hsa-runtime64

Generated by Doxygen 1.9.5

Chapter 1

Deprecated List

Member hsa_agent_extension_supported (uint16_t extension, hsa_agent_t agent, uint16_t version_major, uint16_t version_minor, bool *result)

Member hsa_agent_get_exception_policies (hsa_agent_t agent, hsa_profile_t profile, uint16_t *mask)

Use hsa_isa_get_exception_policies for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, this function uses the first value returned by hsa_agent_iterate_isas.

Member HSA_AGENT_INFO_BASE_PROFILE_DEFAULT_FLOAT_ROUNDING_MODES

Query HSA_ISA_INFO_BASE_PROFILE_DEFAULT_FLOAT_ROUNDING_MODES for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas.

Member HSA_AGENT_INFO_CACHE_SIZE

Query hsa_agent_iterate_caches to retrieve information about the caches present in a given agent.

Member HSA_AGENT_INFO_DEFAULT_FLOAT_ROUNDING_MODE

Query HSA_ISA_INFO_DEFAULT_FLOAT_ROUNDING_MODES for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas.

Member HSA_AGENT_INFO_FAST_F16_OPERATION

Query HSA_ISA_INFO_FAST_F16_OPERATION for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas.

Member HSA_AGENT_INFO_FBARRIER_MAX_SIZE

Query HSA_ISA_INFO_FBARRIER_MAX_SIZE for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas.

Member HSA_AGENT_INFO_GRID_MAX_DIM

Query HSA_ISA_INFO_GRID_MAX_DIM for a given intruction set architecture supported by the agent instead.

Member HSA_AGENT_INFO_GRID_MAX_SIZE

Query HSA_ISA_INFO_GRID_MAX_SIZE for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas.

Member HSA_AGENT_INFO_ISA

An agent may support multiple instruction set architectures. See hsa_agent_iterate_isas. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas.

Member HSA_AGENT_INFO_MACHINE_MODEL

Query HSA_ISA_INFO_MACHINE_MODELS for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas.

Member HSA_AGENT_INFO_NODE

NUMA information is not exposed anywhere else in the API.

Member HSA_AGENT_INFO_PROFILE

Query HSA_ISA_INFO_PROFILES for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas.

Member HSA_AGENT_INFO_QUEUES_MAX

The maximum number of queues is not statically determined.

Member HSA_AGENT_INFO_WAVEFRONT_SIZE

Query HSA_WAVEFRONT_INFO_SIZE for a given wavefront and intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas and the first wavefront enumerated by hsa_isa_iterate_wavefronts for that ISA.

Member HSA_AGENT_INFO_WORKGROUP_MAX_DIM

Query HSA_ISA_INFO_WORKGROUP_MAX_DIM for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas.

Member HSA_AGENT_INFO_WORKGROUP_MAX_SIZE

Query HSA_ISA_INFO_WORKGROUP_MAX_SIZE for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas.

Member hsa_callback_data_t

Member hsa_code_object_deserialize (void ∗serialized_code_object, size_t serialized_code_object_size, const char ∗options, hsa_code_object_t ∗code_object)

Member hsa_code_object_destroy (hsa_code_object_t code_object)

Member hsa_code_object_get_info (hsa_code_object_t code_object, hsa_code_object_info_t attribute, void ∗value)

Member hsa_code_object_get_symbol (hsa_code_object_t code_object, const char ∗symbol_name, hsa↩_code_symbol_t ∗symbol)

Member hsa_code_object_get_symbol_from_name (hsa_code_object_t code_object, const char ∗module_name, const char ∗symbol_name, hsa_code_symbol_t ∗symbol)

Member hsa_code_object_info_t

Member hsa_code_object_iterate_symbols (hsa_code_object_t code_object, hsa_status_t(∗callback)(hsa_code_object_t code_object, hsa_code_symbol_t symbol, void ∗data), void ∗data)

Member hsa_code_object_serialize (hsa_code_object_t code_object, hsa_status_t(∗alloc_callback)(size↩_t size, hsa_callback_data_t data, void ∗∗address), hsa_callback_data_t callback_data, const char ∗options, void ∗∗serialized_code_object, size_t ∗serialized_code_object_size)

Member hsa_code_object_t

Member hsa_code_object_type_t

Member hsa_code_symbol_get_info (hsa_code_symbol_t code_symbol, hsa_code_symbol_info_t attribute, void ∗value)

Member hsa_code_symbol_info_t

Member hsa_code_symbol_t

Member hsa_executable_create (hsa_profile_t profile, hsa_executable_state_t executable_state, const char ∗options, hsa_executable_t ∗executable)

Use hsa_executable_create_alt instead, which allows the application to specify the default floating-point rounding mode of the executable and assumes an unfrozen initial state.

Member hsa_executable_get_symbol (hsa_executable_t executable, const char ∗module_name, const char ∗symbol_name, hsa_agent_t agent, int32_t call_convention, hsa_executable_symbol_t ∗symbol)

Use hsa_executable_get_symbol_by_name instead.

Member hsa_executable_iterate_symbols (hsa_executable_t executable, hsa_status_t(∗callback)(hsa_executable_t exec, hsa_executable_symbol_t symbol, void ∗data), void ∗data)

Member hsa_executable_load_code_object (hsa_executable_t executable, hsa_agent_t agent, hsa_code_object_t code_object, const char ∗options)

Member HSA_EXECUTABLE_SYMBOL_INFO_AGENT

Member HSA_EXECUTABLE_SYMBOL_INFO_INDIRECT_FUNCTION_CALL_CONVENTION

Member HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_CALL_CONVENTION

Member HSA_EXECUTABLE_SYMBOL_INFO_MODULE_NAME

Member HSA_EXECUTABLE_SYMBOL_INFO_MODULE_NAME_LENGTH

Member HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_ALIGNMENT

Member HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_ALLOCATION

Member HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_IS_CONST

Member HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_SEGMENT

Member HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_SIZE

Member HSA_EXT_IMAGE_CAPABILITY_READ_MODIFY_WRITE

Images of this geometry, format, and layout can be accessed from read-modify-write atomic operations in the agent.

Member hsa_isa_compatible (hsa_isa_t code_object_isa, hsa_isa_t agent_isa, bool ∗result)

Use hsa_agent_iterate_isas to query which instructions set architectures are supported by a given agent.

Member hsa_isa_get_info (hsa_isa_t isa, hsa_isa_info_t attribute, uint32_t index, void *value)

The concept of call convention has been deprecated. If the application wants to query the value of an attribute for a given instruction set architecture, use hsa_isa_get_info_alt instead. If the application wants to query an attribute that is specific to a given combination of ISA and wavefront, use hsa_wavefront_get_info.

Member HSA_ISA_INFO_CALL_CONVENTION_COUNT

Member HSA_ISA_INFO_CALL_CONVENTION_INFO_WAVEFRONT_SIZE

Member HSA_ISA_INFO_CALL_CONVENTION_INFO_WAVEFRONTS_PER_COMPUTE_UNIT

Member HSA_PACKET_HEADER_ACQUIRE_FENCE_SCOPE

Renamed as HSA_PACKET_HEADER_SCACQUIRE_FENCE_SCOPE.

Member HSA_PACKET_HEADER_RELEASE_FENCE_SCOPE

Renamed as HSA_PACKET_HEADER_SCRELEASE_FENCE_SCOPE.

Member HSA_PACKET_HEADER_WIDTH_ACQUIRE_FENCE_SCOPE

Use HSA_PACKET_HEADER_WIDTH_SCACQUIRE_FENCE_SCOPE.

Member HSA_PACKET_HEADER_WIDTH_RELEASE_FENCE_SCOPE

Use HSA_PACKET_HEADER_WIDTH_SCRELEASE_FENCE_SCOPE.

Member hsa_queue_add_write_index_acq_rel (const hsa_queue_t *queue, uint64_t value)

Renamed as hsa_queue_add_write_index_scacq_screl.

Member hsa_queue_add_write_index_acquire (const hsa_queue_t *queue, uint64_t value)

Renamed as hsa_queue_add_write_index_scacquire.

Member hsa_queue_add_write_index_release (const hsa_queue_t *queue, uint64_t value)

Renamed as hsa_queue_add_write_index_screlease.

Member hsa_queue_cas_write_index_acq_rel (const hsa_queue_t *queue, uint64_t expected, uint64_t value)

Renamed as hsa_queue_cas_write_index_scacq_screl.

Member hsa_queue_cas_write_index_acquire (const hsa_queue_t *queue, uint64_t expected, uint64_t value)

Renamed as hsa_queue_cas_write_index_scacquire.

Member hsa_queue_cas_write_index_release (const hsa_queue_t *queue, uint64_t expected, uint64_t value)

Renamed as hsa_queue_cas_write_index_screlease.

Member hsa_queue_load_read_index_acquire (const hsa_queue_t *queue)

Renamed as hsa_queue_load_read_index_scacquire.

Member hsa_queue_load_write_index_acquire (const hsa_queue_t *queue)

Renamed as hsa_queue_load_write_index_scacquire.

Member hsa_queue_store_read_index_release (const hsa_queue_t *queue, uint64_t value)

Renamed as hsa_queue_store_read_index_screlease.

Member hsa_queue_store_write_index_release (const hsa_queue_t *queue, uint64_t value)

Renamed as hsa_queue_store_write_index_screlease.

Member hsa_signal_add_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_add_scacq_screl.

Member hsa_signal_add_acquire (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_add_scacquire.

Member hsa_signal_add_release (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_add_screlease.

Member hsa_signal_and_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_and_scacq_screl.

Member hsa_signal_and_acquire (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_and_scacquire.

Member hsa_signal_and_release (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_and_screlease.

Member hsa_signal_cas_acq_rel (hsa_signal_t signal, hsa_signal_value_t expected, hsa_signal_value_t value)

Renamed as hsa_signal_cas_scacq_screl.

Member hsa_signal_cas_acquire (hsa_signal_t signal, hsa_signal_value_t expected, hsa_signal_value_t value)

Renamed as hsa_signal_cas_scacquire.

Member hsa_signal_cas_release (hsa_signal_t signal, hsa_signal_value_t expected, hsa_signal_value_t value)

Renamed as hsa_signal_cas_screlease.

Member hsa_signal_exchange_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_exchange_scacq_screl.

Member hsa_signal_exchange_acquire (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_exchange_scacquire.

Member hsa_signal_exchange_release (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_exchange_screlease.

Member hsa_signal_load_acquire (hsa_signal_t signal)

Renamed as hsa_signal_load_scacquire.

Member hsa_signal_or_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_or_scacq_screl.

Member hsa_signal_or_acquire (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_or_scacquire.

Member hsa_signal_or_release (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_or_screlease.

Member hsa_signal_store_release (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_store_screlease.

Member hsa_signal_subtract_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_subtract_scacq_screl.

Member hsa_signal_subtract_acquire (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_subtract_scacquire.

Member hsa_signal_subtract_release (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_subtract_screlease.

Member hsa_signal_wait_acquire (hsa_signal_t signal, hsa_signal_condition_t condition, hsa_signal_value_t compare_value, uint64_t timeout_hint, hsa_wait_state_t wait_state_hint)

Renamed as hsa_signal_wait_scacquire.

Member hsa_signal_xor_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_xor_scacq_screl.

Member hsa_signal_xor_acquire (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_xor_scacquire.

Member hsa_signal_xor_release (hsa_signal_t signal, hsa_signal_value_t value)

Renamed as hsa_signal_xor_screlease.

Member hsa_system_extension_supported (uint16_t extension, uint16_t version_major, uint16_t version_minor, bool *result)

Member hsa_system_get_extension_table (uint16_t extension, uint16_t version_major, uint16_t version_minor, void *table)

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Module Documentation

5.1 Runtime Notifications

Classes

- struct hsa_dim3_s

  Three-dimensional coordinate.

Typedefs

- typedef struct hsa_dim3_s **hsa_dim3_t**

  Three-dimensional coordinate.
- typedef int hsa_file_t

  POSIX file descriptor.

Enumerations

- enum hsa_status_t {
  HSA_STATUS_SUCCESS = 0x0 , HSA_STATUS_INFO_BREAK = 0x1 , HSA_STATUS_ERROR = 0x1000 ,
  HSA_STATUS_ERROR_INVALID_ARGUMENT = 0x1001 ,
  HSA_STATUS_ERROR_INVALID_QUEUE_CREATION = 0x1002 , HSA_STATUS_ERROR_INVALID_ALLOCATION
  = 0x1003 , HSA_STATUS_ERROR_INVALID_AGENT = 0x1004 , HSA_STATUS_ERROR_INVALID_REGION
  = 0x1005 ,
  HSA_STATUS_ERROR_INVALID_SIGNAL = 0x1006 , HSA_STATUS_ERROR_INVALID_QUEUE = 0x1007 ,
  HSA_STATUS_ERROR_OUT_OF_RESOURCES = 0x1008 , HSA_STATUS_ERROR_INVALID_PACKET_FORMAT
  = 0x1009 ,
  HSA_STATUS_ERROR_RESOURCE_FREE = 0x100A , HSA_STATUS_ERROR_NOT_INITIALIZED =
  0x100B , HSA_STATUS_ERROR_REFCOUNT_OVERFLOW = 0x100C , HSA_STATUS_ERROR_INCOMPATIBLE_ARGUMEN
  = 0x100D ,
  HSA_STATUS_ERROR_INVALID_INDEX = 0x100E , HSA_STATUS_ERROR_INVALID_ISA = 0x100F ,
  HSA_STATUS_ERROR_INVALID_ISA_NAME = 0x1017 , HSA_STATUS_ERROR_INVALID_CODE_OBJECT
  = 0x1010 ,
  HSA_STATUS_ERROR_INVALID_EXECUTABLE = 0x1011 , HSA_STATUS_ERROR_FROZEN_EXECUTABLE
  = 0x1012 , HSA_STATUS_ERROR_INVALID_SYMBOL_NAME = 0x1013 , HSA_STATUS_ERROR_VARIABLE_ALREADY_DE
  = 0x1014 ,
  HSA_STATUS_ERROR_VARIABLE_UNDEFINED = 0x1015 , HSA_STATUS_ERROR_EXCEPTION =
  0x1016 , HSA_STATUS_ERROR_INVALID_CODE_SYMBOL = 0x1018 , HSA_STATUS_ERROR_INVALID_EXECUTABLE_SY

= 0x1019 ,
HSA_STATUS_ERROR_INVALID_FILE = 0x1020 , HSA_STATUS_ERROR_INVALID_CODE_OBJECT_READER = 0x1021 , HSA_STATUS_ERROR_INVALID_CACHE = 0x1022 , HSA_STATUS_ERROR_INVALID_WAVEFRONT = 0x1023 ,
HSA_STATUS_ERROR_INVALID_SIGNAL_GROUP = 0x1024 , HSA_STATUS_ERROR_INVALID_RUNTIME_STATE = 0x1025 , HSA_STATUS_ERROR_FATAL = 0x1026 }

Status codes.

- enum hsa_access_permission_t { HSA_ACCESS_PERMISSION_RO = 1 , HSA_ACCESS_PERMISSION_WO = 2 , HSA_ACCESS_PERMISSION_RW = 3 }

  Access permissions.

Functions

- hsa_status_t HSA_API hsa_status_string (hsa_status_t status, const char **status_string)

  Query additional information about a status code.
- hsa_status_t HSA_API hsa_init ()

  Initialize the HSA runtime.
- hsa_status_t HSA_API hsa_shut_down ()

  Shut down the HSA runtime.

5.1.1 Detailed Description

5.1.2 Typedef Documentation

5.1.2.1 hsa_file_t

```
typedef int hsa_file_t
```

POSIX file descriptor.

Definition at line 336 of file hsa.h.

5.1.3 Enumeration Type Documentation

5.1.3.1 hsa_access_permission_t

```
enum hsa_access_permission_t
```

Access permissions.

Enumerator

HSA_ACCESS_PERMISSION_RO	Read-only access.
HSA_ACCESS_PERMISSION_WO	Write-only access.
HSA_ACCESS_PERMISSION_RW	Read and write access.

Definition at line 318 of file hsa.h.

5.1.3.2 hsa_status_t

enum hsa_status_t

Status codes.

Enumerator

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_INFO_BREAK	A traversal over a list of elements has been interrupted by the application before completing.
HSA_STATUS_ERROR	A generic error has occurred.
HSA_STATUS_ERROR_INVALID_ARGUMENT	One of the actual arguments does not meet a precondition stated in the documentation of the corresponding formal argument.
HSA_STATUS_ERROR_INVALID_QUEUE_↩ CREATION	The requested queue creation is not valid.
HSA_STATUS_ERROR_INVALID_ALLOCATION	The requested allocation is not valid.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_REGION	The memory region is invalid.
HSA_STATUS_ERROR_INVALID_SIGNAL	The signal is invalid.
HSA_STATUS_ERROR_INVALID_QUEUE	The queue is invalid.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the necessary resources. This error may also occur when the HSA runtime needs to spawn threads or create internal OS-specific events.
HSA_STATUS_ERROR_INVALID_PACKET_↩ FORMAT	The AQL packet is malformed.
HSA_STATUS_ERROR_RESOURCE_FREE	An error has been detected while releasing a resource.
HSA_STATUS_ERROR_NOT_INITIALIZED	An API other than hsa_init has been invoked while the reference count of the HSA runtime is 0.
HSA_STATUS_ERROR_REFCOUNT_OVERFLOW	The maximum reference count for the object has been reached.
HSA_STATUS_ERROR_INCOMPATIBLE_↩ ARGUMENTS	The arguments passed to a functions are not compatible.
HSA_STATUS_ERROR_INVALID_INDEX	The index is invalid.
HSA_STATUS_ERROR_INVALID_ISA	The instruction set architecture is invalid.
HSA_STATUS_ERROR_INVALID_ISA_NAME	The instruction set architecture name is invalid.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT	The code object is invalid.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_FROZEN_EXECUTABLE	The executable is frozen.
HSA_STATUS_ERROR_INVALID_SYMBOL_NAME	There is no symbol with the given name.
HSA_STATUS_ERROR_VARIABLE_ALREADY_↩ DEFINED	The variable is already defined.
HSA_STATUS_ERROR_VARIABLE_UNDEFINED	The variable is undefined.
HSA_STATUS_ERROR_EXCEPTION	An HSAIL operation resulted in a hardware exception.
HSA_STATUS_ERROR_INVALID_CODE_SYMBOL	The code object symbol is invalid.

Enumerator

HSA_STATUS_ERROR_INVALID_EXECUTABLE_↩ SYMBOL	The executable symbol is invalid.
HSA_STATUS_ERROR_INVALID_FILE	The file descriptor is invalid.
HSA_STATUS_ERROR_INVALID_CODE_↩ OBJECT_READER	The code object reader is invalid.
HSA_STATUS_ERROR_INVALID_CACHE	The cache is invalid.
HSA_STATUS_ERROR_INVALID_WAVEFRONT	The wavefront is invalid.
HSA_STATUS_ERROR_INVALID_SIGNAL_GROUP	The signal group is invalid.
HSA_STATUS_ERROR_INVALID_RUNTIME_STATE	The HSA runtime is not in the configuration state.
HSA_STATUS_ERROR_FATAL	The queue received an error that may require process termination.

Definition at line 118 of file hsa.h.

5.1.4 Function Documentation

5.1.4.1 hsa_init()

```
hsa_status_t HSA_API hsa_init ( )
```

Initialize the HSA runtime.

Initializes the HSA runtime if it is not already initialized, and increases the reference counter associated with the HSA runtime for the current process. Invocation of any HSA function other than hsa_init results in undefined behavior if the current HSA runtime reference counter is less than one.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_REFCOUNT_OVERFLOW	The HSA runtime reference count reaches INT32_MAX.

5.1.4.2 hsa_shut_down()

```
hsa_status_t HSA_API hsa_shut_down ( )
```

Shut down the HSA runtime.

Decreases the reference count of the HSA runtime instance. When the reference count reaches 0, the HSA runtime is no longer considered valid but the application might call hsa_init to initialize the HSA runtime again.

Once the reference count of the HSA runtime reaches 0, all the resources associated with it (queues, signals, agent information, etc.) are considered invalid and any attempt to reference them in subsequent API calls results in undefined behavior. When the reference count reaches 0, the HSA runtime may release resources associated with it.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.

5.1.4.3 hsa_status_string()

```
hsa_status_t HSA_API hsa_status_string (
            hsa_status_t status,
            const char ** status_string )
```

Query additional information about a status code.

Parameters

in	*status*	Status code.
out	*status_string*	A NUL-terminated string that describes the error status.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`status` is an invalid status code, or `status_string` is NULL.

5.2 System and Agent Information

Classes

- struct hsa_agent_s

  Struct containing an opaque handle to an agent, a device that participates in the HSA memory model. An agent can submit AQL packets for execution, and may also accept AQL packets for execution (agent dispatch packets or kernel dispatch packets launching HSAIL-derived binaries).
- struct hsa_cache_s

  Cache handle.

Typedefs

- typedef struct hsa_agent_s **hsa_agent_t**

  Struct containing an opaque handle to an agent, a device that participates in the HSA memory model. An agent can submit AQL packets for execution, and may also accept AQL packets for execution (agent dispatch packets or kernel dispatch packets launching HSAIL-derived binaries).
- typedef struct hsa_cache_s **hsa_cache_t**

  Cache handle.

Enumerations

- enum hsa_endianness_t { HSA_ENDIANNESS_LITTLE = 0 , HSA_ENDIANNESS_BIG = 1 }

  Endianness. A convention used to interpret the bytes making up a data word.
- enum hsa_machine_model_t { HSA_MACHINE_MODEL_SMALL = 0 , HSA_MACHINE_MODEL_LARGE = 1 }

  Machine model. A machine model determines the size of certain data types in HSA runtime and an agent.
- enum hsa_profile_t { HSA_PROFILE_BASE = 0 , HSA_PROFILE_FULL = 1 }

  Profile. A profile indicates a particular level of feature support. For example, in the base profile the application must use the HSA runtime allocator to reserve shared virtual memory, while in the full profile any host pointer can be shared across all the agents.
- enum hsa_system_info_t { HSA_SYSTEM_INFO_VERSION_MAJOR = 0 , HSA_SYSTEM_INFO_VERSION_MINOR = 1 , HSA_SYSTEM_INFO_TIMEST = 2 , HSA_SYSTEM_INFO_TIMESTAMP_FREQUENCY = 3 , HSA_SYSTEM_INFO_SIGNAL_MAX_WAIT = 4 , HSA_SYSTEM_INFO_ENDIANNESS = 5 , HSA_SYSTEM_INFO_MACHINE = 6 , HSA_SYSTEM_INFO_EXTENSIONS = 7 , HSA_AMD_SYSTEM_INFO_BUILD_VERSION = 0x200 , HSA_AMD_SYSTEM_INFO_SVM_SUPPORTED = 0x201 , HSA_AMD_SYSTEM_INFO_SVM_ACCESSIBLE_BY_DEFAULT = 0x202 }

  System attributes.
- enum hsa_extension_t { HSA_EXTENSION_FINALIZER = 0 , HSA_EXTENSION_IMAGES = 1 , HSA_EXTENSION_PERFORMANCE_COUNTERS = 2 , HSA_EXTENSION_PROFILING_EVENTS = 3 , HSA_EXTENSION_STD_LAST = 3 , HSA_AMD_FIRST_EXTENSION = 0x200 , HSA_EXTENSION_AMD_PROFILER = 0x200 , HSA_EXTENSION_AMD_LOADER = 0x201 , HSA_EXTENSION_AMD_AQLPROFILE = 0x202 , HSA_AMD_LAST_EXTENSION = 0x202 }

  HSA extensions.
- enum hsa_agent_feature_t { HSA_AGENT_FEATURE_KERNEL_DISPATCH = 1 , HSA_AGENT_FEATURE_AGENT_DISPATC = 2 }

  Agent features.
- enum hsa_device_type_t { HSA_DEVICE_TYPE_CPU = 0 , HSA_DEVICE_TYPE_GPU = 1 , HSA_DEVICE_TYPE_DSP = 2 }

  Hardware device type.
- enum hsa_default_float_rounding_mode_t { HSA_DEFAULT_FLOAT_ROUNDING_MODE_DEFAULT = 0 , HSA_DEFAULT_FLOAT_ROUNDING_MODE_ZERO = 1 , HSA_DEFAULT_FLOAT_ROUNDING_MODE_NEAR = 2 }

  Default floating-point rounding mode.
- enum hsa_agent_info_t { HSA_AGENT_INFO_NAME = 0 , HSA_AGENT_INFO_VENDOR_NAME = 1 , HSA_AGENT_INFO_FEATURE = 2 , HSA_AGENT_INFO_MACHINE_MODEL = 3 , HSA_AGENT_INFO_PROFILE = 4 , HSA_AGENT_INFO_DEFAULT_FLOAT_ROUNDING_MODE = 5 , HSA_AGENT_INFO_BASE_PROFILE_DEFAULT_FLOAT_ROUNDING_MODES = 23 , HSA_AGENT_INFO_FAST_F16_OPE = 24 , HSA_AGENT_INFO_WAVEFRONT_SIZE = 6 , HSA_AGENT_INFO_WORKGROUP_MAX_DIM = 7 , HSA_AGENT_INFO_WORKGROUP_MAX_SIZE = 8 , HSA_AGENT_INFO_GRID_MAX_DIM = 9 , HSA_AGENT_INFO_GRID_MAX_SIZE = 10 , HSA_AGENT_INFO_FBARRIER_MAX_SIZE = 11 , HSA_AGENT_INFO_QUEUES_MAX = 12 , HSA_AGENT_INFO_QUEUE_MIN_SIZE = 13 , HSA_AGENT_INFO_QUEUE_MAX_SIZE = 14 , HSA_AGENT_INFO_QUEUE_TYPE = 15 , HSA_AGENT_INFO_NODE = 16 , HSA_AGENT_INFO_DEVICE = 17 , HSA_AGENT_INFO_CACHE_SIZE = 18 , HSA_AGENT_INFO_ISA = 19 , HSA_AGENT_INFO_EXTENSIONS = 20 , HSA_AGENT_INFO_VERSION_MAJOR = 21 , HSA_AGENT_INFO_VERSION_MINOR = 22 , HSA_AGENT_INFO_LAST = INT32_MAX }

  Agent attributes.
- enum hsa_exception_policy_t { HSA_EXCEPTION_POLICY_BREAK = 1 , HSA_EXCEPTION_POLICY_DETECT = 2 }

  Exception policies applied in the presence of hardware exceptions.

- enum hsa_cache_info_t { HSA_CACHE_INFO_NAME_LENGTH = 0 , HSA_CACHE_INFO_NAME = 1 , HSA_CACHE_INFO_LEVEL = 2 , HSA_CACHE_INFO_SIZE = 3 }

  Cache attributes.

Functions

- hsa_status_t HSA_API hsa_system_get_info (hsa_system_info_t attribute, void *value)

  Get the current value of a system attribute.
- hsa_status_t HSA_API hsa_extension_get_name (uint16_t extension, const char **name)

  Query the name of a given extension.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_system_extension_supported (uint16_t extension, uint16_t version_major, uint16_t version_minor, bool *result)

  Query if a given version of an extension is supported by the HSA implementation.
- hsa_status_t HSA_API hsa_system_major_extension_supported (uint16_t extension, uint16_t version_major, uint16_t *version_minor, bool *result)

  Query if a given version of an extension is supported by the HSA implementation. All minor versions from 0 up to the returned `version_minor` must be supported by the implementation.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_system_get_extension_table (uint16_t extension, uint16_t version_major, uint16_t version_minor, void *table)

  Retrieve the function pointers corresponding to a given version of an extension. Portable applications are expected to invoke the extension API using the returned function pointers.
- hsa_status_t HSA_API hsa_system_get_major_extension_table (uint16_t extension, uint16_t version_major, size_t table_length, void *table)

  Retrieve the function pointers corresponding to a given major version of an extension. Portable applications are expected to invoke the extension API using the returned function pointers.
- hsa_status_t HSA_API hsa_agent_get_info (hsa_agent_t agent, hsa_agent_info_t attribute, void *value)

  Get the current value of an attribute for a given agent.
- hsa_status_t HSA_API hsa_iterate_agents (hsa_status_t(*callback)(hsa_agent_t agent, void *data), void *data)

  Iterate over the available agents, and invoke an application-defined callback on every iteration.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_agent_get_exception_policies (hsa_agent_t agent, hsa_profile_t profile, uint16_t *mask)

  Retrieve the exception policy support for a given combination of agent and profile.
- hsa_status_t HSA_API hsa_cache_get_info (hsa_cache_t cache, hsa_cache_info_t attribute, void *value)

  Get the current value of an attribute for a given cache object.
- hsa_status_t HSA_API hsa_agent_iterate_caches (hsa_agent_t agent, hsa_status_t(*callback)(hsa_cache_t cache, void *data), void *data)

  Iterate over the memory caches of a given agent, and invoke an application-defined callback on every iteration.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_agent_extension_supported (uint16_t extension, hsa_agent_t agent, uint16_t version_major, uint16_t version_minor, bool *result)

  Query if a given version of an extension is supported by an agent.
- hsa_status_t HSA_API hsa_agent_major_extension_supported (uint16_t extension, hsa_agent_t agent, uint16_t version_major, uint16_t *version_minor, bool *result)

  Query if a given version of an extension is supported by an agent. All minor versions from 0 up to the returned `version_minor` must be supported.

5.2.1 Detailed Description

5.2.2 Enumeration Type Documentation

5.2.2.1 hsa_agent_feature_t

enum hsa_agent_feature_t

Agent features.

Enumerator

HSA_AGENT_FEATURE_KERNEL_DISPATCH	The agent supports AQL packets of kernel dispatch type. If this feature is enabled, the agent is also a kernel agent.
HSA_AGENT_FEATURE_AGENT_DISPATCH	The agent supports AQL packets of agent dispatch type.

Definition at line 752 of file hsa.h.

5.2.2.2 hsa_agent_info_t

enum hsa_agent_info_t

Agent attributes.

Enumerator

HSA_AGENT_INFO_NAME	Agent name. The type of this attribute is a NUL-terminated char[64]. The name must be at most 63 characters long (not including the NUL terminator) and all array elements not used for the name must be NUL.
HSA_AGENT_INFO_VENDOR_NAME	Name of vendor. The type of this attribute is a NUL-terminated char[64]. The name must be at most 63 characters long (not including the NUL terminator) and all array elements not used for the name must be NUL.
HSA_AGENT_INFO_FEATURE	Agent capability. The type of this attribute is hsa_agent_feature_t.
HSA_AGENT_INFO_MACHINE_MODEL	**Deprecated** Query HSA_ISA_INFO_MACHINE_MODELS for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas. Machine model supported by the agent. The type of this attribute is hsa_machine_model_t.

Enumerator

HSA_AGENT_INFO_PROFILE	**Deprecated** Query HSA_ISA_INFO_PROFILES for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas. Profile supported by the agent. The type of this attribute is hsa_profile_t.
HSA_AGENT_INFO_DEFAULT_FLOAT_↩ ROUNDING_MODE	**Deprecated** Query HSA_ISA_INFO_DEFAULT_FLOAT_ROUNDING_MODES for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas. Default floating-point rounding mode. The type of this attribute is hsa_default_float_rounding_mode_t, but the value HSA_DEFAULT_FLOAT_ROUNDING_MODE_DEFAULT is not allowed.
HSA_AGENT_INFO_BASE_PROFILE_DEFAULT_↩ FLOAT_ROUNDING_MODES	**Deprecated** Query HSA_ISA_INFO_BASE_PROFILE_DEFAULT_FLOAT_ROU for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas. A bit-mask of hsa_default_float_rounding_mode_t values, representing the default floating-point rounding modes supported by the agent in the Base profile. The type of this attribute is uint32_t. The default floating-point rounding mode (HSA_AGENT_INFO_DEFAULT_FLOAT_ROUNDING_MODE) bit must not be set.

Enumerator

HSA_AGENT_INFO_FAST_F16_OPERATION	**Deprecated** Query HSA_ISA_INFO_FAST_F16_OPERATION for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas. Flag indicating that the f16 HSAIL operation is at least as fast as the f32 operation in the current agent. The value of this attribute is undefined if the agent is not a kernel agent. The type of this attribute is bool.
HSA_AGENT_INFO_WAVEFRONT_SIZE	**Deprecated** Query HSA_WAVEFRONT_INFO_SIZE for a given wavefront and intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas and the first wavefront enumerated by hsa_isa_iterate_wavefronts for that ISA. Number of work-items in a wavefront. Must be a power of 2 in the range [1,256]. The value of this attribute is undefined if the agent is not a kernel agent. The type of this attribute is uint32_t.
HSA_AGENT_INFO_WORKGROUP_MAX_DIM	**Deprecated** Query HSA_ISA_INFO_WORKGROUP_MAX_DIM for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas. Maximum number of work-items of each dimension of a work-group. Each maximum must be greater than 0. No maximum can exceed the value of HSA_AGENT_INFO_WORKGROUP_MAX_SIZE. The value of this attribute is undefined if the agent is not a kernel agent. The type of this attribute is uint16_t[3].

Enumerator

HSA_AGENT_INFO_WORKGROUP_MAX_SIZE	**Deprecated** Query HSA_ISA_INFO_WORKGROUP_MAX_SIZE for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas. Maximum total number of work-items in a work-group. The value of this attribute is undefined if the agent is not a kernel agent. The type of this attribute is uint32_t.
HSA_AGENT_INFO_GRID_MAX_DIM	**Deprecated** Query HSA_ISA_INFO_GRID_MAX_DIM for a given intruction set architecture supported by the agent instead. Maximum number of work-items of each dimension of a grid. Each maximum must be greater than 0, and must not be smaller than the corresponding value in HSA_AGENT_INFO_WORKGROUP_MAX_DIM. No maximum can exceed the value of HSA_AGENT_INFO_GRID_MAX_SIZE. The value of this attribute is undefined if the agent is not a kernel agent. The type of this attribute is hsa_dim3_t.
HSA_AGENT_INFO_GRID_MAX_SIZE	**Deprecated** Query HSA_ISA_INFO_GRID_MAX_SIZE for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas. Maximum total number of work-items in a grid. The value of this attribute is undefined if the agent is not a kernel agent. The type of this attribute is uint32_t.
HSA_AGENT_INFO_FBARRIER_MAX_SIZE	**Deprecated** Query HSA_ISA_INFO_FBARRIER_MAX_SIZE for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas. Maximum number of fbarriers per work-group. Must be at least 32. The value of this attribute is undefined if the agent is not a kernel agent. The type of this attribute is uint32_t.

