_props
- struct kbase_gpu_mem_props
- struct kbase_gpu_mmu_props
- struct kbase_gpu_props

Macros

- #define **KBASE_GPU_SPEED_MHZ** 123
- #define **KBASE_GPU_PC_SIZE_LOG2** 24U

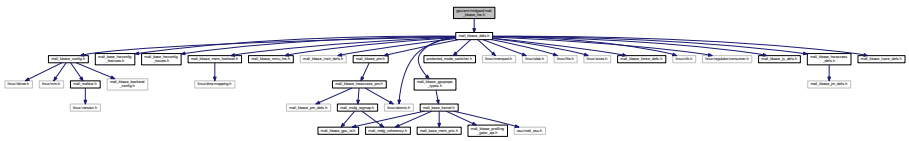
9.6.1 Detailed Description

Base kernel property query APIs

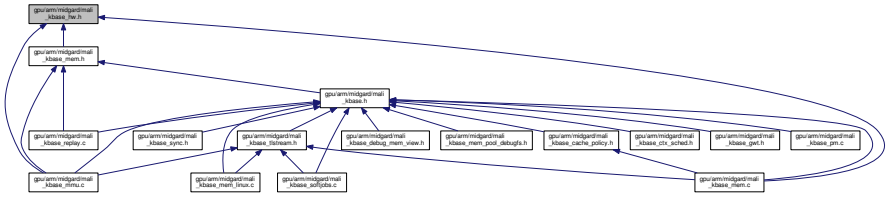
9.7 gpu/arm/midgard/mali_kbase_hw.h File Reference

```
#include "mali_kbase_defs.h"
```

Include dependency graph for mali_kbase_hw.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define kbase_hw_has_issue(kbdev, issue) test_bit(issue, &(kbdev)->hw_issues_mask[0])

  Tell whether a work-around should be enabled.
- #define kbase_hw_has_feature(kbdev, feature) test_bit(feature, &(kbdev)->hw_features_mask[0])

  Tell whether a feature is supported.

Functions

- int kbase_hw_set_issues_mask (struct kbase_device ∗kbdev)
- void kbase_hw_set_features_mask (struct kbase_device ∗kbdev)

  Set the features mask depending on the GPU ID.

9.7.1 Detailed Description

Run-time work-arounds helpers

9.7.2 Function Documentation

9.7.2.1 int kbase_hw_set_issues_mask (struct kbase_device ∗ *kbdev*)

kbase_hw_set_issues_mask - Set the hardware issues mask based on the GPU ID : Device pointer

Return: 0 if the GPU ID was recognized, otherwise -EINVAL.

The GPU ID is read from the .

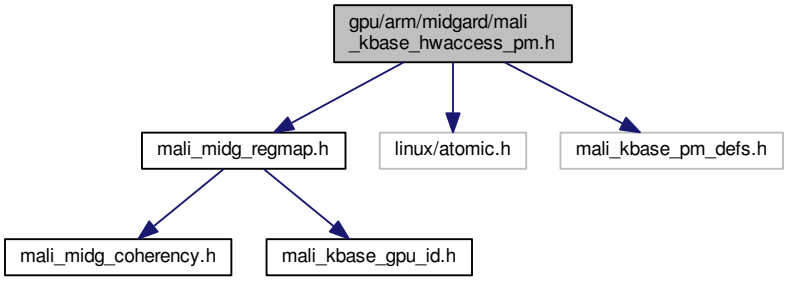
In debugging versions of the driver, unknown versions of a known GPU with a new-format ID will be treated as the most recent known version not later than the actual version. In such circumstances, the GPU ID in will also be replaced with the most recent known version.

Note: The GPU configuration must have been read by kbase_gpuprops_get_props() before calling this function.

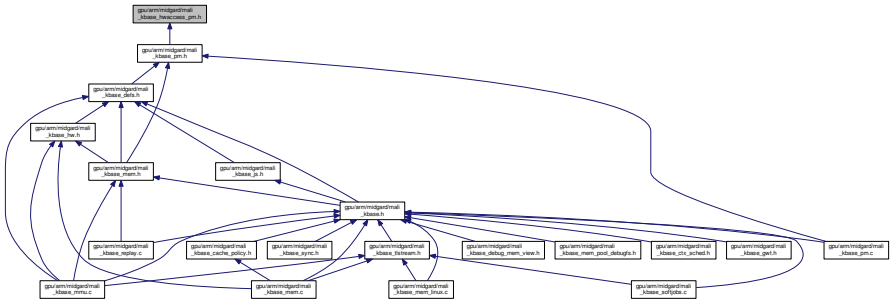
9.8 gpu/arm/midgard/mali_kbase_hwaccess_pm.h File Reference

```
#include <mali_midg_regmap.h>
#include <linux/atomic.h>
#include <mali_kbase_pm_defs.h>
```

Include dependency graph for mali_kbase_hwaccess_pm.h:



This graph shows which files directly or indirectly include this file:



Functions

- int kbase_hwaccess_pm_init (struct kbase_device ∗kbdev)
- void kbase_hwaccess_pm_term (struct kbase_device ∗kbdev)
- int kbase_hwaccess_pm_powerup (struct kbase_device ∗kbdev, unsigned int flags)
- void kbase_hwaccess_pm_halt (struct kbase_device ∗kbdev)
- void kbase_hwaccess_pm_suspend (struct kbase_device ∗kbdev)
- void kbase_hwaccess_pm_resume (struct kbase_device ∗kbdev)
- void kbase_hwaccess_pm_gpu_active (struct kbase_device ∗kbdev)
- void kbase_hwaccess_pm_gpu_idle (struct kbase_device ∗kbdev)
- void kbase_pm_set_debug_core_mask (struct kbase_device ∗kbdev, u64 new_core_mask_js0, u64 new_↩ core_mask_js1, u64 new_core_mask_js2)
- const struct kbase_pm_ca_policy ∗ kbase_pm_ca_get_policy (struct kbase_device ∗kbdev)
- void kbase_pm_ca_set_policy (struct kbase_device ∗kbdev, const struct kbase_pm_ca_policy ∗policy)
- int kbase_pm_ca_list_policies (const struct kbase_pm_ca_policy ∗const ∗∗policies)
- const struct kbase_pm_policy ∗ kbase_pm_get_policy (struct kbase_device ∗kbdev)
- void kbase_pm_set_policy (struct kbase_device ∗kbdev, const struct kbase_pm_policy ∗policy)
- int kbase_pm_list_policies (const struct kbase_pm_policy ∗const ∗∗policies)

9.8.1 Detailed Description

HW access power manager common APIs

9.8.2 Function Documentation

9.8.2.1 void kbase_hwaccess_pm_gpu_active (struct kbase_device ∗ *kbdev*)

Perform any required actions for activating the GPU. Called when the first context goes active.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.8.2.2 void kbase_hwaccess_pm_gpu_idle (struct kbase_device * kbdev)

Perform any required actions for idling the GPU. Called when the last context goes idle.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.8.2.3 void kbase_hwaccess_pm_halt (struct kbase_device * kbdev)

Halt the power management framework.

Should ensure that no new interrupts are generated, but allow any currently running interrupt handlers to complete successfully. The GPU is forced off by the time this function returns, regardless of whether or not the active power policy asks for the GPU to be powered off.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.8.2.4 int kbase_hwaccess_pm_init (struct kbase_device * kbdev)

Initialize the power management framework.

Must be called before any other power management function

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

Returns

0 if the power management framework was successfully initialized.

9.8.2.5 int kbase_hwaccess_pm_powerup (struct kbase_device * kbdev, unsigned int flags)

kbase_hwaccess_pm_powerup - Power up the GPU. : The kbase device structure for the device (must be a valid pointer) : Flags to pass on to kbase_pm_init_hw

Power up GPU after all modules have been initialized and interrupt handlers installed.

Return: 0 if powerup was successful.

9.8.2.6 void kbase_hwaccess_pm_resume (struct kbase_device * kbdev)

Perform any backend-specific actions to resume the GPU from a suspend

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.8.2.7 void kbase_hwaccess_pm_suspend (struct kbase_device * *kbdev*)

Perform any backend-specific actions to suspend the GPU

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.8.2.8 void kbase_hwaccess_pm_term (struct kbase_device * *kbdev*)

Terminate the power management framework.

No power management functions may be called after this (except kbase_pm_init)

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.8.2.9 const struct kbase_pm_ca_policy* kbase_pm_ca_get_policy (struct kbase_device * *kbdev*)

Get the current policy.

Returns the policy that is currently active.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

Returns

The current policy

9.8.2.10 int kbase_pm_ca_list_policies (const struct kbase_pm_ca_policy *const ** *policies*)

Retrieve a static list of the available policies.

Parameters

out	*policies*	An array pointer to take the list of policies. This may be NULL. The contents of this array must not be modified.

Returns

The number of policies

9.8.2.11 void kbase_pm_ca_set_policy (struct kbase_device * *kbdev*, const struct kbase_pm_ca_policy * *policy*)

Change the policy to the one specified.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)
policy	The policy to change to (valid pointer returned from kbase_pm_ca_list_policies)

9.8.2.12 const struct kbase_pm_policy* kbase_pm_get_policy (struct kbase_device * *kbdev*)

Get the current policy.

Returns the policy that is currently active.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

Returns

The current policy

9.8.2.13 int kbase_pm_list_policies (const struct kbase_pm_policy *const ** *policies*)

Retrieve a static list of the available policies.

Parameters

out	*policies*	An array pointer to take the list of policies. This may be NULL. The contents of this array must not be modified.

Returns

The number of policies

9.8.2.14 void kbase_pm_set_debug_core_mask (struct kbase_device * *kbdev*, u64 *new_core_mask_js0*, u64 *new_core_mask_js1*, u64 *new_core_mask_js2*)

Set the debug core mask.

This determines which cores the power manager is allowed to use.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)
new_core_mask_js0	The core mask to use for job slot 0
new_core_mask_js0	The core mask to use for job slot 1
new_core_mask_js0	The core mask to use for job slot 2

9.8.2.15 void kbase_pm_set_policy (struct kbase_device * *kbdev*, const struct kbase_pm_policy * *policy*)

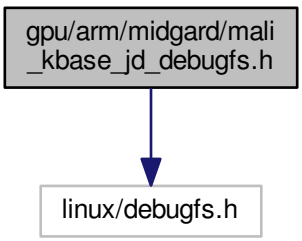
Change the policy to the one specified.

Parameters

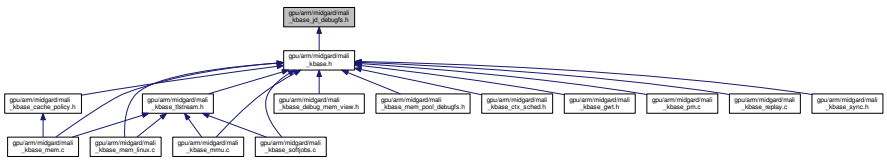
kbdev	The kbase device structure for the device (must be a valid pointer)
policy	The policy to change to (valid pointer returned from kbase_pm_list_policies)

9.9 gpu/arm/midgard/mali_kbase_jd_debugfs.h File Reference

`#include <linux/debugfs.h>`
Include dependency graph for mali_kbase_jd_debugfs.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **MALI_JD_DEBUGFS_VERSION** 2

Functions

- void kbasep_jd_debugfs_ctx_init (struct kbase_context *kctx)

9.9.1 Detailed Description

Header file for job dispatcher-related entries in debugfs

9.9.2 Function Documentation

9.9.2.1 void kbasep_jd_debugfs_ctx_init (struct kbase_context * *kctx*)

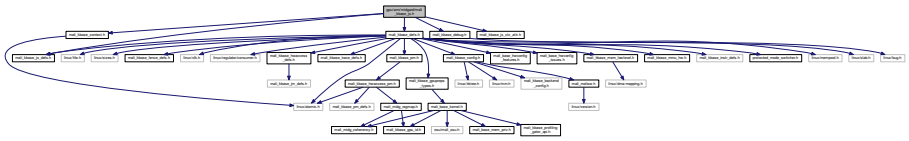
kbasep_jd_debugfs_ctx_init() - Add debugfs entries for JD system

Pointer to kbase_context

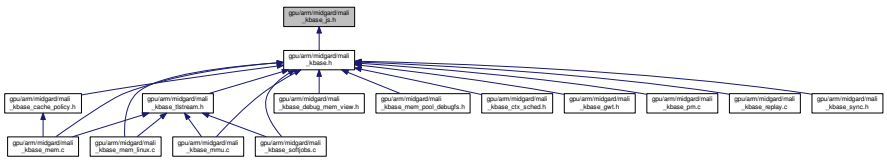
9.10 gpu/arm/midgard/mali_kbase_js.h File Reference

```
#include "mali_kbase_js_defs.h"
#include "mali_kbase_context.h"
#include "mali_kbase_defs.h"
#include "mali_kbase_debug.h"
#include "mali_kbase_js_ctx_attr.h"
```

Include dependency graph for mali_kbase_js.h:



This graph shows which files directly or indirectly include this file:



Functions

- int kbasep_js_devdata_init (struct kbase_device *const kbdev)

  Initialize the Job Scheduler.
- void kbasep_js_devdata_halt (struct kbase_device *kbdev)

  Halt the Job Scheduler.
- void kbasep_js_devdata_term (struct kbase_device *kbdev)

  Terminate the Job Scheduler.
- int kbasep_js_kctx_init (struct kbase_context *const kctx)

  Initialize the Scheduling Component of a struct kbase_context on the Job Scheduler.
- void kbasep_js_kctx_term (struct kbase_context *kctx)

  Terminate the Scheduling Component of a struct kbase_context on the Job Scheduler.
- bool kbasep_js_add_job (struct kbase_context *kctx, struct kbase_jd_atom *atom)

  Add a job chain to the Job Scheduler, and take necessary actions to schedule the context/run the job.
- void kbasep_js_remove_job (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbase_jd_atom *atom)

  Remove a job chain from the Job Scheduler, except for its 'retained state'.
- bool kbasep_js_remove_cancelled_job (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbase_jd_atom *katom)

  Completely remove a job chain from the Job Scheduler, in the case where the job chain was cancelled.
- bool kbasep_js_runpool_retain_ctx (struct kbase_device *kbdev, struct kbase_context *kctx)

  Refcount a context as being busy, preventing it from being scheduled out.
- bool kbasep_js_runpool_retain_ctx_nolock (struct kbase_device *kbdev, struct kbase_context *kctx)

  Refcount a context as being busy, preventing it from being scheduled out.
- struct kbase_context * kbasep_js_runpool_lookup_ctx (struct kbase_device *kbdev, int as_nr)

  Lookup a context in the Run Pool based upon its current address space and ensure that is stays scheduled in.
- void kbasep_js_runpool_requeue_or_kill_ctx (struct kbase_device *kbdev, struct kbase_context *kctx, bool has_pm_ref)

  Handling the requeuing/killing of a context that was evicted from the policy queue or runpool.
- void kbasep_js_runpool_release_ctx (struct kbase_device *kbdev, struct kbase_context *kctx)

  Release a refcount of a context being busy, allowing it to be scheduled out.
- void kbasep_js_runpool_release_ctx_and_katom_retained_state (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbasep_js_atom_retained_state *katom_retained_state)

  Variant of kbasep_js_runpool_release_ctx() that handles additional actions from completing an atom.
- void kbasep_js_runpool_release_ctx_nolock (struct kbase_device *kbdev, struct kbase_context *kctx)

  Variant of kbase_js_runpool_release_ctx() that assumes that kbasep_js_device_data::runpool_mutex and kbasep_js_kctx_info::ctx::jsctx_mutex are held by the caller, and does not attempt to schedule new contexts.
- void kbasep_js_schedule_privileged_ctx (struct kbase_device *kbdev, struct kbase_context *kctx)

  Schedule in a privileged context.
- void kbasep_js_release_privileged_ctx (struct kbase_device *kbdev, struct kbase_context *kctx)

  Release a privileged context, allowing it to be scheduled out.
- void kbase_js_try_run_jobs (struct kbase_device *kbdev)

  Try to submit the next job on each slot.
- void kbasep_js_suspend (struct kbase_device *kbdev)

  Suspend the job scheduler during a Power Management Suspend event.
- void kbasep_js_resume (struct kbase_device *kbdev)

  Resume the Job Scheduler after a Power Management Resume event.
- bool kbase_js_dep_resolved_submit (struct kbase_context *kctx, struct kbase_jd_atom *katom)

  Submit an atom to the job scheduler.
- void jsctx_ll_flush_to_rb (struct kbase_context *kctx, int prio, int js)
- struct kbase_jd_atom * kbase_js_pull (struct kbase_context *kctx, int js)

  Pull an atom from a context in the job scheduler for execution.