Enumerator

HSA_AGENT_INFO_QUEUES_MAX	**Deprecated** The maximum number of queues is not statically determined. Maximum number of queues that can be active (created but not destroyed) at one time in the agent. The type of this attribute is uint32_t.
HSA_AGENT_INFO_QUEUE_MIN_SIZE	Minimum number of packets that a queue created in the agent can hold. Must be a power of 2 greater than 0. Must not exceed the value of HSA_AGENT_INFO_QUEUE_MAX_SIZE. The type of this attribute is uint32_t.
HSA_AGENT_INFO_QUEUE_MAX_SIZE	Maximum number of packets that a queue created in the agent can hold. Must be a power of 2 greater than 0. The type of this attribute is uint32_t.
HSA_AGENT_INFO_QUEUE_TYPE	Type of a queue created in the agent. The type of this attribute is hsa_queue_type32_t.
HSA_AGENT_INFO_NODE	**Deprecated** NUMA information is not exposed anywhere else in the API. Identifier of the NUMA node associated with the agent. The type of this attribute is uint32_t.
HSA_AGENT_INFO_DEVICE	Type of hardware device associated with the agent. The type of this attribute is hsa_device_type_t.
HSA_AGENT_INFO_CACHE_SIZE	**Deprecated** Query hsa_agent_iterate_caches to retrieve information about the caches present in a given agent. Array of data cache sizes (L1..L4). Each size is expressed in bytes. A size of 0 for a particular level indicates that there is no cache information for that level. The type of this attribute is uint32_t[4].
HSA_AGENT_INFO_ISA	**Deprecated** An agent may support multiple instruction set architectures. See hsa_agent_iterate_isas. If more than one ISA is supported by the agent, the returned value corresponds to the first ISA enumerated by hsa_agent_iterate_isas. Instruction set architecture of the agent. The type of this attribute is hsa_isa_t.
HSA_AGENT_INFO_EXTENSIONS	Bit-mask indicating which extensions are supported by the agent. An extension with an ID of `i` is supported if the bit at position `i` is set. The type of this attribute is uint8_t[128].
HSA_AGENT_INFO_VERSION_MAJOR	Major version of the HSA runtime specification supported by the agent. The type of this attribute is uint16_t.

Enumerator

HSA_AGENT_INFO_VERSION_MINOR	Minor version of the HSA runtime specification supported by the agent. The type of this attribute is uint16_t.
HSA_AGENT_INFO_LAST	This enum does not have a fixed underlying type, thus in C++ post D2338: If the enumeration type does not have a fixed underlying type, the value is unchanged if the original value is within the range of the enumeration values (9.7.1 [dcl.enum]), and otherwise, the behavior is undefined. Thus increase the range of this enum to encompass vendor extensions.

Definition at line 806 of file hsa.h.

5.2.2.3 hsa_cache_info_t

`enum hsa_cache_info_t`

Cache attributes.

Enumerator

HSA_CACHE_INFO_NAME_LENGTH	The length of the cache name in bytes, not including the NUL terminator. The type of this attribute is uint32_t.
HSA_CACHE_INFO_NAME	Human-readable description. The type of this attribute is a NUL-terminated character array with the length equal to the value of HSA_CACHE_INFO_NAME_LENGTH attribute.
HSA_CACHE_INFO_LEVEL	Cache level. A L1 cache must return a value of 1, a L2 must return a value of 2, and so on. The type of this attribute is uint8_t.
HSA_CACHE_INFO_SIZE	Cache size, in bytes. A value of 0 indicates that there is no size information available. The type of this attribute is uint32_t.

Definition at line 1160 of file hsa.h.

5.2.2.4 hsa_default_float_rounding_mode_t

`enum hsa_default_float_rounding_mode_t`

Default floating-point rounding mode.

Enumerator

HSA_DEFAULT_FLOAT_ROUNDING_MODE_↩ DEFAULT	Use a default floating-point rounding mode specified elsewhere.
HSA_DEFAULT_FLOAT_ROUNDING_MODE_ZERO	Operations that specify the default floating-point mode are rounded to zero by default.
HSA_DEFAULT_FLOAT_ROUNDING_MODE_NEAR	Operations that specify the default floating-point mode are rounded to the nearest representable number and that ties should be broken by selecting the value with an even least significant bit.

Definition at line 785 of file hsa.h.

5.2.2.5 hsa_device_type_t

`enum hsa_device_type_t`

Hardware device type.

Enumerator

HSA_DEVICE_TYPE_CPU	CPU device.
HSA_DEVICE_TYPE_GPU	GPU device.
HSA_DEVICE_TYPE_DSP	DSP device.

Definition at line 767 of file hsa.h.

5.2.2.6 hsa_endianness_t

`enum hsa_endianness_t`

Endianness. A convention used to interpret the bytes making up a data word.

Enumerator

HSA_ENDIANNESS_LITTLE	The least significant byte is stored in the smallest address.
HSA_ENDIANNESS_BIG	The most significant byte is stored in the smallest address.

Definition at line 398 of file hsa.h.

5.2.2.7 hsa_exception_policy_t

`enum hsa_exception_policy_t`

Exception policies applied in the presence of hardware exceptions.

Enumerator

HSA_EXCEPTION_POLICY_BREAK	If a hardware exception is detected, a work-item signals an exception.
HSA_EXCEPTION_POLICY_DETECT	If a hardware exception is detected, a hardware status bit is set.

Definition at line 1103 of file hsa.h.

5.2.2.8 hsa_extension_t

enum hsa_extension_t

HSA extensions.

Enumerator

HSA_EXTENSION_FINALIZER	Finalizer extension.
HSA_EXTENSION_IMAGES	Images extension.
HSA_EXTENSION_PERFORMANCE_COUNTERS	Performance counter extension.
HSA_EXTENSION_PROFILING_EVENTS	Profiling events extension.
HSA_EXTENSION_STD_LAST	Extension count.
HSA_AMD_FIRST_EXTENSION	First AMD extension number.
HSA_EXTENSION_AMD_PROFILER	Profiler extension.
HSA_EXTENSION_AMD_LOADER	Loader extension.
HSA_EXTENSION_AMD_AQLPROFILE	AqlProfile extension.
HSA_AMD_LAST_EXTENSION	Last AMD extension.

Definition at line 529 of file hsa.h.

5.2.2.9 hsa_machine_model_t

enum hsa_machine_model_t

Machine model. A machine model determines the size of certain data types in HSA runtime and an agent.

Enumerator

HSA_MACHINE_MODEL_SMALL	Small machine model. Addresses use 32 bits.
HSA_MACHINE_MODEL_LARGE	Large machine model. Addresses use 64 bits.

Definition at line 413 of file hsa.h.

5.2.2.10 hsa_profile_t

enum hsa_profile_t

Profile. A profile indicates a particular level of feature support. For example, in the base profile the application must use the HSA runtime allocator to reserve shared virtual memory, while in the full profile any host pointer can be shared across all the agents.

Enumerator

HSA_PROFILE_BASE	Base profile.
HSA_PROFILE_FULL	Full profile.

Definition at line 430 of file hsa.h.

5.2.2.11 hsa_system_info_t

```
enum hsa_system_info_t
```

System attributes.

Enumerator

HSA_SYSTEM_INFO_VERSION_MAJOR	Major version of the HSA runtime specification supported by the implementation. The type of this attribute is uint16_t.
HSA_SYSTEM_INFO_VERSION_MINOR	Minor version of the HSA runtime specification supported by the implementation. The type of this attribute is uint16_t.
HSA_SYSTEM_INFO_TIMESTAMP	Current timestamp. The value of this attribute monotonically increases at a constant rate. The type of this attribute is uint64_t.
HSA_SYSTEM_INFO_TIMESTAMP_FREQUENCY	Timestamp value increase rate, in Hz. The timestamp (clock) frequency is in the range 1-400MHz. The type of this attribute is uint64_t.
HSA_SYSTEM_INFO_SIGNAL_MAX_WAIT	Maximum duration of a signal wait operation. Expressed as a count based on the timestamp frequency. The type of this attribute is uint64_t.
HSA_SYSTEM_INFO_ENDIANNESS	Endianness of the system. The type of this attribute is hsa_endianness_t.
HSA_SYSTEM_INFO_MACHINE_MODEL	Machine model supported by the HSA runtime. The type of this attribute is hsa_machine_model_t.
HSA_SYSTEM_INFO_EXTENSIONS	Bit-mask indicating which extensions are supported by the implementation. An extension with an ID of `i` is supported if the bit at position `i` is set. The type of this attribute is uint8_t[128].
HSA_AMD_SYSTEM_INFO_BUILD_VERSION	String containing the ROCr build identifier.
HSA_AMD_SYSTEM_INFO_SVM_SUPPORTED	Returns true if hsa_amd_svm_* APIs are supported by the driver. The type of this attribute is bool.
HSA_AMD_SYSTEM_INFO_SVM_ACCESSIBLE_↩ BY_DEFAULT	Returns true if all Agents have access to system allocated memory (such as that allocated by mmap, malloc, or new) by default. If false then system allocated memory may only be made SVM accessible to an Agent by declaration of accessibility with hsa_amd_svm_set_attributes. The type of this attribute is bool.

Definition at line 444 of file hsa.h.

5.2.3 Function Documentation

5.2.3.1 hsa_agent_extension_supported()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_agent_extension_supported (
            uint16_t extension,
            hsa_agent_t agent,
            uint16_t version_major,
            uint16_t version_minor,
            bool * result )
```

Query if a given version of an extension is supported by an agent.

Deprecated

Parameters

in	*extension*	Extension identifier.
in	*agent*	Agent.
in	*version_major*	Major version number.
in	*version_minor*	Minor version number.
out	*result*	Pointer to a memory location where the HSA runtime stores the result of the check. The result is true if the specified version of the extension is supported, and false otherwise. The result must be false if hsa_system_extension_supported returns false for the same extension version.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`extension` is not a valid extension, or `result` is NULL.

5.2.3.2 hsa_agent_get_exception_policies()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_agent_get_exception_policies (
            hsa_agent_t agent,
            hsa_profile_t profile,
            uint16_t * mask )
```

Retrieve the exception policy support for a given combination of agent and profile.

Deprecated Use hsa_isa_get_exception_policies for a given intruction set architecture supported by the agent instead. If more than one ISA is supported by the agent, this function uses the first value returned by hsa_agent_iterate_isas.

Parameters

in	*agent*	Agent.
in	*profile*	Profile.
out	*mask*	Pointer to a memory location where the HSA runtime stores a mask of hsa_exception_policy_t values. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`profile` is not a valid profile, or `mask` is NULL.

5.2.3.3 hsa_agent_get_info()

```
hsa_status_t HSA_API hsa_agent_get_info (
            hsa_agent_t agent,
            hsa_agent_info_t attribute,
            void * value )
```

Get the current value of an attribute for a given agent.

Parameters

in	*agent*	A valid agent.
in	*attribute*	Attribute to query.
out	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid agent attribute, or `value` is NULL.

5.2.3.4 hsa_agent_iterate_caches()

```
hsa_status_t HSA_API hsa_agent_iterate_caches (
            hsa_agent_t agent,
            hsa_status_t(*)(hsa_cache_t cache, void *data) callback,
            void * data )
```

Iterate over the memory caches of a given agent, and invoke an application-defined callback on every iteration.

Caches are visited in ascending order according to the value of the HSA_CACHE_INFO_LEVEL attribute.

Parameters

in	*agent*	A valid agent.
in	*callback*	Callback to be invoked once per cache that is present in the agent. The HSA runtime passes two arguments to the callback: the cache and the application data. If `callback` returns a status other than HSA_STATUS_SUCCESS for a particular iteration, the traversal stops and that value is returned.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`callback` is NULL.

5.2.3.5 hsa_agent_major_extension_supported()

```
hsa_status_t HSA_API hsa_agent_major_extension_supported (
            uint16_t extension,
            hsa_agent_t agent,
            uint16_t version_major,
            uint16_t * version_minor,
            bool * result )
```

Query if a given version of an extension is supported by an agent. All minor versions from 0 up to the returned `version_minor` must be supported.

Parameters

in	*extension*	Extension identifier.
in	*agent*	Agent.
in	*version_major*	Major version number.
out	*version_minor*	Minor version number.
out	*result*	Pointer to a memory location where the HSA runtime stores the result of the check. The result is true if the specified version of the extension is supported, and false otherwise. The result must be false if hsa_system_extension_supported returns false for the same extension version.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.

Return values

HSA_STATUS_ERROR_INVALID_ARGUMENT	`extension` is not a valid extension, or `version_minor` is NULL, or `result` is NULL.

5.2.3.6 hsa_cache_get_info()

```
hsa_status_t HSA_API hsa_cache_get_info (
            hsa_cache_t cache,
            hsa_cache_info_t attribute,
            void * value )
```

Get the current value of an attribute for a given cache object.

Parameters

`in`	*cache*	Cache.
`in`	*attribute*	Attribute to query.
`out`	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_CACHE	The cache is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid instruction set architecture attribute, or `value` is NULL.

5.2.3.7 hsa_extension_get_name()

```
hsa_status_t HSA_API hsa_extension_get_name (
            uint16_t extension,
            const char ** name )
```

Query the name of a given extension.

Parameters

`in`	*extension*	Extension identifier. If the extension is not supported by the implementation (see HSA_SYSTEM_INFO_EXTENSIONS), the behavior is undefined.
`out`	*name*	Pointer to a memory location where the HSA runtime stores the extension name. The extension name is a NUL-terminated string.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`extension` is not a valid extension, or `name` is NULL.

5.2.3.8 hsa_iterate_agents()

```
hsa_status_t HSA_API hsa_iterate_agents (
             hsa_status_t(*)(hsa_agent_t agent, void *data) callback,
             void * data )
```

Iterate over the available agents, and invoke an application-defined callback on every iteration.

Parameters

in	*callback*	Callback to be invoked once per agent. The HSA runtime passes two arguments to the callback: the agent and the application data. If `callback` returns a status other than HSA_STATUS_SUCCESS for a particular iteration, the traversal stops and hsa_iterate_agents returns that status value.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`callback` is NULL.

5.2.3.9 hsa_system_extension_supported()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_system_extension_supported (
             uint16_t extension,
             uint16_t version_major,
             uint16_t version_minor,
             bool * result )
```

Query if a given version of an extension is supported by the HSA implementation.

Deprecated

Parameters

in	*extension*	Extension identifier.
in	*version_major*	Major version number.
in	*version_minor*	Minor version number.
out	*result*	Pointer to a memory location where the HSA runtime stores the result of the check. The result is true if the specified version of the extension is supported, and false otherwise.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`extension` is not a valid extension, or `result` is NULL.

5.2.3.10 hsa_system_get_extension_table()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_system_get_extension_table (
            uint16_t extension,
            uint16_t version_major,
            uint16_t version_minor,
            void * table )
```

Retrieve the function pointers corresponding to a given version of an extension. Portable applications are expected to invoke the extension API using the returned function pointers.

Deprecated

The application is responsible for verifying that the given version of the extension is supported by the HSA implementation (see hsa_system_extension_supported). If the given combination of extension, major version, and minor version is not supported by the implementation, the behavior is undefined.

Parameters

`in`	*extension*	Extension identifier.
`in`	*version_major*	Major version number for which to retrieve the function pointer table.
`in`	*version_minor*	Minor version number for which to retrieve the function pointer table.
`out`	*table*	Pointer to an application-allocated function pointer table that is populated by the HSA runtime. Must not be NULL. The memory associated with table can be reused or freed after the function returns.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`extension` is not a valid extension, or `table` is NULL.

5.2.3.11 hsa_system_get_info()

```
hsa_status_t HSA_API hsa_system_get_info (
            hsa_system_info_t attribute,
            void * value )
```

Get the current value of a system attribute.

Parameters

in	*attribute*	Attribute to query.
out	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid system attribute, or `value` is NULL.

5.2.3.12 hsa_system_get_major_extension_table()

```
hsa_status_t HSA_API hsa_system_get_major_extension_table (
            uint16_t extension,
            uint16_t version_major,
            size_t table_length,
            void * table )
```

Retrieve the function pointers corresponding to a given major version of an extension. Portable applications are expected to invoke the extension API using the returned function pointers.

The application is responsible for verifying that the given major version of the extension is supported by the HSA implementation (see hsa_system_major_extension_supported). If the given combination of extension and major version is not supported by the implementation, the behavior is undefined. Additionally if the length doesn't allow space for a full minor version, it is implementation defined if only some of the function pointers for that minor version get written.

Parameters

in	*extension*	Extension identifier.
in	*version_major*	Major version number for which to retrieve the function pointer table.
in	*table_length*	Size in bytes of the function pointer table to be populated. The implementation will not write more than this many bytes to the table.
out	*table*	Pointer to an application-allocated function pointer table that is populated by the HSA runtime. Must not be NULL. The memory associated with table can be reused or freed after the function returns.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`extension` is not a valid extension, or `table` is NULL.

5.2.3.13 hsa_system_major_extension_supported()

```
hsa_status_t HSA_API hsa_system_major_extension_supported (
            uint16_t extension,
            uint16_t version_major,
            uint16_t * version_minor,
            bool * result )
```

Query if a given version of an extension is supported by the HSA implementation. All minor versions from 0 up to the returned `version_minor` must be supported by the implementation.

Parameters

`in`	*extension*	Extension identifier.
`in`	*version_major*	Major version number.
`out`	*version_minor*	Minor version number.
`out`	*result*	Pointer to a memory location where the HSA runtime stores the result of the check. The result is true if the specified version of the extension is supported, and false otherwise.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`extension` is not a valid extension, or `version_minor` is NULL, or `result` is NULL.

5.3 Signals

Classes

- struct hsa_signal_s

  Signal handle.
- struct hsa_signal_group_s

  Group of signals.

Typedefs

- typedef struct hsa_signal_s **hsa_signal_t**

  Signal handle.
- typedef int32_t hsa_signal_value_t

  Signal value. The value occupies 32 bits in small machine mode, and 64 bits in large machine mode.
- typedef struct hsa_signal_group_s **hsa_signal_group_t**

  Group of signals.

Enumerations

- enum hsa_signal_condition_t { HSA_SIGNAL_CONDITION_EQ = 0 , HSA_SIGNAL_CONDITION_NE = 1 , HSA_SIGNAL_CONDITION_LT = 2 , HSA_SIGNAL_CONDITION_GTE = 3 }

  Wait condition operator.
- enum hsa_wait_state_t { HSA_WAIT_STATE_BLOCKED = 0 , HSA_WAIT_STATE_ACTIVE = 1 }

  State of the application thread during a signal wait.

Functions

- hsa_status_t HSA_API hsa_signal_create (hsa_signal_value_t initial_value, uint32_t num_consumers, const hsa_agent_t *consumers, hsa_signal_t *signal)

  Create a signal.
- hsa_status_t HSA_API hsa_signal_destroy (hsa_signal_t signal)

  Destroy a signal previous created by hsa_signal_create.
- hsa_signal_value_t HSA_API hsa_signal_load_scacquire (hsa_signal_t signal)

  Atomically read the current value of a signal.
- hsa_signal_value_t HSA_API hsa_signal_load_relaxed (hsa_signal_t signal)

  Atomically read the current value of a signal.
- hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_load_acquire (hsa_signal_t signal)
- void HSA_API hsa_signal_store_relaxed (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically set the value of a signal.
- void HSA_API hsa_signal_store_screlease (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically set the value of a signal.
- void HSA_API HSA_DEPRECATED hsa_signal_store_release (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_silent_store_relaxed (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically set the value of a signal without necessarily notifying the the agents waiting on it.
- void HSA_API hsa_signal_silent_store_screlease (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically set the value of a signal without necessarily notifying the the agents waiting on it.
- hsa_signal_value_t HSA_API hsa_signal_exchange_scacq_screl (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically set the value of a signal and return its previous value.
- hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_exchange_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)
- hsa_signal_value_t HSA_API hsa_signal_exchange_scacquire (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically set the value of a signal and return its previous value.
- hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_exchange_acquire (hsa_signal_t signal, hsa_signal_value_t value)
- hsa_signal_value_t HSA_API hsa_signal_exchange_relaxed (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically set the value of a signal and return its previous value.
- hsa_signal_value_t HSA_API hsa_signal_exchange_screlease (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically set the value of a signal and return its previous value.
- hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_exchange_release (hsa_signal_t signal, hsa_signal_value_t value)
- hsa_signal_value_t HSA_API hsa_signal_cas_scacq_screl (hsa_signal_t signal, hsa_signal_value_t expected, hsa_signal_value_t value)

  Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.

- hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_cas_acq_rel (hsa_signal_t signal, hsa_signal_value_t expected, hsa_signal_value_t value)
- hsa_signal_value_t HSA_API hsa_signal_cas_scacquire (hsa_signal_t signal, hsa_signal_value_t expected, hsa_signal_value_t value)

  Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.
- hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_cas_acquire (hsa_signal_t signal, hsa_signal_value_t expected, hsa_signal_value_t value)
- hsa_signal_value_t HSA_API hsa_signal_cas_relaxed (hsa_signal_t signal, hsa_signal_value_t expected, hsa_signal_value_t value)

  Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.
- hsa_signal_value_t HSA_API hsa_signal_cas_screlease (hsa_signal_t signal, hsa_signal_value_t expected, hsa_signal_value_t value)

  Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.
- hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_cas_release (hsa_signal_t signal, hsa_signal_value_t expected, hsa_signal_value_t value)
- void HSA_API hsa_signal_add_scacq_screl (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically increment the value of a signal by a given amount.
- void HSA_API HSA_DEPRECATED hsa_signal_add_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_add_scacquire (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically increment the value of a signal by a given amount.
- void HSA_API HSA_DEPRECATED hsa_signal_add_acquire (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_add_relaxed (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically increment the value of a signal by a given amount.
- void HSA_API hsa_signal_add_screlease (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically increment the value of a signal by a given amount.
- void HSA_API HSA_DEPRECATED hsa_signal_add_release (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_subtract_scacq_screl (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically decrement the value of a signal by a given amount.
- void HSA_API HSA_DEPRECATED hsa_signal_subtract_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_subtract_scacquire (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically decrement the value of a signal by a given amount.
- void HSA_API HSA_DEPRECATED hsa_signal_subtract_acquire (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_subtract_relaxed (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically decrement the value of a signal by a given amount.
- void HSA_API hsa_signal_subtract_screlease (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically decrement the value of a signal by a given amount.
- void HSA_API HSA_DEPRECATED hsa_signal_subtract_release (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_and_scacq_screl (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise AND operation between the value of a signal and a given value.
- void HSA_API HSA_DEPRECATED hsa_signal_and_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_and_scacquire (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise AND operation between the value of a signal and a given value.
- void HSA_API HSA_DEPRECATED hsa_signal_and_acquire (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_and_relaxed (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise AND operation between the value of a signal and a given value.
- void HSA_API hsa_signal_and_screlease (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise AND operation between the value of a signal and a given value.

- void HSA_API HSA_DEPRECATED hsa_signal_and_release (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_or_scacq_screl (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise OR operation between the value of a signal and a given value.
- void HSA_API HSA_DEPRECATED hsa_signal_or_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_or_scacquire (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise OR operation between the value of a signal and a given value.
- void HSA_API HSA_DEPRECATED hsa_signal_or_acquire (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_or_relaxed (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise OR operation between the value of a signal and a given value.
- void HSA_API hsa_signal_or_screlease (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise OR operation between the value of a signal and a given value.
- void HSA_API HSA_DEPRECATED hsa_signal_or_release (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_xor_scacq_screl (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise XOR operation between the value of a signal and a given value.
- void HSA_API HSA_DEPRECATED hsa_signal_xor_acq_rel (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_xor_scacquire (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise XOR operation between the value of a signal and a given value.
- void HSA_API HSA_DEPRECATED hsa_signal_xor_acquire (hsa_signal_t signal, hsa_signal_value_t value)
- void HSA_API hsa_signal_xor_relaxed (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise XOR operation between the value of a signal and a given value.
- void HSA_API hsa_signal_xor_screlease (hsa_signal_t signal, hsa_signal_value_t value)

  Atomically perform a bitwise XOR operation between the value of a signal and a given value.
- void HSA_API HSA_DEPRECATED hsa_signal_xor_release (hsa_signal_t signal, hsa_signal_value_t value)
- hsa_signal_value_t HSA_API hsa_signal_wait_scacquire (hsa_signal_t signal, hsa_signal_condition_t condition, hsa_signal_value_t compare_value, uint64_t timeout_hint, hsa_wait_state_t wait_state_hint)

  Wait until a signal value satisfies a specified condition, or a certain amount of time has elapsed.
- hsa_signal_value_t HSA_API hsa_signal_wait_relaxed (hsa_signal_t signal, hsa_signal_condition_t condition, hsa_signal_value_t compare_value, uint64_t timeout_hint, hsa_wait_state_t wait_state_hint)

  Wait until a signal value satisfies a specified condition, or a certain amount of time has elapsed.
- hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_wait_acquire (hsa_signal_t signal, hsa_signal_condition_t condition, hsa_signal_value_t compare_value, uint64_t timeout_hint, hsa_wait_state_t wait_state_hint)
- hsa_status_t HSA_API hsa_signal_group_create (uint32_t num_signals, const hsa_signal_t *signals, uint32_t num_consumers, const hsa_agent_t *consumers, hsa_signal_group_t *signal_group)

  Create a signal group.
- hsa_status_t HSA_API hsa_signal_group_destroy (hsa_signal_group_t signal_group)

  Destroy a signal group previous created by hsa_signal_group_create.
- hsa_status_t HSA_API hsa_signal_group_wait_any_scacquire (hsa_signal_group_t signal_group, const hsa_signal_condition_t *conditions, const hsa_signal_value_t *compare_values, hsa_wait_state_t wait_state_hint, hsa_signal_t *signal, hsa_signal_value_t *value)

  Wait until the value of at least one of the signals in a signal group satisfies its associated condition.
- hsa_status_t HSA_API hsa_signal_group_wait_any_relaxed (hsa_signal_group_t signal_group, const hsa_signal_condition_t *conditions, const hsa_signal_value_t *compare_values, hsa_wait_state_t wait_state_hint, hsa_signal_t *signal, hsa_signal_value_t *value)

  Wait until the value of at least one of the signals in a signal group satisfies its associated condition.

5.3.1 Detailed Description

5.3.2 Typedef Documentation

5.3.2.1 hsa_signal_value_t

```
typedef int32_t hsa_signal_value_t
```

Signal value. The value occupies 32 bits in small machine mode, and 64 bits in large machine mode.

Definition at line 1340 of file hsa.h.

5.3.3 Enumeration Type Documentation

5.3.3.1 hsa_signal_condition_t

```
enum hsa_signal_condition_t
```

Wait condition operator.

Enumerator

HSA_SIGNAL_CONDITION_EQ	The two operands are equal.
HSA_SIGNAL_CONDITION_NE	The two operands are not equal.
HSA_SIGNAL_CONDITION_LT	The first operand is less than the second operand.
HSA_SIGNAL_CONDITION_GTE	The first operand is greater than or equal to the second operand.

Definition at line 1948 of file hsa.h.

5.3.3.2 hsa_wait_state_t

```
enum hsa_wait_state_t
```

State of the application thread during a signal wait.

Enumerator

HSA_WAIT_STATE_BLOCKED	The application thread may be rescheduled while waiting on the signal.
HSA_WAIT_STATE_ACTIVE	The application thread stays active while waiting on a signal.

Definition at line 1970 of file hsa.h.

5.3.4 Function Documentation

5.3.4.1 hsa_signal_add_acq_rel()

```
void HSA_API HSA_DEPRECATED hsa_signal_add_acq_rel (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_add_scacq_screl.

Atomically increment the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to add to the value of the signal.

5.3.4.2 hsa_signal_add_acquire()

```
void HSA_API HSA_DEPRECATED hsa_signal_add_acquire (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_add_scacquire.

Atomically increment the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to add to the value of the signal.

5.3.4.3 hsa_signal_add_relaxed()

```
void HSA_API hsa_signal_add_relaxed (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Atomically increment the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to add to the value of the signal.

5.3.4.4 hsa_signal_add_release()

```
void HSA_API HSA_DEPRECATED hsa_signal_add_release (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_add_screlease.

Atomically increment the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to add to the value of the signal.

5.3.4.5 hsa_signal_add_scacq_screl()

```
void HSA_API hsa_signal_add_scacq_screl (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Atomically increment the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to add to the value of the signal.

5.3.4.6 hsa_signal_add_scacquire()

```
void HSA_API hsa_signal_add_scacquire (
```

```
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically increment the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to add to the value of the signal.

5.3.4.7 hsa_signal_add_screlease()

```
void HSA_API hsa_signal_add_screlease (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically increment the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to add to the value of the signal.

5.3.4.8 hsa_signal_and_acq_rel()

```
void HSA_API HSA_DEPRECATED hsa_signal_and_acq_rel (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_and_scacq_screl.

Atomically perform a bitwise AND operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to AND with the value of the signal.

5.3.4.9 hsa_signal_and_acquire()

```
void HSA_API HSA_DEPRECATED hsa_signal_and_acquire (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_and_scacquire.

Atomically perform a bitwise AND operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to AND with the value of the signal.

5.3.4.10 hsa_signal_and_relaxed()

```
void HSA_API hsa_signal_and_relaxed (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically perform a bitwise AND operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to AND with the value of the signal.

5.3.4.11 hsa_signal_and_release()

```
void HSA_API HSA_DEPRECATED hsa_signal_and_release (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_and_screlease.

Atomically perform a bitwise AND operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to AND with the value of the signal.

5.3.4.12 hsa_signal_and_scacq_screl()

```
void HSA_API hsa_signal_and_scacq_screl (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Atomically perform a bitwise AND operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to AND with the value of the signal.

5.3.4.13 hsa_signal_and_scacquire()

```
void HSA_API hsa_signal_and_scacquire (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Atomically perform a bitwise AND operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to AND with the value of the signal.

5.3.4.14 hsa_signal_and_screlease()

```
void HSA_API hsa_signal_and_screlease (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically perform a bitwise AND operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to AND with the value of the signal.

5.3.4.15 hsa_signal_cas_acq_rel()

```
hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_cas_acq_rel (
            hsa_signal_t signal,
            hsa_signal_value_t expected,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_cas_scacq_screl.

Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*expected*	Value to compare with.
in	*value*	New value.

Returns

Observed value of the signal.

5.3.4.16 hsa_signal_cas_acquire()

```
hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_cas_acquire (
            hsa_signal_t signal,
            hsa_signal_value_t expected,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_cas_scacquire.

Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*expected*	Value to compare with.
in	*value*	New value.

Returns

Observed value of the signal.

5.3.4.17 hsa_signal_cas_relaxed()

```
hsa_signal_value_t HSA_API hsa_signal_cas_relaxed (
            hsa_signal_t signal,
            hsa_signal_value_t expected,
            hsa_signal_value_t value )
```

Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*expected*	Value to compare with.
in	*value*	New value.

Returns

Observed value of the signal.

5.3.4.18 hsa_signal_cas_release()

```
hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_cas_release (
            hsa_signal_t signal,
            hsa_signal_value_t expected,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_cas_screlease.

Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*expected*	Value to compare with.
in	*value*	New value.

Returns

Observed value of the signal.

5.3.4.19 hsa_signal_cas_scacq_screl()

```
hsa_signal_value_t HSA_API hsa_signal_cas_scacq_screl (
            hsa_signal_t signal,
            hsa_signal_value_t expected,
            hsa_signal_value_t value )
```

Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*expected*	Value to compare with.
in	*value*	New value.

Returns

Observed value of the signal.

5.3.4.20 hsa_signal_cas_scacquire()

```
hsa_signal_value_t HSA_API hsa_signal_cas_scacquire (
            hsa_signal_t signal,
```

```
            hsa_signal_value_t expected,
            hsa_signal_value_t value )
```

Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*expected*	Value to compare with.
in	*value*	New value.

Returns

Observed value of the signal.

5.3.4.21 hsa_signal_cas_screlease()

```
hsa_signal_value_t HSA_API hsa_signal_cas_screlease (
            hsa_signal_t signal,
            hsa_signal_value_t expected,
            hsa_signal_value_t value )
```

Atomically set the value of a signal if the observed value is equal to the expected value. The observed value is returned regardless of whether the replacement was done.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*expected*	Value to compare with.
in	*value*	New value.

Returns

Observed value of the signal.

5.3.4.22 hsa_signal_create()

```
hsa_status_t HSA_API hsa_signal_create (
            hsa_signal_value_t initial_value,
```

```
uint32_t num_consumers,
const hsa_agent_t * consumers,
hsa_signal_t * signal )
```

Create a signal.

Parameters

in	*initial_value*	Initial value of the signal.
in	*num_consumers*	Size of `consumers`. A value of 0 indicates that any agent might wait on the signal.
in	*consumers*	List of agents that might consume (wait on) the signal. If `num_consumers` is 0, this argument is ignored; otherwise, the HSA runtime might use the list to optimize the handling of the signal object. If an agent not listed in `consumers` waits on the returned signal, the behavior is undefined. The memory associated with `consumers` can be reused or freed after the function returns.
out	*signal*	Pointer to a memory location where the HSA runtime will store the newly created signal handle. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`signal` is NULL, `num_consumers` is greater than 0 but `consumers` is NULL, or `consumers` contains duplicates.

5.3.4.23 hsa_signal_destroy()

```
hsa_status_t HSA_API hsa_signal_destroy (
hsa_signal_t signal )
```

Destroy a signal previous created by hsa_signal_create.

Parameters

in	*signal*	Signal.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_SIGNAL	`signal` is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	The handle in `signal` is 0.

5.3.4.24 hsa_signal_exchange_acq_rel()

```
hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_exchange_acq_rel (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_exchange_scacq_screl.

Atomically set the value of a signal and return its previous value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	New value.

Returns

Value of the signal prior to the exchange.

5.3.4.25 hsa_signal_exchange_acquire()

```
hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_exchange_acquire (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_exchange_scacquire.

Atomically set the value of a signal and return its previous value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	New value.

Returns

Value of the signal prior to the exchange.

5.3.4.26 hsa_signal_exchange_relaxed()

```
hsa_signal_value_t HSA_API hsa_signal_exchange_relaxed (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically set the value of a signal and return its previous value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	New value.

Returns

Value of the signal prior to the exchange.

5.3.4.27 hsa_signal_exchange_release()

```
hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_exchange_release (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_exchange_screlease.

Atomically set the value of a signal and return its previous value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	New value.

Returns

Value of the signal prior to the exchange.

5.3.4.28 hsa_signal_exchange_scacq_screl()

```
hsa_signal_value_t HSA_API hsa_signal_exchange_scacq_screl (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically set the value of a signal and return its previous value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	New value.

Returns

Value of the signal prior to the exchange.

5.3.4.29 hsa_signal_exchange_scacquire()

```
hsa_signal_value_t HSA_API hsa_signal_exchange_scacquire (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically set the value of a signal and return its previous value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	New value.

Returns

Value of the signal prior to the exchange.

5.3.4.30 hsa_signal_exchange_screlease()

```
hsa_signal_value_t HSA_API hsa_signal_exchange_screlease (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically set the value of a signal and return its previous value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	New value.