- void kbase_js_unpull (struct kbase_context *kctx, struct kbase_jd_atom *katom)

  Return an atom to the job scheduler ringbuffer.
- bool kbase_js_complete_atom_wq (struct kbase_context *kctx, struct kbase_jd_atom *katom)

  Complete an atom from jd_done_worker(), removing it from the job scheduler ringbuffer.
- struct kbase_jd_atom * kbase_js_complete_atom (struct kbase_jd_atom *katom, ktime_t *end_timestamp)

  Complete an atom.
- void kbase_js_sched (struct kbase_device *kbdev, int js_mask)

  Submit atoms from all available contexts.
- void kbase_js_zap_context (struct kbase_context *kctx)
- bool kbase_js_is_atom_valid (struct kbase_device *kbdev, struct kbase_jd_atom *katom)

  Validate an atom.
- void kbase_js_set_timeouts (struct kbase_device *kbdev)

Variables

- const int **kbasep_js_atom_priority_to_relative** [BASE_JD_NR_PRIO_LEVELS]
- const base_jd_prio **kbasep_js_relative_priority_to_atom** [KBASE_JS_ATOM_SCHED_PRIO_COUNT]

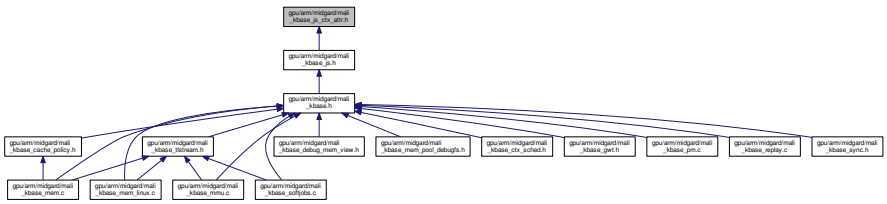
9.10.1 Detailed Description

Job Scheduler APIs.

Job Scheduler Type Definitions

9.11 gpu/arm/midgard/mali_kbase_js_ctx_attr.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void kbasep_js_ctx_attr_set_initial_attrs (struct kbase_device *kbdev, struct kbase_context *kctx)
- void kbasep_js_ctx_attr_runpool_retain_ctx (struct kbase_device *kbdev, struct kbase_context *kctx)
- bool kbasep_js_ctx_attr_runpool_release_ctx (struct kbase_device *kbdev, struct kbase_context *kctx)
- void kbasep_js_ctx_attr_ctx_retain_atom (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbase_jd_atom *katom)
- bool kbasep_js_ctx_attr_ctx_release_atom (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbasep_js_atom_retained_state *katom_retained_state)

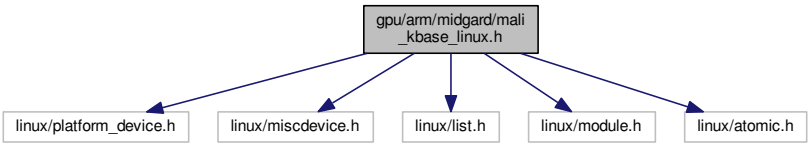
9.11.1 Detailed Description

Job Scheduler Context Attribute APIs

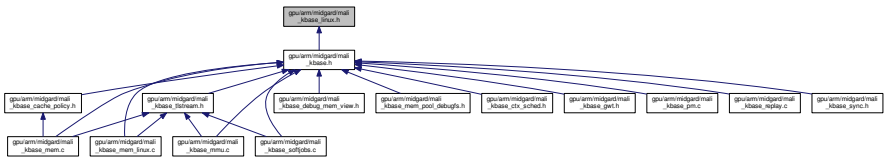
9.12 gpu/arm/midgard/mali_kbase_linux.h File Reference

```
#include <linux/platform_device.h>
#include <linux/miscdevice.h>
#include <linux/list.h>
#include <linux/module.h>
#include <linux/atomic.h>
```

Include dependency graph for mali_kbase_linux.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **KBASE_EXPORT_TEST_API**(func)
- #define **KBASE_EXPORT_SYMBOL**(func) EXPORT_SYMBOL(func)

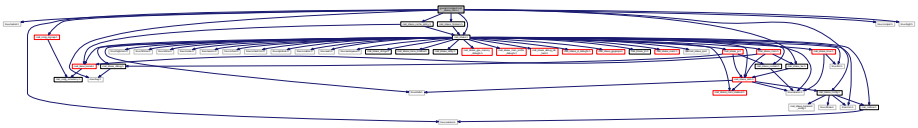
9.12.1 Detailed Description

Base kernel APIs, Linux implementation.

9.13 gpu/arm/midgard/mali_kbase_mem.c File Reference

```
#include <linux/kernel.h>
#include <linux/bug.h>
#include <linux/compat.h>
#include <linux/version.h>
#include <linux/log2.h>
#include <mali_kbase_config.h>
#include <mali_kbase.h>
#include <mali_midg_regmap.h>
#include <mali_kbase_cache_policy.h>
#include <mali_kbase_hw.h>
#include <mali_kbase_tlstream.h>
```

Include dependency graph for mali_kbase_mem.c:



Macros

- #define **KBASE_MSG_PRE** "GPU allocation attempted with "
- #define **KBASE_MSG_PRE_FLAG** KBASE_MSG_PRE "BASE_MEM_TILER_ALIGN_TOP and "

Functions

- struct kbase_va_region * **kbase_region_tracker_find_region_enclosing_address** (struct kbase_context *kctx, u64 gpu_addr)
- **KBASE_EXPORT_TEST_API** (kbase_region_tracker_find_region_enclosing_address)
- struct kbase_va_region * kbase_region_tracker_find_region_base_address (struct kbase_context *kctx, u64 gpu_addr)

  Check that a pointer is actually a valid region.
- **KBASE_EXPORT_TEST_API** (kbase_region_tracker_find_region_base_address)
- **KBASE_EXPORT_TEST_API** (kbase_remove_va_region)
- int kbase_add_va_region (struct kbase_context *kctx, struct kbase_va_region *reg, u64 addr, size_t nr_↩ pages, size_t align)

  Add a VA region to the list.
- **KBASE_EXPORT_TEST_API** (kbase_add_va_region)
- void **kbase_region_tracker_term** (struct kbase_context *kctx)
- int kbase_region_tracker_init (struct kbase_context *kctx)
- int **kbase_region_tracker_init_jit** (struct kbase_context *kctx, u64 jit_va_pages)
- int **kbase_mem_init** (struct kbase_device *kbdev)
- void **kbase_mem_halt** (struct kbase_device *kbdev)
- void **kbase_mem_term** (struct kbase_device *kbdev)
- **KBASE_EXPORT_TEST_API** (kbase_mem_term)
- struct kbase_va_region * kbase_alloc_free_region (struct kbase_context *kctx, u64 start_pfn, size_t nr_↩ pages, int zone)

  Allocate a free region object.
- **KBASE_EXPORT_TEST_API** (kbase_alloc_free_region)
- void kbase_free_alloced_region (struct kbase_va_region *reg)

Free a region object.

- **KBASE_EXPORT_TEST_API** (kbase_free_alloced_region)
- int kbase_gpu_mmap (struct kbase_context *kctx, struct kbase_va_region *reg, u64 addr, size_t nr_pages, size_t align)

  Register region and map it on the GPU.
- **KBASE_EXPORT_TEST_API** (kbase_gpu_mmap)
- int kbase_gpu_munmap (struct kbase_context *kctx, struct kbase_va_region *reg)

  Remove the region from the GPU and unregister it.
- int kbasep_find_enclosing_cpu_mapping_offset (struct kbase_context *kctx, unsigned long uaddr, size_t size, u64 *offset)
- **KBASE_EXPORT_TEST_API** (kbasep_find_enclosing_cpu_mapping_offset)
- int kbasep_find_enclosing_gpu_mapping_start_and_offset (struct kbase_context *kctx, u64 gpu_addr, size_t size, u64 *start, u64 *offset)
- **KBASE_EXPORT_TEST_API** (kbasep_find_enclosing_gpu_mapping_start_and_offset)
- void **kbase_sync_single** (struct kbase_context *kctx, struct tagged_addr t_cpu_pa, struct tagged_addr t_gpu_pa, off_t offset, size_t size, enum kbase_sync_type sync_fn)
- int kbase_sync_now (struct kbase_context *kctx, struct basep_syncset *sset)
- **KBASE_EXPORT_TEST_API** (kbase_sync_now)
- int **kbase_mem_free_region** (struct kbase_context *kctx, struct kbase_va_region *reg)
- **KBASE_EXPORT_TEST_API** (kbase_mem_free_region)
- int kbase_mem_free (struct kbase_context *kctx, u64 gpu_addr)

  Free the region from the GPU and unregister it.
- **KBASE_EXPORT_TEST_API** (kbase_mem_free)
- int kbase_update_region_flags (struct kbase_context *kctx, struct kbase_va_region *reg, unsigned long flags)
- int kbase_alloc_phy_pages_helper (struct kbase_mem_phy_alloc *alloc, size_t nr_pages_requested)

  Allocates physical pages.
- int kbase_free_phy_pages_helper (struct kbase_mem_phy_alloc *alloc, size_t nr_pages_to_free)

  Free physical pages.
- void **kbase_mem_kref_free** (struct kref *kref)
- **KBASE_EXPORT_TEST_API** (kbase_mem_kref_free)
- int **kbase_alloc_phy_pages** (struct kbase_va_region *reg, size_t vsize, size_t size)
- **KBASE_EXPORT_TEST_API** (kbase_alloc_phy_pages)
- bool **kbase_check_alloc_flags** (unsigned long flags)
- bool **kbase_check_import_flags** (unsigned long flags)
- int kbase_check_alloc_sizes (struct kbase_context *kctx, unsigned long flags, u64 va_pages, u64 commit_pages, u64 large_extent)
- void kbase_gpu_vm_lock (struct kbase_context *kctx)

  Acquire the per-context region list lock.
- **KBASE_EXPORT_TEST_API** (kbase_gpu_vm_lock)
- void kbase_gpu_vm_unlock (struct kbase_context *kctx)

  Release the per-context region list lock.
- **KBASE_EXPORT_TEST_API** (kbase_gpu_vm_unlock)
- int kbase_jit_init (struct kbase_context *kctx)
- struct kbase_va_region * kbase_jit_allocate (struct kbase_context *kctx, struct base_jit_alloc_info *info)
- void kbase_jit_free (struct kbase_context *kctx, struct kbase_va_region *reg)
- void kbase_jit_backing_lost (struct kbase_va_region *reg)
- bool kbase_jit_evict (struct kbase_context *kctx)
- void kbase_jit_term (struct kbase_context *kctx)
- struct kbase_mem_phy_alloc * kbase_map_external_resource (struct kbase_context *kctx, struct kbase_va_region *reg, struct mm_struct *locked_mm)
- void kbase_unmap_external_resource (struct kbase_context *kctx, struct kbase_va_region *reg, struct kbase_mem_phy_alloc *alloc)

- struct kbase_ctx_ext_res_meta * kbase_sticky_resource_acquire (struct kbase_context *kctx, u64 gpu_addr)
- bool kbase_sticky_resource_release (struct kbase_context *kctx, struct kbase_ctx_ext_res_meta *meta, u64 gpu_addr)
- int kbase_sticky_resource_init (struct kbase_context *kctx)
- void kbase_sticky_resource_term (struct kbase_context *kctx)

9.13.1 Detailed Description

Base kernel memory APIs

9.13.2 Function Documentation

9.13.2.1 struct kbase_va_region* kbase_alloc_free_region (struct kbase_context * *kctx*, u64 *start_pfn*, size_t *nr_pages*, int *zone*)

Allocate a free region object.

The allocated object is not part of any list yet, and is flagged as KBASE_REG_FREE. No mapping is allocated yet.

zone is KBASE_REG_ZONE_CUSTOM_VA, KBASE_REG_ZONE_SAME_VA, or KBASE_REG_ZONE_EXEC

9.13.2.2 int kbase_alloc_phy_pages_helper (struct kbase_mem_phy_alloc * *alloc*, size_t *nr_pages_requested*)

Allocates physical pages.

Allocates *nr_pages_requested* and updates the alloc object.

Parameters

in	*alloc*	allocation object to add pages to
in	*nr_pages_requested*	number of physical pages to allocate

Returns

0 if all pages have been successfully allocated. Error code otherwise

9.13.2.3 int kbase_check_alloc_sizes (struct kbase_context * *kctx*, unsigned long *flags*, u64 *va_pages*, u64 *commit_pages*, u64 *extent*)

kbase_check_alloc_sizes - check user space sizes parameters for an allocation

: kbase context : The flags passed from user space : The size of the requested region, in pages. : Number of pages to commit initially. : Number of pages to grow by on GPU page fault and/or alignment (depending on flags)

Makes checks on the size parameters passed in from user space for a memory allocation call, with respect to the flags requested.

Return: 0 if sizes are valid for these flags, negative error code otherwise

9.13.2.4 void kbase_free_alloced_region (struct kbase_va_region * *reg*)

Free a region object.

The described region must be freed of any mapping.

If the region is not flagged as KBASE_REG_FREE, the region's alloc object will be released. It is a bug if no alloc object exists for non-free regions.

9.13.2.5 int kbase_free_phy_pages_helper (struct kbase_mem_phy_alloc * *alloc*, size_t *nr_pages_to_free*)

Free physical pages.

Frees *nr_pages* and updates the alloc object.

Parameters

in	*alloc*	allocation object to free pages from
in	*nr_pages_to_free*	number of physical pages to free

9.13.2.6 int kbase_gpu_mmap (struct kbase_context * *kctx*, struct kbase_va_region * *reg*, u64 *addr*, size_t *nr_pages*, size_t *align*)

Register region and map it on the GPU.

Call kbase_add_va_region() and map the region on the GPU.

9.13.2.7 int kbase_gpu_munmap (struct kbase_context * *kctx*, struct kbase_va_region * *reg*)

Remove the region from the GPU and unregister it.

Must be called with context lock held.

9.13.2.8 struct kbase_va_region* kbase_jit_allocate (struct kbase_context * *kctx*, struct base_jit_alloc_info * *info*)

kbase_jit_allocate - Allocate JIT memory : kbase context : JIT allocation information

Return: JIT allocation on success or NULL on failure.

9.13.2.9 void kbase_jit_backing_lost (struct kbase_va_region * *reg*)

kbase_jit_backing_lost - Inform JIT that an allocation has lost backing : JIT allocation

9.13.2.10 bool kbase_jit_evict (struct kbase_context * *kctx*)

kbase_jit_evict - Evict a JIT allocation from the pool : kbase context

Evict the least recently used JIT allocation from the pool. This can be required if normal VA allocations are failing due to VA exhaustion.

Return: True if a JIT allocation was freed, false otherwise.

9.13.2.11 void kbase_jit_free (struct kbase_context * *kctx,* struct kbase_va_region * *reg*)

kbase_jit_free - Free a JIT allocation : kbase context : JIT allocation

Frees a JIT allocation and places it into the free pool for later reuse.

9.13.2.12 int kbase_jit_init (struct kbase_context * *kctx*)

kbase_jit_init - Initialize the JIT memory pool management : kbase context

Returns zero on success or negative error number on failure.

9.13.2.13 void kbase_jit_term (struct kbase_context * *kctx*)

kbase_jit_term - Terminate the JIT memory pool management : kbase context

9.13.2.14 struct kbase_mem_phy_alloc* kbase_map_external_resource (struct kbase_context * *kctx,* struct kbase_va_region * *reg,* struct mm_struct * *locked_mm*)

kbase_map_external_resource - Map an external resource to the GPU. : kbase context. : The region to map. : The mm_struct which has been locked for this operation.

Return: The physical allocation which backs the region on success or NULL on failure.

9.13.2.15 int kbase_mem_free (struct kbase_context * *kctx,* u64 *gpu_addr*)

Free the region from the GPU and unregister it.

This function implements the free operation on a memory segment. It will loudly fail if called with outstanding mappings.

9.13.2.16 struct kbase_va_region* kbase_region_tracker_find_region_base_address (struct kbase_context * *kctx,* u64 *gpu_addr*)

Check that a pointer is actually a valid region.

Must be called with context lock held.

9.13.2.17 int kbase_region_tracker_init (struct kbase_context * *kctx*)

Initialize the region tracker data structure.

9.13.2.18 struct kbase_ctx_ext_res_meta* kbase_sticky_resource_acquire (struct kbase_context * *kctx,* u64 *gpu_addr*)

kbase_sticky_resource_acquire - Acquire a reference on a sticky resource. : kbase context. : The GPU address of the external resource.

Return: The metadata object which represents the binding between the external resource and the kbase context on success or NULL on failure.

9.13.2.19 int kbase_sticky_resource_init (struct kbase_context * *kctx*)

kbase_sticky_resource_init - Initialize sticky resource management. : kbase context

Returns zero on success or negative error number on failure.

9.13.2.20 bool kbase_sticky_resource_release (struct kbase_context * *kctx,* struct kbase_ctx_ext_res_meta * *meta,* u64 *gpu_addr*)

kbase_sticky_resource_release - Release a reference on a sticky resource. : kbase context. : Binding metadata. : GPU address of the external resource.

If meta is NULL then gpu_addr will be used to scan the metadata list and find the matching metadata (if any), otherwise the provided meta will be used and gpu_addr will be ignored.

Return: True if the release found the metadata and the reference was dropped.

9.13.2.21 void kbase_sticky_resource_term (struct kbase_context * *kctx*)

kbase_sticky_resource_term - Terminate sticky resource management. : kbase context

9.13.2.22 int kbase_sync_now (struct kbase_context * *kctx,* struct basep_syncset * *sset*)

kbase_sync_now - Perform cache maintenance on a memory region

: The kbase context of the region : A syncset structure describing the region and direction of the synchronisation required

Return: 0 on success or error code

9.13.2.23 void kbase_unmap_external_resource (struct kbase_context * *kctx,* struct kbase_va_region * *reg,* struct kbase_mem_phy_alloc * *alloc*)

kbase_unmap_external_resource - Unmap an external resource from the GPU. : kbase context. : The region to unmap or NULL if it has already been released. : The physical allocation being unmapped.

9.13.2.24 int kbase_update_region_flags (struct kbase_context * *kctx*, struct kbase_va_region * *reg*, unsigned long *flags*)

kbase_update_region_flags - Convert user space flags to kernel region flags

: kbase context : The region to update the flags on : The flags passed from user space

The user space flag BASE_MEM_COHERENT_SYSTEM_REQUIRED will be rejected and this function will fail if the system does not support system coherency.

Return: 0 if successful, -EINVAL if the flags are not supported

9.13.2.25 int kbasep_find_enclosing_cpu_mapping_offset (struct kbase_context * *kctx*, unsigned long *uaddr*, size_t *size*, u64 * *offset*)

kbasep_find_enclosing_cpu_mapping_offset() - Find the offset of the CPU mapping of a memory allocation containing a given address range

Searches for a CPU mapping of any part of any region that fully encloses the CPU virtual address range specified by and . Returns a failure indication if only part of the address range lies within a CPU mapping.

: The kernel base context used for the allocation. : Start of the CPU virtual address range. : Size of the CPU virtual address range (in bytes). : The offset from the start of the allocation to the specified CPU virtual address.

Return: 0 if offset was obtained successfully. Error code otherwise.

9.13.2.26 int kbasep_find_enclosing_gpu_mapping_start_and_offset (struct kbase_context * *kctx*, u64 *gpu_addr*, size_t *size*, u64 * *start*, u64 * *offset*)

kbasep_find_enclosing_gpu_mapping_start_and_offset() - Find the address of the start of GPU virtual memory region which encloses for the length in bytes

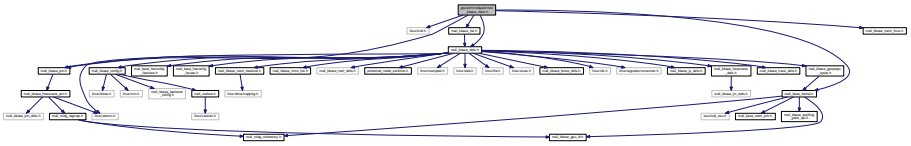
Searches for the memory region in GPU virtual memory space which contains the region defined by the and , where is the beginning and the length in bytes of the provided region. If found, the location of the start address of the GPU virtual memory region is passed in pointer and the location of the offset of the region into the GPU virtual memory region is passed in pointer.

: The kernel base context within which the memory is searched. : GPU virtual address for which the region is sought; defines the beginning of the provided region. : The length (in bytes) of the provided region for which the GPU virtual memory region is sought. : Pointer to the location where the address of the start of the found GPU virtual memory region is. : Pointer to the location where the offset of into the found GPU virtual memory region is.

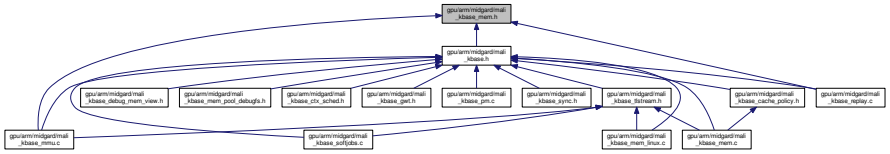
9.14 gpu/arm/midgard/mali_kbase_mem.h File Reference

```
#include <linux/kref.h>
#include "mali_base_kernel.h"
#include <mali_kbase_hw.h>
#include "mali_kbase_pm.h"
#include "mali_kbase_defs.h"
#include "mali_kbase_mem_linux.h"
```

Include dependency graph for mali_kbase_mem.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct kbase_cpu_mapping
- struct kbase_aliased
- struct kbase_mem_phy_alloc
- struct kbase_va_region

Macros

- #define **KBASEP_TMEM_GROWABLE_BLOCKSIZE_PAGES_LOG2_HW_ISSUE_8316** (2) /∗ round to 4 pages ∗/
- #define **KBASEP_TMEM_GROWABLE_BLOCKSIZE_PAGES_LOG2_HW_ISSUE_9630** (3) /∗ round to 8 pages ∗/
- #define **KBASEP_TMEM_GROWABLE_BLOCKSIZE_PAGES_LOG2** (0) /∗ round to 1 page ∗/
- #define **KBASEP_TMEM_GROWABLE_BLOCKSIZE_PAGES** (1u << KBASEP_TMEM_GROWABLE_↩ BLOCKSIZE_PAGES_LOG2)
- #define **KBASEP_TMEM_GROWABLE_BLOCKSIZE_PAGES_HW_ISSUE_8316** (1u << KBASEP_TM↩ EM_GROWABLE_BLOCKSIZE_PAGES_LOG2_HW_ISSUE_8316)
- #define **KBASEP_TMEM_GROWABLE_BLOCKSIZE_PAGES_HW_ISSUE_9630** (1u << KBASEP_TM↩ EM_GROWABLE_BLOCKSIZE_PAGES_LOG2_HW_ISSUE_9630)
- #define KBASE_MEM_PHY_ALLOC_ACCESSED_CACHED (1ul << 0)

  Physical pages tracking object properties.
- #define **KBASE_MEM_PHY_ALLOC_LARGE** (1ul << 1)
- #define **PINNED_ON_IMPORT** (1<<31)
- #define **KBASE_REG_FREE** (1ul << 0)

- #define **KBASE_REG_CPU_WR** (1ul << 1)
- #define **KBASE_REG_GPU_WR** (1ul << 2)
- #define **KBASE_REG_GPU_NX** (1ul << 3)
- #define **KBASE_REG_CPU_CACHED** (1ul << 4)
- #define **KBASE_REG_GPU_CACHED** (1ul << 5)
- #define **KBASE_REG_GROWABLE** (1ul << 6)
- #define **KBASE_REG_PF_GROW** (1ul << 7)
- #define **KBASE_REG_SHARE_IN** (1ul << 9)
- #define **KBASE_REG_SHARE_BOTH** (1ul << 10)
- #define **KBASE_REG_ZONE_MASK** (3ul << 11)
- #define **KBASE_REG_ZONE**(x) (((x) & 3) << 11)
- #define **KBASE_REG_GPU_RD** (1ul<<13)
- #define **KBASE_REG_CPU_RD** (1ul<<14)
- #define **KBASE_REG_MEMATTR_MASK** (7ul << 16)
- #define **KBASE_REG_MEMATTR_INDEX**(x) (((x) & 7) << 16)
- #define **KBASE_REG_MEMATTR_VALUE**(x) (((x) & KBASE_REG_MEMATTR_MASK) >> 16)
- #define **KBASE_REG_SECURE** (1ul << 19)
- #define **KBASE_REG_DONT_NEED** (1ul << 20)
- #define **KBASE_REG_IMPORT_PAD** (1ul << 21)
- #define **KBASE_REG_RESERVED_BIT_22** (1ul << 22)
- #define **KBASE_REG_TILER_ALIGN_TOP** (1ul << 23)
- #define **KBASE_REG_JIT** (1ul << 24)
- #define **KBASE_REG_ZONE_SAME_VA** KBASE_REG_ZONE(0)
- #define **KBASE_REG_ZONE_EXEC** KBASE_REG_ZONE(1)
- #define **KBASE_REG_ZONE_EXEC_BASE** (0x101000000ULL >> PAGE_SHIFT)
- #define **KBASE_REG_ZONE_EXEC_SIZE** ((16ULL * 1024 * 1024) >> PAGE_SHIFT)
- #define **KBASE_REG_ZONE_CUSTOM_VA** KBASE_REG_ZONE(2)
- #define **KBASE_REG_ZONE_CUSTOM_VA_BASE** (KBASE_REG_ZONE_EXEC_BASE + KBASE_REG↩ _ZONE_EXEC_SIZE) /* Starting after KBASE_REG_ZONE_EXEC */
- #define **KBASE_REG_ZONE_CUSTOM_VA_SIZE** (((1ULL << 44) >> PAGE_SHIFT) - KBASE_REG_↩ ZONE_CUSTOM_VA_BASE)
- #define **KBASE_MEM_PHY_ALLOC_LARGE_THRESHOLD** ((size_t)(4*1024)) /* size above which vmalloc is used over kmalloc */
- #define **KBASE_MEM_POOL_MAX_SIZE_KBDEV** (SZ_64M >> PAGE_SHIFT)
- #define **KBASE_MEM_POOL_MAX_SIZE_KCTX** (SZ_64M >> PAGE_SHIFT)
- #define **KBASE_MEM_POOL_2MB_PAGE_TABLE_ORDER** 9
- #define **KBASE_MEM_POOL_4KB_PAGE_TABLE_ORDER** 0

Enumerations

- enum **kbase_memory_type** { **KBASE_MEM_TYPE_NATIVE**, **KBASE_MEM_TYPE_IMPORTED_UMP**, **KBASE_MEM_TYPE_IMPOR↩ TED_UMM**, **KBASE_MEM_TYPE_IMPORTED_USER_BUF**, **KBASE_MEM_TYPE_ALIAS**, **KBASE_MEM_TYPE_TB**, **KBASE_MEM_TYPE_RAW** }

Functions

- void **kbase_mem_kref_free** (struct kref *kref)
- int **kbase_mem_init** (struct kbase_device *kbdev)
- void **kbase_mem_halt** (struct kbase_device *kbdev)
- void **kbase_mem_term** (struct kbase_device *kbdev)
- int kbase_mem_pool_init (struct kbase_mem_pool *pool, size_t max_size, size_t order, struct kbase_device *kbdev, struct kbase_mem_pool *next_pool)

- void kbase_mem_pool_term (struct kbase_mem_pool *pool)
- struct page * kbase_mem_pool_alloc (struct kbase_mem_pool *pool)
- void **kbase_mem_pool_free** (struct kbase_mem_pool *pool, struct page *page, bool dirty)
- int kbase_mem_pool_alloc_pages (struct kbase_mem_pool *pool, size_t nr_pages, struct tagged_addr *pages, bool partial_allowed)
- void kbase_mem_pool_free_pages (struct kbase_mem_pool *pool, size_t nr_pages, struct tagged_addr *pages, bool dirty, bool reclaimed)
- void kbase_mem_pool_set_max_size (struct kbase_mem_pool *pool, size_t max_size)
- int kbase_mem_pool_grow (struct kbase_mem_pool *pool, size_t nr_to_grow)
- void kbase_mem_pool_trim (struct kbase_mem_pool *pool, size_t new_size)
- struct page * kbase_mem_alloc_page (struct kbase_mem_pool *pool)
- int kbase_region_tracker_init (struct kbase_context *kctx)
- int **kbase_region_tracker_init_jit** (struct kbase_context *kctx, u64 jit_va_pages)
- void **kbase_region_tracker_term** (struct kbase_context *kctx)
- struct kbase_va_region * **kbase_region_tracker_find_region_enclosing_address** (struct kbase_context *kctx, u64 gpu_addr)
- struct kbase_va_region * kbase_region_tracker_find_region_base_address (struct kbase_context *kctx, u64 gpu_addr)

  Check that a pointer is actually a valid region.
- struct kbase_va_region * kbase_alloc_free_region (struct kbase_context *kctx, u64 start_pfn, size_t nr_↩ pages, int zone)

  Allocate a free region object.
- void kbase_free_alloced_region (struct kbase_va_region *reg)

  Free a region object.
- int kbase_add_va_region (struct kbase_context *kctx, struct kbase_va_region *reg, u64 addr, size_t nr_↩ pages, size_t align)

  Add a VA region to the list.
- bool **kbase_check_alloc_flags** (unsigned long flags)
- bool **kbase_check_import_flags** (unsigned long flags)
- int kbase_check_alloc_sizes (struct kbase_context *kctx, unsigned long flags, u64 va_pages, u64 commit↩ _pages, u64 extent)
- int kbase_update_region_flags (struct kbase_context *kctx, struct kbase_va_region *reg, unsigned long flags)
- void kbase_gpu_vm_lock (struct kbase_context *kctx)

  Acquire the per-context region list lock.
- void kbase_gpu_vm_unlock (struct kbase_context *kctx)

  Release the per-context region list lock.
- int **kbase_alloc_phy_pages** (struct kbase_va_region *reg, size_t vsize, size_t size)
- int **kbase_mmu_init** (struct kbase_context *kctx)
- void **kbase_mmu_term** (struct kbase_context *kctx)
- phys_addr_t **kbase_mmu_alloc_pgd** (struct kbase_context *kctx)
- void **kbase_mmu_free_pgd** (struct kbase_context *kctx)
- int **kbase_mmu_insert_pages_no_flush** (struct kbase_context *kctx, u64 vpfn, struct tagged_addr *phys, size_t nr, unsigned long flags)
- int **kbase_mmu_insert_pages** (struct kbase_context *kctx, u64 vpfn, struct tagged_addr *phys, size_t nr, unsigned long flags)
- int **kbase_mmu_insert_single_page** (struct kbase_context *kctx, u64 vpfn, struct tagged_addr phys, size↩ _t nr, unsigned long flags)
- int **kbase_mmu_teardown_pages** (struct kbase_context *kctx, u64 vpfn, size_t nr)
- int kbase_mmu_update_pages_no_flush (struct kbase_context *kctx, u64 vpfn, struct tagged_addr *phys, size_t nr, unsigned long flags)
- int **kbase_mmu_update_pages** (struct kbase_context *kctx, u64 vpfn, struct tagged_addr *phys, size_t nr, unsigned long flags)
- int kbase_gpu_mmap (struct kbase_context *kctx, struct kbase_va_region *reg, u64 addr, size_t nr_pages, size_t align)

Register region and map it on the GPU.