Returns

Value of the signal prior to the exchange.

5.3.4.31 hsa_signal_group_create()

```
hsa_status_t HSA_API hsa_signal_group_create (
            uint32_t num_signals,
            const hsa_signal_t * signals,
            uint32_t num_consumers,
            const hsa_agent_t * consumers,
            hsa_signal_group_t * signal_group )
```

Create a signal group.

Parameters

in	*num_signals*	Number of elements in `signals`. Must not be 0.
in	*signals*	List of signals in the group. The list must not contain any repeated elements. Must not be NULL.
in	*num_consumers*	Number of elements in `consumers`. Must not be 0.
in	*consumers*	List of agents that might consume (wait on) the signal group. The list must not contain repeated elements, and must be a subset of the set of agents that are allowed to wait on all the signals in the group. If an agent not listed in `consumers` waits on the returned group, the behavior is undefined. The memory associated with `consumers` can be reused or freed after the function returns. Must not be NULL.
out	*signal_group*	Pointer to newly created signal group. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`num_signals` is 0, `signals` is NULL, `num_consumers` is 0, `consumers` is NULL, or `signal_group` is NULL.

5.3.4.32 hsa_signal_group_destroy()

```
hsa_status_t HSA_API hsa_signal_group_destroy (
            hsa_signal_group_t signal_group )
```

Destroy a signal group previous created by hsa_signal_group_create.

Parameters

in	*signal_group*	Signal group.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_SIGNAL_GROUP	`signal_group` is invalid.

5.3.4.33 hsa_signal_group_wait_any_relaxed()

```
hsa_status_t HSA_API hsa_signal_group_wait_any_relaxed (
            hsa_signal_group_t signal_group,
            const hsa_signal_condition_t * conditions,
            const hsa_signal_value_t * compare_values,
            hsa_wait_state_t wait_state_hint,
            hsa_signal_t * signal,
            hsa_signal_value_t * value )
```

Wait until the value of at least one of the signals in a signal group satisfies its associated condition.

The function is guaranteed to return if the value of at least one of the signals in the group satisfies its associated condition at some point in time during the wait, but the signal value returned to the application may no longer satisfy the condition. The application must ensure that signals in the group are used in such way that wait wakeup conditions are not invalidated before dependent threads have woken up.

When this operation internally loads the value of the passed signal, it uses the memory order indicated in the function name.

Parameters

in	*signal_group*	Signal group.
in	*conditions*	List of conditions. Each condition, and the value at the same index in `compare_values`, is used to compare the value of the signal at that index in `signal_group` (the signal passed by the application to hsa_signal_group_create at that particular index). The size of `conditions` must not be smaller than the number of signals in `signal_group`; any extra elements are ignored. Must not be NULL.
in	*compare_values*	List of comparison values. The size of `compare_values` must not be smaller than the number of signals in `signal_group`; any extra elements are ignored. Must not be NULL.
in	*wait_state_hint*	Hint used by the application to indicate the preferred waiting state. The actual waiting state is decided by the HSA runtime and may not match the provided hint. A value of HSA_WAIT_STATE_ACTIVE may improve the latency of response to a signal update by avoiding rescheduling overhead.
out	*signal*	Signal in the group that satisfied the associated condition. If several signals satisfied their condition, the function can return any of those signals. Must not be NULL.
out	*value*	Observed value for `signal`, which might no longer satisfy the specified condition. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_INVALID_SIGNAL_GROUP	`signal_group` is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`conditions` is NULL, `compare_values` is NULL, `signal` is NULL, or `value` is NULL.

5.3.4.34 hsa_signal_group_wait_any_scacquire()

```
hsa_status_t HSA_API hsa_signal_group_wait_any_scacquire (
            hsa_signal_group_t signal_group,
            const hsa_signal_condition_t * conditions,
            const hsa_signal_value_t * compare_values,
            hsa_wait_state_t wait_state_hint,
            hsa_signal_t * signal,
            hsa_signal_value_t * value )
```

Wait until the value of at least one of the signals in a signal group satisfies its associated condition.

The function is guaranteed to return if the value of at least one of the signals in the group satisfies its associated condition at some point in time during the wait, but the signal value returned to the application may no longer satisfy the condition. The application must ensure that signals in the group are used in such way that wait wakeup conditions are not invalidated before dependent threads have woken up.

When this operation internally loads the value of the passed signal, it uses the memory order indicated in the function name.

Parameters

`in`	*signal_group*	Signal group.
`in`	*conditions*	List of conditions. Each condition, and the value at the same index in `compare_values`, is used to compare the value of the signal at that index in `signal_group` (the signal passed by the application to hsa_signal_group_create at that particular index). The size of `conditions` must not be smaller than the number of signals in `signal_group`; any extra elements are ignored. Must not be NULL.
`in`	*compare_values*	List of comparison values. The size of `compare_values` must not be smaller than the number of signals in `signal_group`; any extra elements are ignored. Must not be NULL.
`in`	*wait_state_hint*	Hint used by the application to indicate the preferred waiting state. The actual waiting state is decided by the HSA runtime and may not match the provided hint. A value of HSA_WAIT_STATE_ACTIVE may improve the latency of response to a signal update by avoiding rescheduling overhead.
`out`	*signal*	Signal in the group that satisfied the associated condition. If several signals satisfied their condition, the function can return any of those signals. Must not be NULL.
`out`	*value*	Observed value for `signal`, which might no longer satisfy the specified condition. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_INVALID_SIGNAL_GROUP	`signal_group` is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`conditions` is NULL, `compare_values` is NULL, `signal` is NULL, or `value` is NULL.

5.3.4.35 hsa_signal_load_acquire()

```
hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_load_acquire (
            hsa_signal_t signal )
```

Deprecated Renamed as hsa_signal_load_scacquire.

Atomically read the current value of a signal.

Parameters

in	*signal*	Signal.

Returns

Value of the signal.

5.3.4.36 hsa_signal_load_relaxed()

```
hsa_signal_value_t HSA_API hsa_signal_load_relaxed (
            hsa_signal_t signal )
```

Atomically read the current value of a signal.

Parameters

in	*signal*	Signal.

Returns

Value of the signal.

5.3.4.37 hsa_signal_load_scacquire()

```
hsa_signal_value_t HSA_API hsa_signal_load_scacquire (
            hsa_signal_t signal )
```

Atomically read the current value of a signal.

Parameters

in	*signal*	Signal.

Returns

Value of the signal.

5.3.4.38 hsa_signal_or_acq_rel()

```
void HSA_API HSA_DEPRECATED hsa_signal_or_acq_rel (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_or_scacq_screl.

Atomically perform a bitwise OR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to OR with the value of the signal.

5.3.4.39 hsa_signal_or_acquire()

```
void HSA_API HSA_DEPRECATED hsa_signal_or_acquire (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_or_scacquire.

Atomically perform a bitwise OR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to OR with the value of the signal.

5.3.4.40 hsa_signal_or_relaxed()

```
void HSA_API hsa_signal_or_relaxed (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Atomically perform a bitwise OR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to OR with the value of the signal.

5.3.4.41 hsa_signal_or_release()

```
void HSA_API HSA_DEPRECATED hsa_signal_or_release (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_or_screlease.

Atomically perform a bitwise OR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to OR with the value of the signal.

5.3.4.42 hsa_signal_or_scacq_screl()

```
void HSA_API hsa_signal_or_scacq_screl (
```

```
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically perform a bitwise OR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to OR with the value of the signal.

5.3.4.43 hsa_signal_or_scacquire()

```
void HSA_API hsa_signal_or_scacquire (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically perform a bitwise OR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to OR with the value of the signal.

5.3.4.44 hsa_signal_or_screlease()

```
void HSA_API hsa_signal_or_screlease (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically perform a bitwise OR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to OR with the value of the signal.

5.3.4.45 hsa_signal_silent_store_relaxed()

```
void HSA_API hsa_signal_silent_store_relaxed (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Atomically set the value of a signal without necessarily notifying the the agents waiting on it.

The agents waiting on `signal` may not wake up even when the new value satisfies their wait condition. If the application wants to update the signal and there is no need to notify any agent, invoking this function can be more efficient than calling the non-silent counterpart.

Parameters

in	*signal*	Signal.
in	*value*	New signal value.

5.3.4.46 hsa_signal_silent_store_screlease()

```
void HSA_API hsa_signal_silent_store_screlease (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Atomically set the value of a signal without necessarily notifying the the agents waiting on it.

The agents waiting on `signal` may not wake up even when the new value satisfies their wait condition. If the application wants to update the signal and there is no need to notify any agent, invoking this function can be more efficient than calling the non-silent counterpart.

Parameters

in	*signal*	Signal.
in	*value*	New signal value.

5.3.4.47 hsa_signal_store_relaxed()

```
void HSA_API hsa_signal_store_relaxed (
             hsa_signal_t signal,
             hsa_signal_value_t value )
```

Atomically set the value of a signal.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal.
in	*value*	New signal value.

5.3.4.48 hsa_signal_store_release()

```
void HSA_API HSA_DEPRECATED hsa_signal_store_release (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_store_screlease.

Atomically set the value of a signal.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal.
in	*value*	New signal value.

5.3.4.49 hsa_signal_store_screlease()

```
void HSA_API hsa_signal_store_screlease (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically set the value of a signal.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal.
in	*value*	New signal value.

5.3.4.50 hsa_signal_subtract_acq_rel()

```
void HSA_API HSA_DEPRECATED hsa_signal_subtract_acq_rel (
```

```
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_subtract_scacq_screl.

Atomically decrement the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to subtract from the value of the signal.

5.3.4.51 hsa_signal_subtract_acquire()

```
void HSA_API HSA_DEPRECATED hsa_signal_subtract_acquire (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_subtract_scacquire.

Atomically decrement the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to subtract from the value of the signal.

5.3.4.52 hsa_signal_subtract_relaxed()

```
void HSA_API hsa_signal_subtract_relaxed (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically decrement the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to subtract from the value of the signal.

5.3.4.53 hsa_signal_subtract_release()

```
void HSA_API HSA_DEPRECATED hsa_signal_subtract_release (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_subtract_screlease.

Atomically decrement the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to subtract from the value of the signal.

5.3.4.54 hsa_signal_subtract_scacq_screl()

```
void HSA_API hsa_signal_subtract_scacq_screl (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically decrement the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to subtract from the value of the signal.

5.3.4.55 hsa_signal_subtract_scacquire()

```
void HSA_API hsa_signal_subtract_scacquire (
```

```
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically decrement the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to subtract from the value of the signal.

5.3.4.56 hsa_signal_subtract_screlease()

```
void HSA_API hsa_signal_subtract_screlease (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically decrement the value of a signal by a given amount.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to subtract from the value of the signal.

5.3.4.57 hsa_signal_wait_acquire()

```
hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_wait_acquire (
            hsa_signal_t signal,
            hsa_signal_condition_t condition,
            hsa_signal_value_t compare_value,
            uint64_t timeout_hint,
            hsa_wait_state_t wait_state_hint )
```

Deprecated Renamed as hsa_signal_wait_scacquire.

Wait until a signal value satisfies a specified condition, or a certain amount of time has elapsed.

A wait operation can spuriously resume at any time sooner than the timeout (for example, due to system or other external factors) even when the condition has not been met.

The function is guaranteed to return if the signal value satisfies the condition at some point in time during the wait, but the value returned to the application might not satisfy the condition. The application must ensure that signals are used in such way that wait wakeup conditions are not invalidated before dependent threads have woken up.

When the wait operation internally loads the value of the passed signal, it uses the memory order indicated in the function name.

Parameters

in	*signal*	Signal.
in	*condition*	Condition used to compare the signal value with `compare_value`.
in	*compare_value*	Value to compare with.
in	*timeout_hint*	Maximum duration of the wait. Specified in the same unit as the system timestamp. The operation might block for a shorter or longer time even if the condition is not met. A value of UINT64_MAX indicates no maximum.
in	*wait_state_hint*	Hint used by the application to indicate the preferred waiting state. The actual waiting state is ultimately decided by HSA runtime and may not match the provided hint. A value of HSA_WAIT_STATE_ACTIVE may improve the latency of response to a signal update by avoiding rescheduling overhead.

Returns

Observed value of the signal, which might not satisfy the specified condition.

5.3.4.58 hsa_signal_wait_relaxed()

```
hsa_signal_value_t HSA_API hsa_signal_wait_relaxed (
            hsa_signal_t signal,
            hsa_signal_condition_t condition,
            hsa_signal_value_t compare_value,
            uint64_t timeout_hint,
            hsa_wait_state_t wait_state_hint )
```

Wait until a signal value satisfies a specified condition, or a certain amount of time has elapsed.

A wait operation can spuriously resume at any time sooner than the timeout (for example, due to system or other external factors) even when the condition has not been met.

The function is guaranteed to return if the signal value satisfies the condition at some point in time during the wait, but the value returned to the application might not satisfy the condition. The application must ensure that signals are used in such way that wait wakeup conditions are not invalidated before dependent threads have woken up.

When the wait operation internally loads the value of the passed signal, it uses the memory order indicated in the function name.

Parameters

in	*signal*	Signal.
in	*condition*	Condition used to compare the signal value with `compare_value`.
in	*compare_value*	Value to compare with.
in	*timeout_hint*	Maximum duration of the wait. Specified in the same unit as the system timestamp. The operation might block for a shorter or longer time even if the condition is not met. A value of UINT64_MAX indicates no maximum.
in	*wait_state_hint*	Hint used by the application to indicate the preferred waiting state. The actual waiting state is ultimately decided by HSA runtime and may not match the provided hint. A value of HSA_WAIT_STATE_ACTIVE may improve the latency of response to a signal update by avoiding rescheduling overhead.

Returns

Observed value of the signal, which might not satisfy the specified condition.

5.3.4.59 hsa_signal_wait_scacquire()

```
hsa_signal_value_t HSA_API hsa_signal_wait_scacquire (
            hsa_signal_t signal,
            hsa_signal_condition_t condition,
            hsa_signal_value_t compare_value,
            uint64_t timeout_hint,
            hsa_wait_state_t wait_state_hint )
```

Wait until a signal value satisfies a specified condition, or a certain amount of time has elapsed.

A wait operation can spuriously resume at any time sooner than the timeout (for example, due to system or other external factors) even when the condition has not been met.

The function is guaranteed to return if the signal value satisfies the condition at some point in time during the wait, but the value returned to the application might not satisfy the condition. The application must ensure that signals are used in such way that wait wakeup conditions are not invalidated before dependent threads have woken up.

When the wait operation internally loads the value of the passed signal, it uses the memory order indicated in the function name.

Parameters

in	*signal*	Signal.
in	*condition*	Condition used to compare the signal value with `compare_value`.
in	*compare_value*	Value to compare with.
in	*timeout_hint*	Maximum duration of the wait. Specified in the same unit as the system timestamp. The operation might block for a shorter or longer time even if the condition is not met. A value of UINT64_MAX indicates no maximum.
in	*wait_state_hint*	Hint used by the application to indicate the preferred waiting state. The actual waiting state is ultimately decided by HSA runtime and may not match the provided hint. A value of HSA_WAIT_STATE_ACTIVE may improve the latency of response to a signal update by avoiding rescheduling overhead.

Returns

Observed value of the signal, which might not satisfy the specified condition.

5.3.4.60 hsa_signal_xor_acq_rel()

```
void HSA_API HSA_DEPRECATED hsa_signal_xor_acq_rel (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_xor_scacq_screl.

Atomically perform a bitwise XOR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to XOR with the value of the signal.

5.3.4.61 hsa_signal_xor_acquire()

```
void HSA_API HSA_DEPRECATED hsa_signal_xor_acquire (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_xor_scacquire.

Atomically perform a bitwise XOR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to XOR with the value of the signal.

5.3.4.62 hsa_signal_xor_relaxed()

```
void HSA_API hsa_signal_xor_relaxed (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically perform a bitwise XOR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to XOR with the value of the signal.

5.3.4.63 hsa_signal_xor_release()

```
void HSA_API HSA_DEPRECATED hsa_signal_xor_release (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Deprecated Renamed as hsa_signal_xor_screlease.

Atomically perform a bitwise XOR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to XOR with the value of the signal.

5.3.4.64 hsa_signal_xor_scacq_screl()

```
void HSA_API hsa_signal_xor_scacq_screl (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically perform a bitwise XOR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to XOR with the value of the signal.

5.3.4.65 hsa_signal_xor_scacquire()

```
void HSA_API hsa_signal_xor_scacquire (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically perform a bitwise XOR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to XOR with the value of the signal.

5.3.4.66 hsa_signal_xor_screlease()

```
void HSA_API hsa_signal_xor_screlease (
            hsa_signal_t signal,
            hsa_signal_value_t value )
```

Atomically perform a bitwise XOR operation between the value of a signal and a given value.

If the value of the signal is changed, all the agents waiting on `signal` for which `value` satisfies their wait condition are awakened.

Parameters

in	*signal*	Signal. If `signal` is a queue doorbell signal, the behavior is undefined.
in	*value*	Value to XOR with the value of the signal.

5.4 Memory

Classes

- struct hsa_region_s

  A memory region represents a block of virtual memory with certain properties. For example, the HSA runtime represents fine-grained memory in the global segment using a region. A region might be associated with more than one agent.

Typedefs

- typedef struct hsa_region_s **hsa_region_t**

  A memory region represents a block of virtual memory with certain properties. For example, the HSA runtime represents fine-grained memory in the global segment using a region. A region might be associated with more than one agent.

Enumerations

- enum hsa_region_segment_t {
  HSA_REGION_SEGMENT_GLOBAL = 0 , HSA_REGION_SEGMENT_READONLY = 1 , HSA_REGION_SEGMENT_PRIVATE
  = 2 , HSA_REGION_SEGMENT_GROUP = 3 ,
  HSA_REGION_SEGMENT_KERNARG = 4 }

  Memory segments associated with a region.

- enum hsa_region_global_flag_t { HSA_REGION_GLOBAL_FLAG_KERNARG = 1 , HSA_REGION_GLOBAL_FLAG_FINE_GR = 2 , HSA_REGION_GLOBAL_FLAG_COARSE_GRAINED = 4 }

  Global region flags.
- enum hsa_region_info_t { HSA_REGION_INFO_SEGMENT = 0 , HSA_REGION_INFO_GLOBAL_FLAGS = 1 , HSA_REGION_INFO_SIZE = 2 , HSA_REGION_INFO_ALLOC_MAX_SIZE = 4 , HSA_REGION_INFO_ALLOC_MAX_PRIVATE_WORKGROUP_SIZE = 8 , HSA_REGION_INFO_RUNTIME_ALLOC_ALLOWE = 5 , HSA_REGION_INFO_RUNTIME_ALLOC_GRANULE = 6 , HSA_REGION_INFO_RUNTIME_ALLOC_ALIGNMENT = 7 }

  Attributes of a memory region.

Functions

- hsa_status_t HSA_API hsa_region_get_info (hsa_region_t region, hsa_region_info_t attribute, void ∗value)

  Get the current value of an attribute of a region.
- hsa_status_t HSA_API hsa_agent_iterate_regions (hsa_agent_t agent, hsa_status_t(∗callback)(hsa_region_t region, void ∗data), void ∗data)

  Iterate over the memory regions associated with a given agent, and invoke an application-defined callback on every iteration.
- hsa_status_t HSA_API hsa_memory_allocate (hsa_region_t region, size_t size, void ∗∗ptr)

  Allocate a block of memory in a given region.
- hsa_status_t HSA_API hsa_memory_free (void ∗ptr)

  Deallocate a block of memory previously allocated using hsa_memory_allocate.
- hsa_status_t HSA_API hsa_memory_copy (void ∗dst, const void ∗src, size_t size)

  Copy a block of memory from the location pointed to by `src` to the memory block pointed to by `dst`.
- hsa_status_t HSA_API hsa_memory_assign_agent (void ∗ptr, hsa_agent_t agent, hsa_access_permission_t access)

  Change the ownership of a global, coarse-grained buffer.
- hsa_status_t HSA_API hsa_memory_register (void ∗ptr, size_t size)

  Register a global, fine-grained buffer.
- hsa_status_t HSA_API hsa_memory_deregister (void ∗ptr, size_t size)

  Deregister memory previously registered using hsa_memory_register.

5.4.1 Detailed Description

5.4.2 Enumeration Type Documentation

5.4.2.1 hsa_region_global_flag_t

`enum hsa_region_global_flag_t`

Global region flags.

Enumerator

HSA_REGION_GLOBAL_FLAG_KERNARG	The application can use memory in the region to store kernel arguments, and provide the values for the kernarg segment of a kernel dispatch. If this flag is set, then HSA_REGION_GLOBAL_FLAG_FINE_GRAINED must be set.
HSA_REGION_GLOBAL_FLAG_FINE_GRAINED	Updates to memory in this region are immediately visible to all the agents under the terms of the HSA memory model. If this flag is set, then HSA_REGION_GLOBAL_FLAG_COARSE_GRAINED must not be set.
HSA_REGION_GLOBAL_FLAG_COARSE_↩ GRAINED	Updates to memory in this region can be performed by a single agent at a time. If a different agent in the system is allowed to access the region, the application must explicitely invoke hsa_memory_assign_agent in order to transfer ownership to that agent for a particular buffer.

Definition at line 3197 of file hsa.h.

5.4.2.2 hsa_region_info_t

```
enum hsa_region_info_t
```

Attributes of a memory region.

Enumerator

HSA_REGION_INFO_SEGMENT	Segment where memory in the region can be used. The type of this attribute is hsa_region_segment_t.
HSA_REGION_INFO_GLOBAL_FLAGS	Flag mask. The value of this attribute is undefined if the value of HSA_REGION_INFO_SEGMENT is not HSA_REGION_SEGMENT_GLOBAL. The type of this attribute is uint32_t, a bit-field of hsa_region_global_flag_t values.
HSA_REGION_INFO_SIZE	Size of this region, in bytes. The type of this attribute is size_t.
HSA_REGION_INFO_ALLOC_MAX_SIZE	Maximum allocation size in this region, in bytes. Must not exceed the value of HSA_REGION_INFO_SIZE. The type of this attribute is size_t. If the region is in the global or readonly segments, this is the maximum size that the application can pass to hsa_memory_allocate. If the region is in the group segment, this is the maximum size (per work-group) that can be requested for a given kernel dispatch. If the region is in the private segment, this is the maximum size (per work-item) that can be requested for a specific kernel dispatch, and must be at least 256 bytes.

Enumerator

HSA_REGION_INFO_ALLOC_MAX_PRIVATE_↩ WORKGROUP_SIZE	Maximum size (per work-group) of private memory that can be requested for a specific kernel dispatch. Must be at least 65536 bytes. The type of this attribute is uint32_t. The value of this attribute is undefined if the region is not in the private segment.
HSA_REGION_INFO_RUNTIME_ALLOC_ALLOWED	Indicates whether memory in this region can be allocated using hsa_memory_allocate. The type of this attribute is bool. The value of this flag is always false for regions in the group and private segments.
HSA_REGION_INFO_RUNTIME_ALLOC_GRANULE	Allocation granularity of buffers allocated by hsa_memory_allocate in this region. The size of a buffer allocated in this region is a multiple of the value of this attribute. The value of this attribute is only defined if HSA_REGION_INFO_RUNTIME_ALLOC_ALLOWED is true for this region. The type of this attribute is size_t.
HSA_REGION_INFO_RUNTIME_ALLOC_↩ ALIGNMENT	Alignment of buffers allocated by hsa_memory_allocate in this region. The value of this attribute is only defined if HSA_REGION_INFO_RUNTIME_ALLOC_ALLOWED is true for this region, and must be a power of 2. The type of this attribute is size_t.

Definition at line 3222 of file hsa.h.

5.4.2.3 hsa_region_segment_t

```
enum hsa_region_segment_t
```

Memory segments associated with a region.

Enumerator

HSA_REGION_SEGMENT_GLOBAL	Global segment. Used to hold data that is shared by all agents.
HSA_REGION_SEGMENT_READONLY	Read-only segment. Used to hold data that remains constant during the execution of a kernel.
HSA_REGION_SEGMENT_PRIVATE	Private segment. Used to hold data that is local to a single work-item.
HSA_REGION_SEGMENT_GROUP	Group segment. Used to hold data that is shared by the work-items of a work-group.
HSA_REGION_SEGMENT_KERNARG	Kernarg segment. Used to store kernel arguments.

Definition at line 3169 of file hsa.h.

5.4.3 Function Documentation

5.4.3.1 hsa_agent_iterate_regions()

```
hsa_status_t HSA_API hsa_agent_iterate_regions (
             hsa_agent_t agent,
             hsa_status_t(*)(hsa_region_t region, void *data) callback,
             void * data )
```

Iterate over the memory regions associated with a given agent, and invoke an application-defined callback on every iteration.

Parameters

in	*agent*	A valid agent.
in	*callback*	Callback to be invoked once per region that is accessible from the agent. The HSA runtime passes two arguments to the callback, the region and the application data. If `callback` returns a status other than HSA_STATUS_SUCCESS for a particular iteration, the traversal stops and hsa_agent_iterate_regions returns that status value.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`callback` is NULL.

5.4.3.2 hsa_memory_allocate()

```
hsa_status_t HSA_API hsa_memory_allocate (
             hsa_region_t region,
             size_t size,
             void ** ptr )
```

Allocate a block of memory in a given region.

Parameters

in	*region*	Region where to allocate memory from. The region must have the HSA_REGION_INFO_RUNTIME_ALLOC_ALLOWED flag set.
in	*size*	Allocation size, in bytes. Must not be zero. This value is rounded up to the nearest multiple of HSA_REGION_INFO_RUNTIME_ALLOC_GRANULE in `region`.
out	*ptr*	Pointer to the location where to store the base address of the allocated block. The returned base address is aligned to the value of HSA_REGION_INFO_RUNTIME_ALLOC_ALIGNMENT in `region`. If the allocation fails, the returned value is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_REGION	The region is invalid.
HSA_STATUS_ERROR_INVALID_ALLOCATION	The host is not allowed to allocate memory in `region`, or `size` is greater than the value of HSA_REGION_INFO_ALLOC_MAX_SIZE in `region`.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`ptr` is NULL, or `size` is 0.

5.4.3.3 hsa_memory_assign_agent()

```
hsa_status_t HSA_API hsa_memory_assign_agent (
             void * ptr,
             hsa_agent_t agent,
             hsa_access_permission_t access )
```

Change the ownership of a global, coarse-grained buffer.

The contents of a coarse-grained buffer are visible to an agent only after ownership has been explicitely transferred to that agent. Once the operation completes, the previous owner cannot longer access the data in the buffer.

An implementation of the HSA runtime is allowed, but not required, to change the physical location of the buffer when ownership is transferred to a different agent. In general the application must not assume this behavior. The virtual location (address) of the passed buffer is never modified.

Parameters

`in`	*ptr*	Base address of a global buffer. The pointer must match an address previously returned by hsa_memory_allocate. The size of the buffer affected by the ownership change is identical to the size of that previous allocation. If `ptr` points to a fine-grained global buffer, no operation is performed and the function returns success. If `ptr` does not point to global memory, the behavior is undefined.
`in`	*agent*	Agent that becomes the owner of the buffer. The application is responsible for ensuring that `agent` has access to the region that contains the buffer. It is allowed to change ownership to an agent that is already the owner of the buffer, with the same or different access permissions.
`in`	*access*	Access permissions requested for the new owner.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`ptr` is NULL, or `access` is not a valid access value.

5.4.3.4 hsa_memory_copy()

```
hsa_status_t HSA_API hsa_memory_copy (
            void * dst,
            const void * src,
            size_t size )
```

Copy a block of memory from the location pointed to by `src` to the memory block pointed to by `dst`.

Parameters

`out`	*dst*	Buffer where the content is to be copied. If `dst` is in coarse-grained memory, the copied data is only visible to the agent currently assigned (hsa_memory_assign_agent) to `dst`.
`in`	*src*	A valid pointer to the source of data to be copied. The source buffer must not overlap with the destination buffer. If the source buffer is in coarse-grained memory then it must be assigned to an agent, from which the data will be retrieved.
`in`	*size*	Number of bytes to copy. If `size` is 0, no copy is performed and the function returns success. Copying a number of bytes larger than the size of the buffers pointed by `dst` or `src` results in undefined behavior.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ARGUMENT	The source or destination pointers are NULL.

5.4.3.5 hsa_memory_deregister()

```
hsa_status_t HSA_API hsa_memory_deregister (
            void * ptr,
            size_t size )
```

Deregister memory previously registered using hsa_memory_register.

If the memory interval being deregistered does not match a previous registration (start and end addresses), the behavior is undefined.

Parameters

`in`	*ptr*	A pointer to the base of the buffer to be deregistered. If a NULL pointer is passed, no operation is performed.
`in`	*size*	Size of the buffer to be deregistered.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.

5.4.3.6 hsa_memory_free()

```
hsa_status_t HSA_API hsa_memory_free (
            void * ptr )
```

Deallocate a block of memory previously allocated using hsa_memory_allocate.

Parameters

in	*ptr*	Pointer to a memory block. If `ptr` does not match a value previously returned by hsa_memory_allocate, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.

5.4.3.7 hsa_memory_register()

```
hsa_status_t HSA_API hsa_memory_register (
            void * ptr,
            size_t size )
```

Register a global, fine-grained buffer.

Registering a buffer serves as an indication to the HSA runtime that the memory might be accessed from a kernel agent other than the host. Registration is a performance hint that allows the HSA runtime implementation to know which buffers will be accessed by some of the kernel agents ahead of time.

Registration is only recommended for buffers in the global segment that have not been allocated using the HSA allocator (hsa_memory_allocate), but an OS allocator instead. Registering an OS-allocated buffer in the base profile is equivalent to a no-op.

Registrations should not overlap.

Parameters

in	*ptr*	A buffer in global, fine-grained memory. If a NULL pointer is passed, no operation is performed. If the buffer has been allocated using hsa_memory_allocate, or has already been registered, no operation is performed.
in	*size*	Requested registration size in bytes. A size of 0 is only allowed if `ptr` is NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.

Return values

HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`size` is 0 but `ptr` is not NULL.

5.4.3.8 hsa_region_get_info()

```
hsa_status_t HSA_API hsa_region_get_info (
            hsa_region_t region,
            hsa_region_info_t attribute,
            void * value )
```

Get the current value of an attribute of a region.

Parameters

`in`	*region*	A valid region.
`in`	*attribute*	Attribute to query.
`out`	*value*	Pointer to a application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_REGION	The region is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid region attribute, or `value` is NULL.

5.5 Queues

Classes

- struct hsa_queue_s

  User mode queue.

Typedefs

- typedef uint32_t hsa_queue_type32_t

  A fixed-size type used to represent hsa_queue_type_t constants.
- typedef struct hsa_queue_s hsa_queue_t

  User mode queue.

Enumerations

- enum hsa_queue_type_t { HSA_QUEUE_TYPE_MULTI = 0 , HSA_QUEUE_TYPE_SINGLE = 1 , HSA_QUEUE_TYPE_COOPERATIVE = 2 }

  Queue type. Intended to be used for dynamic queue protocol determination.
- enum hsa_queue_feature_t { HSA_QUEUE_FEATURE_KERNEL_DISPATCH = 1 , HSA_QUEUE_FEATURE_AGENT_DISPATC = 2 }

  Queue features.

Functions

- hsa_status_t HSA_API hsa_queue_create (hsa_agent_t agent, uint32_t size, hsa_queue_type32_t type, void(*callback)(hsa_status_t status, hsa_queue_t *source, void *data), void *data, uint32_t private_↩ segment_size, uint32_t group_segment_size, hsa_queue_t **queue)

  Create a user mode queue.
- hsa_status_t HSA_API hsa_soft_queue_create (hsa_region_t region, uint32_t size, hsa_queue_type32_t type, uint32_t features, hsa_signal_t doorbell_signal, hsa_queue_t **queue)

  Create a queue for which the application or a kernel is responsible for processing the AQL packets.
- hsa_status_t HSA_API hsa_queue_destroy (hsa_queue_t *queue)

  Destroy a user mode queue.
- hsa_status_t HSA_API hsa_queue_inactivate (hsa_queue_t *queue)

  Inactivate a queue.
- uint64_t HSA_API HSA_DEPRECATED hsa_queue_load_read_index_acquire (const hsa_queue_t *queue)
- uint64_t HSA_API hsa_queue_load_read_index_scacquire (const hsa_queue_t *queue)

  Atomically load the read index of a queue.
- uint64_t HSA_API hsa_queue_load_read_index_relaxed (const hsa_queue_t *queue)

  Atomically load the read index of a queue.
- uint64_t HSA_API HSA_DEPRECATED hsa_queue_load_write_index_acquire (const hsa_queue_t *queue)
- uint64_t HSA_API hsa_queue_load_write_index_scacquire (const hsa_queue_t *queue)

  Atomically load the write index of a queue.
- uint64_t HSA_API hsa_queue_load_write_index_relaxed (const hsa_queue_t *queue)

  Atomically load the write index of a queue.
- void HSA_API hsa_queue_store_write_index_relaxed (const hsa_queue_t *queue, uint64_t value)

  Atomically set the write index of a queue.
- void HSA_API HSA_DEPRECATED hsa_queue_store_write_index_release (const hsa_queue_t *queue, uint64_t value)
- void HSA_API hsa_queue_store_write_index_screlease (const hsa_queue_t *queue, uint64_t value)

  Atomically set the write index of a queue.
- uint64_t HSA_API HSA_DEPRECATED hsa_queue_cas_write_index_acq_rel (const hsa_queue_t *queue, uint64_t expected, uint64_t value)
- uint64_t HSA_API hsa_queue_cas_write_index_scacq_screl (const hsa_queue_t *queue, uint64_t expected, uint64_t value)

  Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.
- uint64_t HSA_API HSA_DEPRECATED hsa_queue_cas_write_index_acquire (const hsa_queue_t *queue, uint64_t expected, uint64_t value)
- uint64_t HSA_API hsa_queue_cas_write_index_scacquire (const hsa_queue_t *queue, uint64_t expected, uint64_t value)

  Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.
- uint64_t HSA_API hsa_queue_cas_write_index_relaxed (const hsa_queue_t *queue, uint64_t expected, uint64_t value)

Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.