- int kbase_gpu_munmap (struct kbase_context *kctx, struct kbase_va_region *reg)

  Remove the region from the GPU and unregister it.
- void kbase_mmu_update (struct kbase_context *kctx)
- void kbase_mmu_disable (struct kbase_context *kctx)
- void kbase_mmu_disable_as (struct kbase_device *kbdev, int as_nr)
- void **kbase_mmu_interrupt** (struct kbase_device *kbdev, u32 irq_stat)
- void * kbase_mmu_dump (struct kbase_context *kctx, int nr_pages)
- int kbase_sync_now (struct kbase_context *kctx, struct basep_syncset *sset)
- void **kbase_sync_single** (struct kbase_context *kctx, struct tagged_addr cpu_pa, struct tagged_addr gpu_pa, off_t offset, size_t size, enum kbase_sync_type sync_fn)
- void **kbase_pre_job_sync** (struct kbase_context *kctx, struct base_syncset *syncsets, size_t nr)
- void **kbase_post_job_sync** (struct kbase_context *kctx, struct base_syncset *syncsets, size_t nr)
- int kbase_mem_free (struct kbase_context *kctx, u64 gpu_addr)

  Free the region from the GPU and unregister it.
- int **kbase_mem_free_region** (struct kbase_context *kctx, struct kbase_va_region *reg)
- void **kbase_os_mem_map_lock** (struct kbase_context *kctx)
- void **kbase_os_mem_map_unlock** (struct kbase_context *kctx)
- void kbasep_os_process_page_usage_update (struct kbase_context *kctx, int pages)

  Update the memory allocation counters for the current process.
- int kbasep_find_enclosing_cpu_mapping_offset (struct kbase_context *kctx, unsigned long uaddr, size_t size, u64 *offset)
- int kbasep_find_enclosing_gpu_mapping_start_and_offset (struct kbase_context *kctx, u64 gpu_addr, size_t size, u64 *start, u64 *offset)
- enum hrtimer_restart **kbasep_as_poke_timer_callback** (struct hrtimer *timer)
- void kbase_as_poking_timer_retain_atom (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbase_jd_atom *katom)
- void kbase_as_poking_timer_release_atom (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbase_jd_atom *katom)
- int kbase_alloc_phy_pages_helper (struct kbase_mem_phy_alloc *alloc, size_t nr_pages_requested)

  Allocates physical pages.
- int kbase_free_phy_pages_helper (struct kbase_mem_phy_alloc *alloc, size_t nr_pages_to_free)

  Free physical pages.
- void kbase_mmu_interrupt_process (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbase_as *as)

  Process a bus or page fault.
- void page_fault_worker (struct work_struct *data)

  Process a page fault.
- void bus_fault_worker (struct work_struct *data)

  Process a bus fault.
- void kbase_flush_mmu_wqs (struct kbase_device *kbdev)

  Flush MMU workqueues.
- void **kbase_sync_single_for_device** (struct kbase_device *kbdev, dma_addr_t handle, size_t size, enum dma_data_direction dir)
- void **kbase_sync_single_for_cpu** (struct kbase_device *kbdev, dma_addr_t handle, size_t size, enum dma_data_direction dir)
- int kbase_jit_init (struct kbase_context *kctx)
- struct kbase_va_region * kbase_jit_allocate (struct kbase_context *kctx, struct base_jit_alloc_info *info)
- void kbase_jit_free (struct kbase_context *kctx, struct kbase_va_region *reg)
- void kbase_jit_backing_lost (struct kbase_va_region *reg)
- bool kbase_jit_evict (struct kbase_context *kctx)
- void kbase_jit_term (struct kbase_context *kctx)
- struct kbase_mem_phy_alloc * kbase_map_external_resource (struct kbase_context *kctx, struct kbase_va_region *reg, struct mm_struct *locked_mm)

- void kbase_unmap_external_resource (struct kbase_context *kctx, struct kbase_va_region *reg, struct kbase_mem_phy_alloc *alloc)
- int kbase_sticky_resource_init (struct kbase_context *kctx)
- struct kbase_ctx_ext_res_meta * kbase_sticky_resource_acquire (struct kbase_context *kctx, u64 gpu_↩ addr)
- bool kbase_sticky_resource_release (struct kbase_context *kctx, struct kbase_ctx_ext_res_meta *meta, u64 gpu_addr)
- void kbase_sticky_resource_term (struct kbase_context *kctx)

9.14.1 Detailed Description

Base kernel memory APIs

9.14.2 Function Documentation

9.14.2.1 void bus_fault_worker (struct work_struct * *data*)

Process a bus fault.

Parameters

in	*data*	work_struct passed by queue_work()

9.14.2.2 struct kbase_va_region* kbase_alloc_free_region (struct kbase_context * *kctx*, u64 *start_pfn*, size_t *nr_pages*, int *zone*)

Allocate a free region object.

The allocated object is not part of any list yet, and is flagged as KBASE_REG_FREE. No mapping is allocated yet.

zone is KBASE_REG_ZONE_CUSTOM_VA, KBASE_REG_ZONE_SAME_VA, or KBASE_REG_ZONE_EXEC

9.14.2.3 int kbase_alloc_phy_pages_helper (struct kbase_mem_phy_alloc * *alloc*, size_t *nr_pages_requested*)

Allocates physical pages.

Allocates *nr_pages_requested* and updates the alloc object.

Parameters

in	*alloc*	allocation object to add pages to
in	*nr_pages_requested*	number of physical pages to allocate

Returns

0 if all pages have been successfully allocated. Error code otherwise

9.14.2.4 void kbase_as_poking_timer_release_atom (struct kbase_device * *kbdev,* struct kbase_context * *kctx,* struct kbase_jd_atom * *katom*)

If an atom holds a poking timer, release it and wait for it to finish

This must only be called on a context that's scheduled in, and an atom that still has a JS reference on the context

This must **not** be called from atomic context, since it can sleep.

9.14.2.5 void kbase_as_poking_timer_retain_atom (struct kbase_device * *kbdev,* struct kbase_context * *kctx,* struct kbase_jd_atom * *katom*)

Retain the poking timer on an atom's context (if the atom hasn't already done so), and start the timer (if it's not already started).

This must only be called on a context that's scheduled in, and an atom that's running on the GPU.

The caller must hold hwaccess_lock

This can be called safely from atomic context

9.14.2.6 int kbase_check_alloc_sizes (struct kbase_context * *kctx,* unsigned long *flags,* u64 *va_pages,* u64 *commit_pages,* u64 *extent*)

kbase_check_alloc_sizes - check user space sizes parameters for an allocation

: kbase context : The flags passed from user space : The size of the requested region, in pages. : Number of pages to commit initially. : Number of pages to grow by on GPU page fault and/or alignment (depending on flags)

Makes checks on the size parameters passed in from user space for a memory allocation call, with respect to the flags requested.

Return: 0 if sizes are valid for these flags, negative error code otherwise

9.14.2.7 void kbase_flush_mmu_wqs (struct kbase_device * *kbdev*)

Flush MMU workqueues.

This function will cause any outstanding page or bus faults to be processed. It should be called prior to powering off the GPU.

Parameters

in	*kbdev*	Device pointer

9.14.2.8 void kbase_free_alloced_region (struct kbase_va_region * *reg*)

Free a region object.

The described region must be freed of any mapping.

If the region is not flagged as KBASE_REG_FREE, the region's alloc object will be released. It is a bug if no alloc object exists for non-free regions.

9.14.2.9 int kbase_free_phy_pages_helper (struct kbase_mem_phy_alloc * *alloc*, size_t *nr_pages_to_free*)

Free physical pages.

Frees *nr_pages* and updates the alloc object.

Parameters

in	*alloc*	allocation object to free pages from
in	*nr_pages_to_free*	number of physical pages to free

9.14.2.10 int kbase_gpu_mmap (struct kbase_context * *kctx*, struct kbase_va_region * *reg*, u64 *addr*, size_t *nr_pages*, size_t *align*)

Register region and map it on the GPU.

Call kbase_add_va_region() and map the region on the GPU.

9.14.2.11 int kbase_gpu_munmap (struct kbase_context * *kctx*, struct kbase_va_region * *reg*)

Remove the region from the GPU and unregister it.

Must be called with context lock held.

9.14.2.12 struct kbase_va_region* kbase_jit_allocate (struct kbase_context * *kctx*, struct base_jit_alloc_info * *info*)

kbase_jit_allocate - Allocate JIT memory : kbase context : JIT allocation information

Return: JIT allocation on success or NULL on failure.

9.14.2.13 void kbase_jit_backing_lost (struct kbase_va_region * *reg*)

kbase_jit_backing_lost - Inform JIT that an allocation has lost backing : JIT allocation

9.14.2.14 bool kbase_jit_evict (struct kbase_context * *kctx*)

kbase_jit_evict - Evict a JIT allocation from the pool : kbase context

Evict the least recently used JIT allocation from the pool. This can be required if normal VA allocations are failing due to VA exhaustion.

Return: True if a JIT allocation was freed, false otherwise.

9.14.2.15 void kbase_jit_free (struct kbase_context * kctx, struct kbase_va_region * reg)

kbase_jit_free - Free a JIT allocation : kbase context : JIT allocation

Frees a JIT allocation and places it into the free pool for later reuse.

9.14.2.16 int kbase_jit_init (struct kbase_context * kctx)

kbase_jit_init - Initialize the JIT memory pool management : kbase context

Returns zero on success or negative error number on failure.

9.14.2.17 void kbase_jit_term (struct kbase_context * kctx)

kbase_jit_term - Terminate the JIT memory pool management : kbase context

9.14.2.18 struct kbase_mem_phy_alloc* kbase_map_external_resource (struct kbase_context * kctx, struct kbase_va_region * reg, struct mm_struct * locked_mm)

kbase_map_external_resource - Map an external resource to the GPU. : kbase context. : The region to map. : The mm_struct which has been locked for this operation.

Return: The physical allocation which backs the region on success or NULL on failure.

9.14.2.19 struct page* kbase_mem_alloc_page (struct kbase_mem_pool * pool)

kbase_mem_alloc_page - Allocate a new page for a device : Memory pool to allocate a page from

Most uses should use kbase_mem_pool_alloc to allocate a page. However that function can fail in the event the pool is empty.

Return: A new page or NULL if no memory

9.14.2.20 int kbase_mem_free (struct kbase_context * kctx, u64 gpu_addr)

Free the region from the GPU and unregister it.

This function implements the free operation on a memory segment. It will loudly fail if called with outstanding mappings.

9.14.2.21 struct page* kbase_mem_pool_alloc (struct kbase_mem_pool * pool)

kbase_mem_pool_alloc - Allocate a page from memory pool : Memory pool to allocate from

Allocations from the pool are made as follows:

1. If there are free pages in the pool, allocate a page from .
2. Otherwise, if is not NULL and has free pages, allocate a page from .
3. Return NULL if no memory in the pool

Return: Pointer to allocated page, or NULL if allocation failed.

9.14.2.22 int kbase_mem_pool_alloc_pages (struct kbase_mem_pool * *pool*, size_t *nr_pages*, struct tagged_addr * *pages*, bool *partial_allowed*)

kbase_mem_pool_alloc_pages - Allocate pages from memory pool : Memory pool to allocate from : Number of pages to allocate : Pointer to array where the physical address of the allocated pages will be stored. : If fewer pages allocated is allowed

Like kbase_mem_pool_alloc() but optimized for allocating many pages.

Return: On success number of pages allocated (could be less than nr_pages if partial_allowed). On error an error code.

9.14.2.23 void kbase_mem_pool_free_pages (struct kbase_mem_pool * *pool*, size_t *nr_pages*, struct tagged_addr * *pages*, bool *dirty*, bool *reclaimed*)

kbase_mem_pool_free_pages - Free pages to memory pool : Memory pool where pages should be freed : Number of pages to free : Pointer to array holding the physical addresses of the pages to free. : Whether any pages may be dirty in the cache. : Whether the pages where reclaimable and thus should bypass the pool and go straight to the kernel.

Like kbase_mem_pool_free() but optimized for freeing many pages.

9.14.2.24 int kbase_mem_pool_grow (struct kbase_mem_pool * *pool*, size_t *nr_to_grow*)

kbase_mem_pool_grow - Grow the pool : Memory pool to grow : Number of pages to add to the pool

Adds pages to the pool. Note that this may cause the pool to become larger than the maximum size specified.

Returns: 0 on success, -ENOMEM if unable to allocate sufficent pages

9.14.2.25 int kbase_mem_pool_init (struct kbase_mem_pool * *pool*, size_t *max_size*, size_t *order*, struct kbase_device * *kbdev*, struct kbase_mem_pool * *next_pool*)

kbase_mem_pool_init - Create a memory pool for a kbase device : Memory pool to initialize : Maximum number of free pages the pool can hold : Page order for physical page size (order=0=>4kB, order=9=>2MB) : Kbase device where memory is used : Pointer to the next pool or NULL.

Allocations from are in whole pages. Each has a free list where pages can be quickly allocated from. The free list is initially empty and filled whenever pages are freed back to the pool. The number of free pages in the pool will in general not exceed , but the pool may in certain corner cases grow above .

If is not NULL, we will allocate from before going to the kernel allocator. Similarily pages can spill over to when is full. Pages are zeroed before they spill over to another pool, to prevent leaking information between applications.

A shrinker is registered so that Linux mm can reclaim pages from the pool as needed.

Return: 0 on success, negative -errno on error

9.14.2.26 void kbase_mem_pool_set_max_size (struct kbase_mem_pool * *pool*, size_t *max_size*)

kbase_mem_pool_set_max_size - Set maximum number of free pages in memory pool : Memory pool to inspect : Maximum number of free pages the pool can hold

If is reduced, the pool will be shrunk to adhere to the new limit. For details see kbase_mem_pool_shrink().

9.14.2.27 void kbase_mem_pool_term (struct kbase_mem_pool * *pool*)

kbase_mem_pool_term - Destroy a memory pool : Memory pool to destroy

Pages in the pool will spill over to (if available) or freed to the kernel.

9.14.2.28 void kbase_mem_pool_trim (struct kbase_mem_pool * *pool*, size_t *new_size*)

kbase_mem_pool_trim - Grow or shrink the pool to a new size : Memory pool to trim : New number of pages in the pool

If $>$, fill the pool with new pages from the kernel, but not above the max_size for the pool. If $<$, shrink the pool by freeing pages to the kernel.

9.14.2.29 void kbase_mmu_disable (struct kbase_context * *kctx*)

kbase_mmu_disable() - Disable the MMU for a previously active kbase context. : Kbase context

Disable and perform the required cache maintenance to remove the all data from provided kbase context from the GPU caches.

The caller has the following locking conditions:

- It must hold kbase_device->mmu_hw_mutex
- It must hold the hwaccess_lock

9.14.2.30 void kbase_mmu_disable_as (struct kbase_device * *kbdev*, int *as_nr*)

kbase_mmu_disable_as() - Set the MMU to unmapped mode for the specified address space. : Kbase device : The address space number to set to unmapped.

This function must only be called during reset/power-up and it used to ensure the registers are in a known state.

The caller must hold kbdev->mmu_hw_mutex.

9.14.2.31 void* kbase_mmu_dump (struct kbase_context * *kctx*, int *nr_pages*)

Dump the MMU tables to a buffer

This function allocates a buffer (of `nr_pages` pages) to hold a dump of the MMU tables and fills it. If the buffer is too small then the return value will be NULL.

The GPU vm lock must be held when calling this function.

The buffer returned should be freed with vfree when it is no longer required.

Parameters

in	*kctx*	The kbase context to dump
in	*nr_pages*	The number of pages to allocate for the buffer.

Returns

The address of the buffer containing the MMU dump or NULL on error (including if the `nr_pages` is too small)

9.14.2.32 void kbase_mmu_interrupt_process (struct kbase_device * *kbdev*, struct kbase_context * *kctx*, struct kbase_as * *as*)

Process a bus or page fault.

This function will process a fault on a specific address space

Parameters

in	*kbdev*	The kbase_device the fault happened on
in	*kctx*	The kbase_context for the faulting address space if one was found.
in	*as*	The address space that has the fault

9.14.2.33 void kbase_mmu_update (struct kbase_context * *kctx*)

The caller has the following locking conditions:

- It must hold kbase_device->mmu_hw_mutex
- It must hold the hwaccess_lock

9.14.2.34 int kbase_mmu_update_pages_no_flush (struct kbase_context * *kctx*, u64 *vpfn*, struct tagged_addr * *phys*, size_t *nr*, unsigned long *flags*)

Update the entries for specified number of pages pointed to by 'phys' at GPU PFN 'vpfn'. This call is being triggered as a response to the changes of the mem attributes

Precondition

: The caller is responsible for validating the memory attributes

IMPORTANT: This uses kbasep_js_runpool_release_ctx() when the context is currently scheduled into the runpool, and so potentially uses a lot of locks. These locks must be taken in the correct order with respect to others already held by the caller. Refer to kbasep_js_runpool_release_ctx() for more information.

9.14.2.35 struct kbase_va_region* kbase_region_tracker_find_region_base_address (struct kbase_context * *kctx*, u64 *gpu_addr*)

Check that a pointer is actually a valid region.

Must be called with context lock held.

9.14.2.36 int kbase_region_tracker_init (struct kbase_context * *kctx*)

Initialize the region tracker data structure.

9.14.2.37 struct kbase_ctx_ext_res_meta* kbase_sticky_resource_acquire (struct kbase_context * *kctx*, u64 *gpu_addr*)

kbase_sticky_resource_acquire - Acquire a reference on a sticky resource. : kbase context. : The GPU address of the external resource.

Return: The metadata object which represents the binding between the external resource and the kbase context on success or NULL on failure.

9.14.2.38 int kbase_sticky_resource_init (struct kbase_context * *kctx*)

kbase_sticky_resource_init - Initialize sticky resource management. : kbase context

Returns zero on success or negative error number on failure.

9.14.2.39 bool kbase_sticky_resource_release (struct kbase_context * *kctx*, struct kbase_ctx_ext_res_meta * *meta*, u64 *gpu_addr*)

kbase_sticky_resource_release - Release a reference on a sticky resource. : kbase context. : Binding metadata. : GPU address of the external resource.

If meta is NULL then gpu_addr will be used to scan the metadata list and find the matching metadata (if any), otherwise the provided meta will be used and gpu_addr will be ignored.

Return: True if the release found the metadata and the reference was dropped.

9.14.2.40 void kbase_sticky_resource_term (struct kbase_context * *kctx*)

kbase_sticky_resource_term - Terminate sticky resource management. : kbase context

9.14.2.41 int kbase_sync_now (struct kbase_context * *kctx*, struct basep_syncset * *sset*)

kbase_sync_now - Perform cache maintenance on a memory region

: The kbase context of the region : A syncset structure describing the region and direction of the synchronisation required

Return: 0 on success or error code

9.14.2.42 void kbase_unmap_external_resource (struct kbase_context * *kctx*, struct kbase_va_region * *reg*, struct kbase_mem_phy_alloc * *alloc*)

kbase_unmap_external_resource - Unmap an external resource from the GPU. : kbase context. : The region to unmap or NULL if it has already been released. : The physical allocation being unmapped.

9.14.2.43 int kbase_update_region_flags (struct kbase_context * *kctx*, struct kbase_va_region * *reg*, unsigned long *flags*)

kbase_update_region_flags - Convert user space flags to kernel region flags

: kbase context : The region to update the flags on : The flags passed from user space

The user space flag BASE_MEM_COHERENT_SYSTEM_REQUIRED will be rejected and this function will fail if the system does not support system coherency.

Return: 0 if successful, -EINVAL if the flags are not supported

9.14.2.44 int kbasep_find_enclosing_cpu_mapping_offset (struct kbase_context * *kctx*, unsigned long *uaddr*, size_t *size*, u64 * *offset*)

kbasep_find_enclosing_cpu_mapping_offset() - Find the offset of the CPU mapping of a memory allocation containing a given address range

Searches for a CPU mapping of any part of any region that fully encloses the CPU virtual address range specified by and . Returns a failure indication if only part of the address range lies within a CPU mapping.

: The kernel base context used for the allocation. : Start of the CPU virtual address range. : Size of the CPU virtual address range (in bytes). : The offset from the start of the allocation to the specified CPU virtual address.

Return: 0 if offset was obtained successfully. Error code otherwise.

9.14.2.45 int kbasep_find_enclosing_gpu_mapping_start_and_offset (struct kbase_context * *kctx*, u64 *gpu_addr*, size_t *size*, u64 * *start*, u64 * *offset*)

kbasep_find_enclosing_gpu_mapping_start_and_offset() - Find the address of the start of GPU virtual memory region which encloses for the length in bytes

Searches for the memory region in GPU virtual memory space which contains the region defined by the and , where is the beginning and the length in bytes of the provided region. If found, the location of the start address of the GPU virtual memory region is passed in pointer and the location of the offset of the region into the GPU virtual memory region is passed in pointer.

: The kernel base context within which the memory is searched. : GPU virtual address for which the region is sought; defines the beginning of the provided region. : The length (in bytes) of the provided region for which the GPU virtual memory region is sought. : Pointer to the location where the address of the start of the found GPU virtual memory region is. : Pointer to the location where the offset of into the found GPU virtual memory region is.

9.14.2.46 void kbasep_os_process_page_usage_update (struct kbase_context * *kctx*, int *pages*)

Update the memory allocation counters for the current process.

OS specific call to updates the current memory allocation counters for the current process with the supplied delta.

Parameters

in	*kctx*	The kbase context
in	*pages*	The desired delta to apply to the memory usage counters.

9.14.2.47 void page_fault_worker (struct work_struct * *data*)

Process a page fault.

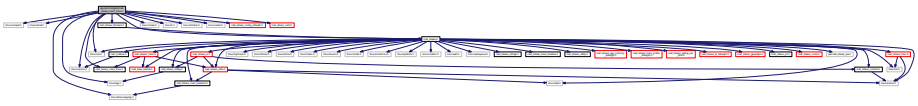
Parameters

in	*data*	work_struct passed by queue_work()

9.15 gpu/arm/midgard/mali_kbase_mem_linux.c File Reference

```
#include <linux/compat.h>
#include <linux/kernel.h>
#include <linux/bug.h>
#include <linux/mm.h>
#include <linux/mman.h>
#include <linux/fs.h>
#include <linux/version.h>
#include <linux/dma-mapping.h>
#include <linux/shrinker.h>
#include <linux/cache.h>
#include <mali_kbase.h>
#include <mali_kbase_mem_linux.h>
#include <mali_kbase_config_defaults.h>
#include <mali_kbase_tlstream.h>
#include <mali_kbase_ioctl.h>
```

Include dependency graph for mali_kbase_mem_linux.c:



Macros

- #define **KBASE_MEM_IMPORT_HAVE_PAGES** (1UL << BASE_MEM_FLAGS_NR_BITS)

Functions

- struct kbase_va_region * **kbase_mem_alloc** (struct kbase_context *kctx, u64 va_pages, u64 commit_↩ pages, u64 extent, u64 *flags, u64 *gpu_va)
- **KBASE_EXPORT_TEST_API** (kbase_mem_alloc)
- int **kbase_mem_query** (struct kbase_context *kctx, u64 gpu_addr, int query, u64 *const out)

- int kbase_mem_evictable_init (struct kbase_context *kctx)
- void kbase_mem_evictable_deinit (struct kbase_context *kctx)
- int kbase_mem_evictable_make (struct kbase_mem_phy_alloc *gpu_alloc)
- bool kbase_mem_evictable_unmake (struct kbase_mem_phy_alloc *gpu_alloc)
- int **kbase_mem_flags_change** (struct kbase_context *kctx, u64 gpu_addr, unsigned int flags, unsigned int mask)
- u64 **kbase_mem_alias** (struct kbase_context *kctx, u64 *flags, u64 stride, u64 nents, struct base_mem_aliasing_info *ai, u64 *num_pages)
- int **kbase_mem_import** (struct kbase_context *kctx, enum base_mem_import_type type, void __user *phandle, u32 padding, u64 *gpu_va, u64 *va_pages, u64 *flags)
- int kbase_mem_grow_gpu_mapping (struct kbase_context *kctx, struct kbase_va_region *reg, u64 new_pages, u64 old_pages)
- int kbase_mem_commit (struct kbase_context *kctx, u64 gpu_addr, u64 new_pages)
- **KBASE_EXPORT_TEST_API** (kbase_cpu_vm_close)
- void **kbase_os_mem_map_lock** (struct kbase_context *kctx)
- void **kbase_os_mem_map_unlock** (struct kbase_context *kctx)
- int **kbase_mmap** (struct file *file, struct vm_area_struct *vma)
- **KBASE_EXPORT_TEST_API** (kbase_mmap)
- void * kbase_vmap_prot (struct kbase_context *kctx, u64 gpu_addr, size_t size, unsigned long prot_request, struct kbase_vmap_struct *map)
- void * kbase_vmap (struct kbase_context *kctx, u64 gpu_addr, size_t size, struct kbase_vmap_struct *map)
- **KBASE_EXPORT_TEST_API** (kbase_vmap)
- void kbase_vunmap (struct kbase_context *kctx, struct kbase_vmap_struct *map)
- **KBASE_EXPORT_TEST_API** (kbase_vunmap)
- void kbasep_os_process_page_usage_update (struct kbase_context *kctx, int pages)

  Update the memory allocation counters for the current process.
- void * kbase_va_alloc (struct kbase_context *kctx, u32 size, struct kbase_hwc_dma_mapping *handle)

  Allocate memory from kernel space and map it onto the GPU.
- **KBASE_EXPORT_SYMBOL** (kbase_va_alloc)
- void kbase_va_free (struct kbase_context *kctx, struct kbase_hwc_dma_mapping *handle)

  Free/unmap memory allocated by kbase_va_alloc.
- **KBASE_EXPORT_SYMBOL** (kbase_va_free)

Variables

- const struct vm_operations_struct **kbase_vm_ops**

9.15.1 Detailed Description

Base kernel memory APIs, Linux implementation.

9.15.2 Function Documentation

9.15.2.1 int kbase_mem_commit (struct kbase_context * *kctx*, u64 *gpu_addr*, u64 *new_pages*)

kbase_mem_commit - Change the physical backing size of a region

: The kernel context : Handle to the memory region : Number of physical pages to back the region with

Return: 0 on success or error code

9.15.2.2 void kbase_mem_evictable_deinit (struct kbase_context * *kctx*)

kbase_mem_evictable_deinit - De-initialize the Ephemeral memory eviction mechanism. : The kbase context to de-initialize.

9.15.2.3 int kbase_mem_evictable_init (struct kbase_context * *kctx*)

kbase_mem_evictable_init - Initialize the Ephemeral memory the eviction mechanism. : The kbase context to initialize.

Return: Zero on success or -errno on failure.

9.15.2.4 int kbase_mem_evictable_make (struct kbase_mem_phy_alloc * *gpu_alloc*)

kbase_mem_evictable_make - Make a physical allocation eligible for eviction : The physical allocation to make evictable

Return: 0 on success, -errno on error.

Take the provided region and make all the physical pages within it reclaimable by the kernel, updating the per-process VM stats as well. Remove any CPU mappings (as these can't be removed in the shrinker callback as mmap_sem might already be taken) but leave the GPU mapping intact as and until the shrinker reclaims the allocation.

Note: Must be called with the region lock of the containing context.

9.15.2.5 bool kbase_mem_evictable_unmake (struct kbase_mem_phy_alloc * *alloc*)

kbase_mem_evictable_unmake - Remove a physical allocations eligibility for eviction. : The physical allocation to remove eviction eligibility from.

Return: True if the allocation had its backing restored and false if it hasn't.

Make the physical pages in the region no longer reclaimable and update the per-process stats, if the shrinker has already evicted the memory then re-allocate it if the region is still alive.

Note: Must be called with the region lock of the containing context.

9.15.2.6 int kbase_mem_grow_gpu_mapping (struct kbase_context * *kctx,* struct kbase_va_region * *reg,* u64 *new_pages,* u64 *old_pages*)

kbase_mem_grow_gpu_mapping - Grow the GPU mapping of an allocation : Context the region belongs to : The GPU region : The number of pages after the grow : The number of pages before the grow

Return: 0 on success, -errno on error.

Expand the GPU mapping to encompass the new psychical pages which have been added to the allocation.

Note: Caller must be holding the region lock.

9.15.2.7 void* kbase_va_alloc (struct kbase_context * *kctx,* u32 *size,* struct kbase_hwc_dma_mapping * *handle*)

Allocate memory from kernel space and map it onto the GPU.

Parameters

kctx	The context used for the allocation/mapping
size	The size of the allocation in bytes
handle	An opaque structure used to contain the state needed to free the memory

Returns

the VA for kernel space and GPU MMU

9.15.2.8 void kbase_va_free (struct kbase_context * *kctx*, struct kbase_hwc_dma_mapping * *handle*)

Free/unmap memory allocated by kbase_va_alloc.

Parameters

kctx	The context used for the allocation/mapping
handle	An opaque structure returned by the kbase_va_alloc function.

9.15.2.9 void* kbase_vmap (struct kbase_context * *kctx*, u64 *gpu_addr*, size_t *size*, struct kbase_vmap_struct * *map*)

kbase_vmap - Map a GPU VA range into the kernel safely : Context the VA range belongs to : Start address of VA range : Size of VA range : Structure to be given to kbase_vunmap() on freeing

Return: Kernel-accessible CPU pointer to the VA range, or NULL on error

Map a GPU VA Range into the kernel. The VA range must be contained within a GPU memory region. Appropriate CPU cache-flushing operations are made as required, dependent on the CPU mapping for the memory region.

This is safer than using kmap() on the pages directly, because the pages here are refcounted to prevent freeing (and hence reuse elsewhere in the system) until an kbase_vunmap()

kbase_vmap_prot() should be used in preference, since kbase_vmap() makes no checks to ensure the security of e.g. imported user bufs from RO SHM.

Note: All cache maintenance operations shall be ignored if the memory region has been imported.

9.15.2.10 void* kbase_vmap_prot (struct kbase_context * *kctx*, u64 *gpu_addr*, size_t *size*, unsigned long *prot_request*, struct kbase_vmap_struct * *map*)

kbase_vmap_prot - Map a GPU VA range into the kernel safely, only if the requested access permissions are supported : Context the VA range belongs to : Start address of VA range : Size of VA range : Flags indicating how the caller will then access the memory : Structure to be given to kbase_vunmap() on freeing

Return: Kernel-accessible CPU pointer to the VA range, or NULL on error

Map a GPU VA Range into the kernel. The VA range must be contained within a GPU memory region. Appropriate CPU cache-flushing operations are made as required, dependent on the CPU mapping for the memory region.

This is safer than using kmap() on the pages directly, because the pages here are refcounted to prevent freeing (and hence reuse elsewhere in the system) until an kbase_vunmap()

The flags in should use KBASE_REG_{CPU,GPU}_{RD,WR}, to check whether the region should allow the intended access, and return an error if disallowed. This is essential for security of imported memory, particularly a user buf from SHM mapped into the process as RO. In that case, write access must be checked if the intention is for kernel to write to the memory.

The checks are also there to help catch access errors on memory where security is not a concern: imported memory that is always RW, and memory that was allocated and owned by the process attached to . In this case, it helps to identify memory that was was mapped with the wrong access type.

Note: KBASE_REG_GPU_{RD,WR} flags are currently supported for legacy cases where either the security of memory is solely dependent on those flags, or when userspace code was expecting only the GPU to access the memory (e.g. HW workarounds).