- uint64_t HSA_API HSA_DEPRECATED hsa_queue_cas_write_index_release (const hsa_queue_t *queue, uint64_t expected, uint64_t value)
- uint64_t HSA_API hsa_queue_cas_write_index_screlease (const hsa_queue_t *queue, uint64_t expected, uint64_t value)

  Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.
- uint64_t HSA_API HSA_DEPRECATED hsa_queue_add_write_index_acq_rel (const hsa_queue_t *queue, uint64_t value)
- uint64_t HSA_API hsa_queue_add_write_index_scacq_screl (const hsa_queue_t *queue, uint64_t value)

  Atomically increment the write index of a queue by an offset.
- uint64_t HSA_API HSA_DEPRECATED hsa_queue_add_write_index_acquire (const hsa_queue_t *queue, uint64_t value)
- uint64_t HSA_API hsa_queue_add_write_index_scacquire (const hsa_queue_t *queue, uint64_t value)

  Atomically increment the write index of a queue by an offset.
- uint64_t HSA_API hsa_queue_add_write_index_relaxed (const hsa_queue_t *queue, uint64_t value)

  Atomically increment the write index of a queue by an offset.
- uint64_t HSA_API HSA_DEPRECATED hsa_queue_add_write_index_release (const hsa_queue_t *queue, uint64_t value)
- uint64_t HSA_API hsa_queue_add_write_index_screlease (const hsa_queue_t *queue, uint64_t value)

  Atomically increment the write index of a queue by an offset.
- void HSA_API hsa_queue_store_read_index_relaxed (const hsa_queue_t *queue, uint64_t value)

  Atomically set the read index of a queue.
- void HSA_API HSA_DEPRECATED hsa_queue_store_read_index_release (const hsa_queue_t *queue, uint64_t value)
- void HSA_API hsa_queue_store_read_index_screlease (const hsa_queue_t *queue, uint64_t value)

  Atomically set the read index of a queue.

5.5.1 Detailed Description

5.5.2 Typedef Documentation

5.5.2.1 hsa_queue_t

```
typedef struct hsa_queue_s hsa_queue_t
```

User mode queue.

The queue structure is read-only and allocated by the HSA runtime, but agents can directly modify the contents of the buffer pointed by *base_address*, or use HSA runtime APIs to access the doorbell signal.

5.5.2.2 hsa_queue_type32_t

```
typedef uint32_t hsa_queue_type32_t
```

A fixed-size type used to represent hsa_queue_type_t constants.

Definition at line 2242 of file hsa.h.

5.5.3 Enumeration Type Documentation

5.5.3.1 hsa_queue_feature_t

`enum hsa_queue_feature_t`

Queue features.

Enumerator

HSA_QUEUE_FEATURE_KERNEL_DISPATCH	Queue supports kernel dispatch packets.
HSA_QUEUE_FEATURE_AGENT_DISPATCH	Queue supports agent dispatch packets.

Definition at line 2247 of file hsa.h.

5.5.3.2 hsa_queue_type_t

`enum hsa_queue_type_t`

Queue type. Intended to be used for dynamic queue protocol determination.

Enumerator

HSA_QUEUE_TYPE_MULTI	Queue supports multiple producers. Use of multiproducer queue mechanics is required.
HSA_QUEUE_TYPE_SINGLE	Queue only supports a single producer. In some scenarios, the application may want to limit the submission of AQL packets to a single agent. Queues that support a single producer may be more efficient than queues supporting multiple producers. Use of multiproducer queue mechanics is not supported.
HSA_QUEUE_TYPE_COOPERATIVE	Queue supports multiple producers and cooperative dispatches. Cooperative dispatches are able to use GWS synchronization. Queues of this type may be limited in number. The runtime may return the same queue to serve multiple hsa_queue_create calls when this type is given. Callers must inspect the returned queue to discover queue size. Queues of this type are reference counted and require a matching number of hsa_queue_destroy calls to release. Use of multiproducer queue mechanics is required. See HSA_AMD_AGENT_INFO_COOPERATIVE_QUEUES to query agent support for this type.

Definition at line 2212 of file hsa.h.

5.5.4 Function Documentation

5.5.4.1 hsa_queue_add_write_index_acq_rel()

```
uint64_t HSA_API HSA_DEPRECATED hsa_queue_add_write_index_acq_rel (
            const hsa_queue_t * queue,
            uint64_t value )
```

Deprecated Renamed as hsa_queue_add_write_index_scacq_screl.

Atomically increment the write index of a queue by an offset.

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to add to the write index.

Returns

Previous value of the write index.

5.5.4.2 hsa_queue_add_write_index_acquire()

```
uint64_t HSA_API HSA_DEPRECATED hsa_queue_add_write_index_acquire (
            const hsa_queue_t * queue,
            uint64_t value )
```

Deprecated Renamed as hsa_queue_add_write_index_scacquire.

Atomically increment the write index of a queue by an offset.

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to add to the write index.

Returns

Previous value of the write index.

5.5.4.3 hsa_queue_add_write_index_relaxed()

```
uint64_t HSA_API hsa_queue_add_write_index_relaxed (
            const hsa_queue_t * queue,
            uint64_t value )
```

Atomically increment the write index of a queue by an offset.

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to add to the write index.

Returns

Previous value of the write index.

5.5.4.4 hsa_queue_add_write_index_release()

```
uint64_t HSA_API HSA_DEPRECATED hsa_queue_add_write_index_release (
            const hsa_queue_t * queue,
            uint64_t value )
```

Deprecated Renamed as hsa_queue_add_write_index_screlease.

Atomically increment the write index of a queue by an offset.

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to add to the write index.

Returns

Previous value of the write index.

5.5.4.5 hsa_queue_add_write_index_scacq_screl()

```
uint64_t HSA_API hsa_queue_add_write_index_scacq_screl (
            const hsa_queue_t * queue,
            uint64_t value )
```

Atomically increment the write index of a queue by an offset.

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to add to the write index.

Returns

Previous value of the write index.

5.5.4.6 hsa_queue_add_write_index_scacquire()

```
uint64_t HSA_API hsa_queue_add_write_index_scacquire (
            const hsa_queue_t * queue,
            uint64_t value )
```

Atomically increment the write index of a queue by an offset.

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to add to the write index.

Returns

Previous value of the write index.

5.5.4.7 hsa_queue_add_write_index_screlease()

```
uint64_t HSA_API hsa_queue_add_write_index_screlease (
            const hsa_queue_t * queue,
            uint64_t value )
```

Atomically increment the write index of a queue by an offset.

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to add to the write index.

Returns

Previous value of the write index.

5.5.4.8 hsa_queue_cas_write_index_acq_rel()

```
uint64_t HSA_API HSA_DEPRECATED hsa_queue_cas_write_index_acq_rel (
            const hsa_queue_t * queue,
            uint64_t expected,
            uint64_t value )
```

Deprecated Renamed as hsa_queue_cas_write_index_scacq_screl.

Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.

Parameters

in	*queue*	Pointer to a queue.
in	*expected*	Expected value.
in	*value*	Value to assign to the write index if `expected` matches the observed write index. Must be greater than `expected`.

Returns

Previous value of the write index.

5.5.4.9 hsa_queue_cas_write_index_acquire()

```
uint64_t HSA_API HSA_DEPRECATED hsa_queue_cas_write_index_acquire (
            const hsa_queue_t * queue,
            uint64_t expected,
            uint64_t value )
```

Deprecated Renamed as hsa_queue_cas_write_index_scacquire.

Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.

Parameters

in	*queue*	Pointer to a queue.
in	*expected*	Expected value.
in	*value*	Value to assign to the write index if `expected` matches the observed write index. Must be greater than `expected`.

Returns

Previous value of the write index.

5.5.4.10 hsa_queue_cas_write_index_relaxed()

```
uint64_t HSA_API hsa_queue_cas_write_index_relaxed (
            const hsa_queue_t * queue,
```

```
            uint64_t expected,
            uint64_t value )
```

Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.

Parameters

in	*queue*	Pointer to a queue.
in	*expected*	Expected value.
in	*value*	Value to assign to the write index if `expected` matches the observed write index. Must be greater than `expected`.

Returns

Previous value of the write index.

5.5.4.11 hsa_queue_cas_write_index_release()

```
uint64_t HSA_API HSA_DEPRECATED hsa_queue_cas_write_index_release (
            const hsa_queue_t * queue,
            uint64_t expected,
            uint64_t value )
```

Deprecated Renamed as hsa_queue_cas_write_index_screlease.

Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.

Parameters

in	*queue*	Pointer to a queue.
in	*expected*	Expected value.
in	*value*	Value to assign to the write index if `expected` matches the observed write index. Must be greater than `expected`.

Returns

Previous value of the write index.

5.5.4.12 hsa_queue_cas_write_index_scacq_screl()

```
uint64_t HSA_API hsa_queue_cas_write_index_scacq_screl (
            const hsa_queue_t * queue,
```

```
            uint64_t expected,
            uint64_t value )
```

Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.

Parameters

in	*queue*	Pointer to a queue.
in	*expected*	Expected value.
in	*value*	Value to assign to the write index if `expected` matches the observed write index. Must be greater than `expected`.

Returns

Previous value of the write index.

5.5.4.13 hsa_queue_cas_write_index_scacquire()

```
uint64_t HSA_API hsa_queue_cas_write_index_scacquire (
            const hsa_queue_t * queue,
            uint64_t expected,
            uint64_t value )
```

Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.

Parameters

in	*queue*	Pointer to a queue.
in	*expected*	Expected value.
in	*value*	Value to assign to the write index if `expected` matches the observed write index. Must be greater than `expected`.

Returns

Previous value of the write index.

5.5.4.14 hsa_queue_cas_write_index_screlease()

```
uint64_t HSA_API hsa_queue_cas_write_index_screlease (
            const hsa_queue_t * queue,
            uint64_t expected,
            uint64_t value )
```

Atomically set the write index of a queue if the observed value is equal to the expected value. The application can inspect the returned value to determine if the replacement was done.

Parameters

in	*queue*	Pointer to a queue.
in	*expected*	Expected value.
in	*value*	Value to assign to the write index if `expected` matches the observed write index. Must be greater than `expected`.

Returns

Previous value of the write index.

5.5.4.15 hsa_queue_create()

```
hsa_status_t HSA_API hsa_queue_create (
            hsa_agent_t agent,
            uint32_t size,
            hsa_queue_type32_t type,
            void(*)(hsa_status_t status, hsa_queue_t *source, void *data) callback,
            void * data,
            uint32_t private_segment_size,
            uint32_t group_segment_size,
            hsa_queue_t ** queue )
```

Create a user mode queue.

The HSA runtime creates the queue structure, the underlying packet buffer, the completion signal, and the write and read indexes. The initial value of the write and read indexes is 0. The type of every packet in the buffer is initialized to HSA_PACKET_TYPE_INVALID.

The application should only rely on the error code returned to determine if the queue is valid.

Parameters

in	*agent*	Agent where to create the queue.
in	*size*	Number of packets the queue is expected to hold. Must be a power of 2 between 1 and the value of HSA_AGENT_INFO_QUEUE_MAX_SIZE in `agent`. The size of the newly created queue is the maximum of `size` and the value of HSA_AGENT_INFO_QUEUE_MIN_SIZE in `agent`.
in	*type*	Type of the queue, a bitwise OR of hsa_queue_type_t values. If the value of HSA_AGENT_INFO_QUEUE_TYPE in `agent` is HSA_QUEUE_TYPE_SINGLE, then `type` must also be HSA_QUEUE_TYPE_SINGLE.
in	*callback*	Callback invoked by the HSA runtime for every asynchronous event related to the newly created queue. May be NULL. The HSA runtime passes three arguments to the callback: a code identifying the event that triggered the invocation, a pointer to the queue where the event originated, and the application data.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Parameters

in	*private_segment_size*	Hint indicating the maximum expected private segment usage per work-item, in bytes. There may be performance degradation if the application places a kernel dispatch packet in the queue and the corresponding private segment usage exceeds `private_segment_size`. If the application does not want to specify any particular value for this argument, `private_segment_size` must be UINT32_MAX. If the queue does not support kernel dispatch packets, this argument is ignored.
in	*group_segment_size*	Hint indicating the maximum expected group segment usage per work-group, in bytes. There may be performance degradation if the application places a kernel dispatch packet in the queue and the corresponding group segment usage exceeds `group_segment_size`. If the application does not want to specify any particular value for this argument, `group_segment_size` must be UINT32_MAX. If the queue does not support kernel dispatch packets, this argument is ignored.
out	*queue*	Memory location where the HSA runtime stores a pointer to the newly created queue.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_QUEUE_CREATION	`agent` does not support queues of the given type.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`size` is not a power of two, `size` is 0, `type` is an invalid queue type, or `queue` is NULL.

5.5.4.16 hsa_queue_destroy()

```
hsa_status_t HSA_API hsa_queue_destroy (
            hsa_queue_t * queue )
```

Destroy a user mode queue.

When a queue is destroyed, the state of the AQL packets that have not been yet fully processed (their completion phase has not finished) becomes undefined. It is the responsibility of the application to ensure that all pending queue operations are finished if their results are required.

The resources allocated by the HSA runtime during queue creation (queue structure, ring buffer, doorbell signal) are released. The queue should not be accessed after being destroyed.

Parameters

in	*queue*	Pointer to a queue created using hsa_queue_create.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_QUEUE	The queue is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`queue` is NULL.

5.5.4.17 hsa_queue_inactivate()

```
hsa_status_t HSA_API hsa_queue_inactivate (
            hsa_queue_t * queue )
```

Inactivate a queue.

Inactivating the queue aborts any pending executions and prevent any new packets from being processed. Any more packets written to the queue once it is inactivated will be ignored by the packet processor.

Parameters

`in`	*queue*	Pointer to a queue.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_QUEUE	The queue is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`queue` is NULL.

5.5.4.18 hsa_queue_load_read_index_acquire()

```
uint64_t HSA_API HSA_DEPRECATED hsa_queue_load_read_index_acquire (
            const hsa_queue_t * queue )
```

Deprecated Renamed as hsa_queue_load_read_index_scacquire.

Atomically load the read index of a queue.

Parameters

`in`	*queue*	Pointer to a queue.

Returns

Read index of the queue pointed by `queue`.

5.5.4.19 hsa_queue_load_read_index_relaxed()

```
uint64_t HSA_API hsa_queue_load_read_index_relaxed (
            const hsa_queue_t * queue )
```

Atomically load the read index of a queue.

Parameters

in	*queue*	Pointer to a queue.

Returns

Read index of the queue pointed by `queue`.

5.5.4.20 hsa_queue_load_read_index_scacquire()

```
uint64_t HSA_API hsa_queue_load_read_index_scacquire (
            const hsa_queue_t * queue )
```

Atomically load the read index of a queue.

Parameters

in	*queue*	Pointer to a queue.

Returns

Read index of the queue pointed by `queue`.

5.5.4.21 hsa_queue_load_write_index_acquire()

```
uint64_t HSA_API HSA_DEPRECATED hsa_queue_load_write_index_acquire (
            const hsa_queue_t * queue )
```

Deprecated Renamed as hsa_queue_load_write_index_scacquire.

Atomically load the write index of a queue.

Parameters

in	*queue*	Pointer to a queue.

Returns

Write index of the queue pointed by `queue`.

5.5.4.22 hsa_queue_load_write_index_relaxed()

```
uint64_t HSA_API hsa_queue_load_write_index_relaxed (
            const hsa_queue_t * queue )
```

Atomically load the write index of a queue.

Parameters

in	*queue*	Pointer to a queue.

Returns

Write index of the queue pointed by `queue`.

5.5.4.23 hsa_queue_load_write_index_scacquire()

```
uint64_t HSA_API hsa_queue_load_write_index_scacquire (
            const hsa_queue_t * queue )
```

Atomically load the write index of a queue.

Parameters

in	*queue*	Pointer to a queue.

Returns

Write index of the queue pointed by `queue`.

5.5.4.24 hsa_queue_store_read_index_relaxed()

```
void HSA_API hsa_queue_store_read_index_relaxed (
            const hsa_queue_t * queue,
            uint64_t value )
```

Atomically set the read index of a queue.

Modifications of the read index are not allowed and result in undefined behavior if the queue is associated with an agent for which only the corresponding packet processor is permitted to update the read index.

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to assign to the read index.

5.5.4.25 hsa_queue_store_read_index_release()

```
void HSA_API HSA_DEPRECATED hsa_queue_store_read_index_release (
             const hsa_queue_t * queue,
             uint64_t value )
```

Deprecated Renamed as hsa_queue_store_read_index_screlease.

Atomically set the read index of a queue.

Modifications of the read index are not allowed and result in undefined behavior if the queue is associated with an agent for which only the corresponding packet processor is permitted to update the read index.

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to assign to the read index.

5.5.4.26 hsa_queue_store_read_index_screlease()

```
void HSA_API hsa_queue_store_read_index_screlease (
             const hsa_queue_t * queue,
             uint64_t value )
```

Atomically set the read index of a queue.

Modifications of the read index are not allowed and result in undefined behavior if the queue is associated with an agent for which only the corresponding packet processor is permitted to update the read index.

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to assign to the read index.

5.5.4.27 hsa_queue_store_write_index_relaxed()

```
void HSA_API hsa_queue_store_write_index_relaxed (
             const hsa_queue_t * queue,
             uint64_t value )
```

Atomically set the write index of a queue.

It is recommended that the application uses this function to update the write index when there is a single agent submitting work to the queue (the queue type is HSA_QUEUE_TYPE_SINGLE).

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to assign to the write index.

5.5.4.28 hsa_queue_store_write_index_release()

```
void HSA_API HSA_DEPRECATED hsa_queue_store_write_index_release (
             const hsa_queue_t * queue,
             uint64_t value )
```

Deprecated Renamed as hsa_queue_store_write_index_screlease.

Atomically set the write index of a queue.

It is recommended that the application uses this function to update the write index when there is a single agent submitting work to the queue (the queue type is HSA_QUEUE_TYPE_SINGLE).

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to assign to the write index.

5.5.4.29 hsa_queue_store_write_index_screlease()

```
void HSA_API hsa_queue_store_write_index_screlease (
             const hsa_queue_t * queue,
             uint64_t value )
```

Atomically set the write index of a queue.

It is recommended that the application uses this function to update the write index when there is a single agent submitting work to the queue (the queue type is HSA_QUEUE_TYPE_SINGLE).

Parameters

in	*queue*	Pointer to a queue.
in	*value*	Value to assign to the write index.

5.5.4.30 hsa_soft_queue_create()

```
hsa_status_t HSA_API hsa_soft_queue_create (
            hsa_region_t region,
            uint32_t size,
            hsa_queue_type32_t type,
            uint32_t features,
            hsa_signal_t doorbell_signal,
            hsa_queue_t ** queue )
```

Create a queue for which the application or a kernel is responsible for processing the AQL packets.

The application can use this function to create queues where AQL packets are not parsed by the packet processor associated with an agent, but rather by a unit of execution running on that agent (for example, a thread in the host application).

The application is responsible for ensuring that all the producers and consumers of the resulting queue can access the provided doorbell signal and memory region. The application is also responsible for ensuring that the unit of execution processing the queue packets supports the indicated features (AQL packet types).

When the queue is created, the HSA runtime allocates the packet buffer using `region`, and the write and read indexes. The initial value of the write and read indexes is 0, and the type of every packet in the buffer is initialized to HSA_PACKET_TYPE_INVALID. The value of the *size*, *type*, *features*, and *doorbell_signal* fields in the returned queue match the values passed by the application.

Parameters

in	*region*	Memory region that the HSA runtime should use to allocate the AQL packet buffer and any other queue metadata.
in	*size*	Number of packets the queue is expected to hold. Must be a power of 2 greater than 0.
in	*type*	Queue type.
in	*features*	Supported queue features. This is a bit-field of hsa_queue_feature_t values.
in	*doorbell_signal*	Doorbell signal that the HSA runtime must associate with the returned queue. The signal handle must not be 0.
out	*queue*	Memory location where the HSA runtime stores a pointer to the newly created queue. The application should not rely on the value returned for this argument but only in the status code to determine if the queue is valid. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.

Return values

HSA_STATUS_ERROR_INVALID_ARGUMENT	`size` is not a power of two, `size` is 0, `type` is an invalid queue type, the doorbell signal handle is 0, or `queue` is NULL.

5.6 Architected Queuing Language

Classes

- struct hsa_kernel_dispatch_packet_s

  AQL kernel dispatch packet.
- struct hsa_agent_dispatch_packet_s

  Agent dispatch packet.
- struct hsa_barrier_and_packet_s

  Barrier-AND packet.
- struct hsa_barrier_or_packet_s

  Barrier-OR packet.
- struct hsa_amd_packet_header_s

  AMD vendor specific AQL packet header.
- struct hsa_amd_barrier_value_packet_s

  AMD barrier value packet. Halts packet processing and waits for (signal_value & mask) cond value to be satisfied, where signal_value is the value of the signal signal.

Typedefs

- typedef struct hsa_kernel_dispatch_packet_s **hsa_kernel_dispatch_packet_t**

  AQL kernel dispatch packet.
- typedef struct hsa_agent_dispatch_packet_s **hsa_agent_dispatch_packet_t**

  Agent dispatch packet.
- typedef struct hsa_barrier_and_packet_s **hsa_barrier_and_packet_t**

  Barrier-AND packet.
- typedef struct hsa_barrier_or_packet_s **hsa_barrier_or_packet_t**

  Barrier-OR packet.
- typedef uint32_t hsa_signal_condition32_t

  A fixed-size type used to represent hsa_signal_condition_t constants.
- typedef uint8_t hsa_amd_packet_type8_t

  A fixed-size type used to represent hsa_amd_packet_type_t constants.
- typedef struct hsa_amd_packet_header_s **hsa_amd_vendor_packet_header_t**

  AMD vendor specific AQL packet header.
- typedef struct hsa_amd_barrier_value_packet_s **hsa_amd_barrier_value_packet_t**

  AMD barrier value packet. Halts packet processing and waits for (signal_value & ::mask) ::cond ::value to be satisfied, where signal_value is the value of the signal ::signal.

Enumerations

- enum hsa_packet_type_t { HSA_PACKET_TYPE_VENDOR_SPECIFIC = 0 , HSA_PACKET_TYPE_INVALID = 1 , HSA_PACKET_TYPE_KERNEL_DISPA = 2 , HSA_PACKET_TYPE_BARRIER_AND = 3 , HSA_PACKET_TYPE_AGENT_DISPATCH = 4 , HSA_PACKET_TYPE_BARRIER_OR = 5 }

  Packet type.
- enum hsa_fence_scope_t { HSA_FENCE_SCOPE_NONE = 0 , HSA_FENCE_SCOPE_AGENT = 1 , HSA_FENCE_SCOPE_SYSTEM = 2 }

  Scope of the memory fence operation associated with a packet.
- enum hsa_packet_header_t { HSA_PACKET_HEADER_TYPE = 0 , HSA_PACKET_HEADER_BARRIER = 8 , HSA_PACKET_HEADER_SCACQUIRE_FENC = 9 , HSA_PACKET_HEADER_ACQUIRE_FENCE_SCOPE = 9 , HSA_PACKET_HEADER_SCRELEASE_FENCE_SCOPE = 11 , HSA_PACKET_HEADER_RELEASE_FENCE_SCOPE = 11 }

  Sub-fields of the header field that is present in any AQL packet. The offset (with respect to the address of header) of a sub-field is identical to its enumeration constant. The width of each sub-field is determined by the corresponding value in hsa_packet_header_width_t. The offset and the width are expressed in bits.
- enum hsa_packet_header_width_t { **HSA_PACKET_HEADER_WIDTH_TYPE** = 8 , **HSA_PACKET_HEADER_WIDTH_BARRIER** = 1 , **HSA_↩ PACKET_HEADER_WIDTH_SCACQUIRE_FENCE_SCOPE** = 2 , HSA_PACKET_HEADER_WIDTH_ACQUIRE_FENCE_SC = 2 , **HSA_PACKET_HEADER_WIDTH_SCRELEASE_FENCE_SCOPE** = 2 , HSA_PACKET_HEADER_WIDTH_RELEASE_FENC = 2 }

  Width (in bits) of the sub-fields in hsa_packet_header_t.
- enum hsa_kernel_dispatch_packet_setup_t { HSA_KERNEL_DISPATCH_PACKET_SETUP_DIMENSIONS = 0 }

  Sub-fields of the kernel dispatch packet setup field. The offset (with respect to the address of setup) of a sub-field is identical to its enumeration constant. The width of each sub-field is determined by the corresponding value in hsa_kernel_dispatch_packet_setup_width_t. The offset and the width are expressed in bits.
- enum hsa_kernel_dispatch_packet_setup_width_t { **HSA_KERNEL_DISPATCH_PACKET_SETUP_↩ WIDTH_DIMENSIONS** = 2 }

  Width (in bits) of the sub-fields in hsa_kernel_dispatch_packet_setup_t.
- enum hsa_amd_packet_type_t { HSA_AMD_PACKET_TYPE_BARRIER_VALUE = 2 }

  AMD vendor specific packet type.

5.6.1 Detailed Description

5.6.2 Typedef Documentation

5.6.2.1 hsa_amd_packet_type8_t

```
typedef uint8_t hsa_amd_packet_type8_t
```

A fixed-size type used to represent hsa_amd_packet_type_t constants.

Definition at line 82 of file hsa_ext_amd.h.

5.6.2.2 hsa_signal_condition32_t

```
typedef uint32_t hsa_signal_condition32_t
```

A fixed-size type used to represent hsa_signal_condition_t constants.

Definition at line 65 of file hsa_ext_amd.h.

5.6.3 Enumeration Type Documentation

5.6.3.1 hsa_amd_packet_type_t

```
enum hsa_amd_packet_type_t
```

AMD vendor specific packet type.

Enumerator

HSA_AMD_PACKET_TYPE_BARRIER_VALUE	Packet used by agents to delay processing of subsequent packets until a configurable condition is satisfied by an HSA signal. Only kernel dispatch queues created from AMD GPU Agents support this packet.

Definition at line 70 of file hsa_ext_amd.h.

5.6.3.2 hsa_fence_scope_t

```
enum hsa_fence_scope_t
```

Scope of the memory fence operation associated with a packet.

Enumerator

HSA_FENCE_SCOPE_NONE	No scope (no fence is applied). The packet relies on external fences to ensure visibility of memory updates.
HSA_FENCE_SCOPE_AGENT	The fence is applied with agent scope for the global segment.
HSA_FENCE_SCOPE_SYSTEM	The fence is applied across both agent and system scope for the global segment.

Definition at line 2807 of file hsa.h.

5.6.3.3 hsa_kernel_dispatch_packet_setup_t

enum hsa_kernel_dispatch_packet_setup_t

Sub-fields of the kernel dispatch packet *setup* field. The offset (with respect to the address of *setup*) of a sub-field is identical to its enumeration constant. The width of each sub-field is determined by the corresponding value in hsa_kernel_dispatch_packet_setup_width_t. The offset and the width are expressed in bits.

Enumerator

HSA_KERNEL_DISPATCH_PACKET_SETUP_↩ DIMENSIONS	Number of dimensions of the grid. Valid values are 1, 2, or 3.

Definition at line 2899 of file hsa.h.

5.6.3.4 hsa_kernel_dispatch_packet_setup_width_t

enum hsa_kernel_dispatch_packet_setup_width_t

Width (in bits) of the sub-fields in hsa_kernel_dispatch_packet_setup_t.

Definition at line 2911 of file hsa.h.

5.6.3.5 hsa_packet_header_t

enum hsa_packet_header_t

Sub-fields of the *header* field that is present in any AQL packet. The offset (with respect to the address of *header*) of a sub-field is identical to its enumeration constant. The width of each sub-field is determined by the corresponding value in hsa_packet_header_width_t. The offset and the width are expressed in bits.

Enumerator

HSA_PACKET_HEADER_TYPE	Packet type. The value of this sub-field must be one of hsa_packet_type_t. If the type is HSA_PACKET_TYPE_VENDOR_SPECIFIC, the packet layout is vendor-specific.
HSA_PACKET_HEADER_BARRIER	Barrier bit. If the barrier bit is set, the processing of the current packet only launches when all preceding packets (within the same queue) are complete.
HSA_PACKET_HEADER_SCACQUIRE_FENCE_↩ SCOPE	Acquire fence scope. The value of this sub-field determines the scope and type of the memory fence operation applied before the packet enters the active phase. An acquire fence ensures that any subsequent global segment or image loads by any unit of execution that belongs to a dispatch that has not yet entered the active phase on any queue of the same kernel agent, sees any data previously released at the scopes specified by the acquire fence. The value of this sub-field must be one of hsa_fence_scope_t.

Enumerator

HSA_PACKET_HEADER_ACQUIRE_FENCE_↩ SCOPE	**Deprecated** Renamed as HSA_PACKET_HEADER_SCACQUIRE_FENCE_SCOPE.
HSA_PACKET_HEADER_SCRELEASE_FENCE_↩ SCOPE	Release fence scope, The value of this sub-field determines the scope and type of the memory fence operation applied after kernel completion but before the packet is completed. A release fence makes any global segment or image data that was stored by any unit of execution that belonged to a dispatch that has completed the active phase on any queue of the same kernel agent visible in all the scopes specified by the release fence. The value of this sub-field must be one of hsa_fence_scope_t.
HSA_PACKET_HEADER_RELEASE_FENCE_↩ SCOPE	**Deprecated** Renamed as HSA_PACKET_HEADER_SCRELEASE_FENCE_SCOPE.

Definition at line 2831 of file hsa.h.

5.6.3.6 hsa_packet_header_width_t

`enum hsa_packet_header_width_t`

Width (in bits) of the sub-fields in hsa_packet_header_t.

Enumerator

HSA_PACKET_HEADER_WIDTH_ACQUIRE_↩ FENCE_SCOPE	**Deprecated** Use HSA_PACKET_HEADER_WIDTH_↩ SCACQUIRE_FENCE_SCOPE.
HSA_PACKET_HEADER_WIDTH_RELEASE_↩ FENCE_SCOPE	**Deprecated** Use HSA_PACKET_HEADER_WIDTH_↩ SCRELEASE_FENCE_SCOPE.

Definition at line 2877 of file hsa.h.

5.6.3.7 hsa_packet_type_t

`enum hsa_packet_type_t`

Packet type.

Enumerator

HSA_PACKET_TYPE_VENDOR_SPECIFIC	Vendor-specific packet.
HSA_PACKET_TYPE_INVALID	The packet has been processed in the past, but has not been reassigned to the packet processor. A packet processor must not process a packet of this type. All queues support this packet type.
HSA_PACKET_TYPE_KERNEL_DISPATCH	Packet used by agents for dispatching jobs to kernel agents. Not all queues support packets of this type (see hsa_queue_feature_t).
HSA_PACKET_TYPE_BARRIER_AND	Packet used by agents to delay processing of subsequent packets, and to express complex dependencies between multiple packets. All queues support this packet type.
HSA_PACKET_TYPE_AGENT_DISPATCH	Packet used by agents for dispatching jobs to agents. Not all queues support packets of this type (see hsa_queue_feature_t).
HSA_PACKET_TYPE_BARRIER_OR	Packet used by agents to delay processing of subsequent packets, and to express complex dependencies between multiple packets. All queues support this packet type.

Definition at line 2769 of file hsa.h.

5.7 Instruction Set Architecture.

Classes

- struct hsa_isa_s

  Instruction set architecture.
- struct hsa_wavefront_s

  Wavefront handle.

Typedefs

- typedef struct hsa_isa_s **hsa_isa_t**

  Instruction set architecture.
- typedef struct hsa_wavefront_s **hsa_wavefront_t**

  Wavefront handle.

Enumerations

- enum hsa_isa_info_t { HSA_ISA_INFO_NAME_LENGTH = 0 , HSA_ISA_INFO_NAME = 1 , HSA_ISA_INFO_CALL_CONVENTION_COUNT = 2 , HSA_ISA_INFO_CALL_CONVENTION_INFO_WAVEFRONT_SIZE = 3 , HSA_ISA_INFO_CALL_CONVENTION_INFO_WAVEFRONTS_PER_COMPUTE_UNIT = 4 , HSA_ISA_INFO_MACHINE_MO = 5 , HSA_ISA_INFO_PROFILES = 6 , HSA_ISA_INFO_DEFAULT_FLOAT_ROUNDING_MODES = 7 , HSA_ISA_INFO_BASE_PROFILE_DEFAULT_FLOAT_ROUNDING_MODES = 8 , HSA_ISA_INFO_FAST_F16_OPERATION = 9 , HSA_ISA_INFO_WORKGROUP_MAX_DIM = 12 , HSA_ISA_INFO_WORKGROUP_MAX_SIZE = 13 , HSA_ISA_INFO_GRID_MAX_DIM = 14 , HSA_ISA_INFO_GRID_MAX_SIZE = 16 , HSA_ISA_INFO_FBARRIER_MAX_SIZE = 17 }

Instruction set architecture attributes.

- enum hsa_fp_type_t { HSA_FP_TYPE_16 = 1 , HSA_FP_TYPE_32 = 2 , HSA_FP_TYPE_64 = 4 }

  Floating-point types.
- enum hsa_flush_mode_t { HSA_FLUSH_MODE_FTZ = 1 , HSA_FLUSH_MODE_NON_FTZ = 2 }

  Flush to zero modes.
- enum hsa_round_method_t { HSA_ROUND_METHOD_SINGLE = 1 , HSA_ROUND_METHOD_DOUBLE = 2 }

  Round methods.
- enum hsa_wavefront_info_t { HSA_WAVEFRONT_INFO_SIZE = 0 }

  Wavefront attributes.

Functions

- hsa_status_t HSA_API hsa_isa_from_name (const char *name, hsa_isa_t *isa)

  Retrieve a reference to an instruction set architecture handle out of a symbolic name.
- hsa_status_t HSA_API hsa_agent_iterate_isas (hsa_agent_t agent, hsa_status_t(*callback)(hsa_isa_t isa, void *data), void *data)

  Iterate over the instruction sets supported by the given agent, and invoke an application-defined callback on every iteration. The iterator is deterministic: if an agent supports several instruction set architectures, they are traversed in the same order in every invocation of this function.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_isa_get_info (hsa_isa_t isa, hsa_isa_info_t attribute, uint32_t index, void *value)

  Get the current value of an attribute for a given instruction set architecture (ISA).
- hsa_status_t HSA_API hsa_isa_get_info_alt (hsa_isa_t isa, hsa_isa_info_t attribute, void *value)

  Get the current value of an attribute for a given instruction set architecture (ISA).
- hsa_status_t HSA_API hsa_isa_get_exception_policies (hsa_isa_t isa, hsa_profile_t profile, uint16_t *mask)

  Retrieve the exception policy support for a given combination of instruction set architecture and profile.
- hsa_status_t HSA_API hsa_isa_get_round_method (hsa_isa_t isa, hsa_fp_type_t fp_type, hsa_flush_mode_t flush_mode, hsa_round_method_t *round_method)

  Retrieve the round method (single or double) used to implement the floating-point multiply add instruction (mad) for a given combination of instruction set architecture, floating-point type, and flush to zero modifier.
- hsa_status_t HSA_API hsa_wavefront_get_info (hsa_wavefront_t wavefront, hsa_wavefront_info_t attribute, void *value)

  Get the current value of a wavefront attribute.
- hsa_status_t HSA_API hsa_isa_iterate_wavefronts (hsa_isa_t isa, hsa_status_t(*callback)(hsa_wavefront_t wavefront, void *data), void *data)

  Iterate over the different wavefronts supported by an instruction set architecture, and invoke an application-defined callback on every iteration.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_isa_compatible (hsa_isa_t code_object_isa, hsa_isa_t agent_isa, bool *result)

  Check if the instruction set architecture of a code object can be executed on an agent associated with another architecture.