All cache maintenance operations shall be ignored if the memory region has been imported.

9.15.2.11 void kbase_vunmap (struct kbase_context * *kctx*, struct kbase_vmap_struct * *map*)

kbase_vunmap - Unmap a GPU VA range from the kernel : Context the VA range belongs to : Structure describing the mapping from the corresponding kbase_vmap() call

Unmaps a GPU VA range from the kernel, given its structure obtained from kbase_vmap(). Appropriate CPU cache-flushing operations are made as required, dependent on the CPU mapping for the memory region.

The reference taken on pages during kbase_vmap() is released.

Note: All cache maintenance operations shall be ignored if the memory region has been imported.

9.15.2.12 void kbasep_os_process_page_usage_update (struct kbase_context * *kctx*, int *pages*)

Update the memory allocation counters for the current process.

OS specific call to updates the current memory allocation counters for the current process with the supplied delta.

Parameters

in	*kctx*	The kbase context
in	*pages*	The desired delta to apply to the memory usage counters.

9.15.3 Variable Documentation

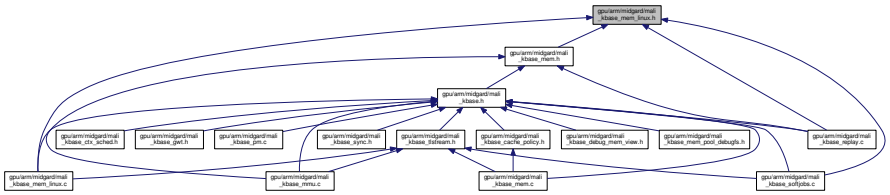
9.15.3.1 const struct vm_operations_struct kbase_vm_ops

Initial value:

```
= {
    .open  = kbase_cpu_vm_open,
    .close = kbase_cpu_vm_close,
    .fault = kbase_cpu_vm_fault
}
```

9.16 gpu/arm/midgard/mali_kbase_mem_linux.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct kbase_hwc_dma_mapping
- struct kbase_vmap_struct

Functions

- struct kbase_va_region * **kbase_mem_alloc** (struct kbase_context *kctx, u64 va_pages, u64 commit_↩ pages, u64 extent, u64 *flags, u64 *gpu_va)
- int **kbase_mem_query** (struct kbase_context *kctx, u64 gpu_addr, int query, u64 *const pages)
- int **kbase_mem_import** (struct kbase_context *kctx, enum base_mem_import_type type, void __user *phandle, u32 padding, u64 *gpu_va, u64 *va_pages, u64 *flags)
- u64 **kbase_mem_alias** (struct kbase_context *kctx, u64 *flags, u64 stride, u64 nents, struct base_mem_↩ aliasing_info *ai, u64 *num_pages)
- int **kbase_mem_flags_change** (struct kbase_context *kctx, u64 gpu_addr, unsigned int flags, unsigned int mask)
- int kbase_mem_commit (struct kbase_context *kctx, u64 gpu_addr, u64 new_pages)
- int **kbase_mmap** (struct file *file, struct vm_area_struct *vma)
- int kbase_mem_evictable_init (struct kbase_context *kctx)
- void kbase_mem_evictable_deinit (struct kbase_context *kctx)
- int kbase_mem_grow_gpu_mapping (struct kbase_context *kctx, struct kbase_va_region *reg, u64 new_↩ pages, u64 old_pages)
- int kbase_mem_evictable_make (struct kbase_mem_phy_alloc *gpu_alloc)
- bool kbase_mem_evictable_unmake (struct kbase_mem_phy_alloc *alloc)
- void * kbase_vmap_prot (struct kbase_context *kctx, u64 gpu_addr, size_t size, unsigned long prot_request, struct kbase_vmap_struct *map)
- void * kbase_vmap (struct kbase_context *kctx, u64 gpu_addr, size_t size, struct kbase_vmap_struct *map)
- void kbase_vunmap (struct kbase_context *kctx, struct kbase_vmap_struct *map)
- void * kbase_va_alloc (struct kbase_context *kctx, u32 size, struct kbase_hwc_dma_mapping *handle)

  Allocate memory from kernel space and map it onto the GPU.
- void kbase_va_free (struct kbase_context *kctx, struct kbase_hwc_dma_mapping *handle)

  Free/unmap memory allocated by kbase_va_alloc.

Variables

- const struct vm_operations_struct **kbase_vm_ops**

9.16.1 Detailed Description

Base kernel memory APIs, Linux implementation.

9.16.2 Function Documentation

9.16.2.1 int kbase_mem_commit (struct kbase_context * *kctx,* u64 *gpu_addr,* u64 *new_pages*)

kbase_mem_commit - Change the physical backing size of a region

: The kernel context : Handle to the memory region : Number of physical pages to back the region with

Return: 0 on success or error code

9.16.2.2 void kbase_mem_evictable_deinit (struct kbase_context * *kctx*)

kbase_mem_evictable_deinit - De-initialize the Ephemeral memory eviction mechanism. : The kbase context to de-initialize.

9.16.2.3 int kbase_mem_evictable_init (struct kbase_context * *kctx*)

kbase_mem_evictable_init - Initialize the Ephemeral memory the eviction mechanism. : The kbase context to initialize.

Return: Zero on success or -errno on failure.

9.16.2.4 int kbase_mem_evictable_make (struct kbase_mem_phy_alloc * *gpu_alloc*)

kbase_mem_evictable_make - Make a physical allocation eligible for eviction : The physical allocation to make evictable

Return: 0 on success, -errno on error.

Take the provided region and make all the physical pages within it reclaimable by the kernel, updating the per-process VM stats as well. Remove any CPU mappings (as these can't be removed in the shrinker callback as mmap_sem might already be taken) but leave the GPU mapping intact as and until the shrinker reclaims the allocation.

Note: Must be called with the region lock of the containing context.

9.16.2.5 bool kbase_mem_evictable_unmake (struct kbase_mem_phy_alloc * *alloc*)

kbase_mem_evictable_unmake - Remove a physical allocations eligibility for eviction. : The physical allocation to remove eviction eligibility from.

Return: True if the allocation had its backing restored and false if it hasn't.

Make the physical pages in the region no longer reclaimable and update the per-process stats, if the shrinker has already evicted the memory then re-allocate it if the region is still alive.

Note: Must be called with the region lock of the containing context.

9.16.2.6 int kbase_mem_grow_gpu_mapping (struct kbase_context * *kctx*, struct kbase_va_region * *reg*, u64 *new_pages*, u64 *old_pages*)

kbase_mem_grow_gpu_mapping - Grow the GPU mapping of an allocation : Context the region belongs to : The GPU region : The number of pages after the grow : The number of pages before the grow

Return: 0 on success, -errno on error.

Expand the GPU mapping to encompass the new psychical pages which have been added to the allocation.

Note: Caller must be holding the region lock.

9.16.2.7 void* kbase_va_alloc (struct kbase_context * *kctx*, u32 *size*, struct kbase_hwc_dma_mapping * *handle*)

Allocate memory from kernel space and map it onto the GPU.

Parameters

kctx	The context used for the allocation/mapping
size	The size of the allocation in bytes
handle	An opaque structure used to contain the state needed to free the memory

Returns

the VA for kernel space and GPU MMU

9.16.2.8 void kbase_va_free (struct kbase_context * *kctx*, struct kbase_hwc_dma_mapping * *handle*)

Free/unmap memory allocated by kbase_va_alloc.

Parameters

kctx	The context used for the allocation/mapping
handle	An opaque structure returned by the kbase_va_alloc function.

9.16.2.9 void* kbase_vmap (struct kbase_context * *kctx*, u64 *gpu_addr*, size_t *size*, struct kbase_vmap_struct * *map*)

kbase_vmap - Map a GPU VA range into the kernel safely : Context the VA range belongs to : Start address of VA range : Size of VA range : Structure to be given to kbase_vunmap() on freeing

Return: Kernel-accessible CPU pointer to the VA range, or NULL on error

Map a GPU VA Range into the kernel. The VA range must be contained within a GPU memory region. Appropriate CPU cache-flushing operations are made as required, dependent on the CPU mapping for the memory region.

This is safer than using kmap() on the pages directly, because the pages here are refcounted to prevent freeing (and hence reuse elsewhere in the system) until an kbase_vunmap()

kbase_vmap_prot() should be used in preference, since kbase_vmap() makes no checks to ensure the security of e.g. imported user bufs from RO SHM.

Note: All cache maintenance operations shall be ignored if the memory region has been imported.

9.16.2.10 void* kbase_vmap_prot (struct kbase_context * *kctx*, u64 *gpu_addr*, size_t *size*, unsigned long *prot_request*, struct kbase_vmap_struct * *map*)

kbase_vmap_prot - Map a GPU VA range into the kernel safely, only if the requested access permissions are supported : Context the VA range belongs to : Start address of VA range : Size of VA range : Flags indicating how the caller will then access the memory : Structure to be given to kbase_vunmap() on freeing

Return: Kernel-accessible CPU pointer to the VA range, or NULL on error

Map a GPU VA Range into the kernel. The VA range must be contained within a GPU memory region. Appropriate CPU cache-flushing operations are made as required, dependent on the CPU mapping for the memory region.

This is safer than using kmap() on the pages directly, because the pages here are refcounted to prevent freeing (and hence reuse elsewhere in the system) until an kbase_vunmap()

The flags in should use KBASE_REG_{CPU,GPU}_{RD,WR}, to check whether the region should allow the intended access, and return an error if disallowed. This is essential for security of imported memory, particularly a user buf from SHM mapped into the process as RO. In that case, write access must be checked if the intention is for kernel to write to the memory.

The checks are also there to help catch access errors on memory where security is not a concern: imported memory that is always RW, and memory that was allocated and owned by the process attached to . In this case, it helps to identify memory that was was mapped with the wrong access type.

Note: KBASE_REG_GPU_{RD,WR} flags are currently supported for legacy cases where either the security of memory is solely dependent on those flags, or when userspace code was expecting only the GPU to access the memory (e.g. HW workarounds).

All cache maintenance operations shall be ignored if the memory region has been imported.

9.16.2.11 void kbase_vunmap (struct kbase_context * *kctx*, struct kbase_vmap_struct * *map*)

kbase_vunmap - Unmap a GPU VA range from the kernel : Context the VA range belongs to : Structure describing the mapping from the corresponding kbase_vmap() call

Unmaps a GPU VA range from the kernel, given its structure obtained from kbase_vmap(). Appropriate CPU cache-flushing operations are made as required, dependent on the CPU mapping for the memory region.

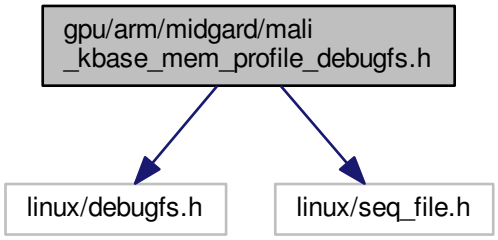
The reference taken on pages during kbase_vmap() is released.

Note: All cache maintenance operations shall be ignored if the memory region has been imported.

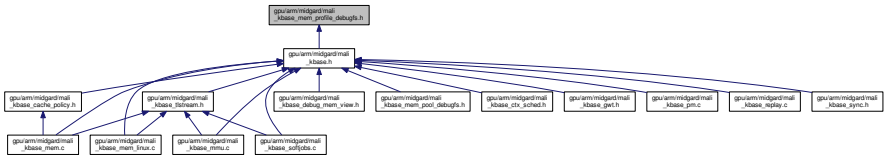
9.17 gpu/arm/midgard/mali_kbase_mem_profile_debugfs.h File Reference

```
#include <linux/debugfs.h>
#include <linux/seq_file.h>
```

Include dependency graph for mali_kbase_mem_profile_debugfs.h:



This graph shows which files directly or indirectly include this file:



Functions

- void kbasep_mem_profile_debugfs_remove (struct kbase_context ∗kctx)

  Remove entry from Mali memory profile debugfs.
- int kbasep_mem_profile_debugfs_insert (struct kbase_context ∗kctx, char ∗data, size_t size)

  Insert `data` *to the debugfs file so it can be read by userspace.*

9.17.1 Detailed Description

Header file for mem profiles entries in debugfs

9.17.2 Function Documentation

9.17.2.1 int kbasep_mem_profile_debugfs_insert (struct kbase_context ∗ *kctx,* char ∗ *data,* size_t *size*)

Insert `data` to the debugfs file so it can be read by userspace.

The function takes ownership of `data` and frees it later when new data is inserted.

If the debugfs entry corresponding to the `kctx` doesn't exist, an attempt will be made to create it.

Parameters

kctx	The context whose debugfs file `data` should be inserted to
data	A NULL-terminated string to be inserted to the debugfs file, without the trailing new line character
size	The length of the `data` string

Returns

0 if `data` inserted correctly -EAGAIN in case of error

Postcondition

mem_profile_initialized will be set to `true` the first time this function succeeds.

9.18 gpu/arm/midgard/mali_kbase_mem_profile_debugfs_buf_size.h File Reference

Macros

- #define KBASE_MEM_PROFILE_MAX_BUF_SIZE ((size_t) (64 + ((80 + (56 * 64)) * 15) + 56))

9.18.1 Detailed Description

Header file for the size of the buffer to accumulate the histogram report text in

9.18.2 Macro Definition Documentation

9.18.2.1 #define KBASE_MEM_PROFILE_MAX_BUF_SIZE ((size_t) (64 + ((80 + (56 * 64)) * 15) + 56))

The size of the buffer to accumulate the histogram report text in

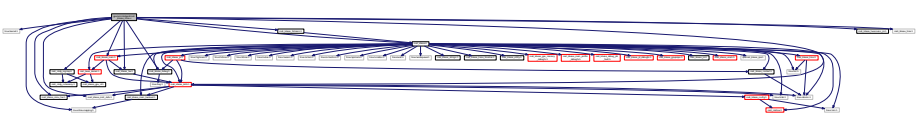
See also

CCTXP_HIST_BUF_SIZE_MAX_LENGTH_REPORT

9.19 gpu/arm/midgard/mali_kbase_mmu.c File Reference

```
#include <linux/kernel.h>
#include <linux/dma-mapping.h>
#include <mali_kbase.h>
#include <mali_midg_regmap.h>
#include <mali_kbase_tlstream.h>
#include <mali_kbase_instr_defs.h>
#include <mali_kbase_debug.h>
#include <mali_kbase_defs.h>
#include <mali_kbase_hw.h>
#include <mali_kbase_mmu_hw.h>
#include <mali_kbase_hwaccess_jm.h>
#include <mali_kbase_time.h>
#include <mali_kbase_mem.h>
```

Include dependency graph for mali_kbase_mmu.c:



Macros

- #define **beenthere**(kctx, f, a...) dev_dbg(kctx->kbdev->dev, "%s:" f, __func__, ##a)
- #define **KBASE_MMU_PAGE_ENTRIES** 512
- #define **mmu_get_bottom_pgd**(kctx, vpfn, out_pgd) mmu_get_pgd_at_level((kctx), (vpfn), MIDGARD_M↩ MU_BOTTOMLEVEL, (out_pgd))

Functions

- void page_fault_worker (struct work_struct *data)

  Process a page fault.
- phys_addr_t **kbase_mmu_alloc_pgd** (struct kbase_context *kctx)
- **KBASE_EXPORT_TEST_API** (kbase_mmu_alloc_pgd)
- int **kbase_mmu_insert_single_page** (struct kbase_context *kctx, u64 vpfn, struct tagged_addr phys, size↩ _t nr, unsigned long flags)
- int **kbase_mmu_insert_pages_no_flush** (struct kbase_context *kctx, const u64 start_vpfn, struct tagged↩ _addr *phys, size_t nr, unsigned long flags)
- int **kbase_mmu_insert_pages** (struct kbase_context *kctx, u64 vpfn, struct tagged_addr *phys, size_t nr, unsigned long flags)
- **KBASE_EXPORT_TEST_API** (kbase_mmu_insert_pages)
- void kbase_mmu_update (struct kbase_context *kctx)
- **KBASE_EXPORT_TEST_API** (kbase_mmu_update)
- void kbase_mmu_disable_as (struct kbase_device *kbdev, int as_nr)
- void kbase_mmu_disable (struct kbase_context *kctx)
- **KBASE_EXPORT_TEST_API** (kbase_mmu_disable)
- int **kbase_mmu_teardown_pages** (struct kbase_context *kctx, u64 vpfn, size_t nr)
- **KBASE_EXPORT_TEST_API** (kbase_mmu_teardown_pages)
- int kbase_mmu_update_pages_no_flush (struct kbase_context *kctx, u64 vpfn, struct tagged_addr *phys, size_t nr, unsigned long flags)
- int **kbase_mmu_update_pages** (struct kbase_context *kctx, u64 vpfn, struct tagged_addr *phys, size_t nr, unsigned long flags)
- int **kbase_mmu_init** (struct kbase_context *kctx)
- void **kbase_mmu_term** (struct kbase_context *kctx)
- void **kbase_mmu_free_pgd** (struct kbase_context *kctx)
- **KBASE_EXPORT_TEST_API** (kbase_mmu_free_pgd)
- void * kbase_mmu_dump (struct kbase_context *kctx, int nr_pages)
- **KBASE_EXPORT_TEST_API** (kbase_mmu_dump)
- void bus_fault_worker (struct work_struct *data)

  Process a bus fault.
- const char * kbase_exception_name (struct kbase_device *kbdev, u32 exception_code)
- void **kbasep_as_do_poke** (struct work_struct *work)
- enum hrtimer_restart **kbasep_as_poke_timer_callback** (struct hrtimer *timer)
- void kbase_as_poking_timer_retain_atom (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbase_jd_atom *katom)
- void kbase_as_poking_timer_release_atom (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbase_jd_atom *katom)
- void kbase_mmu_interrupt_process (struct kbase_device *kbdev, struct kbase_context *kctx, struct kbase↩ _as *as)

  Process a bus or page fault.
- void kbase_flush_mmu_wqs (struct kbase_device *kbdev)

  Flush MMU workqueues.