5.7.1 Detailed Description

5.7.2 Enumeration Type Documentation

5.7.2.1 hsa_flush_mode_t

```
enum hsa_flush_mode_t
```

Flush to zero modes.

Enumerator

HSA_FLUSH_MODE_FTZ	Flush to zero.
HSA_FLUSH_MODE_NON_FTZ	Do not flush to zero.

Definition at line 3839 of file hsa.h.

5.7.2.2 hsa_fp_type_t

`enum hsa_fp_type_t`

Floating-point types.

Enumerator

HSA_FP_TYPE_16	16-bit floating-point type.
HSA_FP_TYPE_32	32-bit floating-point type.
HSA_FP_TYPE_64	64-bit floating-point type.

Definition at line 3821 of file hsa.h.

5.7.2.3 hsa_isa_info_t

`enum hsa_isa_info_t`

Instruction set architecture attributes.

Enumerator

HSA_ISA_INFO_NAME_LENGTH	The length of the ISA name in bytes, not including the NUL terminator. The type of this attribute is uint32_t.
HSA_ISA_INFO_NAME	Human-readable description. The type of this attribute is character array with the length equal to the value of HSA_ISA_INFO_NAME_LENGTH attribute.
HSA_ISA_INFO_CALL_CONVENTION_COUNT	**Deprecated** Number of call conventions supported by the instruction set architecture. Must be greater than zero. The type of this attribute is uint32_t.
HSA_ISA_INFO_CALL_CONVENTION_INFO_↩ WAVEFRONT_SIZE	**Deprecated** Number of work-items in a wavefront for a given call convention. Must be a power of 2 in the range [1,256]. The type of this attribute is uint32_t.

Enumerator

HSA_ISA_INFO_CALL_CONVENTION_INFO_↩ WAVEFRONTS_PER_COMPUTE_UNIT	**Deprecated** Number of wavefronts per compute unit for a given call convention. In practice, other factors (for example, the amount of group memory used by a work-group) may further limit the number of wavefronts per compute unit. The type of this attribute is uint32_t.
HSA_ISA_INFO_MACHINE_MODELS	Machine models supported by the instruction set architecture. The type of this attribute is a bool[2]. If the ISA supports the small machine model, the element at index HSA_MACHINE_MODEL_SMALL is true. If the ISA supports the large model, the element at index HSA_MACHINE_MODEL_LARGE is true.
HSA_ISA_INFO_PROFILES	Profiles supported by the instruction set architecture. The type of this attribute is a bool[2]. If the ISA supports the base profile, the element at index HSA_PROFILE_BASE is true. If the ISA supports the full profile, the element at index HSA_PROFILE_FULL is true.
HSA_ISA_INFO_DEFAULT_FLOAT_ROUNDING_↩ MODES	Default floating-point rounding modes supported by the instruction set architecture. The type of this attribute is a bool[3]. The value at a given index is true if the corresponding rounding mode in hsa_default_float_rounding_mode_t is supported. At least one default mode has to be supported. If the default mode is supported, then HSA_ISA_INFO_BASE_PROFILE_DEFAULT_FLOAT_ROUNDING_MOD must report that both the zero and the near roundings modes are supported.
HSA_ISA_INFO_BASE_PROFILE_DEFAULT_↩ FLOAT_ROUNDING_MODES	Default floating-point rounding modes supported by the instruction set architecture in the Base profile. The type of this attribute is a bool[3]. The value at a given index is true if the corresponding rounding mode in hsa_default_float_rounding_mode_t is supported. The value at index HSA_DEFAULT_FLOAT_↩ ROUNDING_MODE_DEFAULT must be false. At least one of the values at indexes HSA_DEFAULT_FLOAT_ROUNDING_MODE_ZERO or HSA_DEFAULT_FLOAT_ROUNDING_MODE_NEAR must be true.
HSA_ISA_INFO_FAST_F16_OPERATION	Flag indicating that the f16 HSAIL operation is at least as fast as the f32 operation in the instruction set architecture. The type of this attribute is bool.
HSA_ISA_INFO_WORKGROUP_MAX_DIM	Maximum number of work-items of each dimension of a work-group. Each maximum must be greater than 0. No maximum can exceed the value of HSA_ISA_INFO_WORKGROUP_MAX_SIZE. The type of this attribute is uint16_t[3].
HSA_ISA_INFO_WORKGROUP_MAX_SIZE	Maximum total number of work-items in a work-group. The type of this attribute is uint32_t.

Enumerator

HSA_ISA_INFO_GRID_MAX_DIM	Maximum number of work-items of each dimension of a grid. Each maximum must be greater than 0, and must not be smaller than the corresponding value in HSA_ISA_INFO_WORKGROUP_MAX_DIM. No maximum can exceed the value of HSA_ISA_INFO_GRID_MAX_SIZE. The type of this attribute is hsa_dim3_t.
HSA_ISA_INFO_GRID_MAX_SIZE	Maximum total number of work-items in a grid. The type of this attribute is uint64_t.
HSA_ISA_INFO_FBARRIER_MAX_SIZE	Maximum number of fbarriers per work-group. Must be at least 32. The type of this attribute is uint32_t.

Definition at line 3611 of file hsa.h.

5.7.2.4 hsa_round_method_t

`enum hsa_round_method_t`

Round methods.

Enumerator

HSA_ROUND_METHOD_SINGLE	Single round method.
HSA_ROUND_METHOD_DOUBLE	Double round method.

Definition at line 3853 of file hsa.h.

5.7.2.5 hsa_wavefront_info_t

`enum hsa_wavefront_info_t`

Wavefront attributes.

Enumerator

HSA_WAVEFRONT_INFO_SIZE	Number of work-items in the wavefront. Must be a power of 2 in the range [1,256]. The type of this attribute is uint32_t.

Definition at line 3911 of file hsa.h.

5.7.3 Function Documentation

5.7.3.1 hsa_agent_iterate_isas()

```
hsa_status_t HSA_API hsa_agent_iterate_isas (
            hsa_agent_t agent,
            hsa_status_t(*)(hsa_isa_t isa, void *data) callback,
            void * data )
```

Iterate over the instruction sets supported by the given agent, and invoke an application-defined callback on every iteration. The iterator is deterministic: if an agent supports several instruction set architectures, they are traversed in the same order in every invocation of this function.

Parameters

in	*agent*	A valid agent.
in	*callback*	Callback to be invoked once per instruction set architecture. The HSA runtime passes two arguments to the callback: the ISA and the application data. If `callback` returns a status other than HSA_STATUS_SUCCESS for a particular iteration, the traversal stops and that status value is returned.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`callback` is NULL.

5.7.3.2 hsa_isa_compatible()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_isa_compatible (
            hsa_isa_t code_object_isa,
            hsa_isa_t agent_isa,
            bool * result )
```

Check if the instruction set architecture of a code object can be executed on an agent associated with another architecture.

Deprecated Use hsa_agent_iterate_isas to query which instructions set architectures are supported by a given agent.

Parameters

in	*code_object_isa*	Instruction set architecture associated with a code object.
in	*agent_isa*	Instruction set architecture associated with an agent.
out	*result*	Pointer to a memory location where the HSA runtime stores the result of the check. If the two architectures are compatible, the result is true; if they are incompatible, the result is false.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ISA	`code_object_isa` or `agent_isa` are invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`result` is NULL.

5.7.3.3 hsa_isa_from_name()

```
hsa_status_t HSA_API hsa_isa_from_name (
            const char * name,
            hsa_isa_t * isa )
```

Retrieve a reference to an instruction set architecture handle out of a symbolic name.

Parameters

`in`	*name*	Vendor-specific name associated with a a particular instruction set architecture. `name` must start with the vendor name and a colon (for example, "AMD:"). The rest of the name is vendor-specific. Must be a NUL-terminated string.
`out`	*isa*	Memory location where the HSA runtime stores the ISA handle corresponding to the given name. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ISA_NAME	The given name does not correspond to any instruction set architecture.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`name` is NULL, or `isa` is NULL.

5.7.3.4 hsa_isa_get_exception_policies()

```
hsa_status_t HSA_API hsa_isa_get_exception_policies (
            hsa_isa_t isa,
            hsa_profile_t profile,
            uint16_t * mask )
```

Retrieve the exception policy support for a given combination of instruction set architecture and profile.

Parameters

`in`	*isa*	A valid instruction set architecture.
`in`	*profile*	Profile.
`out`	*mask*	Pointer to a memory location where the HSA runtime stores a mask of hsa_exception_policy_t values. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ISA	The instruction set architecture is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`profile` is not a valid profile, or `mask` is NULL.

5.7.3.5 hsa_isa_get_info()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_isa_get_info (
            hsa_isa_t isa,
            hsa_isa_info_t attribute,
            uint32_t index,
            void * value )
```

Get the current value of an attribute for a given instruction set architecture (ISA).

Deprecated The concept of call convention has been deprecated. If the application wants to query the value of an attribute for a given instruction set architecture, use hsa_isa_get_info_alt instead. If the application wants to query an attribute that is specific to a given combination of ISA and wavefront, use hsa_wavefront_get_info.

Parameters

`in`	*isa*	A valid instruction set architecture.
`in`	*attribute*	Attribute to query.
`in`	*index*	Call convention index. Used only for call convention attributes, otherwise ignored. Must have a value between 0 (inclusive) and the value of the attribute HSA_ISA_INFO_CALL_CONVENTION_COUNT (not inclusive) in `isa`.
`out`	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ISA	The instruction set architecture is invalid.
HSA_STATUS_ERROR_INVALID_INDEX	The index is out of range.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid instruction set architecture attribute, or `value` is NULL.

5.7.3.6 hsa_isa_get_info_alt()

```
hsa_status_t HSA_API hsa_isa_get_info_alt (
```

```
            hsa_isa_t isa,
            hsa_isa_info_t attribute,
            void * value )
```

Get the current value of an attribute for a given instruction set architecture (ISA).

Parameters

in	*isa*	A valid instruction set architecture.
in	*attribute*	Attribute to query.
out	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ISA	The instruction set architecture is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid instruction set architecture attribute, or `value` is NULL.

5.7.3.7 hsa_isa_get_round_method()

```
hsa_status_t HSA_API hsa_isa_get_round_method (
            hsa_isa_t isa,
            hsa_fp_type_t fp_type,
            hsa_flush_mode_t flush_mode,
            hsa_round_method_t * round_method )
```

Retrieve the round method (single or double) used to implement the floating-point multiply add instruction (mad) for a given combination of instruction set architecture, floating-point type, and flush to zero modifier.

Parameters

in	*isa*	Instruction set architecture.
in	*fp_type*	Floating-point type.
in	*flush_mode*	Flush to zero modifier.
out	*round_method*	Pointer to a memory location where the HSA runtime stores the round method used by the implementation. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ISA	The instruction set architecture is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`fp_type` is not a valid floating-point type, or `flush_mode` is not a valid flush to zero modifier, or `round_method` is NULL.

5.7.3.8 hsa_isa_iterate_wavefronts()

```
hsa_status_t HSA_API hsa_isa_iterate_wavefronts (
            hsa_isa_t isa,
            hsa_status_t(*)(hsa_wavefront_t wavefront, void *data) callback,
            void * data )
```

Iterate over the different wavefronts supported by an instruction set architecture, and invoke an application-defined callback on every iteration.

Parameters

in	*isa*	Instruction set architecture.
in	*callback*	Callback to be invoked once per wavefront that is supported by the agent. The HSA runtime passes two arguments to the callback: the wavefront handle and the application data. If `callback` returns a status other than HSA_STATUS_SUCCESS for a particular iteration, the traversal stops and that value is returned.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_ISA	The instruction set architecture is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`callback` is NULL.

5.7.3.9 hsa_wavefront_get_info()

```
hsa_status_t HSA_API hsa_wavefront_get_info (
            hsa_wavefront_t wavefront,
            hsa_wavefront_info_t attribute,
            void * value )
```

Get the current value of a wavefront attribute.

Parameters

in	*wavefront*	A wavefront.
in	*attribute*	Attribute to query.
out	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.

Return values

HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_WAVEFRONT	The wavefront is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid wavefront attribute, or `value` is NULL.

5.8 Executable

Classes

- struct hsa_code_object_reader_s

  Code object reader handle. A code object reader is used to load a code object from file (when created using hsa_code_object_reader_create_from_file), or from memory (if created using hsa_code_object_reader_create_from_memory).
- struct hsa_executable_s

  Struct containing an opaque handle to an executable, which contains ISA for finalized kernels and indirect functions together with the allocated global or readonly segment variables they reference.
- struct hsa_loaded_code_object_s

  Loaded code object handle.
- struct hsa_executable_symbol_s

  Executable symbol handle.

Typedefs

- typedef struct hsa_code_object_reader_s **hsa_code_object_reader_t**

  Code object reader handle. A code object reader is used to load a code object from file (when created using hsa_code_object_reader_create_from_file), or from memory (if created using hsa_code_object_reader_create_from_memory).
- typedef struct hsa_executable_s **hsa_executable_t**

  Struct containing an opaque handle to an executable, which contains ISA for finalized kernels and indirect functions together with the allocated global or readonly segment variables they reference.
- typedef struct hsa_loaded_code_object_s **hsa_loaded_code_object_t**

  Loaded code object handle.
- typedef struct hsa_executable_symbol_s hsa_executable_symbol_t

  Executable symbol handle.

Enumerations

- enum hsa_executable_state_t { HSA_EXECUTABLE_STATE_UNFROZEN = 0 , HSA_EXECUTABLE_STATE_FROZEN = 1 }

  Executable state.
- enum hsa_executable_info_t { HSA_EXECUTABLE_INFO_PROFILE = 1 , HSA_EXECUTABLE_INFO_STATE = 2 , HSA_EXECUTABLE_INFO_DEFAULT_FLOAT_ROUNDING_MODE = 3 }

  Executable attributes.
- enum hsa_symbol_kind_t { HSA_SYMBOL_KIND_VARIABLE = 0 , HSA_SYMBOL_KIND_KERNEL = 1 , HSA_SYMBOL_KIND_INDIRECT_FUNCTION = 2 }

  Symbol type.
- enum hsa_symbol_linkage_t { HSA_SYMBOL_LINKAGE_MODULE = 0 , HSA_SYMBOL_LINKAGE_PROGRAM = 1 }

Linkage type of a symbol.

- enum hsa_variable_allocation_t { HSA_VARIABLE_ALLOCATION_AGENT = 0 , HSA_VARIABLE_ALLOCATION_PROGRAM = 1 }

  Allocation type of a variable.

- enum hsa_variable_segment_t { HSA_VARIABLE_SEGMENT_GLOBAL = 0 , HSA_VARIABLE_SEGMENT_READONLY = 1 }

  Memory segment associated with a variable.

- enum hsa_executable_symbol_info_t { HSA_EXECUTABLE_SYMBOL_INFO_TYPE = 0 , HSA_EXECUTABLE_SYMBOL_INFO_NAME_LENGTH = 1 , HSA_EXECUTABLE_SYMBOL_INFO_NAME = 2 , HSA_EXECUTABLE_SYMBOL_INFO_MODULE_NAME_LENGTH = 3 , HSA_EXECUTABLE_SYMBOL_INFO_MODULE_NAME = 4 , HSA_EXECUTABLE_SYMBOL_INFO_AGENT = 20 , HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_ADDRESS = 21 , HSA_EXECUTABLE_SYMBOL_INFO_LINKAGE = 5 , HSA_EXECUTABLE_SYMBOL_INFO_IS_DEFINITION = 17 , HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_ALLOCATIO = 6 , HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_SEGMENT = 7 , HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_AL = 8 , HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_SIZE = 9 , HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_IS_CONST = 10 , HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_OBJECT = 22 , HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_KERN = 11 , HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_KERNARG_SEGMENT_ALIGNMENT = 12 , HSA_EXECUTABLE_SYMBOL_ = 13 , HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_PRIVATE_SEGMENT_SIZE = 14 , HSA_EXECUTABLE_SYMBOL_INF = 15 , HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_CALL_CONVENTION = 18 , HSA_EXECUTABLE_SYMBOL_INFO_INDIREC = 23 , HSA_EXECUTABLE_SYMBOL_INFO_INDIRECT_FUNCTION_CALL_CONVENTION = 16 }

  Executable symbol attributes.

Functions

- hsa_status_t HSA_API hsa_code_object_reader_create_from_file (hsa_file_t file, hsa_code_object_reader_t *code_object_reader)

  Create a code object reader to operate on a file.

- hsa_status_t HSA_API hsa_code_object_reader_create_from_memory (const void *code_object, size_t size, hsa_code_object_reader_t *code_object_reader)

  Create a code object reader to operate on memory.

- hsa_status_t HSA_API hsa_code_object_reader_destroy (hsa_code_object_reader_t code_object_reader)

  Destroy a code object reader.

- hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_create (hsa_profile_t profile, hsa_executable_state_t executable_state, const char *options, hsa_executable_t *executable)

  Create an empty executable.

- hsa_status_t HSA_API hsa_executable_create_alt (hsa_profile_t profile, hsa_default_float_rounding_mode_t default_float_rounding_mode, const char *options, hsa_executable_t *executable)

  Create an empty executable.

- hsa_status_t HSA_API hsa_executable_destroy (hsa_executable_t executable)

  Destroy an executable.

- hsa_status_t HSA_API hsa_executable_load_program_code_object (hsa_executable_t executable, hsa_code_object_reader_t code_object_reader, const char *options, hsa_loaded_code_object_t *loaded_code_object)

  Load a program code object into an executable.

- hsa_status_t HSA_API hsa_executable_load_agent_code_object (hsa_executable_t executable, hsa_agent_t agent, hsa_code_object_reader_t code_object_reader, const char *options, hsa_loaded_code_object_t *loaded_code_object)

  Load an agent code object into an executable.

- hsa_status_t HSA_API hsa_executable_freeze (hsa_executable_t executable, const char *options)

Freeze the executable.

- hsa_status_t HSA_API hsa_executable_get_info (hsa_executable_t executable, hsa_executable_info_t attribute, void ∗value)

  Get the current value of an attribute for a given executable.

- hsa_status_t HSA_API hsa_executable_global_variable_define (hsa_executable_t executable, const char ∗variable_name, void ∗address)

  Define an external global variable with program allocation.

- hsa_status_t HSA_API hsa_executable_agent_global_variable_define (hsa_executable_t executable, hsa_agent_t agent, const char ∗variable_name, void ∗address)

  Define an external global variable with agent allocation.

- hsa_status_t HSA_API hsa_executable_readonly_variable_define (hsa_executable_t executable, hsa_agent_t agent, const char ∗variable_name, void ∗address)

  Define an external readonly variable.

- hsa_status_t HSA_API hsa_executable_validate (hsa_executable_t executable, uint32_t ∗result)

  Validate an executable. Checks that all code objects have matching machine model, profile, and default floating-point rounding mode. Checks that all declarations have definitions. Checks declaration-definition compatibility (see the HSA Programming Reference Manual for compatibility rules). Invoking this function is equivalent to invoking hsa_executable_validate_alt with no options.

- hsa_status_t HSA_API hsa_executable_validate_alt (hsa_executable_t executable, const char ∗options, uint32_t ∗result)

  Validate an executable. Checks that all code objects have matching machine model, profile, and default floating-point rounding mode. Checks that all declarations have definitions. Checks declaration-definition compatibility (see the HSA Programming Reference Manual for compatibility rules).

- hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_get_symbol (hsa_executable_t executable, const char ∗module_name, const char ∗symbol_name, hsa_agent_t agent, int32_t call_convention, hsa_executable_symbol_t ∗symbol)

  Get the symbol handle for a given a symbol name.

- hsa_status_t HSA_API hsa_executable_get_symbol_by_name (hsa_executable_t executable, const char ∗symbol_name, const hsa_agent_t ∗agent, hsa_executable_symbol_t ∗symbol)

  Retrieve the symbol handle corresponding to a given a symbol name.

- hsa_status_t HSA_API hsa_executable_symbol_get_info (hsa_executable_symbol_t executable_symbol, hsa_executable_symbol_info_t attribute, void ∗value)

  Get the current value of an attribute for a given executable symbol.

- hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_iterate_symbols (hsa_executable_t executable, hsa_status_t(∗callback)(hsa_executable_t exec, hsa_executable_symbol_t symbol, void ∗data), void ∗data)

  Iterate over the symbols in a executable, and invoke an application-defined callback on every iteration.

- hsa_status_t HSA_API hsa_executable_iterate_agent_symbols (hsa_executable_t executable, hsa_agent_t agent, hsa_status_t(∗callback)(hsa_executable_t exec, hsa_agent_t agent, hsa_executable_symbol_t symbol, void ∗data), void ∗data)

  Iterate over the kernels, indirect functions, and agent allocation variables in an executable for a given agent, and invoke an application- defined callback on every iteration.

- hsa_status_t HSA_API hsa_executable_iterate_program_symbols (hsa_executable_t executable, hsa_status_t(∗callback)(hsa_ exec, hsa_executable_symbol_t symbol, void ∗data), void ∗data)

  Iterate over the program allocation variables in an executable, and invoke an application-defined callback on every iteration.

5.8.1 Detailed Description

5.8.2 Typedef Documentation

5.8.2.1 hsa_executable_symbol_t

```
typedef struct hsa_executable_symbol_s hsa_executable_symbol_t
```

Executable symbol handle.

The lifetime of an executable object symbol matches that of the executable associated with it. An operation on a symbol whose associated executable has been destroyed results in undefined behavior.

5.8.3 Enumeration Type Documentation

5.8.3.1 hsa_executable_info_t

```
enum hsa_executable_info_t
```

Executable attributes.

Enumerator

HSA_EXECUTABLE_INFO_PROFILE	Profile this executable is created for. The type of this attribute is hsa_profile_t.
HSA_EXECUTABLE_INFO_STATE	Executable state. The type of this attribute is hsa_executable_state_t.
HSA_EXECUTABLE_INFO_DEFAULT_FLOAT_↩ ROUNDING_MODE	Default floating-point rounding mode specified when executable was created. The type of this attribute is hsa_default_float_rounding_mode_t.

Definition at line 4401 of file hsa.h.

5.8.3.2 hsa_executable_state_t

```
enum hsa_executable_state_t
```

Executable state.

Enumerator

HSA_EXECUTABLE_STATE_UNFROZEN	Executable state, which allows the user to load code objects and define external variables. Variable addresses, kernel code handles, and indirect function code handles are not available in query operations until the executable is frozen (zero always returned).
HSA_EXECUTABLE_STATE_FROZEN	Executable state, which allows the user to query variable addresses, kernel code handles, and indirect function code handles using query operations. Loading new code objects, as well as defining external variables, is not allowed in this state.

Definition at line 4121 of file hsa.h.

5.8.3.3 hsa_executable_symbol_info_t

`enum hsa_executable_symbol_info_t`

Executable symbol attributes.

Enumerator

HSA_EXECUTABLE_SYMBOL_INFO_TYPE	The kind of the symbol. The type of this attribute is hsa_symbol_kind_t.
HSA_EXECUTABLE_SYMBOL_INFO_NAME_LENGTH	The length of the symbol name in bytes, not including the NUL terminator. The type of this attribute is uint32_t.
HSA_EXECUTABLE_SYMBOL_INFO_NAME	The name of the symbol. The type of this attribute is character array with the length equal to the value of HSA_EXECUTABLE_SYMBOL_INFO_NAME_LENGTH attribute.
HSA_EXECUTABLE_SYMBOL_INFO_MODULE_NAME_LENGTH	**Deprecated** The length of the module name in bytes (not including the NUL terminator) to which this symbol belongs if this symbol has module linkage, otherwise 0 is returned. The type of this attribute is uint32_t.
HSA_EXECUTABLE_SYMBOL_INFO_MODULE_NAME	**Deprecated** The module name to which this symbol belongs if this symbol has module linkage, otherwise an empty string is returned. The type of this attribute is character array with the length equal to the value of HSA_EXECUTABLE_SYMBOL_INFO_MODULE_NAME_LENGTH attribute.
HSA_EXECUTABLE_SYMBOL_INFO_AGENT	**Deprecated** Agent associated with this symbol. If the symbol is a variable, the value of this attribute is only defined if HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_ALLOCATION is HSA_VARIABLE_ALLOCATION_AGENT. The type of this attribute is hsa_agent_t.
HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_ADDRESS	The address of the variable. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is uint64_t. If executable's state is HSA_EXECUTABLE_STATE_UNFROZEN, then 0 is returned.
HSA_EXECUTABLE_SYMBOL_INFO_LINKAGE	The linkage kind of the symbol. The type of this attribute is hsa_symbol_linkage_t.
HSA_EXECUTABLE_SYMBOL_INFO_IS_DEFINITION	Indicates whether the symbol corresponds to a definition. The type of this attribute is bool.

Enumerator

HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE↩ _ALLOCATION	**Deprecated** The allocation kind of the variable. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is hsa_variable_allocation_t.
HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE↩ _SEGMENT	**Deprecated** The segment kind of the variable. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is hsa_variable_segment_t.
HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE↩ _ALIGNMENT	**Deprecated** Alignment of the symbol in memory. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is uint32_t. The current alignment of the variable in memory may be greater than the value specified in the source program variable declaration.
HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE↩ _SIZE	**Deprecated** Size of the variable. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is uint32_t. A value of 0 is returned if the variable is an external variable and has an unknown dimension.
HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE↩ _IS_CONST	**Deprecated** Indicates whether the variable is constant. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is bool.
HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_↩ OBJECT	Kernel object handle, used in the kernel dispatch packet. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint64_t. If the state of the executable is HSA_EXECUTABLE_STATE_UNFROZEN, then 0 is returned.
HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_↩ KERNARG_SEGMENT_SIZE	Size of kernarg segment memory that is required to hold the values of the kernel arguments, in bytes. Must be a multiple of 16. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint32_t.
HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_↩ KERNARG_SEGMENT_ALIGNMENT	Alignment (in bytes) of the buffer used to pass arguments to the kernel, which is the maximum of 16 and the maximum alignment of any of the kernel arguments. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint32_t.

Enumerator

HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_↩ GROUP_SEGMENT_SIZE	Size of static group segment memory required by the kernel (per work-group), in bytes. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint32_t. The reported amount does not include any dynamically allocated group segment memory that may be requested by the application when a kernel is dispatched.
HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_↩ PRIVATE_SEGMENT_SIZE	Size of static private, spill, and arg segment memory required by this kernel (per work-item), in bytes. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint32_t. If the value of HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_DYNAMIC_CALLSTACK is true, the kernel may use more private memory than the reported value, and the application must add the dynamic call stack usage to *private_segment_size* when populating a kernel dispatch packet.
HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_↩ DYNAMIC_CALLSTACK	Dynamic callstack flag. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is bool. If this flag is set (the value is true), the kernel uses a dynamically sized call stack. This can happen if recursive calls, calls to indirect functions, or the HSAIL alloca instruction are present in the kernel.
HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_↩ CALL_CONVENTION	**Deprecated** Call convention of the kernel. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint32_t.
HSA_EXECUTABLE_SYMBOL_INFO_INDIRECT_↩ FUNCTION_OBJECT	Indirect function object handle. The value of this attribute is undefined if the symbol is not an indirect function, or the associated agent does not support the Full Profile. The type of this attribute depends on the machine model: the type is uint32_t for small machine model, and uint64_t for large model. If the state of the executable is HSA_EXECUTABLE_STATE_UNFROZEN, then 0 is returned.
HSA_EXECUTABLE_SYMBOL_INFO_INDIRECT_↩ FUNCTION_CALL_CONVENTION	**Deprecated** Call convention of the indirect function. The value of this attribute is undefined if the symbol is not an indirect function, or the associated agent does not support the Full Profile. The type of this attribute is uint32_t.

Definition at line 4801 of file hsa.h.

5.8.3.4 hsa_symbol_kind_t

`enum hsa_symbol_kind_t`

Symbol type.

Enumerator

HSA_SYMBOL_KIND_VARIABLE	Variable.
HSA_SYMBOL_KIND_KERNEL	Kernel.
HSA_SYMBOL_KIND_INDIRECT_FUNCTION	Indirect function.

Definition at line 4741 of file hsa.h.

5.8.3.5 hsa_symbol_linkage_t

`enum hsa_symbol_linkage_t`

Linkage type of a symbol.

Enumerator

HSA_SYMBOL_LINKAGE_MODULE	Module linkage.
HSA_SYMBOL_LINKAGE_PROGRAM	Program linkage.

Definition at line 4759 of file hsa.h.

5.8.3.6 hsa_variable_allocation_t

`enum hsa_variable_allocation_t`

Allocation type of a variable.

Enumerator

HSA_VARIABLE_ALLOCATION_AGENT	Agent allocation.
HSA_VARIABLE_ALLOCATION_PROGRAM	Program allocation.

Definition at line 4773 of file hsa.h.

5.8.3.7 hsa_variable_segment_t

`enum hsa_variable_segment_t`

Memory segment associated with a variable.

Enumerator

HSA_VARIABLE_SEGMENT_GLOBAL	Global memory segment.
HSA_VARIABLE_SEGMENT_READONLY	Readonly memory segment.

Definition at line 4787 of file hsa.h.

5.8.4 Function Documentation

5.8.4.1 hsa_code_object_reader_create_from_file()

```
hsa_status_t HSA_API hsa_code_object_reader_create_from_file (
            hsa_file_t file,
            hsa_code_object_reader_t * code_object_reader )
```

Create a code object reader to operate on a file.

Parameters

in	*file*	File descriptor. The file must have been opened by application with at least read permissions prior calling this function. The file must contain a vendor-specific code object.

The file is owned and managed by the application; the lifetime of the file descriptor must exceed that of any associated code object reader.

Parameters

out	*code_object_reader*	Memory location to store the newly created code object reader handle. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_FILE	`file` is invalid.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`code_object_reader` is NULL.

5.8.4.2 hsa_code_object_reader_create_from_memory()

```
hsa_status_t HSA_API hsa_code_object_reader_create_from_memory (
            const void * code_object,
```

```
            size_t size,
            hsa_code_object_reader_t * code_object_reader )
```

Create a code object reader to operate on memory.

Parameters

in	*code_object*	Memory buffer that contains a vendor-specific code object. The buffer is owned and managed by the application; the lifetime of the buffer must exceed that of any associated code object reader.
in	*size*	Size of the buffer pointed to by `code_object`. Must not be 0.
out	*code_object_reader*	Memory location to store newly created code object reader handle. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`code_object` is NULL, `size` is zero, or `code_object_reader` is NULL.

5.8.4.3 hsa_code_object_reader_destroy()

```
hsa_status_t HSA_API hsa_code_object_reader_destroy (
            hsa_code_object_reader_t code_object_reader )
```

Destroy a code object reader.

The code object reader handle becomes invalid after completion of this function. Any file or memory used to create the code object read is not closed, removed, or deallocated by this function.

Parameters

in	*code_object_reader*	Code object reader to destroy.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT_READER	`code_object_reader` is invalid.

5.8.4.4 hsa_executable_agent_global_variable_define()

```
hsa_status_t HSA_API hsa_executable_agent_global_variable_define (
            hsa_executable_t executable,
```

```
            hsa_agent_t agent,
            const char * variable_name,
            void * address )
```

Define an external global variable with agent allocation.

This function allows the application to provide the definition of a variable in the global segment memory with agent allocation. The variable must be defined before loading a code object into an executable. In addition, code objects loaded must not define the variable.

Parameters

in	*executable*	Executable. Must not be in frozen state.
in	*agent*	Agent for which the variable is being defined.
in	*variable_name*	Name of the variable. The Programmer's Reference Manual describes the standard name mangling scheme.
in	*address*	Address where the variable is defined. This address must have been previously allocated using hsa_memory_allocate in a global region that is only visible to `agent`. The application cannot deallocate the buffer pointed by `address` before `executable` is destroyed.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_INVALID_AGENT	`agent` is invalid.
HSA_STATUS_ERROR_VARIABLE_ALREADY_DEFINED	The variable is already defined.
HSA_STATUS_ERROR_INVALID_SYMBOL_NAME	There is no variable with the `variable_name`.
HSA_STATUS_ERROR_FROZEN_EXECUTABLE	`executable` is frozen.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`variable_name` is NULL.

5.8.4.5 hsa_executable_create()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_create (
            hsa_profile_t profile,
            hsa_executable_state_t executable_state,
            const char * options,
            hsa_executable_t * executable )
```

Create an empty executable.

Deprecated Use hsa_executable_create_alt instead, which allows the application to specify the default floating-point rounding mode of the executable and assumes an unfrozen initial state.

Parameters

in	*profile*	Profile used in the executable.
in	*executable_state*	Executable state. If the state is HSA_EXECUTABLE_STATE_FROZEN, the resulting executable is useless because no code objects can be loaded, and no variables can be defined.
in	*options*	Standard and vendor-specific options. Unknown options are ignored. A standard option begins with the "-hsa_" prefix. Options beginning with the "-hsa_ext_<extension_name>_" prefix are reserved for extensions. A vendor-specific option begins with the "-<vendor_name>_" prefix. Must be a NUL-terminated string. May be NULL.
out	*executable*	Memory location where the HSA runtime stores the newly created executable handle.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`profile` is invalid, or `executable` is NULL.

5.8.4.6 hsa_executable_create_alt()

```
hsa_status_t HSA_API hsa_executable_create_alt (
            hsa_profile_t profile,
            hsa_default_float_rounding_mode_t default_float_rounding_mode,
            const char * options,
            hsa_executable_t * executable )
```

Create an empty executable.

Parameters

in	*profile*	Profile used in the executable.
in	*default_float_rounding_mode*	Default floating-point rounding mode used in the executable. Allowed rounding modes are near and zero (default is not allowed).
in	*options*	Standard and vendor-specific options. Unknown options are ignored. A standard option begins with the "-hsa_" prefix. Options beginning with the "-hsa_ext_<extension_name>_" prefix are reserved for extensions. A vendor-specific option begins with the "-<vendor_name>_" prefix. Must be a NUL-terminated string. May be NULL.
out	*executable*	Memory location where the HSA runtime stores newly created executable handle. The initial state of the executable is HSA_EXECUTABLE_STATE_UNFROZEN.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.

Return values

HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`profile` is invalid, or `executable` is NULL.

5.8.4.7 hsa_executable_destroy()

```
hsa_status_t HSA_API hsa_executable_destroy (
            hsa_executable_t executable )
```

Destroy an executable.

An executable handle becomes invalid after the executable has been destroyed. Code object handles that were loaded into this executable are still valid after the executable has been destroyed, and can be used as intended. Resources allocated outside and associated with this executable (such as external global or readonly variables) can be released after the executable has been destroyed.

Executable should not be destroyed while kernels are in flight.

Parameters

in	*executable*	Executable.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.

5.8.4.8 hsa_executable_freeze()

```
hsa_status_t HSA_API hsa_executable_freeze (
            hsa_executable_t executable,
            const char * options )
```

Freeze the executable.

No modifications to executable can be made after freezing: no code objects can be loaded to the executable, and no external variables can be defined. Freezing the executable does not prevent querying the executable's attributes. The application must define all the external variables in an executable before freezing it.

Parameters

in	*executable*	Executable.
in	*options*	Standard and vendor-specific options. Unknown options are ignored. A standard option begins with the "-hsa_" prefix. Options beginning with the "-hsa_ext_<extension_name>_" prefix are reserved for extensions. A vendor-specific option begins with the "-<vendor_name>_" prefix. Must be a NUL-terminated string. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_VARIABLE_UNDEFINED	One or more variables are undefined in the executable.
HSA_STATUS_ERROR_FROZEN_EXECUTABLE	`executable` is already frozen.

5.8.4.9 hsa_executable_get_info()

```
hsa_status_t HSA_API hsa_executable_get_info (
            hsa_executable_t executable,
            hsa_executable_info_t attribute,
            void * value )
```

Get the current value of an attribute for a given executable.

Parameters

`in`	*executable*	Executable.
`in`	*attribute*	Attribute to query.
`out`	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid executable attribute, or `value` is NULL.

5.8.4.10 hsa_executable_get_symbol()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_get_symbol (
            hsa_executable_t executable,
            const char * module_name,
            const char * symbol_name,
            hsa_agent_t agent,
            int32_t call_convention,
            hsa_executable_symbol_t * symbol )
```

Get the symbol handle for a given a symbol name.