9.19.1 Detailed Description

Base kernel MMU management.

9.19.2 Function Documentation

9.19.2.1 void bus_fault_worker (struct work_struct * *data*)

Process a bus fault.

Parameters

in	*data*	work_struct passed by queue_work()

9.19.2.2 void kbase_as_poking_timer_release_atom (struct **kbase_device** * *kbdev,* struct **kbase_context** * *kctx,* struct **kbase_jd_atom** * *katom*)

If an atom holds a poking timer, release it and wait for it to finish

This must only be called on a context that's scheduled in, and an atom that still has a JS reference on the context

This must **not** be called from atomic context, since it can sleep.

9.19.2.3 void kbase_as_poking_timer_retain_atom (struct **kbase_device** * *kbdev,* struct **kbase_context** * *kctx,* struct **kbase_jd_atom** * *katom*)

Retain the poking timer on an atom's context (if the atom hasn't already done so), and start the timer (if it's not already started).

This must only be called on a context that's scheduled in, and an atom that's running on the GPU.

The caller must hold hwaccess_lock

This can be called safely from atomic context

9.19.2.4 const char* kbase_exception_name (struct **kbase_device** * *kbdev,* u32 *exception_code*)

Returns the name associated with a Mali exception code

This function is called from the interrupt handler when a GPU fault occurs. It reports the details of the fault using KBASE_DEBUG_PRINT_WARN.

Parameters

in	*kbdev*	The kbase device that the GPU fault occurred from.
in	*exception_code*	exception code

Returns

name associated with the exception code

9.19.2.5 void kbase_flush_mmu_wqs (struct kbase_device * *kbdev*)

Flush MMU workqueues.

This function will cause any outstanding page or bus faults to be processed. It should be called prior to powering off the GPU.

Parameters

in	*kbdev*	Device pointer

9.19.2.6 void kbase_mmu_disable (struct kbase_context * *kctx*)

kbase_mmu_disable() - Disable the MMU for a previously active kbase context. : Kbase context

Disable and perform the required cache maintenance to remove the all data from provided kbase context from the GPU caches.

The caller has the following locking conditions:

- It must hold kbase_device->mmu_hw_mutex
- It must hold the hwaccess_lock

9.19.2.7 void kbase_mmu_disable_as (struct kbase_device * *kbdev*, int *as_nr*)

kbase_mmu_disable_as() - Set the MMU to unmapped mode for the specified address space. : Kbase device : The address space number to set to unmapped.

This function must only be called during reset/power-up and it used to ensure the registers are in a known state.

The caller must hold kbdev->mmu_hw_mutex.

9.19.2.8 void* kbase_mmu_dump (struct kbase_context * *kctx*, int *nr_pages*)

Dump the MMU tables to a buffer

This function allocates a buffer (of `nr_pages` pages) to hold a dump of the MMU tables and fills it. If the buffer is too small then the return value will be NULL.

The GPU vm lock must be held when calling this function.

The buffer returned should be freed with vfree when it is no longer required.

Parameters

in	*kctx*	The kbase context to dump
in	*nr_pages*	The number of pages to allocate for the buffer.

Returns

The address of the buffer containing the MMU dump or NULL on error (including if the `nr_pages` is too small)

9.19.2.9 void kbase_mmu_interrupt_process (struct kbase_device * *kbdev*, struct kbase_context * *kctx*, struct kbase_as * *as*)

Process a bus or page fault.

This function will process a fault on a specific address space

Parameters

in	*kbdev*	The kbase_device the fault happened on
in	*kctx*	The kbase_context for the faulting address space if one was found.
in	*as*	The address space that has the fault

9.19.2.10 void kbase_mmu_update (struct kbase_context * *kctx*)

The caller has the following locking conditions:

- It must hold kbase_device->mmu_hw_mutex
- It must hold the hwaccess_lock

9.19.2.11 int kbase_mmu_update_pages_no_flush (struct kbase_context * *kctx*, u64 *vpfn*, struct tagged_addr * *phys*, size_t *nr*, unsigned long *flags*)

Update the entries for specified number of pages pointed to by 'phys' at GPU PFN 'vpfn'. This call is being triggered as a response to the changes of the mem attributes

Precondition

: The caller is responsible for validating the memory attributes

IMPORTANT: This uses kbasep_js_runpool_release_ctx() when the context is currently scheduled into the runpool, and so potentially uses a lot of locks. These locks must be taken in the correct order with respect to others already held by the caller. Refer to kbasep_js_runpool_release_ctx() for more information.

9.19.2.12 void page_fault_worker (struct work_struct * *data*)

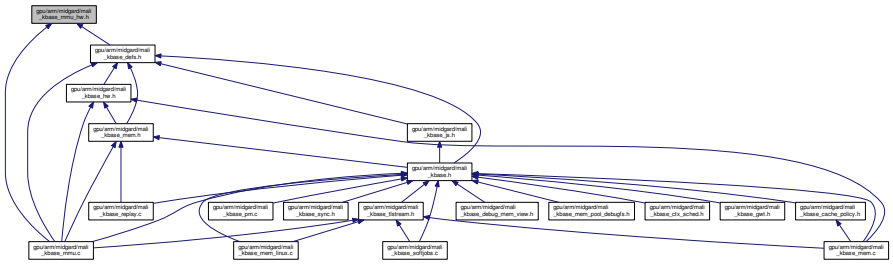
Process a page fault.

Parameters

in	*data*	work_struct passed by queue_work()

9.20 gpu/arm/midgard/mali_kbase_mmu_hw.h File Reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum kbase_mmu_fault_type {
  KBASE_MMU_FAULT_TYPE_UNKNOWN = 0, **KBASE_MMU_FAULT_TYPE_PAGE**, **KBASE_MMU_F↩AULT_TYPE_BUS**, **KBASE_MMU_FAULT_TYPE_PAGE_UNEXPECTED**,
  KBASE_MMU_FAULT_TYPE_BUS_UNEXPECTED }

  MMU fault type descriptor.

Functions

- void kbase_mmu_hw_configure (struct kbase_device ∗kbdev, struct kbase_as ∗as, struct kbase_context ∗kctx)

  Configure an address space for use.
- int kbase_mmu_hw_do_operation (struct kbase_device ∗kbdev, struct kbase_as ∗as, struct kbase_context ∗kctx, u64 vpfn, u32 nr, u32 type, unsigned int handling_irq)

  Issue an operation to the MMU.
- void kbase_mmu_hw_clear_fault (struct kbase_device ∗kbdev, struct kbase_as ∗as, struct kbase_context ∗kctx, enum kbase_mmu_fault_type type)

  Clear a fault that has been previously reported by the MMU.
- void kbase_mmu_hw_enable_fault (struct kbase_device ∗kbdev, struct kbase_as ∗as, struct kbase_context ∗kctx, enum kbase_mmu_fault_type type)

  Enable fault that has been previously reported by the MMU.

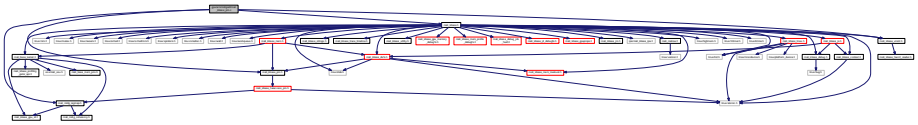
9.20.1 Detailed Description

Interface file for accessing MMU hardware functionality

9.21 gpu/arm/midgard/mali_kbase_pm.c File Reference

```
#include <mali_kbase.h>
#include <mali_midg_regmap.h>
#include <mali_kbase_vinstr.h>
#include <mali_kbase_pm.h>
```

Include dependency graph for mali_kbase_pm.c:



Functions

- int kbase_pm_powerup (struct kbase_device *kbdev, unsigned int flags)
- void kbase_pm_halt (struct kbase_device *kbdev)
- void kbase_pm_context_active (struct kbase_device *kbdev)
- int kbase_pm_context_active_handle_suspend (struct kbase_device *kbdev, enum kbase_pm_suspend_↩ handler suspend_handler)
- **KBASE_EXPORT_TEST_API** (kbase_pm_context_active)
- void kbase_pm_context_idle (struct kbase_device *kbdev)
- **KBASE_EXPORT_TEST_API** (kbase_pm_context_idle)
- void kbase_pm_suspend (struct kbase_device *kbdev)
- void kbase_pm_resume (struct kbase_device *kbdev)

9.21.1 Detailed Description

Base kernel power management APIs

9.21.2 Function Documentation

9.21.2.1 void kbase_pm_context_active (struct kbase_device * *kbdev*)

Increment the count of active contexts.

This function should be called when a context is about to submit a job. It informs the active power policy that the GPU is going to be in use shortly and the policy is expected to start turning on the GPU.

This function will block until the GPU is available.

This function ASSERTS if a suspend is occuring/has occurred whilst this is in use. Use kbase_pm_contect_active↩ _unless_suspending() instead.

Note

a Suspend is only visible to Kernel threads; user-space threads in a syscall cannot witness a suspend, because they are frozen before the suspend begins.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.21.2.2 int kbase_pm_context_active_handle_suspend (struct kbase_device * *kbdev*, enum kbase_pm_suspend_handler *suspend_handler*)

Suspend 'safe' variant of kbase_pm_context_active()

If a suspend is in progress, this allows for various different ways of handling the suspend. Refer to enum kbase_↩ pm_suspend_handler for details.

We returns a status code indicating whether we're allowed to keep the GPU active during the suspend, depending on the handler code. If the status code indicates a failure, the caller must abort whatever operation it was attempting, and potentially queue it up for after the OS has resumed.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)
suspend_handler	The handler code for how to handle a suspend that might occur

Returns

zero Indicates success
non-zero Indicates failure due to the system being suspending/suspended.

9.21.2.3 void kbase_pm_context_idle (struct kbase_device * *kbdev*)

Decrement the reference count of active contexts.

This function should be called when a context becomes idle. After this call the GPU may be turned off by the power policy so the calling code should ensure that it does not access the GPU's registers.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.21.2.4 void kbase_pm_halt (struct kbase_device * *kbdev*)

Halt the power management framework. Should ensure that no new interrupts are generated, but allow any currently running interrupt handlers to complete successfully. The GPU is forced off by the time this function returns, regardless of whether or not the active power policy asks for the GPU to be powered off.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.21.2.5 int kbase_pm_powerup (struct kbase_device * *kbdev,* unsigned int *flags*)

Power up GPU after all modules have been initialized and interrupt handlers installed.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)
flags	Flags to pass on to kbase_pm_init_hw

Returns

0 if powerup was successful.

9.21.2.6 void kbase_pm_resume (struct kbase_device * *kbdev*)

Resume the GPU, allow register accesses to it, and resume running atoms on the GPU.

This is called in response to an OS resume event, and calls into the various kbase components to complete the resume.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.21.2.7 void kbase_pm_suspend (struct kbase_device * *kbdev*)

Suspend the GPU and prevent any further register accesses to it from Kernel threads.

This is called in response to an OS suspend event, and calls into the various kbase components to complete the suspend.

Note

the mechanisms used here rely on all user-space threads being frozen by the OS before we suspend. Otherwise, an IOCTL could occur that powers up the GPU e.g. via atom submission.

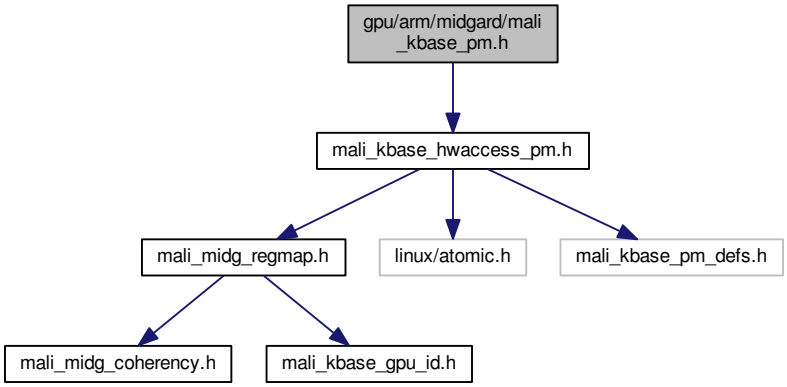
Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

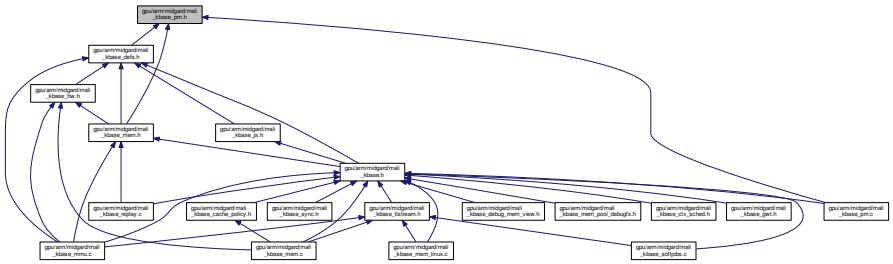
9.22 gpu/arm/midgard/mali_kbase_pm.h File Reference

```
#include "mali_kbase_hwaccess_pm.h"
```

Include dependency graph for mali_kbase_pm.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **PM_ENABLE_IRQS** 0x01
- #define **PM_HW_ISSUES_DETECT** 0x02

Enumerations

- enum kbase_pm_suspend_handler { KBASE_PM_SUSPEND_HANDLER_NOT_POSSIBLE, KBASE_PM↩ _SUSPEND_HANDLER_DONT_INCREASE, KBASE_PM_SUSPEND_HANDLER_DONT_REACTIVATE }

Functions

- int kbase_pm_init (struct kbase_device ∗kbdev)
- int kbase_pm_powerup (struct kbase_device ∗kbdev, unsigned int flags)
- void kbase_pm_halt (struct kbase_device ∗kbdev)
- void kbase_pm_term (struct kbase_device ∗kbdev)
- void kbase_pm_context_active (struct kbase_device ∗kbdev)
- int kbase_pm_context_active_handle_suspend (struct kbase_device ∗kbdev, enum kbase_pm_suspend_↩ handler suspend_handler)
- void kbase_pm_context_idle (struct kbase_device ∗kbdev)
- void kbase_pm_suspend (struct kbase_device ∗kbdev)
- void kbase_pm_resume (struct kbase_device ∗kbdev)
- void kbase_pm_vsync_callback (int buffer_updated, void ∗data)

9.22.1 Detailed Description

Power management API definitions

9.22.2 Enumeration Type Documentation

9.22.2.1 enum kbase_pm_suspend_handler

Handler codes for doing kbase_pm_context_active_handle_suspend()

Enumerator

KBASE_PM_SUSPEND_HANDLER_NOT_POSSIBLE A suspend is not expected/not possible - this is the same as kbase_pm_context_active()

KBASE_PM_SUSPEND_HANDLER_DONT_INCREASE If we're suspending, fail and don't increase the active count

KBASE_PM_SUSPEND_HANDLER_DONT_REACTIVATE If we're suspending, succeed and allow the active count to increase iff it didn't go from 0->1 (i.e., we didn't re-activate the GPU).

This should only be used when there is a bounded time on the activation (e.g. guarantee it's going to be idled very soon after)

9.22.3 Function Documentation

9.22.3.1 void kbase_pm_context_active (struct kbase_device * *kbdev*)

Increment the count of active contexts.

This function should be called when a context is about to submit a job. It informs the active power policy that the GPU is going to be in use shortly and the policy is expected to start turning on the GPU.

This function will block until the GPU is available.

This function ASSERTS if a suspend is occuring/has occurred whilst this is in use. Use kbase_pm_contect_active↩ _unless_suspending() instead.

Note

a Suspend is only visible to Kernel threads; user-space threads in a syscall cannot witness a suspend, because they are frozen before the suspend begins.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.22.3.2 int kbase_pm_context_active_handle_suspend (struct kbase_device * *kbdev*, enum kbase_pm_suspend_handler *suspend_handler*)

Suspend 'safe' variant of kbase_pm_context_active()

If a suspend is in progress, this allows for various different ways of handling the suspend. Refer to enum kbase_↩ pm_suspend_handler for details.

We returns a status code indicating whether we're allowed to keep the GPU active during the suspend, depending on the handler code. If the status code indicates a failure, the caller must abort whatever operation it was attempting, and potentially queue it up for after the OS has resumed.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)
suspend_handler	The handler code for how to handle a suspend that might occur

Returns

zero Indicates success
non-zero Indicates failure due to the system being suspending/suspended.

9.22.3.3 void kbase_pm_context_idle (struct kbase_device * *kbdev*)

Decrement the reference count of active contexts.

This function should be called when a context becomes idle. After this call the GPU may be turned off by the power policy so the calling code should ensure that it does not access the GPU's registers.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.22.3.4 void kbase_pm_halt (struct kbase_device * *kbdev*)

Halt the power management framework. Should ensure that no new interrupts are generated, but allow any currently running interrupt handlers to complete successfully. The GPU is forced off by the time this function returns, regardless of whether or not the active power policy asks for the GPU to be powered off.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.22.3.5 int kbase_pm_init (struct kbase_device * *kbdev*)

Initialize the power management framework.

Must be called before any other power management function

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

Returns

0 if the power management framework was successfully initialized.

9.22.3.6 int kbase_pm_powerup (struct kbase_device * *kbdev*, unsigned int *flags*)

Power up GPU after all modules have been initialized and interrupt handlers installed.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)
flags	Flags to pass on to kbase_pm_init_hw

Returns

0 if powerup was successful.

9.22.3.7 void kbase_pm_resume (struct kbase_device * *kbdev*)

Resume the GPU, allow register accesses to it, and resume running atoms on the GPU.

This is called in response to an OS resume event, and calls into the various kbase components to complete the resume.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.22.3.8 void kbase_pm_suspend (struct kbase_device * *kbdev*)

Suspend the GPU and prevent any further register accesses to it from Kernel threads.

This is called in response to an OS suspend event, and calls into the various kbase components to complete the suspend.

Note

the mechanisms used here rely on all user-space threads being frozen by the OS before we suspend. Otherwise, an IOCTL could occur that powers up the GPU e.g. via atom submission.

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.22.3.9 void kbase_pm_term (struct kbase_device * *kbdev*)

Terminate the power management framework.

No power management functions may be called after this (except kbase_pm_init)

Parameters

kbdev	The kbase device structure for the device (must be a valid pointer)

9.22.3.10 void kbase_pm_vsync_callback (int *buffer_updated,* void * *data*)

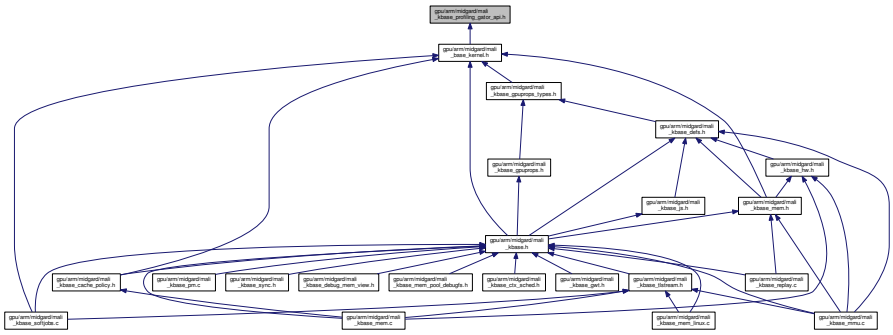
kbase_pm_vsync_callback - vsync callback

: 1 if a new frame was displayed, 0 otherwise : Pointer to the kbase device as returned by kbase_find_device()

Callback function used to notify the power management code that a vsync has occurred on the display.

9.23 gpu/arm/midgard/mali_kbase_profiling_gator_api.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define **FBDUMP_CONTROL_ENABLE** (1)
- #define **FBDUMP_CONTROL_RATE** (2)
- #define **SW_COUNTER_ENABLE** (3)
- #define **FBDUMP_CONTROL_RESIZE_FACTOR** (4)
- #define **FBDUMP_CONTROL_MAX** (5)
- #define **FBDUMP_CONTROL_MIN** FBDUMP_CONTROL_ENABLE

Functions

- void **_mali_profiling_control** (u32 action, u32 value)

9.23.1 Detailed Description

Model interface

9.24 gpu/arm/midgard/mali_kbase_replay.c File Reference

```
#include <linux/dma-mapping.h>
#include <mali_kbase_config.h>
#include <mali_kbase.h>
#include <mali_kbase_mem.h>
#include <mali_kbase_mem_linux.h>
```

Include dependency graph for mali_kbase_replay.c:

Classes

- struct fragment_job

Macros

- #define **JOB_NOT_STARTED** 0
- #define **JOB_TYPE_NULL** (1)
- #define **JOB_TYPE_VERTEX** (5)
- #define **JOB_TYPE_TILER** (7)
- #define **JOB_TYPE_FUSED** (8)
- #define **JOB_TYPE_FRAGMENT** (9)
- #define **JOB_HEADER_32_FBD_OFFSET** (31∗4)
- #define **JOB_HEADER_64_FBD_OFFSET** (44∗4)
- #define **FBD_POINTER_MASK** (∼0x3f)
- #define **SFBD_TILER_OFFSET** (48∗4)
- #define **MFBD_TILER_OFFSET** (14∗4)
- #define **FBD_HIERARCHY_WEIGHTS** 8
- #define **FBD_HIERARCHY_MASK_MASK** 0x1fff
- #define **FBD_TYPE** 1
- #define **HIERARCHY_WEIGHTS** 13
- #define **JOB_HEADER_ID_MAX** 0xffff
- #define **JOB_SOURCE_ID**(status) (((status) >> 16) & 0xFFFF)
- #define **JOB_POLYGON_LIST** (0x03)

Functions

- bool kbase_replay_process (struct kbase_jd_atom ∗katom)

  Process a replay job.

9.24.1 Detailed Description

Replay soft job handlers

9.24.2 Function Documentation

9.24.2.1 bool kbase_replay_process (struct kbase_jd_atom * *katom*)

Process a replay job.

Called from kbase_process_soft_job.

On exit, if the job has completed, katom->event_code will have been updated. If the job has not completed, and is replaying jobs, then the atom status will have been reset to KBASE_JD_ATOM_STATE_QUEUED.

Parameters

in	*katom*	The atom to be processed

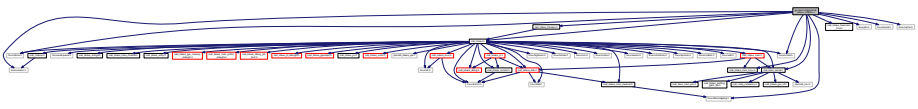
Returns

false if the atom has completed true if the atom is replaying jobs

9.25 gpu/arm/midgard/mali_kbase_softjobs.c File Reference

```
#include <mali_kbase.h>
#include <linux/dma-mapping.h>
#include <mali_base_kernel.h>
#include <mali_kbase_hwaccess_time.h>
#include <mali_kbase_mem_linux.h>
#include <mali_kbase_tlstream.h>
#include <linux/version.h>
#include <linux/ktime.h>
#include <linux/pfn.h>
#include <linux/sched.h>
#include <linux/kernel.h>
#include <linux/cache.h>
```

Include dependency graph for mali_kbase_softjobs.c:



Classes

- struct kbase_debug_copy_buffer

Functions

- void **kbasep_remove_waiting_soft_job** (struct kbase_jd_atom *katom)
- void **kbasep_complete_triggered_soft_events** (struct kbase_context *kctx, u64 evt)
- void **kbasep_soft_job_timeout_worker** (unsigned long data)
- int kbase_soft_event_update (struct kbase_context *kctx, u64 event, unsigned char new_status)
- int **kbase_process_soft_job** (struct kbase_jd_atom *katom)
- void **kbase_cancel_soft_job** (struct kbase_jd_atom *katom)
- int **kbase_prepare_soft_job** (struct kbase_jd_atom *katom)
- void **kbase_finish_soft_job** (struct kbase_jd_atom *katom)
- void **kbase_resume_suspended_soft_jobs** (struct kbase_device *kbdev)

9.25.1 Detailed Description

This file implements the logic behind software only jobs that are executed within the driver rather than being handed over to the GPU.

9.25.2 Function Documentation

9.25.2.1 int kbase_soft_event_update (struct kbase_context * *kctx*, u64 *event*, unsigned char *new_status*)

kbase_soft_event_update() - Update soft event state : Pointer to context : Event to update : New status value of event

Update the event, and wake up any atoms waiting for the event.

Return: 0 on success, a negative error code on failure.

9.26 gpu/arm/midgard/mali_kbase_sync.h File Reference

```
#include <linux/syscalls.h>
#include "mali_kbase.h"
```

Include dependency graph for mali_kbase_sync.h:

Classes

- struct kbase_sync_fence_info

Functions

: Name of stream (only used to ease debugging/visualization)

kbase_sync_fence_stream_create() - Create a stream object

: A file descriptor representing the created stream object

Can map down to a timeline implementation in some implementations. Exposed as a file descriptor. Life-time controlled via the file descriptor:

- *dup to add a ref*
- *close to remove a ref*

return: 0 on success, < 0 on error

- int **kbase_sync_fence_stream_create** (const char *name, int *const out_fd)
- int kbase_sync_fence_out_create (struct kbase_jd_atom *katom, int stream_fd)
- int kbase_sync_fence_in_from_fd (struct kbase_jd_atom *katom, int fd)
- int kbase_sync_fence_validate (int fd)
- enum base_jd_event_code kbase_sync_fence_out_trigger (struct kbase_jd_atom *katom, int result)
- int kbase_sync_fence_in_wait (struct kbase_jd_atom *katom)
- void kbase_sync_fence_in_cancel_wait (struct kbase_jd_atom *katom)
- void kbase_sync_fence_in_remove (struct kbase_jd_atom *katom)
- void kbase_sync_fence_out_remove (struct kbase_jd_atom *katom)
- int kbase_sync_fence_in_info_get (struct kbase_jd_atom *katom, struct kbase_sync_fence_info *info)
- int kbase_sync_fence_out_info_get (struct kbase_jd_atom *katom, struct kbase_sync_fence_info *info)
- const char * kbase_sync_status_string (int status)
- void **kbase_sync_fence_wait_worker** (struct work_struct *data)

9.26.1 Detailed Description

This file contains our internal "API" for explicit fences. It hides the implementation details of the actual explicit fence mechanism used (Android fences or sync file with DMA fences).

9.26.2 Function Documentation

9.26.2.1 void kbase_sync_fence_in_cancel_wait (struct kbase_jd_atom * *katom*)

kbase_sync_fence_in_cancel_wait() - Cancel explicit input fence waits : Atom to cancel wait for

This function is fully responsible for continuing processing of this atom (remove_waiting_soft_job + finish_soft_job + jd_done + js_sched_all)

9.26.2.2 int kbase_sync_fence_in_from_fd (struct kbase_jd_atom * *katom*, int *fd*)

kbase_sync_fence_in_from_fd() Assigns an existing fence to specified atom : Atom to assign the existing explicit fence to : File descriptor to an existing fence

Assigns an explicit input fence to atom. This can later be waited for by calling

return: 0 on success, < 0 on error

9.26.2.3 int kbase_sync_fence_in_info_get (struct kbase_jd_atom * *katom*, struct kbase_sync_fence_info * *info*)

kbase_sync_fence_in_info_get() - Retrieves information about input fence : Atom to get fence information from : Struct to be filled with fence information

return: 0 on success, < 0 on error

9.26.2.4 void kbase_sync_fence_in_remove (struct kbase_jd_atom * *katom*)

kbase_sync_fence_in_remove() - Remove the input fence from the katom : Atom to remove explicit input fence for

This will also release the corresponding reference.

9.26.2.5 int kbase_sync_fence_in_wait (struct kbase_jd_atom * *katom*)

kbase_sync_fence_in_wait() - Wait for explicit input fence to be signaled : Atom with explicit fence to wait for

If the fence is already signaled, then 0 is returned, and the caller must continue processing of the katom.

If the fence isn't already signaled, then this kbase_sync framework will take responsibility to continue the processing once the fence is signaled.

return: 0 if already signaled, otherwise 1

9.26.2.6 int kbase_sync_fence_out_create (struct kbase_jd_atom * *katom*, int *stream_fd*)

kbase_sync_fence_out_create Create an explicit output fence to specified atom : Atom to assign the new explicit fence to : File descriptor for stream object to create fence on

return: Valid file descriptor to fence or < 0 on error

9.26.2.7 int kbase_sync_fence_out_info_get (struct kbase_jd_atom * *katom*, struct kbase_sync_fence_info * *info*)

kbase_sync_fence_out_info_get() - Retrieves information about output fence : Atom to get fence information from : Struct to be filled with fence information

return: 0 on success, < 0 on error

9.26.2.8 void kbase_sync_fence_out_remove (struct kbase_jd_atom * *katom*)

kbase_sync_fence_out_remove() - Remove the output fence from the katom : Atom to remove explicit output fence for

This will also release the corresponding reference.

9.26.2.9 enum base_jd_event_code kbase_sync_fence_out_trigger (struct kbase_jd_atom * *katom*, int *result*)

kbase_sync_fence_out_trigger - Signal explicit output fence attached on katom : Atom with an explicit fence to signal

Returns

: < 0 means signal with error, 0 >= indicates success

Signal output fence attached on katom and remove the fence from the atom.

return: The "next" event code for atom, typically JOB_CANCELLED or EVENT_DONE

9.26.2.10 int kbase_sync_fence_validate (int *fd*)

kbase_sync_fence_validate() - Validate a fd to be a valid fence : File descriptor to check

This function is only usable to catch unintentional user errors early, it does not stop malicious code changing the fd after this function returns.

return 0: if fd is for a valid fence, < 0 if invalid

9.26.2.11 const char* kbase_sync_status_string (int *status*)

kbase_sync_status_string() - Get string matching : Value of fence status.

return: Pointer to string describing .

Index