Deprecated Use hsa_executable_get_symbol_by_name instead.

Parameters

in	*executable*	Executable.
in	*module_name*	Module name. Must be NULL if the symbol has program linkage.
in	*symbol_name*	Symbol name.
in	*agent*	Agent associated with the symbol. If the symbol is independent of any agent (for example, a variable with program allocation), this argument is ignored.
in	*call_convention*	Call convention associated with the symbol. If the symbol does not correspond to an indirect function, this argument is ignored.
out	*symbol*	Memory location where the HSA runtime stores the symbol handle.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_INVALID_SYMBOL_NAME	There is no symbol with a name that matches `symbol_name`.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`symbol_name` is NULL, or `symbol` is NULL.

5.8.4.11 hsa_executable_get_symbol_by_name()

```
hsa_status_t HSA_API hsa_executable_get_symbol_by_name (
            hsa_executable_t executable,
            const char * symbol_name,
            const hsa_agent_t * agent,
            hsa_executable_symbol_t * symbol )
```

Retrieve the symbol handle corresponding to a given a symbol name.

Parameters

in	*executable*	Executable.
in	*symbol_name*	Symbol name. Must be a NUL-terminated character array. The Programmer's Reference Manual describes the standard name mangling scheme.
in	*agent*	Pointer to the agent for which the symbol with the given name is defined. If the symbol corresponding to the given name has program allocation, `agent` must be NULL.
out	*symbol*	Memory location where the HSA runtime stores the symbol handle. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_INVALID_SYMBOL_NAME	There is no symbol with a name that matches `symbol_name`.

Return values

HSA_STATUS_ERROR_INVALID_ARGUMENT	`symbol_name` is NULL, or `symbol` is NULL.

5.8.4.12 hsa_executable_global_variable_define()

```
hsa_status_t HSA_API hsa_executable_global_variable_define (
            hsa_executable_t executable,
            const char * variable_name,
            void * address )
```

Define an external global variable with program allocation.

This function allows the application to provide the definition of a variable in the global segment memory with program allocation. The variable must be defined before loading a code object into an executable. In addition, code objects loaded must not define the variable.

Parameters

`in`	*executable*	Executable. Must not be in frozen state.
`in`	*variable_name*	Name of the variable. The Programmer's Reference Manual describes the standard name mangling scheme.
`in`	*address*	Address where the variable is defined. This address must be in global memory and can be read and written by any agent in the system. The application cannot deallocate the buffer pointed by `address` before `executable` is destroyed.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_VARIABLE_ALREADY_DEFINED	The variable is already defined.
HSA_STATUS_ERROR_INVALID_SYMBOL_NAME	There is no variable with the `variable_name`.
HSA_STATUS_ERROR_FROZEN_EXECUTABLE	`executable` is frozen.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`variable_name` is NULL.

5.8.4.13 hsa_executable_iterate_agent_symbols()

```
hsa_status_t HSA_API hsa_executable_iterate_agent_symbols (
            hsa_executable_t executable,
            hsa_agent_t agent,
            hsa_status_t(*)(hsa_executable_t exec, hsa_agent_t agent, hsa_executable_symbol_t
symbol, void *data) callback,
            void * data )
```

Iterate over the kernels, indirect functions, and agent allocation variables in an executable for a given agent, and invoke an application- defined callback on every iteration.

Parameters

in	*executable*	Executable.
in	*agent*	Agent.
in	*callback*	Callback to be invoked once per executable symbol. The HSA runtime passes three arguments to the callback: the executable, a symbol, and the application data. If `callback` returns a status other than HSA_STATUS_SUCCESS for a particular iteration, the traversal stops and hsa_executable_iterate_symbols returns that status value.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`callback` is NULL.

5.8.4.14 hsa_executable_iterate_program_symbols()

```
hsa_status_t HSA_API hsa_executable_iterate_program_symbols (
            hsa_executable_t executable,
            hsa_status_t(*)(hsa_executable_t exec, hsa_executable_symbol_t symbol, void *data)
callback,
            void * data )
```

Iterate over the program allocation variables in an executable, and invoke an application-defined callback on every iteration.

Parameters

in	*executable*	Executable.
in	*callback*	Callback to be invoked once per executable symbol. The HSA runtime passes three arguments to the callback: the executable, a symbol, and the application data. If `callback` returns a status other than HSA_STATUS_SUCCESS for a particular iteration, the traversal stops and hsa_executable_iterate_symbols returns that status value.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`callback` is NULL.

5.8.4.15 hsa_executable_iterate_symbols()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_iterate_symbols (
            hsa_executable_t executable,
            hsa_status_t(*)(hsa_executable_t exec, hsa_executable_symbol_t symbol, void *data)
callback,
            void * data )
```

Iterate over the symbols in a executable, and invoke an application-defined callback on every iteration.

Deprecated

Parameters

in	*executable*	Executable.
in	*callback*	Callback to be invoked once per executable symbol. The HSA runtime passes three arguments to the callback: the executable, a symbol, and the application data. If `callback` returns a status other than HSA_STATUS_SUCCESS for a particular iteration, the traversal stops and hsa_executable_iterate_symbols returns that status value.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`callback` is NULL.

5.8.4.16 hsa_executable_load_agent_code_object()

```
hsa_status_t HSA_API hsa_executable_load_agent_code_object (
            hsa_executable_t executable,
            hsa_agent_t agent,
            hsa_code_object_reader_t code_object_reader,
            const char * options,
            hsa_loaded_code_object_t * loaded_code_object )
```

Load an agent code object into an executable.

The agent code object contains all defined agent allocation variables, functions, indirect functions, and kernels in a given program for a given instruction set architecture.

Any module linkage declaration must have been defined either by a define variable or by loading a code object that has a symbol with module linkage definition.

The default floating-point rounding mode of the code object associated with `code_object_reader` must match that of the executable (HSA_EXECUTABLE_INFO_DEFAULT_FLOAT_ROUNDING_MODE), or be default (in which case the value of HSA_EXECUTABLE_INFO_DEFAULT_FLOAT_ROUNDING_MODE is used). If the agent code object uses extensions, the implementation and the agent must support them for this operation to return successfully.

Parameters

in	*executable*	Executable.
in	*agent*	Agent to load code object for. A code object can be loaded into an executable at most once for a given agent. The instruction set architecture of the code object must be supported by the agent.
in	*code_object_reader*	A code object reader that holds the code object to load. If a code object reader is destroyed before all the associated executables are destroyed, the behavior is undefined.
in	*options*	Standard and vendor-specific options. Unknown options are ignored. A standard option begins with the "-hsa_" prefix. Options beginning with the "-hsa_ext_<extension_name>_" prefix are reserved for extensions. A vendor-specific option begins with the "-<vendor_name>_" prefix. Must be a NUL-terminated string. May be NULL.
out	*loaded_code_object*	Pointer to a memory location where the HSA runtime stores the loaded code object handle. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_FROZEN_EXECUTABLE	The executable is frozen.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT_READER	`code_object_reader` is invalid.
HSA_STATUS_ERROR_INCOMPATIBLE_ARGUMENTS	The code object read by `code_object_reader` is not compatible with the agent (for example, the agent does not support the instruction set architecture of the code object), the executable (for example, there is a default floating-point mode mismatch between the two), or the implementation.

5.8.4.17 hsa_executable_load_program_code_object()

```
hsa_status_t HSA_API hsa_executable_load_program_code_object (
            hsa_executable_t executable,
            hsa_code_object_reader_t code_object_reader,
            const char * options,
            hsa_loaded_code_object_t * loaded_code_object )
```

Load a program code object into an executable.

A program code object contains information about resources that are accessible by all kernel agents that run the executable, and can be loaded at most once into an executable.

If the program code object uses extensions, the implementation must support them for this operation to return successfully.

Parameters

in	*executable*	Executable.
in	*code_object_reader*	A code object reader that holds the program code object to load. If a code object reader is destroyed before all the associated executables are destroyed, the behavior is undefined.
in	*options*	Standard and vendor-specific options. Unknown options are ignored. A standard option begins with the "-hsa_" prefix. Options beginning with the "-hsa_ext_<extension_name>_" prefix are reserved for extensions. A vendor-specific option begins with the "-<vendor_name>_" prefix. Must be a NUL-terminated string. May be NULL.
out	*loaded_code_object*	Pointer to a memory location where the HSA runtime stores the loaded code object handle. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_FROZEN_EXECUTABLE	The executable is frozen.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT_READER	`code_object_reader` is invalid.
HSA_STATUS_ERROR_INCOMPATIBLE_ARGUMENTS	The program code object is not compatible with the executable or the implementation (for example, the code object uses an extension that is not supported by the implementation).

5.8.4.18 hsa_executable_readonly_variable_define()

```
hsa_status_t HSA_API hsa_executable_readonly_variable_define (
            hsa_executable_t executable,
            hsa_agent_t agent,
            const char * variable_name,
            void * address )
```

Define an external readonly variable.

This function allows the application to provide the definition of a variable in the readonly segment memory. The variable must be defined before loading a code object into an executable. In addition, code objects loaded must not define the variable.

Parameters

in	*executable*	Executable. Must not be in frozen state.
in	*agent*	Agent for which the variable is being defined.
in	*variable_name*	Name of the variable. The Programmer's Reference Manual describes the standard name mangling scheme.
in	*address*	Address where the variable is defined. This address must have been previously allocated using hsa_memory_allocate in a readonly region associated with `agent`. The application cannot deallocate the buffer pointed by `address` before `executable` is destroyed.
in	*address*	Address where the variable is defined. The buffer pointed by `address` is owned by the application, and cannot be deallocated before `executable` is destroyed.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	Executable is invalid.
HSA_STATUS_ERROR_INVALID_AGENT	`agent` is invalid.
HSA_STATUS_ERROR_VARIABLE_ALREADY_DEFINED	The variable is already defined.
HSA_STATUS_ERROR_INVALID_SYMBOL_NAME	There is no variable with the `variable_name`.
HSA_STATUS_ERROR_FROZEN_EXECUTABLE	`executable` is frozen.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`variable_name` is NULL.

5.8.4.19 hsa_executable_symbol_get_info()

```
hsa_status_t HSA_API hsa_executable_symbol_get_info (
            hsa_executable_symbol_t executable_symbol,
            hsa_executable_symbol_info_t attribute,
            void * value )
```

Get the current value of an attribute for a given executable symbol.

Parameters

`in`	*executable_symbol*	Executable symbol.
`in`	*attribute*	Attribute to query.
`out`	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE_SYMBOL	The executable symbol is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid executable symbol attribute, or `value` is NULL.

5.8.4.20 hsa_executable_validate()

```
hsa_status_t HSA_API hsa_executable_validate (
            hsa_executable_t executable,
            uint32_t * result )
```

Validate an executable. Checks that all code objects have matching machine model, profile, and default floating-point rounding mode. Checks that all declarations have definitions. Checks declaration-definition compatibility (see the HSA Programming Reference Manual for compatibility rules). Invoking this function is equivalent to invoking hsa_executable_validate_alt with no options.

Parameters

in	*executable*	Executable. Must be in frozen state.
out	*result*	Memory location where the HSA runtime stores the validation result. If the executable passes validation, the result is 0.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	`executable` is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`result` is NULL.

5.8.4.21 hsa_executable_validate_alt()

```
hsa_status_t HSA_API hsa_executable_validate_alt (
            hsa_executable_t executable,
            const char * options,
            uint32_t * result )
```

Validate an executable. Checks that all code objects have matching machine model, profile, and default floating-point rounding mode. Checks that all declarations have definitions. Checks declaration-definition compatibility (see the HSA Programming Reference Manual for compatibility rules).

Parameters

in	*executable*	Executable. Must be in frozen state.
in	*options*	Standard and vendor-specific options. Unknown options are ignored. A standard option begins with the "-hsa_" prefix. Options beginning with the "-hsa_ext_<extension_name>_" prefix are reserved for extensions. A vendor-specific option begins with the "-<vendor_name>_" prefix. Must be a NUL-terminated string. May be NULL.
out	*result*	Memory location where the HSA runtime stores the validation result. If the executable passes validation, the result is 0.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	`executable` is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`result` is NULL.

5.9 Code Objects (deprecated).

Classes

- struct hsa_code_object_s

  Struct containing an opaque handle to a code object, which contains ISA for finalized kernels and indirect functions together with information about the global or readonly segment variables they reference.
- struct hsa_callback_data_s

  Application data handle that is passed to the serialization and deserialization functions.
- struct hsa_code_symbol_s

  Code object symbol handle.

Typedefs

- typedef struct hsa_code_object_s hsa_code_object_t

  Struct containing an opaque handle to a code object, which contains ISA for finalized kernels and indirect functions together with information about the global or readonly segment variables they reference.
- typedef struct hsa_callback_data_s hsa_callback_data_t

  Application data handle that is passed to the serialization and deserialization functions.
- typedef struct hsa_code_symbol_s hsa_code_symbol_t

  Code object symbol handle.

Enumerations

- enum hsa_code_object_type_t { HSA_CODE_OBJECT_TYPE_PROGRAM = 0 }

  Code object type.
- enum hsa_code_object_info_t {
  HSA_CODE_OBJECT_INFO_VERSION = 0 , HSA_CODE_OBJECT_INFO_TYPE = 1 , HSA_CODE_OBJECT_INFO_ISA = 2 , HSA_CODE_OBJECT_INFO_MACHINE_MODEL = 3 ,
  HSA_CODE_OBJECT_INFO_PROFILE = 4 , HSA_CODE_OBJECT_INFO_DEFAULT_FLOAT_ROUNDING_MODE = 5 }

  Code object attributes.
- enum hsa_code_symbol_info_t {
  HSA_CODE_SYMBOL_INFO_TYPE = 0 , HSA_CODE_SYMBOL_INFO_NAME_LENGTH = 1 ,
  HSA_CODE_SYMBOL_INFO_NAME = 2 , HSA_CODE_SYMBOL_INFO_MODULE_NAME_LENGTH = 3 ,
  HSA_CODE_SYMBOL_INFO_MODULE_NAME = 4 , HSA_CODE_SYMBOL_INFO_LINKAGE = 5 ,
  HSA_CODE_SYMBOL_INFO_IS_DEFINITION = 17 , HSA_CODE_SYMBOL_INFO_VARIABLE_ALLOCATION = 6 ,
  HSA_CODE_SYMBOL_INFO_VARIABLE_SEGMENT = 7 , HSA_CODE_SYMBOL_INFO_VARIABLE_ALIGNMENT = 8 , HSA_CODE_SYMBOL_INFO_VARIABLE_SIZE = 9 , HSA_CODE_SYMBOL_INFO_VARIABLE_IS_CONST = 10 ,
  HSA_CODE_SYMBOL_INFO_KERNEL_KERNARG_SEGMENT_SIZE = 11 , HSA_CODE_SYMBOL_INFO_KERNEL_KERNA = 12 , HSA_CODE_SYMBOL_INFO_KERNEL_GROUP_SEGMENT_SIZE = 13 , HSA_CODE_SYMBOL_INFO_KERNEL_PRI = 14 ,
  HSA_CODE_SYMBOL_INFO_KERNEL_DYNAMIC_CALLSTACK = 15 , HSA_CODE_SYMBOL_INFO_KERNEL_CALL_CONV = 18 , HSA_CODE_SYMBOL_INFO_INDIRECT_FUNCTION_CALL_CONVENTION = 16 }

  Code object symbol attributes.

Functions

- hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_serialize (hsa_code_object_t code_object, hsa_status_t(∗alloc_callback)(size_t size, hsa_callback_data_t data, void ∗∗address), hsa_callback_data_t callback_data, const char ∗options, void ∗∗serialized_code_object, size_t ∗serialized_code_object_size)

  Serialize a code object. Can be used for offline finalization, install-time finalization, disk code caching, etc.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_deserialize (void ∗serialized_code_object, size_t serialized_code_object_size, const char ∗options, hsa_code_object_t ∗code_object)

  Deserialize a code object.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_destroy (hsa_code_object_t code_object)

  Destroy a code object.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_get_info (hsa_code_object_t code_object, hsa_code_object_info_t attribute, void ∗value)

  Get the current value of an attribute for a given code object.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_load_code_object (hsa_executable_t executable, hsa_agent_t agent, hsa_code_object_t code_object, const char ∗options)

  Load code object into the executable.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_get_symbol (hsa_code_object_t code_↩ object, const char ∗symbol_name, hsa_code_symbol_t ∗symbol)

  Get the symbol handle within a code object for a given a symbol name.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_get_symbol_from_name (hsa_code_object_t code_object, const char ∗module_name, const char ∗symbol_name, hsa_code_symbol_t ∗symbol)

  Get the symbol handle within a code object for a given a symbol name.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_code_symbol_get_info (hsa_code_symbol_t code_↩ symbol, hsa_code_symbol_info_t attribute, void ∗value)

  Get the current value of an attribute for a given code symbol.
- hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_iterate_symbols (hsa_code_object_t code↩ _object, hsa_status_t(∗callback)(hsa_code_object_t code_object, hsa_code_symbol_t symbol, void ∗data), void ∗data)

  Iterate over the symbols in a code object, and invoke an application-defined callback on every iteration.

5.9.1 Detailed Description

5.9.2 Typedef Documentation

5.9.2.1 hsa_callback_data_t

```
typedef struct hsa_callback_data_s hsa_callback_data_t
```

Application data handle that is passed to the serialization and deserialization functions.

Deprecated

5.9.2.2 hsa_code_object_t

```
typedef struct hsa_code_object_s hsa_code_object_t
```

Struct containing an opaque handle to a code object, which contains ISA for finalized kernels and indirect functions together with information about the global or readonly segment variables they reference.

Deprecated

5.9.2.3 hsa_code_symbol_t

```
typedef struct hsa_code_symbol_s hsa_code_symbol_t
```

Code object symbol handle.

Deprecated

The lifetime of a code object symbol matches that of the code object associated with it. An operation on a symbol whose associated code object has been destroyed results in undefined behavior.

5.9.3 Enumeration Type Documentation

5.9.3.1 hsa_code_object_info_t

```
enum hsa_code_object_info_t
```

Code object attributes.

Deprecated

Enumerator

HSA_CODE_OBJECT_INFO_VERSION	The version of the code object. The type of this attribute is a NUL-terminated char[64]. The name must be at most 63 characters long (not including the NUL terminator) and all array elements not used for the name must be NUL.
HSA_CODE_OBJECT_INFO_TYPE	Type of code object. The type of this attribute is hsa_code_object_type_t.
HSA_CODE_OBJECT_INFO_ISA	Instruction set architecture this code object is produced for. The type of this attribute is hsa_isa_t.
HSA_CODE_OBJECT_INFO_MACHINE_MODEL	Machine model this code object is produced for. The type of this attribute is hsa_machine_model_t.
HSA_CODE_OBJECT_INFO_PROFILE	Profile this code object is produced for. The type of this attribute is hsa_profile_t.
HSA_CODE_OBJECT_INFO_DEFAULT_FLOAT_↩ ROUNDING_MODE	Default floating-point rounding mode used when the code object is produced. The type of this attribute is

Definition at line 5282 of file hsa.h.

5.9.3.2 hsa_code_object_type_t

`enum hsa_code_object_type_t`

Code object type.

Deprecated

Enumerator

HSA_CODE_OBJECT_TYPE_PROGRAM	Produces code object that contains ISA for all kernels and indirect functions in HSA source.

Definition at line 5269 of file hsa.h.

5.9.3.3 hsa_code_symbol_info_t

`enum hsa_code_symbol_info_t`

Code object symbol attributes.

Deprecated

Enumerator

HSA_CODE_SYMBOL_INFO_TYPE	The type of the symbol. The type of this attribute is hsa_symbol_kind_t.
HSA_CODE_SYMBOL_INFO_NAME_LENGTH	The length of the symbol name in bytes, not including the NUL terminator. The type of this attribute is uint32_t.
HSA_CODE_SYMBOL_INFO_NAME	The name of the symbol. The type of this attribute is character array with the length equal to the value of HSA_CODE_SYMBOL_INFO_NAME_LENGTH attribute.
HSA_CODE_SYMBOL_INFO_MODULE_NAME_↩ LENGTH	The length of the module name in bytes (not including the NUL terminator) to which this symbol belongs if this symbol has module linkage, otherwise 0 is returned. The type of this attribute is uint32_t.
HSA_CODE_SYMBOL_INFO_MODULE_NAME	The module name to which this symbol belongs if this symbol has module linkage, otherwise an empty string is returned. The type of this attribute is character array with the length equal to the value of HSA_CODE_SYMBOL_INFO_MODULE_NAME_LENGTH attribute.

Enumerator

HSA_CODE_SYMBOL_INFO_LINKAGE	The linkage kind of the symbol. The type of this attribute is hsa_symbol_linkage_t.
HSA_CODE_SYMBOL_INFO_IS_DEFINITION	Indicates whether the symbol corresponds to a definition. The type of this attribute is bool.
HSA_CODE_SYMBOL_INFO_VARIABLE_↩ ALLOCATION	The allocation kind of the variable. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is hsa_variable_allocation_t.
HSA_CODE_SYMBOL_INFO_VARIABLE_↩ SEGMENT	The segment kind of the variable. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is hsa_variable_segment_t.
HSA_CODE_SYMBOL_INFO_VARIABLE_↩ ALIGNMENT	Alignment of the symbol in memory. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is uint32_t. The current alignment of the variable in memory may be greater than the value specified in the source program variable declaration.
HSA_CODE_SYMBOL_INFO_VARIABLE_SIZE	Size of the variable. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is uint32_t. A size of 0 is returned if the variable is an external variable and has an unknown dimension.
HSA_CODE_SYMBOL_INFO_VARIABLE_IS_↩ CONST	Indicates whether the variable is constant. The value of this attribute is undefined if the symbol is not a variable. The type of this attribute is bool.
HSA_CODE_SYMBOL_INFO_KERNEL_↩ KERNARG_SEGMENT_SIZE	Size of kernarg segment memory that is required to hold the values of the kernel arguments, in bytes. Must be a multiple of 16. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint32_t.
HSA_CODE_SYMBOL_INFO_KERNEL_↩ KERNARG_SEGMENT_ALIGNMENT	Alignment (in bytes) of the buffer used to pass arguments to the kernel, which is the maximum of 16 and the maximum alignment of any of the kernel arguments. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint32_t.
HSA_CODE_SYMBOL_INFO_KERNEL_GROUP_↩ SEGMENT_SIZE	Size of static group segment memory required by the kernel (per work-group), in bytes. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint32_t. The reported amount does not include any dynamically allocated group segment memory that may be requested by the application when a kernel is dispatched.
HSA_CODE_SYMBOL_INFO_KERNEL_PRIVATE↩ _SEGMENT_SIZE	Size of static private, spill, and arg segment memory required by this kernel (per work-item), in bytes. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint32_t. If the value of HSA_CODE_SYMBOL_INFO_KERNEL_DYNAMIC_CALLSTACK is true, the kernel may use more private memory than the reported value, and the application must add the dynamic call stack usage to *private_segment_size* when populating a kernel dispatch packet.

Enumerator

HSA_CODE_SYMBOL_INFO_KERNEL_↩ DYNAMIC_CALLSTACK	Dynamic callstack flag. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is bool. If this flag is set (the value is true), the kernel uses a dynamically sized call stack. This can happen if recursive calls, calls to indirect functions, or the HSAIL alloca instruction are present in the kernel.
HSA_CODE_SYMBOL_INFO_KERNEL_CALL_↩ CONVENTION	Call convention of the kernel. The value of this attribute is undefined if the symbol is not a kernel. The type of this attribute is uint32_t.
HSA_CODE_SYMBOL_INFO_INDIRECT_↩ FUNCTION_CALL_CONVENTION	Call convention of the indirect function. The value of this attribute is undefined if the symbol is not an indirect function. The type of this attribute is uint32_t.

Definition at line 5489 of file hsa.h.

5.9.4 Function Documentation

5.9.4.1 hsa_code_object_deserialize()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_deserialize (
            void * serialized_code_object,
            size_t serialized_code_object_size,
            const char * options,
            hsa_code_object_t * code_object )
```

Deserialize a code object.

Deprecated

Parameters

in	*serialized_code_object*	A serialized code object. Must not be NULL.
in	*serialized_code_object_size*	The size (in bytes) of `serialized_code_object`. Must not be 0.
in	*options*	Standard and vendor-specific options. Unknown options are ignored. A standard option begins with the "-hsa_" prefix. Options beginning with the "-hsa_ext_<extension_name>_" prefix are reserved for extensions. A vendor-specific option begins with the "-<vendor_name>_" prefix. Must be a NUL-terminated string. May be NULL.
out	*code_object*	Memory location where the HSA runtime stores the deserialized code object.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`serialized_code_object`, or `code_object` are NULL, or `serialized_code_object_size` is 0.

5.9.4.2 hsa_code_object_destroy()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_destroy (
            hsa_code_object_t code_object )
```

Destroy a code object.

Deprecated

The lifetime of a code object must exceed that of any executable where it has been loaded. If an executable that loaded `code_object` has not been destroyed, the behavior is undefined.

Parameters

`in`	*code_object*	Code object. The handle becomes invalid after it has been destroyed.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT	`code_object` is invalid.

5.9.4.3 hsa_code_object_get_info()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_get_info (
            hsa_code_object_t code_object,
            hsa_code_object_info_t attribute,
            void * value )
```

Get the current value of an attribute for a given code object.

Deprecated

Parameters

in	*code_object*	Code object.
in	*attribute*	Attribute to query.
out	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT	`code_object` is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid code object attribute, or `value` is NULL.

5.9.4.4 hsa_code_object_get_symbol()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_get_symbol (
            hsa_code_object_t code_object,
            const char * symbol_name,
            hsa_code_symbol_t * symbol )
```

Get the symbol handle within a code object for a given a symbol name.

Deprecated

Parameters

in	*code_object*	Code object.
in	*symbol_name*	Symbol name.
out	*symbol*	Memory location where the HSA runtime stores the symbol handle.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT	`code_object` is invalid.
HSA_STATUS_ERROR_INVALID_SYMBOL_NAME	There is no symbol with a name that matches `symbol_name`.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`symbol_name` is NULL, or `symbol` is NULL.

5.9.4.5 hsa_code_object_get_symbol_from_name()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_get_symbol_from_name (
            hsa_code_object_t code_object,
            const char * module_name,
            const char * symbol_name,
            hsa_code_symbol_t * symbol )
```

Get the symbol handle within a code object for a given a symbol name.

Deprecated

Parameters

in	*code_object*	Code object.
in	*module_name*	Module name. Must be NULL if the symbol has program linkage.
in	*symbol_name*	Symbol name.
out	*symbol*	Memory location where the HSA runtime stores the symbol handle.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT	`code_object` is invalid.
HSA_STATUS_ERROR_INVALID_SYMBOL_NAME	There is no symbol with a name that matches `symbol_name`.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`symbol_name` is NULL, or `symbol` is NULL.

5.9.4.6 hsa_code_object_iterate_symbols()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_iterate_symbols (
            hsa_code_object_t code_object,
            hsa_status_t(*)(hsa_code_object_t code_object, hsa_code_symbol_t symbol, void
*data) callback,
            void * data )
```

Iterate over the symbols in a code object, and invoke an application-defined callback on every iteration.

Deprecated

Parameters

in	*code_object*	Code object.
in	*callback*	Callback to be invoked once per code object symbol. The HSA runtime passes three arguments to the callback: the code object, a symbol, and the application data. If `callback` returns a status other than HSA_STATUS_SUCCESS for a particular iteration, the traversal stops and hsa_code_object_iterate_symbols returns that status value.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT	`code_object` is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`callback` is NULL.

5.9.4.7 hsa_code_object_serialize()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_serialize (
            hsa_code_object_t code_object,
            hsa_status_t(*)(size_t size, hsa_callback_data_t data, void **address) alloc_↩
callback,
            hsa_callback_data_t callback_data,
            const char * options,
            void ** serialized_code_object,
            size_t * serialized_code_object_size )
```

Serialize a code object. Can be used for offline finalization, install-time finalization, disk code caching, etc.

Deprecated

Parameters

in	*code_object*	Code object.
in	*alloc_callback*	Callback function for memory allocation. Must not be NULL. The HSA runtime passes three arguments to the callback: the allocation size, the application data, and a pointer to a memory location where the application stores the allocation result. The HSA runtime invokes `alloc_callback` once to allocate a buffer that contains the serialized version of `code_object`. If the callback returns a status code other than HSA_STATUS_SUCCESS, this function returns the same code.
in	*callback_data*	Application data that is passed to `alloc_callback`. May be NULL.
in	*options*	Standard and vendor-specific options. Unknown options are ignored. A standard option begins with the "-hsa_" prefix. Options beginning with the "-hsa_ext_<extension_name>_" prefix are reserved for extensions. A vendor-specific option begins with the "-<vendor_name>_" prefix. Must be a NUL-terminated string. May be NULL.
out	*serialized_code_object*	Memory location where the HSA runtime stores a pointer to the serialized code object. Must not be NULL.
out	*serialized_code_object_size*	Memory location where the HSA runtime stores the size (in bytes) of `serialized_code_object`. The returned value matches the allocation size passed by the HSA runtime to `alloc_callback`. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT	`code_object` is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`alloc_callback`, `serialized_code_object`, or `serialized_code_object_size` are NULL.

5.9.4.8 hsa_code_symbol_get_info()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_code_symbol_get_info (
            hsa_code_symbol_t code_symbol,
            hsa_code_symbol_info_t attribute,
            void * value )
```

Get the current value of an attribute for a given code symbol.

Deprecated

Parameters

`in`	*code_symbol*	Code symbol.
`in`	*attribute*	Attribute to query.
`out`	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_CODE_SYMBOL	The code symbol is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid code symbol attribute, or `value` is NULL.

5.9.4.9 hsa_executable_load_code_object()

```
hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_load_code_object (
            hsa_executable_t executable,
            hsa_agent_t agent,
```

```
hsa_code_object_t code_object,
const char * options )
```

Load code object into the executable.

Deprecated

Every global or readonly variable that is external must be defined before loading the code object. An internal global or readonly variable is allocated once the code object, that is being loaded, references this variable and this variable is not allocated.

Any module linkage declaration must have been defined either by a define variable or by loading a code object that has a symbol with module linkage definition.

Parameters

in	*executable*	Executable.
in	*agent*	Agent to load code object for. The agent must support the default floating-point rounding mode used by `code_object`.
in	*code_object*	Code object to load. The lifetime of the code object must exceed that of the executable: if `code_object` is destroyed before `executable`, the behavior is undefined.
in	*options*	Standard and vendor-specific options. Unknown options are ignored. A standard option begins with the "-hsa_" prefix. Options beginning with the "-hsa_ext_<extension_name>_" prefix are reserved for extensions. A vendor-specific option begins with the "-<vendor_name>_" prefix. Must be a NUL-terminated string. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_EXECUTABLE	The executable is invalid.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_CODE_OBJECT	`code_object` is invalid.
HSA_STATUS_ERROR_INCOMPATIBLE_ARGUMENTS	`agent` is not compatible with `code_object` (for example, `agent` does not support the default floating-point rounding mode specified by `code_object`), or `code_object` is not compatible with `executable` (for example, `code_object` and `executable` have different machine models or profiles).
HSA_STATUS_ERROR_FROZEN_EXECUTABLE	`executable` is frozen.

5.10 Finalization Extensions

Enumerations

- enum { HSA_EXT_STATUS_ERROR_INVALID_PROGRAM = 0x2000 , HSA_EXT_STATUS_ERROR_INVALID_MODULE

= 0x2001 , HSA_EXT_STATUS_ERROR_INCOMPATIBLE_MODULE = 0x2002 , HSA_EXT_STATUS_ERROR_MODULE_ALR
= 0x2003 ,
HSA_EXT_STATUS_ERROR_SYMBOL_MISMATCH = 0x2004 , HSA_EXT_STATUS_ERROR_FINALIZATION_FAILED
= 0x2005 , HSA_EXT_STATUS_ERROR_DIRECTIVE_MISMATCH = 0x2006 }

Enumeration constants added to hsa_status_t by this extension.

5.10.1 Detailed Description

5.10.2 Enumeration Type Documentation

5.10.2.1 anonymous enum

```
anonymous enum
```

Enumeration constants added to hsa_status_t by this extension.

Enumerator

HSA_EXT_STATUS_ERROR_INVALID_PROGRAM	The HSAIL program is invalid.
HSA_EXT_STATUS_ERROR_INVALID_MODULE	The HSAIL module is invalid.
HSA_EXT_STATUS_ERROR_INCOMPATIBLE_↩ MODULE	Machine model or profile of the HSAIL module do not match the machine model or profile of the HSAIL program.
HSA_EXT_STATUS_ERROR_MODULE_↩ ALREADY_INCLUDED	The HSAIL module is already a part of the HSAIL program.
HSA_EXT_STATUS_ERROR_SYMBOL_MISMATCH	Compatibility mismatch between symbol declaration and symbol definition.
HSA_EXT_STATUS_ERROR_FINALIZATION_↩ FAILED	The finalization encountered an error while finalizing a kernel or indirect function.
HSA_EXT_STATUS_ERROR_DIRECTIVE_↩ MISMATCH	Mismatch between a directive in the control directive structure and in the HSAIL kernel.

Definition at line 69 of file hsa_ext_finalize.h.

5.11 Finalization Program

Classes

- struct hsa_ext_program_s

  An opaque handle to a HSAIL program, which groups a set of HSAIL modules that collectively define functions and variables used by kernels and indirect functions.
- struct hsa_ext_control_directives_s

  Control directives specify low-level information about the finalization process.

Typedefs

- typedef BrigModule_t hsa_ext_module_t

  HSAIL (BRIG) module. The HSA Programmer's Reference Manual contains the definition of the BrigModule_t type.
- typedef struct hsa_ext_program_s **hsa_ext_program_t**

  An opaque handle to a HSAIL program, which groups a set of HSAIL modules that collectively define functions and variables used by kernels and indirect functions.
- typedef struct hsa_ext_control_directives_s **hsa_ext_control_directives_t**

  Control directives specify low-level information about the finalization process.

Enumerations

- enum hsa_ext_program_info_t { HSA_EXT_PROGRAM_INFO_MACHINE_MODEL = 0 , HSA_EXT_PROGRAM_INFO_PROFI = 1 , HSA_EXT_PROGRAM_INFO_DEFAULT_FLOAT_ROUNDING_MODE = 2 }

  HSAIL program attributes.
- enum hsa_ext_finalizer_call_convention_t { HSA_EXT_FINALIZER_CALL_CONVENTION_AUTO = -1 }

  Finalizer-determined call convention.

Functions

- hsa_status_t HSA_API hsa_ext_program_create (hsa_machine_model_t machine_model, hsa_profile_t profile, hsa_default_float_rounding_mode_t default_float_rounding_mode, const char ∗options, hsa_ext_program_t ∗program)

  Create an empty HSAIL program.
- hsa_status_t HSA_API hsa_ext_program_destroy (hsa_ext_program_t program)

  Destroy a HSAIL program.
- hsa_status_t HSA_API hsa_ext_program_add_module (hsa_ext_program_t program, hsa_ext_module_t module)

  Add a HSAIL module to an existing HSAIL program.
- hsa_status_t HSA_API hsa_ext_program_iterate_modules (hsa_ext_program_t program, hsa_status_t(∗callback)(hsa_ext_prog program, hsa_ext_module_t module, void ∗data), void ∗data)

  Iterate over the HSAIL modules in a program, and invoke an application-defined callback on every iteration.
- hsa_status_t HSA_API hsa_ext_program_get_info (hsa_ext_program_t program, hsa_ext_program_info_t attribute, void ∗value)

  Get the current value of an attribute for a given HSAIL program.
- hsa_status_t HSA_API hsa_ext_program_finalize (hsa_ext_program_t program, hsa_isa_t isa, int32_t call↩ _convention, hsa_ext_control_directives_t control_directives, const char ∗options, hsa_code_object_type_t code_object_type, hsa_code_object_t ∗code_object)

  Finalize an HSAIL program for a given instruction set architecture.

5.11.1 Detailed Description

5.11.2 Typedef Documentation

5.11.2.1 hsa_ext_module_t

```
typedef BrigModule_t hsa_ext_module_t
```

HSAIL (BRIG) module. The HSA Programmer's Reference Manual contains the definition of the BrigModule_t type.

Definition at line 113 of file hsa_ext_finalize.h.

5.11.3 Enumeration Type Documentation

5.11.3.1 hsa_ext_finalizer_call_convention_t

```
enum hsa_ext_finalizer_call_convention_t
```

Finalizer-determined call convention.

Enumerator

HSA_EXT_FINALIZER_CALL_CONVENTION_AUTO	Finalizer-determined call convention.

Definition at line 311 of file hsa_ext_finalize.h.

5.11.3.2 hsa_ext_program_info_t

```
enum hsa_ext_program_info_t
```

HSAIL program attributes.

Enumerator

HSA_EXT_PROGRAM_INFO_MACHINE_MODEL	Machine model specified when the HSAIL program was created. The type of this attribute is hsa_machine_model_t.
HSA_EXT_PROGRAM_INFO_PROFILE	Profile specified when the HSAIL program was created. The type of this attribute is hsa_profile_t.
HSA_EXT_PROGRAM_INFO_DEFAULT_FLOAT_↩ ROUNDING_MODE	Default float rounding mode specified when the HSAIL program was created. The type of this attribute is hsa_default_float_rounding_mode_t.

Definition at line 264 of file hsa_ext_finalize.h.

5.11.4 Function Documentation

5.11.4.1 hsa_ext_program_add_module()

```
hsa_status_t HSA_API hsa_ext_program_add_module (
             hsa_ext_program_t program,
             hsa_ext_module_t module )
```

Add a HSAIL module to an existing HSAIL program.

The HSA runtime does not perform a deep copy of the HSAIL module upon addition. Instead, it stores a pointer to the HSAIL module. The ownership of the HSAIL module belongs to the application, which must ensure that `module` is not released before destroying the HSAIL program.

The HSAIL module is successfully added to the HSAIL program if `module` is valid, if all the declarations and definitions for the same symbol are compatible, and if `module` specify machine model and profile that matches the HSAIL program.

Parameters

`in`	*program*	HSAIL program.
`in`	*module*	HSAIL module. The application can add the same HSAIL module to `program` at most once. The HSAIL module must specify the same machine model and profile as `program`. If the floating-mode rounding mode of `module` is not default, then it should match that of `program`.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	There is a failure to allocate resources required for the operation.
HSA_EXT_STATUS_ERROR_INVALID_PROGRAM	The HSAIL program is invalid.
HSA_EXT_STATUS_ERROR_INVALID_MODULE	The HSAIL module is invalid.
HSA_EXT_STATUS_ERROR_INCOMPATIBLE_MODULE	The machine model of `module` does not match machine model of `program`, or the profile of `module` does not match profile of `program`.
HSA_EXT_STATUS_ERROR_MODULE_ALREADY_INCLUDED	The HSAIL module is already a part of the HSAIL program.
HSA_EXT_STATUS_ERROR_SYMBOL_MISMATCH	Symbol declaration and symbol definition compatibility mismatch. See the symbol compatibility rules in the HSA Programming Reference Manual.

5.11.4.2 hsa_ext_program_create()

```
hsa_status_t HSA_API hsa_ext_program_create (
             hsa_machine_model_t machine_model,
             hsa_profile_t profile,
             hsa_default_float_rounding_mode_t default_float_rounding_mode,
             const char * options,
             hsa_ext_program_t * program )
```

Create an empty HSAIL program.

Parameters

in	*machine_model*	Machine model used in the HSAIL program.
in	*profile*	Profile used in the HSAIL program.
in	*default_float_rounding_mode*	Default float rounding mode used in the HSAIL program.
in	*options*	Vendor-specific options. May be NULL.
out	*program*	Memory location where the HSA runtime stores the newly created HSAIL program handle.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	There is a failure to allocate resources required for the operation.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`machine_model` is invalid, `profile` is invalid, `default_float_rounding_mode` is invalid, or `program` is NULL.

5.11.4.3 hsa_ext_program_destroy()

```
hsa_status_t HSA_API hsa_ext_program_destroy (
            hsa_ext_program_t program )
```

Destroy a HSAIL program.

The HSAIL program handle becomes invalid after it has been destroyed. Code object handles produced by hsa_ext_program_finalize are still valid after the HSAIL program has been destroyed, and can be used as intended. Resources allocated outside and associated with the HSAIL program (such as HSAIL modules that are added to the HSAIL program) can be released after the finalization program has been destroyed.

Parameters

in	*program*	HSAIL program.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_EXT_STATUS_ERROR_INVALID_PROGRAM	The HSAIL program is invalid.

5.11.4.4 hsa_ext_program_finalize()

```
hsa_status_t HSA_API hsa_ext_program_finalize (
            hsa_ext_program_t program,
```

```
            hsa_isa_t isa,
            int32_t call_convention,
            hsa_ext_control_directives_t control_directives,
            const char * options,
            hsa_code_object_type_t code_object_type,
            hsa_code_object_t * code_object )
```

Finalize an HSAIL program for a given instruction set architecture.

Finalize all of the kernels and indirect functions that belong to the same HSAIL program for a specific instruction set architecture (ISA). The transitive closure of all functions specified by call or scall must be defined. Kernels and indirect functions that are being finalized must be defined. Kernels and indirect functions that are referenced in kernels and indirect functions being finalized may or may not be defined, but must be declared. All the global/readonly segment variables that are referenced in kernels and indirect functions being finalized may or may not be defined, but must be declared.

Parameters

in	*program*	HSAIL program.
in	*isa*	Instruction set architecture to finalize for.
in	*call_convention*	A call convention used in a finalization. Must have a value between HSA_EXT_FINALIZER_CALL_CONVENTION_AUTO (inclusive) and the value of the attribute HSA_ISA_INFO_CALL_CONVENTION_COUNT in `isa` (not inclusive).
in	*control_directives*	Low-level control directives that influence the finalization process.
in	*options*	Vendor-specific options. May be NULL.
in	*code_object_type*	Type of code object to produce.
out	*code_object*	Code object generated by the Finalizer, which contains the machine code for the kernels and indirect functions in the HSAIL program. The code object is independent of the HSAIL module that was used to generate it.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	There is a failure to allocate resources required for the operation.
HSA_EXT_STATUS_ERROR_INVALID_PROGRAM	The HSAIL program is invalid.
HSA_STATUS_ERROR_INVALID_ISA	`isa` is invalid.
HSA_EXT_STATUS_ERROR_DIRECTIVE_MISMATCH	The directive in the control directive structure and in the HSAIL kernel mismatch, or if the same directive is used with a different value in one of the functions used by this kernel.
HSA_EXT_STATUS_ERROR_FINALIZATION_FAILED	The Finalizer encountered an error while compiling a kernel or an indirect function.

5.11.4.5 hsa_ext_program_get_info()

```
hsa_status_t HSA_API hsa_ext_program_get_info (
            hsa_ext_program_t program,
```

```
            hsa_ext_program_info_t attribute,
            void * value )
```

Get the current value of an attribute for a given HSAIL program.

Parameters

in	*program*	HSAIL program.
in	*attribute*	Attribute to query.
out	*value*	Pointer to an application-allocated buffer where to store the value of the attribute. If the buffer passed by the application is not large enough to hold the value of `attribute`, the behaviour is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_EXT_STATUS_ERROR_INVALID_PROGRAM	The HSAIL program is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`attribute` is an invalid HSAIL program attribute, or `value` is NULL.

5.11.4.6 hsa_ext_program_iterate_modules()

```
hsa_status_t HSA_API hsa_ext_program_iterate_modules (
            hsa_ext_program_t program,
            hsa_status_t(*)(hsa_ext_program_t program, hsa_ext_module_t module, void *data)
callback,
            void * data )
```

Iterate over the HSAIL modules in a program, and invoke an application-defined callback on every iteration.

Parameters

in	*program*	HSAIL program.
in	*callback*	Callback to be invoked once per HSAIL module in the program. The HSA runtime passes three arguments to the callback: the program, a HSAIL module, and the application data. If `callback` returns a status other than HSA_STATUS_SUCCESS for a particular iteration, the traversal stops and hsa_ext_program_iterate_modules returns that status value.
in	*data*	Application data that is passed to `callback` on every iteration. May be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_EXT_STATUS_ERROR_INVALID_PROGRAM	The program is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`callback` is NULL.

5.12 Images and Samplers

Classes

- struct hsa_ext_image_s

  Image handle, populated by hsa_ext_image_create or hsa_ext_image_create_with_layout. Image handles are only unique within an agent, not across agents.
- struct hsa_ext_image_format_s

  Image format.
- struct hsa_ext_image_descriptor_s

  Implementation independent image descriptor.
- struct hsa_ext_image_data_info_s

  Agent specific image size and alignment requirements, populated by hsa_ext_image_data_get_info and hsa_ext_image_data_get_info_with_layout.
- struct hsa_ext_image_region_s

  Image region.
- struct hsa_ext_sampler_s

  Sampler handle. Samplers are populated by hsa_ext_sampler_create. Sampler handles are only unique within an agent, not across agents.
- struct hsa_ext_sampler_descriptor_s

  Implementation independent sampler descriptor.
- struct hsa_ext_images_1_00_pfn_s

  The function pointer table for the images v1.00 extension. Can be returned by hsa_system_get_extension_table or hsa_system_get_major_extension_table.
- struct hsa_ext_images_1_pfn_s

  The function pointer table for the images v1 extension. Can be returned by hsa_system_get_extension_table or hsa_system_get_major_extension_table.

Typedefs

- typedef struct hsa_ext_image_s hsa_ext_image_t

  Image handle, populated by hsa_ext_image_create or hsa_ext_image_create_with_layout. Image handles are only unique within an agent, not across agents.
- typedef uint32_t hsa_ext_image_channel_type32_t

  A fixed-size type used to represent hsa_ext_image_channel_type_t constants.
- typedef uint32_t hsa_ext_image_channel_order32_t

  A fixed-size type used to represent hsa_ext_image_channel_order_t constants.
- typedef struct hsa_ext_image_format_s **hsa_ext_image_format_t**

  Image format.
- typedef struct hsa_ext_image_descriptor_s **hsa_ext_image_descriptor_t**

  Implementation independent image descriptor.
- typedef struct hsa_ext_image_data_info_s **hsa_ext_image_data_info_t**

  Agent specific image size and alignment requirements, populated by hsa_ext_image_data_get_info and hsa_ext_image_data_get_info_with_layout.
- typedef struct hsa_ext_image_region_s **hsa_ext_image_region_t**

  Image region.
- typedef struct hsa_ext_sampler_s **hsa_ext_sampler_t**

  Sampler handle. Samplers are populated by hsa_ext_sampler_create. Sampler handles are only unique within an agent, not across agents.
- typedef uint32_t hsa_ext_sampler_addressing_mode32_t

  A fixed-size type used to represent hsa_ext_sampler_addressing_mode_t constants.

- typedef uint32_t hsa_ext_sampler_coordinate_mode32_t

  A fixed-size type used to represent hsa_ext_sampler_coordinate_mode_t constants.
- typedef uint32_t hsa_ext_sampler_filter_mode32_t

  A fixed-size type used to represent hsa_ext_sampler_filter_mode_t constants.
- typedef struct hsa_ext_sampler_descriptor_s **hsa_ext_sampler_descriptor_t**

  Implementation independent sampler descriptor.
- typedef struct hsa_ext_images_1_00_pfn_s **hsa_ext_images_1_00_pfn_t**

  The function pointer table for the images v1.00 extension. Can be returned by hsa_system_get_extension_table or hsa_system_get_major_extension_table.
- typedef struct hsa_ext_images_1_pfn_s **hsa_ext_images_1_pfn_t**

  The function pointer table for the images v1 extension. Can be returned by hsa_system_get_extension_table or hsa_system_get_major_extension_table.

Enumerations

- enum { HSA_EXT_STATUS_ERROR_IMAGE_FORMAT_UNSUPPORTED = 0x3000 , HSA_EXT_STATUS_ERROR_IMAGE_S = 0x3001 , HSA_EXT_STATUS_ERROR_IMAGE_PITCH_UNSUPPORTED = 0x3002 , HSA_EXT_STATUS_ERROR_SAMPLE = 0x3003 }

  Enumeration constants added to hsa_status_t by this extension.
- enum {
  HSA_EXT_AGENT_INFO_IMAGE_1D_MAX_ELEMENTS = 0x3000 , HSA_EXT_AGENT_INFO_IMAGE_1DA_MAX_ELEMEN = 0x3001 , HSA_EXT_AGENT_INFO_IMAGE_1DB_MAX_ELEMENTS = 0x3002 , HSA_EXT_AGENT_INFO_IMAGE_2D_MAX = 0x3003 ,
  HSA_EXT_AGENT_INFO_IMAGE_2DA_MAX_ELEMENTS = 0x3004 , HSA_EXT_AGENT_INFO_IMAGE_2DDEPTH_MAX_E = 0x3005 , HSA_EXT_AGENT_INFO_IMAGE_2DADEPTH_MAX_ELEMENTS = 0x3006 , HSA_EXT_AGENT_INFO_IMAGE_3 = 0x3007 ,
  HSA_EXT_AGENT_INFO_IMAGE_ARRAY_MAX_LAYERS = 0x3008 , HSA_EXT_AGENT_INFO_MAX_IMAGE_RD_HANDLE = 0x3009 , HSA_EXT_AGENT_INFO_MAX_IMAGE_RORW_HANDLES = 0x300A , HSA_EXT_AGENT_INFO_MAX_SAMPLE = 0x300B ,
  HSA_EXT_AGENT_INFO_IMAGE_LINEAR_ROW_PITCH_ALIGNMENT = 0x300C }

  Enumeration constants added to hsa_agent_info_t by this extension.
- enum hsa_ext_image_geometry_t {
  HSA_EXT_IMAGE_GEOMETRY_1D = 0 , HSA_EXT_IMAGE_GEOMETRY_2D = 1 , HSA_EXT_IMAGE_GEOMETRY_3D = 2 , HSA_EXT_IMAGE_GEOMETRY_1DA = 3 ,
  HSA_EXT_IMAGE_GEOMETRY_2DA = 4 , HSA_EXT_IMAGE_GEOMETRY_1DB = 5 , HSA_EXT_IMAGE_GEOMETRY_2DD = 6 , HSA_EXT_IMAGE_GEOMETRY_2DADEPTH = 7 }

  Geometry associated with the image. This specifies the number of image dimensions and whether the image is an image array. See the Image Geometry section in the HSA Programming Reference Manual for definitions on each geometry. The enumeration values match the BRIG type `hsa_ext_brig_image_geometry_t`.
- enum hsa_ext_image_channel_type_t {
  HSA_EXT_IMAGE_CHANNEL_TYPE_SNORM_INT8 = 0 , **HSA_EXT_IMAGE_CHANNEL_TYPE_SNORM_INT16** = 1 , **HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_INT8** = 2 , **HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_INT16** = 3 ,
  HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_INT24 = 4 , **HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_SHORT_555** = 5 , **HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_SHORT_565** = 6 , **HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_SHORT_101010** = 7 ,
  HSA_EXT_IMAGE_CHANNEL_TYPE_SIGNED_INT8 = 8 , **HSA_EXT_IMAGE_CHANNEL_TYPE_SIGNED_INT16** = 9 , **HSA_EXT_IMAGE_CHANNEL_TYPE_SIGNED_INT32** = 10 , **HSA_EXT_IMAGE_CHANNEL_TYPE_UNSIGNED_INT8** = 11 ,
  HSA_EXT_IMAGE_CHANNEL_TYPE_UNSIGNED_INT16 = 12 , **HSA_EXT_IMAGE_CHANNEL_TYPE_UNSIGNED_INT32** = 13 , **HSA_EXT_IMAGE_CHANNEL_TYPE_HALF_FLOAT** = 14 , **HSA_EXT_IMAGE_CHANNEL_TYPE_FLOAT** = 15 }

  Channel type associated with the elements of an image. See the Channel Type section in the HSA Programming Reference Manual for definitions on each channel type. The enumeration values and definition match the BRIG type `hsa_ext_brig_image_channel_type_t`.

- enum hsa_ext_image_channel_order_t {
  HSA_EXT_IMAGE_CHANNEL_ORDER_A = 0 , **HSA_EXT_IMAGE_CHANNEL_ORDER_R** = 1 , **HSA_EXT_IMAGE_CHANNEL_ORDER_RX** = 2 , **HSA_EXT_IMAGE_CHANNEL_ORDER_RG** = 3 , **HSA_EXT_IMAGE_CHANNEL_ORDER_RGX** = 4 , **HSA_EXT_IMAGE_CHANNEL_ORDER_RA** = 5 , **HSA_EXT_IMAGE_CHANNEL_ORDER_RGB** = 6 , **HSA_EXT_IMAGE_CHANNEL_ORDER_RGBX** = 7 , **HSA_EXT_IMAGE_CHANNEL_ORDER_RGBA** = 8 , **HSA_EXT_IMAGE_CHANNEL_ORDER_BGRA** = 9 , **HSA_EXT_IMAGE_CHANNEL_ORDER_ARGB** = 10 , **HSA_EXT_IMAGE_CHANNEL_ORDER_ABGR** = 11 , **HSA_EXT_IMAGE_CHANNEL_ORDER_SRGB** = 12 , **HSA_EXT_IMAGE_CHANNEL_ORDER_SRGBX** = 13 , **HSA_EXT_IMAGE_CHANNEL_ORDER_SRGBA** = 14 , **HSA_EXT_IMAGE_CHANNEL_ORDER_SBGRA** = 15 , **HSA_EXT_IMAGE_CHANNEL_ORDER_INTENSITY** = 16 , **HSA_EXT_IMAGE_CHANNEL_ORDER_LUMINANCE** = 17 , **HSA_EXT_IMAGE_CHANNEL_ORDER_DEPTH** = 18 , **HSA_EXT_IMAGE_CHANNEL_ORDER_DEPTH_STENCIL** = 19 }

  Channel order associated with the elements of an image. See the Channel Order section in the HSA Programming Reference Manual for definitions on each channel order. The enumeration values match the BRIG type `hsa_ext_brig_image_channel_order_t`.
- enum hsa_ext_image_capability_t {
  HSA_EXT_IMAGE_CAPABILITY_NOT_SUPPORTED = 0x0 , HSA_EXT_IMAGE_CAPABILITY_READ_ONLY = 0x1 , HSA_EXT_IMAGE_CAPABILITY_WRITE_ONLY = 0x2 , HSA_EXT_IMAGE_CAPABILITY_READ_WRITE = 0x4 , HSA_EXT_IMAGE_CAPABILITY_READ_MODIFY_WRITE = 0x8 , HSA_EXT_IMAGE_CAPABILITY_ACCESS_INVARIANT_D = 0x10 }

  Image capability.
- enum hsa_ext_image_data_layout_t { HSA_EXT_IMAGE_DATA_LAYOUT_OPAQUE = 0x0 , HSA_EXT_IMAGE_DATA_LAYOUT = 0x1 }

  Image data layout.
- enum hsa_ext_sampler_addressing_mode_t {
  HSA_EXT_SAMPLER_ADDRESSING_MODE_UNDEFINED = 0 , HSA_EXT_SAMPLER_ADDRESSING_MODE_CLAMP_TO = 1 , HSA_EXT_SAMPLER_ADDRESSING_MODE_CLAMP_TO_BORDER = 2 , HSA_EXT_SAMPLER_ADDRESSING_MOD = 3 , HSA_EXT_SAMPLER_ADDRESSING_MODE_MIRRORED_REPEAT = 4 }

  Sampler address modes. The sampler address mode describes the processing of out-of-range image coordinates. See the Addressing Mode section in the HSA Programming Reference Manual for definitions on each address mode. The values match the BRIG type `hsa_ext_brig_sampler_addressing_t`.
- enum hsa_ext_sampler_coordinate_mode_t { HSA_EXT_SAMPLER_COORDINATE_MODE_UNNORMALIZED = 0 , HSA_EXT_SAMPLER_COORDINATE_MODE_NORMALIZED = 1 }

  Sampler coordinate normalization modes. See the Coordinate Normalization Mode section in the HSA Programming Reference Manual for definitions on each coordinate normalization mode. The values match the BRIG type `hsa_ext_brig_sampler_coord_normalization_t`.
- enum hsa_ext_sampler_filter_mode_t { HSA_EXT_SAMPLER_FILTER_MODE_NEAREST = 0 , HSA_EXT_SAMPLER_FILTER = 1 }

  Sampler filter modes. See the Filter Mode section in the HSA Programming Reference Manual for definitions on each address mode. The enumeration values match the BRIG type `hsa_ext_brig_sampler_filter_t`.

Functions

- hsa_status_t HSA_API hsa_ext_image_get_capability (hsa_agent_t agent, hsa_ext_image_geometry_t geometry, const hsa_ext_image_format_t *image_format, uint32_t *capability_mask)

  Retrieve the supported image capabilities for a given combination of agent, geometry, and image format for an image created with an opaque image data layout.
- hsa_status_t HSA_API hsa_ext_image_get_capability_with_layout (hsa_agent_t agent, hsa_ext_image_geometry_t geometry, const hsa_ext_image_format_t *image_format, hsa_ext_image_data_layout_t image_data_layout, uint32_t *capability_mask)

  Retrieve the supported image capabilities for a given combination of agent, geometry, image format, and image layout for an image created with an explicit image data layout.

- hsa_status_t HSA_API hsa_ext_image_data_get_info (hsa_agent_t agent, const hsa_ext_image_descriptor_t *image_descriptor, hsa_access_permission_t access_permission, hsa_ext_image_data_info_t *image_data_info)

  Retrieve the image data requirements for a given combination of agent, image descriptor, and access permission for an image created with an opaque image data layout.
- hsa_status_t HSA_API hsa_ext_image_data_get_info_with_layout (hsa_agent_t agent, const hsa_ext_image_descriptor_t *image_descriptor, hsa_access_permission_t access_permission, hsa_ext_image_data_layout_t image_data_layout, size_t image_data_row_pitch, size_t image_data_slice_pitch, hsa_ext_image_data_info_t *image_data_info)

  Retrieve the image data requirements for a given combination of image descriptor, access permission, image data layout, image data row pitch, and image data slice pitch for an image created with an explicit image data layout.
- hsa_status_t HSA_API hsa_ext_image_create (hsa_agent_t agent, const hsa_ext_image_descriptor_t *image_descriptor, const void *image_data, hsa_access_permission_t access_permission, hsa_ext_image_t *image)

  Creates an agent specific image handle to an image with an opaque image data layout.
- hsa_status_t HSA_API hsa_ext_image_create_with_layout (hsa_agent_t agent, const hsa_ext_image_descriptor_t *image_descriptor, const void *image_data, hsa_access_permission_t access_permission, hsa_ext_image_data_layout_t image_data_layout, size_t image_data_row_pitch, size_t image_data_slice_pitch, hsa_ext_image_t *image)

  Creates an agent specific image handle to an image with an explicit image data layout.
- hsa_status_t HSA_API hsa_ext_image_destroy (hsa_agent_t agent, hsa_ext_image_t image)

  Destroy an image handle previously created using hsa_ext_image_create or hsa_ext_image_create_with_layout.
- hsa_status_t HSA_API hsa_ext_image_copy (hsa_agent_t agent, hsa_ext_image_t src_image, const hsa_dim3_t *src_offset, hsa_ext_image_t dst_image, const hsa_dim3_t *dst_offset, const hsa_dim3_t *range)

  Copies a portion of one image (the source) to another image (the destination).
- hsa_status_t HSA_API hsa_ext_image_import (hsa_agent_t agent, const void *src_memory, size_t src_row_pitch, size_t src_slice_pitch, hsa_ext_image_t dst_image, const hsa_ext_image_region_t *image_region)

  Import a linearly organized image data from memory directly to an image handle.
- hsa_status_t HSA_API hsa_ext_image_export (hsa_agent_t agent, hsa_ext_image_t src_image, void *dst_memory, size_t dst_row_pitch, size_t dst_slice_pitch, const hsa_ext_image_region_t *image_region)

  Export the image data to linearly organized memory.
- hsa_status_t HSA_API hsa_ext_image_clear (hsa_agent_t agent, hsa_ext_image_t image, const void *data, const hsa_ext_image_region_t *image_region)

  Clear a region of an image so that every image element has the specified value.
- hsa_status_t HSA_API hsa_ext_sampler_create (hsa_agent_t agent, const hsa_ext_sampler_descriptor_t *sampler_descriptor, hsa_ext_sampler_t *sampler)

  Create an agent specific sampler handle for a given agent independent sampler descriptor and agent.
- hsa_status_t HSA_API hsa_ext_sampler_destroy (hsa_agent_t agent, hsa_ext_sampler_t sampler)

  Destroy a sampler handle previously created using hsa_ext_sampler_create.

5.12.1 Detailed Description

5.12.2 Macro Definition Documentation

5.12.2.1 hsa_ext_images_1

```
#define hsa_ext_images_1
```

Definition at line 1352 of file hsa_ext_image.h.

5.12.2.2 hsa_ext_images_1_00

```
#define hsa_ext_images_1_00
```

Definition at line 1281 of file hsa_ext_image.h.

5.12.3 Typedef Documentation

5.12.3.1 hsa_ext_image_channel_order32_t

```
typedef uint32_t hsa_ext_image_channel_order32_t
```

A fixed-size type used to represent hsa_ext_image_channel_order_t constants.

Definition at line 305 of file hsa_ext_image.h.

5.12.3.2 hsa_ext_image_channel_type32_t

```
typedef uint32_t hsa_ext_image_channel_type32_t
```

A fixed-size type used to represent hsa_ext_image_channel_type_t constants.

Definition at line 269 of file hsa_ext_image.h.

5.12.3.3 hsa_ext_image_t

```
typedef struct hsa_ext_image_s hsa_ext_image_t
```

Image handle, populated by hsa_ext_image_create or hsa_ext_image_create_with_layout. Image handles are only unique within an agent, not across agents.

5.12.3.4 hsa_ext_sampler_addressing_mode32_t

```
typedef uint32_t hsa_ext_sampler_addressing_mode32_t
```

A fixed-size type used to represent hsa_ext_sampler_addressing_mode_t constants.

Definition at line 1145 of file hsa_ext_image.h.

5.12.3.5 hsa_ext_sampler_coordinate_mode32_t

```
typedef uint32_t hsa_ext_sampler_coordinate_mode32_t
```

A fixed-size type used to represent hsa_ext_sampler_coordinate_mode_t constants.

Definition at line 1172 of file hsa_ext_image.h.

5.12.3.6 hsa_ext_sampler_filter_mode32_t

```
typedef uint32_t hsa_ext_sampler_filter_mode32_t
```

A fixed-size type used to represent hsa_ext_sampler_filter_mode_t constants.

Definition at line 1200 of file hsa_ext_image.h.

5.12.4 Enumeration Type Documentation

5.12.4.1 anonymous enum

```
anonymous enum
```

Enumeration constants added to hsa_status_t by this extension.

Remarks

Additions to hsa_status_t

Enumerator

HSA_EXT_STATUS_ERROR_IMAGE_FORMAT_↩ UNSUPPORTED	Image format is not supported.
HSA_EXT_STATUS_ERROR_IMAGE_SIZE_↩ UNSUPPORTED	Image size is not supported.
HSA_EXT_STATUS_ERROR_IMAGE_PITCH_↩ UNSUPPORTED	Image pitch is not supported or invalid.
HSA_EXT_STATUS_ERROR_SAMPLER_DESCRIPTOR_↩ UNSUPPORTED	Sampler descriptor is not supported or invalid.

Definition at line 68 of file hsa_ext_image.h.

5.12.4.2 anonymous enum

```
anonymous enum
```

Enumeration constants added to hsa_agent_info_t by this extension.

Remarks

Additions to hsa_agent_info_t

Enumerator

HSA_EXT_AGENT_INFO_IMAGE_1D_MAX_↩ ELEMENTS	Maximum number of elements in 1D images. Must be at least 16384. The type of this attribute is size_t.
HSA_EXT_AGENT_INFO_IMAGE_1DA_MAX_↩ ELEMENTS	Maximum number of elements in 1DA images. Must be at least 16384. The type of this attribute is size_t.
HSA_EXT_AGENT_INFO_IMAGE_1DB_MAX_↩ ELEMENTS	Maximum number of elements in 1DB images. Must be at least 65536. The type of this attribute is size_t.
HSA_EXT_AGENT_INFO_IMAGE_2D_MAX_↩ ELEMENTS	Maximum dimensions (width, height) of 2D images, in image elements. The X and Y maximums must be at least 16384. The type of this attribute is size_t[2].
HSA_EXT_AGENT_INFO_IMAGE_2DA_MAX_↩ ELEMENTS	Maximum dimensions (width, height) of 2DA images, in image elements. The X and Y maximums must be at least 16384. The type of this attribute is size_t[2].
HSA_EXT_AGENT_INFO_IMAGE_2DDEPTH_↩ MAX_ELEMENTS	Maximum dimensions (width, height) of 2DDEPTH images, in image elements. The X and Y maximums must be at least 16384. The type of this attribute is size_t[2].
HSA_EXT_AGENT_INFO_IMAGE_2DADEPTH_↩ MAX_ELEMENTS	Maximum dimensions (width, height) of 2DADEPTH images, in image elements. The X and Y maximums must be at least 16384. The type of this attribute is size_t[2].
HSA_EXT_AGENT_INFO_IMAGE_3D_MAX_↩ ELEMENTS	Maximum dimensions (width, height, depth) of 3D images, in image elements. The maximum along any dimension must be at least 2048. The type of this attribute is size_t[3].
HSA_EXT_AGENT_INFO_IMAGE_ARRAY_MAX_↩ LAYERS	Maximum number of image layers in a image array. Must be at least 2048. The type of this attribute is size_t.
HSA_EXT_AGENT_INFO_MAX_IMAGE_RD_↩ HANDLES	Maximum number of read-only image handles that can be created for an agent at any one time. Must be at least 128. The type of this attribute is size_t.
HSA_EXT_AGENT_INFO_MAX_IMAGE_RORW_↩ HANDLES	Maximum number of write-only and read-write image handles (combined) that can be created for an agent at any one time. Must be at least 64. The type of this attribute is size_t.
HSA_EXT_AGENT_INFO_MAX_SAMPLER_↩ HANDLERS	Maximum number of sampler handlers that can be created for an agent at any one time. Must be at least 16. The type of this attribute is size_t.
HSA_EXT_AGENT_INFO_IMAGE_LINEAR_ROW↩ _PITCH_ALIGNMENT	Image pitch alignment. The agent only supports linear image data layouts with a row pitch that is a multiple of this value. Must be a power of 2. The type of this attribute is size_t.

Definition at line 93 of file hsa_ext_image.h.

5.12.4.3 hsa_ext_image_capability_t

`enum hsa_ext_image_capability_t`

Image capability.

Enumerator

HSA_EXT_IMAGE_CAPABILITY_NOT_SUPPORTED	Images of this geometry, format, and layout are not supported by the agent.
HSA_EXT_IMAGE_CAPABILITY_READ_ONLY	Read-only images of this geometry, format, and layout are supported by the agent.
HSA_EXT_IMAGE_CAPABILITY_WRITE_ONLY	Write-only images of this geometry, format, and layout are supported by the agent.
HSA_EXT_IMAGE_CAPABILITY_READ_WRITE	Read-write images of this geometry, format, and layout are supported by the agent.
HSA_EXT_IMAGE_CAPABILITY_READ_MODIFY_WRITE	**Deprecated** Images of this geometry, format, and layout can be accessed from read-modify-write atomic operations in the agent.
HSA_EXT_IMAGE_CAPABILITY_ACCESS_INVARIANT_DATA_LAYOUT	Images of this geometry, format, and layout are guaranteed to have a consistent data layout regardless of how they are accessed by the associated agent.

Definition at line 362 of file hsa_ext_image.h.

5.12.4.4 hsa_ext_image_channel_order_t

`enum hsa_ext_image_channel_order_t`

Channel order associated with the elements of an image. See the *Channel Order* section in the *HSA Programming Reference Manual* for definitions on each channel order. The enumeration values match the BRIG type `hsa_ext_brig_image_channel_order_t`.

Definition at line 279 of file hsa_ext_image.h.

5.12.4.5 hsa_ext_image_channel_type_t

`enum hsa_ext_image_channel_type_t`

Channel type associated with the elements of an image. See the *Channel Type* section in the *HSA Programming Reference Manual* for definitions on each channel type. The enumeration values and definition match the BRIG type `hsa_ext_brig_image_channel_type_t`.

Definition at line 247 of file hsa_ext_image.h.

5.12.4.6 hsa_ext_image_data_layout_t

enum hsa_ext_image_data_layout_t

Image data layout.

An image data layout denotes such aspects of image data layout as tiling and organization of channels in memory. Some image data layouts may only apply to specific image geometries, formats, and access permissions. Different agents may support different image layout identifiers, including vendor specific layouts. Note that an agent may not support the same image data layout for different access permissions to images with the same image geometry, size, and format. If multiple agents support the same image data layout then it is possible to use separate image handles for each agent that references the same image data.

Enumerator

HSA_EXT_IMAGE_DATA_LAYOUT_OPAQUE	An implementation specific opaque image data layout which can vary depending on the agent, geometry, image format, image size, and access permissions.
HSA_EXT_IMAGE_DATA_LAYOUT_LINEAR	The image data layout is specified by the following rules in ascending byte address order. For a 3D image, 2DA image array, or 1DA image array, the image data is stored as a linear sequence of adjacent 2D image slices, 2D images, or 1D images respectively, spaced according to the slice pitch. Each 2D image is stored as a linear sequence of adjacent image rows, spaced according to the row pitch. Each 1D or 1DB image is stored as a single image row. Each image row is stored as a linear sequence of image elements. Each image element is stored as a linear sequence of image components specified by the left to right channel order definition. Each image component is stored using the memory type specified by the channel type. The 1DB image geometry always uses the linear image data layout.

Definition at line 411 of file hsa_ext_image.h.

5.12.4.7 hsa_ext_image_geometry_t

enum hsa_ext_image_geometry_t

Geometry associated with the image. This specifies the number of image dimensions and whether the image is an image array. See the *Image Geometry* section in the *HSA Programming Reference Manual* for definitions on each geometry. The enumeration values match the BRIG type `hsa_ext_brig_image_geometry_t`.

Enumerator

HSA_EXT_IMAGE_GEOMETRY_1D	One-dimensional image addressed by width coordinate.
HSA_EXT_IMAGE_GEOMETRY_2D	Two-dimensional image addressed by width and height coordinates.
HSA_EXT_IMAGE_GEOMETRY_3D	Three-dimensional image addressed by width, height, and depth coordinates.

Enumerator

HSA_EXT_IMAGE_GEOMETRY_1DA	Array of one-dimensional images with the same size and format. 1D arrays are addressed by width and index coordinate.
HSA_EXT_IMAGE_GEOMETRY_2DA	Array of two-dimensional images with the same size and format. 2D arrays are addressed by width, height, and index coordinates.
HSA_EXT_IMAGE_GEOMETRY_1DB	One-dimensional image addressed by width coordinate. It has specific restrictions compared to HSA_EXT_IMAGE_GEOMETRY_1D. An image with an opaque image data layout will always use a linear image data layout, and one with an explicit image data layout must specify HSA_EXT_IMAGE_DATA_LAYOUT_LINEAR.
HSA_EXT_IMAGE_GEOMETRY_2DDEPTH	Two-dimensional depth image addressed by width and height coordinates.
HSA_EXT_IMAGE_GEOMETRY_2DADEPTH	Array of two-dimensional depth images with the same size and format. 2D arrays are addressed by width, height, and index coordinates.

Definition at line 191 of file hsa_ext_image.h.

5.12.4.8 hsa_ext_sampler_addressing_mode_t

`enum hsa_ext_sampler_addressing_mode_t`

Sampler address modes. The sampler address mode describes the processing of out-of-range image coordinates. See the *Addressing Mode* section in the *HSA Programming Reference Manual* for definitions on each address mode. The values match the BRIG type `hsa_ext_brig_sampler_addressing_t`.

Enumerator

HSA_EXT_SAMPLER_ADDRESSING_MODE_↩ UNDEFINED	Out-of-range coordinates are not handled.
HSA_EXT_SAMPLER_ADDRESSING_MODE_↩ CLAMP_TO_EDGE	Clamp out-of-range coordinates to the image edge.
HSA_EXT_SAMPLER_ADDRESSING_MODE_↩ CLAMP_TO_BORDER	Clamp out-of-range coordinates to the image border color.
HSA_EXT_SAMPLER_ADDRESSING_MODE_↩ REPEAT	Wrap out-of-range coordinates back into the valid coordinate range so the image appears as repeated tiles.
HSA_EXT_SAMPLER_ADDRESSING_MODE_↩ MIRRORED_REPEAT	Mirror out-of-range coordinates back into the valid coordinate range so the image appears as repeated tiles with every other tile a reflection.

Definition at line 1111 of file hsa_ext_image.h.

5.12.4.9 hsa_ext_sampler_coordinate_mode_t

`enum hsa_ext_sampler_coordinate_mode_t`

Sampler coordinate normalization modes. See the *Coordinate Normalization Mode* section in the *HSA Programming Reference Manual* for definitions on each coordinate normalization mode. The values match the BRIG type `hsa_ext_brig_sampler_coord_normalization_t`.

Enumerator

HSA_EXT_SAMPLER_COORDINATE_MODE_UNNORMALIZED	Coordinates are used to directly address an image element.
HSA_EXT_SAMPLER_COORDINATE_MODE_NORMALIZED	Coordinates are scaled by the image dimension size before being used to address an image element.

Definition at line 1154 of file hsa_ext_image.h.

5.12.4.10 hsa_ext_sampler_filter_mode_t

```
enum hsa_ext_sampler_filter_mode_t
```

Sampler filter modes. See the *Filter Mode* section in the *HSA Programming Reference Manual* for definitions on each address mode. The enumeration values match the BRIG type `hsa_ext_brig_sampler_filter_t`.

Enumerator

HSA_EXT_SAMPLER_FILTER_MODE_NEAREST	Filter to the image element nearest (in Manhattan distance) to the specified coordinate.
HSA_EXT_SAMPLER_FILTER_MODE_LINEAR	Filter to the image element calculated by combining the elements in a 2x2 square block or 2x2x2 cube block around the specified coordinate. The elements are combined using linear interpolation.

Definition at line 1181 of file hsa_ext_image.h.

5.12.5 Function Documentation

5.12.5.1 hsa_ext_image_clear()

```
hsa_status_t HSA_API hsa_ext_image_clear (
            hsa_agent_t agent,
            hsa_ext_image_t image,
            const void * data,
            const hsa_ext_image_region_t * image_region )
```

Clear a region of an image so that every image element has the specified value.

Parameters

in	*agent*	Agent associated with the image handle.

Parameters

in	*image*	Image handle for image to be cleared.
in	*data*	The value to which to set each image element being cleared. It is specified as an array of image component values. The number of array elements must match the number of access components for the image channel order. The type of each array element must match the image access type of the image channel type. When the value is used to set the value of an image element, the conversion method corresponding to the image channel type is used. See the *Channel Order* section and *Channel Type* section in the *HSA Programming Reference Manual* for more information. Must not be NULL.
in	*image_region*	Pointer to the image region to clear. Must not be NULL. If the region references an out-out-bounds element, the behavior is undefined.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`data` is NULL, or `image_region` is NULL.

5.12.5.2 hsa_ext_image_copy()

```
hsa_status_t HSA_API hsa_ext_image_copy (
             hsa_agent_t agent,
             hsa_ext_image_t src_image,
             const hsa_dim3_t * src_offset,
             hsa_ext_image_t dst_image,
             const hsa_dim3_t * dst_offset,
             const hsa_dim3_t * range )
```

Copies a portion of one image (the source) to another image (the destination).

The source and destination image formats should be the same, with the exception that s-form channel orders match the corresponding non-s-form channel order and vice versa. For example, it is allowed to copy a source image with a channel order of HSA_EXT_IMAGE_CHANNEL_ORDER_SRGB to a destination image with a channel order of HSA_EXT_IMAGE_CHANNEL_ORDER_RGB.

The source and destination images do not have to be of the same geometry and appropriate scaling is performed by the HSA runtime. It is possible to copy subregions between any combinations of source and destination geometries, provided that the dimensions of the subregions are the same. For example, it is allowed to copy a rectangular region from a 2D image to a slice of a 3D image.

If the source and destination image data overlap, or the combination of offset and range references an out-out-bounds element in any of the images, the behavior is undefined.

Parameters

in	*agent*	Agent associated with both the source and destination image handles.
in	*src_image*	Image handle of source image. The agent associated with the source image handle must be identical to that of the destination image.

Parameters

in	*src_offset*	Pointer to the offset within the source image where to copy the data from. Must not be NULL.
in	*dst_image*	Image handle of destination image.
in	*dst_offset*	Pointer to the offset within the destination image where to copy the data. Must not be NULL.
in	*range*	Dimensions of the image portion to be copied. The HSA runtime computes the size of the image data to be copied using this argument. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`src_offset` is NULL, `dst_offset` is NULL, or `range` is NULL.

5.12.5.3 hsa_ext_image_create()

```
hsa_status_t HSA_API hsa_ext_image_create (
             hsa_agent_t agent,
             const hsa_ext_image_descriptor_t * image_descriptor,
             const void * image_data,
             hsa_access_permission_t access_permission,
             hsa_ext_image_t * image )
```

Creates an agent specific image handle to an image with an opaque image data layout.

Images with an opaque image data layout created with different access permissions but matching image descriptors and same agent can share the same image data if HSA_EXT_IMAGE_CAPABILITY_ACCESS_INVARIANT_DATA_LAYOUT is reported by hsa_ext_image_get_capability for the image format specified in the image descriptor. Image descriptors match if they have the same values, with the exception that s-form channel orders match the corresponding non-s-form channel order and vice versa.

If necessary, an application can use image operations (import, export, copy, clear) to prepare the image for the intended use regardless of the access permissions.

Parameters

in	*agent*	agent to be associated with the image handle created.
in	*image_descriptor*	Pointer to an image descriptor. Must not be NULL.
in	*image_data*	Image data buffer that must have been allocated according to the size and alignment requirements dictated by hsa_ext_image_data_get_info. Must not be NULL.

Any previous memory contents are preserved upon creation. The application is responsible for ensuring that the lifetime of the image data exceeds that of all the associated images.

Parameters

in	*access_permission*	Access permission of the image when accessed by agent. The access permission defines how the agent is allowed to access the image using the image handle created and must match the corresponding HSAIL image handle type. The agent must support the image format specified in `image_descriptor` for the given `access_permission`.
out	*image*	Pointer to a memory location where the HSA runtime stores the newly created image handle. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_EXT_STATUS_ERROR_IMAGE_FORMAT_UNSUPPORTED	The agent does not have the capability to support the image format contained in `image_descriptor` using the specified `access_permission`.
HSA_EXT_STATUS_ERROR_IMAGE_SIZE_UNSUPPORTED	The agent does not support the image dimensions specified by `image_descriptor` using the specified `access_permission`.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.

support the creation of more image handles with the given `access_permission`).

Return values

HSA_STATUS_ERROR_INVALID_ARGUMENT	`image_descriptor` is NULL, `image_data` is NULL, `image_data` does not have a valid alignment, `access_permission` is not a valid access permission value, or `image` is NULL.

5.12.5.4 hsa_ext_image_create_with_layout()

```
hsa_status_t HSA_API hsa_ext_image_create_with_layout (
             hsa_agent_t agent,
             const hsa_ext_image_descriptor_t * image_descriptor,
             const void * image_data,
             hsa_access_permission_t access_permission,
             hsa_ext_image_data_layout_t image_data_layout,
             size_t image_data_row_pitch,
             size_t image_data_slice_pitch,
             hsa_ext_image_t * image )
```

Creates an agent specific image handle to an image with an explicit image data layout.

Images with an explicit image data layout created with different access permissions but matching image descriptors and matching image layout can share the same image data if HSA_EXT_IMAGE_CAPABILITY_ACCESS_INVARIANT_DATA_LAYOUT is reported by hsa_ext_image_get_capability_with_layout for the image format specified in the image descriptor

and specified image data layout. Image descriptors match if they have the same values, with the exception that s-form channel orders match the corresponding non-s-form channel order and vice versa. Image layouts match if they are the same image data layout and use the same image row and slice values.

If necessary, an application can use image operations (import, export, copy, clear) to prepare the image for the intended use regardless of the access permissions.

Parameters

in	*agent*	agent to be associated with the image handle created.
in	*image_descriptor*	Pointer to an image descriptor. Must not be NULL.
in	*image_data*	Image data buffer that must have been allocated according to the size and alignment requirements dictated by hsa_ext_image_data_get_info_with_layout. Must not be NULL.

Any previous memory contents are preserved upon creation. The application is responsible for ensuring that the lifetime of the image data exceeds that of all the associated images.

Parameters

in	*access_permission*	Access permission of the image when accessed by the agent. The access permission defines how the agent is allowed to access the image and must match the corresponding HSAIL image handle type. The agent must support the image format specified in `image_descriptor` for the given `access_permission` and `image_data_layout`.
in	*image_data_layout*	The image data layout to use for the `image_data`. It is invalid to use HSA_EXT_IMAGE_DATA_LAYOUT_OPAQUE; use hsa_ext_image_create instead.
in	*image_data_row_pitch*	The size in bytes for a single row of the image in the image data. If 0 is specified then the default row pitch value is used: image width $*$ image element byte size. The value used must be greater than or equal to the default row pitch, and be a multiple of the image element byte size. For the linear image layout it must also be a multiple of the image linear row pitch alignment for the agents that will access the image data using image instructions.
in	*image_data_slice_pitch*	The size in bytes of a single slice of a 3D image, or the size in bytes of each image layer in an image array in the image data. If 0 is specified then the default slice pitch value is used: row pitch $*$ height if geometry is HSA_EXT_IMAGE_GEOMETRY_3D, HSA_EXT_IMAGE_GEOMETRY_2DA, or HSA_EXT_IMAGE_GEOMETRY_2DADEPTH; row pitch if geometry is HSA_EXT_IMAGE_GEOMETRY_1DA; and 0 otherwise. The value used must be 0 if the default slice pitch is 0, be greater than or equal to the default slice pitch, and be a multiple of the row pitch.
out	*image*	Pointer to a memory location where the HSA runtime stores the newly created image handle. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.

Return values

HSA_EXT_STATUS_ERROR_IMAGE_FORMAT_UNSUPPORTED	The agent does not have the capability to support the image format contained in the image descriptor using the specified `access_permission` and `image_data_layout`.
HSA_EXT_STATUS_ERROR_IMAGE_SIZE_UNSUPPORTED	The agent does not support the image dimensions specified by `image_descriptor` using the specified `access_permission` and `image_data_layout`.
HSA_EXT_STATUS_ERROR_IMAGE_PITCH_UNSUPPORTED	The agent does not support the row and slice pitch specified by `image_data_row_pitch` and `image_data_slice_pitch`, or the values are invalid.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.

support the creation of more image handles with the given `access_permission`).

Return values

HSA_STATUS_ERROR_INVALID_ARGUMENT	`image_descriptor` is NULL, `image_data` is NULL, `image_data` does not have a valid alignment, `image_data_layout` is HSA_EXT_IMAGE_DATA_LAYOUT_OPAQUE, or `image` is NULL.

5.12.5.5 hsa_ext_image_data_get_info()

```
hsa_status_t HSA_API hsa_ext_image_data_get_info (
             hsa_agent_t agent,
             const hsa_ext_image_descriptor_t * image_descriptor,
             hsa_access_permission_t access_permission,
             hsa_ext_image_data_info_t * image_data_info )
```

Retrieve the image data requirements for a given combination of agent, image descriptor, and access permission for an image created with an opaque image data layout.

The optimal image data size and alignment requirements may vary depending on the image attributes specified in `image_descriptor`, the `access_permission`, and the `agent`. Also, different implementations of the HSA runtime may return different requirements for the same input values.

The implementation must return the same image data requirements for different access permissions with matching image descriptors as long as hsa_ext_image_get_capability reports HSA_EXT_IMAGE_CAPABILITY_ACCESS_INVARIANT_DATA_L
Image descriptors match if they have the same values, with the exception that s-form channel orders match the corresponding non-s-form channel order and vice versa.

Parameters

`in`	*agent*	Agent to be associated with the image handle.
`in`	*image_descriptor*	Pointer to an image descriptor. Must not be NULL.

Parameters

in	*access_permission*	Access permission of the image when accessed by `agent`. The access permission defines how the agent is allowed to access the image and must match the corresponding HSAIL image handle type. The `agent` must support the image format specified in `image_descriptor` for the given `access_permission`.
out	*image_data_info*	Memory location where the runtime stores the size and alignment requirements. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_EXT_STATUS_ERROR_IMAGE_FORMAT_UNSUPPORTED	`agent` does not support the image format specified by `image_descriptor` with the specified `access_permission`.
HSA_EXT_STATUS_ERROR_IMAGE_SIZE_UNSUPPORTED	The agent does not support the image dimensions specified by `image_descriptor` with the specified `access_permission`.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`image_descriptor` is NULL, `access_permission` is not a valid access permission value, or `image_data_info` is NULL.

5.12.5.6 hsa_ext_image_data_get_info_with_layout()

```
hsa_status_t HSA_API hsa_ext_image_data_get_info_with_layout (
             hsa_agent_t agent,
             const hsa_ext_image_descriptor_t * image_descriptor,
             hsa_access_permission_t access_permission,
             hsa_ext_image_data_layout_t image_data_layout,
             size_t image_data_row_pitch,
             size_t image_data_slice_pitch,
             hsa_ext_image_data_info_t * image_data_info )
```

Retrieve the image data requirements for a given combination of image descriptor, access permission, image data layout, image data row pitch, and image data slice pitch for an image created with an explicit image data layout.

The image data size and alignment requirements may vary depending on the image attributes specified in `image_descriptor`, the `access_permission`, and the image layout. However, different implementations of the HSA runtime will return the same requirements for the same input values.

The implementation must return the same image data requirements for different access permissions with matching image descriptors and matching image layouts as long as hsa_ext_image_get_capability reports HSA_EXT_IMAGE_CAPABILITY_ACCESS_INVARIANT_DATA_LAYOUT. Image descriptors match if they have the same values, with the exception that s-form channel orders match the corresponding non-s-form channel order and vice versa. Image layouts match if they are the same image data layout and use the same image row and slice pitch values.

Parameters

in	*image_descriptor*	Pointer to an image descriptor. Must not be NULL.
in	*access_permission*	Access permission of the image when accessed by an agent. The access permission defines how the agent is allowed to access the image and must match the corresponding HSAIL image handle type.
in	*image_data_layout*	The image data layout to use. It is invalid to use HSA_EXT_IMAGE_DATA_LAYOUT_OPAQUE; use hsa_ext_image_data_get_info instead.
in	*image_data_row_pitch*	The size in bytes for a single row of the image in the image data. If 0 is specified then the default row pitch value is used: image width * image element byte size. The value used must be greater than or equal to the default row pitch, and be a multiple of the image element byte size. For the linear image layout it must also be a multiple of the image linear row pitch alignment for the agents that will access the image data using image instructions.
in	*image_data_slice_pitch*	The size in bytes of a single slice of a 3D image, or the size in bytes of each image layer in an image array in the image data. If 0 is specified then the default slice pitch value is used: row pitch * height if geometry is HSA_EXT_IMAGE_GEOMETRY_3D, HSA_EXT_IMAGE_GEOMETRY_2DA, or HSA_EXT_IMAGE_GEOMETRY_2DADEPTH; row pitch if geometry is HSA_EXT_IMAGE_GEOMETRY_1DA; and 0 otherwise. The value used must be 0 if the default slice pitch is 0, be greater than or equal to the default slice pitch, and be a multiple of the row pitch.
out	*image_data_info*	Memory location where the runtime stores the size and alignment requirements. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_EXT_STATUS_ERROR_IMAGE_FORMAT_UNSUPPORTED	The image format specified by `image_descriptor` is not supported for the `access_permission` and `image_data_layout` specified.
HSA_EXT_STATUS_ERROR_IMAGE_SIZE_UNSUPPORTED	The image dimensions specified by `image_descriptor` are not supported for the `access_permission` and `image_data_layout` specified.
HSA_EXT_STATUS_ERROR_IMAGE_PITCH_UNSUPPORTED	The row and slice pitch specified by `image_data_row_pitch` and `image_data_slice_pitch` are invalid or not supported.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`image_descriptor` is NULL, `image_data_layout` is HSA_EXT_IMAGE_DATA_LAYOUT_OPAQUE, or `image_data_info` is NULL.

5.12.5.7 hsa_ext_image_destroy()

```
hsa_status_t HSA_API hsa_ext_image_destroy (
```

```
            hsa_agent_t agent,
            hsa_ext_image_t image )
```

Destroy an image handle previously created using hsa_ext_image_create or hsa_ext_image_create_with_layout.

Destroying the image handle does not free the associated image data, or modify its contents. The application should not destroy an image handle while there are references to it queued for execution or currently being used in a kernel dispatch.

Parameters

in	*agent*	Agent associated with the image handle.
in	*image*	Image handle to destroy.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.

5.12.5.8 hsa_ext_image_export()

```
hsa_status_t HSA_API hsa_ext_image_export (
            hsa_agent_t agent,
            hsa_ext_image_t src_image,
            void * dst_memory,
            size_t dst_row_pitch,
            size_t dst_slice_pitch,
            const hsa_ext_image_region_t * image_region )
```

Export the image data to linearly organized memory.

The operation updates the destination memory with the image data of `src_image`. The size of the data exported to memory is implicitly derived from the image region.

It is the application's responsibility to avoid out of bounds memory access.

None of the destination memory or source image data memory can overlap. Overlapping of any of the source and destination image data memory within the export operation produces undefined results.

Parameters

in	*agent*	Agent associated with the image handle.
in	*src_image*	Image handle of source image.
in	*dst_memory*	Destination memory. Must not be NULL.
in	*dst_row_pitch*	The size in bytes of a single row of the image in the destination memory. If the value is smaller than the source image region width * image element byte size, then region width * image element byte size is used.

Parameters

in	*dst_slice_pitch*	The size in bytes of a single 2D slice of a 3D image, or the size in bytes of each image in an image array in the destination memory. If the geometry is HSA_EXT_IMAGE_GEOMETRY_1DA and the value is smaller than the value used for `dst_row_pitch`, then the value used for `dst_row_pitch` is used. If the geometry is HSA_EXT_IMAGE_GEOMETRY_3D, HSA_EXT_IMAGE_GEOMETRY_2DA, or HSA_EXT_IMAGE_GEOMETRY_2DADEPTH and the value is smaller than the value used for `dst_row_pitch` * source image region height, then the value used for `dst_row_pitch` * source image region height is used. Otherwise, the value is not used.
in	*image_region*	Pointer to the image region to be exported. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`dst_memory` is NULL, or `image_region` is NULL.

5.12.5.9 hsa_ext_image_get_capability()

```
hsa_status_t HSA_API hsa_ext_image_get_capability (
            hsa_agent_t agent,
            hsa_ext_image_geometry_t geometry,
            const hsa_ext_image_format_t * image_format,
            uint32_t * capability_mask )
```

Retrieve the supported image capabilities for a given combination of agent, geometry, and image format for an image created with an opaque image data layout.

Parameters

in	*agent*	Agent to be associated with the image handle.
in	*geometry*	Geometry.
in	*image_format*	Pointer to an image format. Must not be NULL.
out	*capability_mask*	Pointer to a memory location where the HSA runtime stores a bit-mask of supported image capability (hsa_ext_image_capability_t) values. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`image_format` is NULL, or `capability_mask` is NULL.

5.12.5.10 hsa_ext_image_get_capability_with_layout()

```
hsa_status_t HSA_API hsa_ext_image_get_capability_with_layout (
            hsa_agent_t agent,
            hsa_ext_image_geometry_t geometry,
            const hsa_ext_image_format_t * image_format,
            hsa_ext_image_data_layout_t image_data_layout,
            uint32_t * capability_mask )
```

Retrieve the supported image capabilities for a given combination of agent, geometry, image format, and image layout for an image created with an explicit image data layout.

Parameters

in	*agent*	Agent to be associated with the image handle.
in	*geometry*	Geometry.
in	*image_format*	Pointer to an image format. Must not be NULL.
in	*image_data_layout*	The image data layout. It is invalid to use HSA_EXT_IMAGE_DATA_LAYOUT_OPAQUE; use hsa_ext_image_get_capability instead.
out	*capability_mask*	Pointer to a memory location where the HSA runtime stores a bit-mask of supported image capability (hsa_ext_image_capability_t) values. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`image_format` is NULL, `image_data_layout` is HSA_EXT_IMAGE_DATA_LAYOUT_OPAQUE, or `capability_mask` is NULL.

5.12.5.11 hsa_ext_image_import()

```
hsa_status_t HSA_API hsa_ext_image_import (
            hsa_agent_t agent,
            const void * src_memory,
            size_t src_row_pitch,
            size_t src_slice_pitch,
            hsa_ext_image_t dst_image,
            const hsa_ext_image_region_t * image_region )
```

Import a linearly organized image data from memory directly to an image handle.

This operation updates the image data referenced by the image handle from the source memory. The size of the data imported from memory is implicitly derived from the image region.

It is the application's responsibility to avoid out of bounds memory access.

None of the source memory or destination image data memory can overlap. Overlapping of any of the source and destination image data memory within the import operation produces undefined results.

Parameters

in	*agent*	Agent associated with the image handle.
in	*src_memory*	Source memory. Must not be NULL.
in	*src_row_pitch*	The size in bytes of a single row of the image in the source memory. If the value is smaller than the destination image region width * image element byte size, then region width * image element byte size is used.
in	*src_slice_pitch*	The size in bytes of a single 2D slice of a 3D image, or the size in bytes of each image layer in an image array in the source memory. If the geometry is HSA_EXT_IMAGE_GEOMETRY_1DA and the value is smaller than the value used for `src_row_pitch`, then the value used for `src_row_pitch` is used. If the geometry is HSA_EXT_IMAGE_GEOMETRY_3D, HSA_EXT_IMAGE_GEOMETRY_2DA, or HSA_EXT_IMAGE_GEOMETRY_2DADEPTH and the value is smaller than the value used for `src_row_pitch` * destination image region height, then the value used for `src_row_pitch` * destination image region height is used. Otherwise, the value is not used.
in	*dst_image*	Image handle of destination image.
in	*image_region*	Pointer to the image region to be updated. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`src_memory` is NULL, or `image_region` is NULL.

5.12.5.12 hsa_ext_sampler_create()

```
hsa_status_t HSA_API hsa_ext_sampler_create (
             hsa_agent_t agent,
             const hsa_ext_sampler_descriptor_t * sampler_descriptor,
             hsa_ext_sampler_t * sampler )
```

Create an agent specific sampler handle for a given agent independent sampler descriptor and agent.

Parameters

in	*agent*	Agent to be associated with the sampler handle created.
in	*sampler_descriptor*	Pointer to a sampler descriptor. Must not be NULL.
out	*sampler*	Memory location where the HSA runtime stores the newly created sampler handle. Must not be NULL.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.
HSA_EXT_STATUS_ERROR_SAMPLER_DESCRIPTOR_UNSUPPORTED	The agent does not have the capability to support the properties specified by `sampler_descriptor` or it is invalid.
HSA_STATUS_ERROR_OUT_OF_RESOURCES	The HSA runtime failed to allocate the required resources.
HSA_STATUS_ERROR_INVALID_ARGUMENT	`sampler_descriptor` is NULL, or `sampler` is NULL.

5.12.5.13 hsa_ext_sampler_destroy()

```
hsa_status_t HSA_API hsa_ext_sampler_destroy (
             hsa_agent_t agent,
             hsa_ext_sampler_t sampler )
```

Destroy a sampler handle previously created using hsa_ext_sampler_create.

The sampler handle should not be destroyed while there are references to it queued for execution or currently being used in a kernel dispatch.

Parameters

`in`	*agent*	Agent associated with the sampler handle.
`in`	*sampler*	Sampler handle to destroy.

Return values

HSA_STATUS_SUCCESS	The function has been executed successfully.
HSA_STATUS_ERROR_NOT_INITIALIZED	The HSA runtime has not been initialized.
HSA_STATUS_ERROR_INVALID_AGENT	The agent is invalid.

Chapter 6

Class Documentation

6.1 amd_control_directives_s Struct Reference

Public Attributes

- amd_enabled_control_directive64_t enabled_control_directives
- uint16_t enable_break_exceptions
- uint16_t enable_detect_exceptions
- uint32_t max_dynamic_group_size
- uint64_t max_flat_grid_size
- uint32_t max_flat_workgroup_size
- uint8_t required_dim
- uint8_t reserved1 [3]
- uint64_t required_grid_size [3]
- uint32_t required_workgroup_size [3]
- uint8_t reserved2 [60]

6.1.1 Detailed Description

Definition at line 205 of file amd_hsa_kernel_code.h.

6.1.2 Member Data Documentation

6.1.2.1 enable_break_exceptions

```
uint16_t amd_control_directives_s::enable_break_exceptions
```

Definition at line 207 of file amd_hsa_kernel_code.h.

6.1.2.2 enable_detect_exceptions

```
uint16_t amd_control_directives_s::enable_detect_exceptions
```

Definition at line 208 of file amd_hsa_kernel_code.h.

6.1.2.3 enabled_control_directives

```
amd_enabled_control_directive64_t amd_control_directives_s::enabled_control_directives
```

Definition at line 206 of file amd_hsa_kernel_code.h.

6.1.2.4 max_dynamic_group_size

```
uint32_t amd_control_directives_s::max_dynamic_group_size
```

Definition at line 209 of file amd_hsa_kernel_code.h.

6.1.2.5 max_flat_grid_size

```
uint64_t amd_control_directives_s::max_flat_grid_size
```

Definition at line 210 of file amd_hsa_kernel_code.h.

6.1.2.6 max_flat_workgroup_size

```
uint32_t amd_control_directives_s::max_flat_workgroup_size
```

Definition at line 211 of file amd_hsa_kernel_code.h.

6.1.2.7 required_dim

```
uint8_t amd_control_directives_s::required_dim
```

Definition at line 212 of file amd_hsa_kernel_code.h.

6.1.2.8 required_grid_size

```
uint64_t amd_control_directives_s::required_grid_size[3]
```

Definition at line 214 of file amd_hsa_kernel_code.h.

6.1.2.9 required_workgroup_size

```
uint32_t amd_control_directives_s::required_workgroup_size[3]
```

Definition at line 215 of file amd_hsa_kernel_code.h.

6.1.2.10 reserved1

```
uint8_t amd_control_directives_s::reserved1[3]
```

Definition at line 213 of file amd_hsa_kernel_code.h.

6.1.2.11 reserved2

```
uint8_t amd_control_directives_s::reserved2[60]
```

Definition at line 216 of file amd_hsa_kernel_code.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_kernel_code.h

6.2 amd_kernel_code_s Struct Reference

Collaboration diagram for amd_kernel_code_s:

Public Attributes

- amd_kernel_code_version32_t amd_kernel_code_version_major
- amd_kernel_code_version32_t amd_kernel_code_version_minor
- amd_machine_kind16_t amd_machine_kind
- amd_machine_version16_t amd_machine_version_major
- amd_machine_version16_t amd_machine_version_minor
- amd_machine_version16_t amd_machine_version_stepping
- int64_t kernel_code_entry_byte_offset
- int64_t kernel_code_prefetch_byte_offset
- uint64_t kernel_code_prefetch_byte_size
- uint64_t max_scratch_backing_memory_byte_size
- amd_compute_pgm_rsrc_one32_t compute_pgm_rsrc1
- amd_compute_pgm_rsrc_two32_t compute_pgm_rsrc2
- amd_kernel_code_properties32_t kernel_code_properties
- uint32_t workitem_private_segment_byte_size
- uint32_t workgroup_group_segment_byte_size
- uint32_t gds_segment_byte_size
- uint64_t kernarg_segment_byte_size
- uint32_t workgroup_fbarrier_count
- uint16_t wavefront_sgpr_count
- uint16_t workitem_vgpr_count
- uint16_t reserved_vgpr_first
- uint16_t reserved_vgpr_count
- uint16_t reserved_sgpr_first
- uint16_t reserved_sgpr_count
- uint16_t debug_wavefront_private_segment_offset_sgpr
- uint16_t debug_private_segment_buffer_sgpr
- amd_powertwo8_t kernarg_segment_alignment
- amd_powertwo8_t group_segment_alignment
- amd_powertwo8_t private_segment_alignment
- amd_powertwo8_t wavefront_size
- int32_t call_convention
- uint8_t reserved1 [12]
- uint64_t runtime_loader_kernel_symbol
- amd_control_directives_t control_directives

6.2.1 Detailed Description

Definition at line 223 of file amd_hsa_kernel_code.h.

6.2.2 Member Data Documentation

6.2.2.1 amd_kernel_code_version_major

```
amd_kernel_code_version32_t amd_kernel_code_s::amd_kernel_code_version_major
```

Definition at line 224 of file amd_hsa_kernel_code.h.

6.2.2.2 amd_kernel_code_version_minor

```
amd_kernel_code_version32_t amd_kernel_code_s::amd_kernel_code_version_minor
```

Definition at line 225 of file amd_hsa_kernel_code.h.

6.2.2.3 amd_machine_kind

```
amd_machine_kind16_t amd_kernel_code_s::amd_machine_kind
```

Definition at line 226 of file amd_hsa_kernel_code.h.

6.2.2.4 amd_machine_version_major

```
amd_machine_version16_t amd_kernel_code_s::amd_machine_version_major
```

Definition at line 227 of file amd_hsa_kernel_code.h.

6.2.2.5 amd_machine_version_minor

```
amd_machine_version16_t amd_kernel_code_s::amd_machine_version_minor
```

Definition at line 228 of file amd_hsa_kernel_code.h.

6.2.2.6 amd_machine_version_stepping

```
amd_machine_version16_t amd_kernel_code_s::amd_machine_version_stepping
```

Definition at line 229 of file amd_hsa_kernel_code.h.

6.2.2.7 call_convention

```
int32_t amd_kernel_code_s::call_convention
```

Definition at line 254 of file amd_hsa_kernel_code.h.

6.2.2.8 compute_pgm_rsrc1

```
amd_compute_pgm_rsrc_one32_t amd_kernel_code_s::compute_pgm_rsrc1
```

Definition at line 234 of file amd_hsa_kernel_code.h.

6.2.2.9 compute_pgm_rsrc2

```
amd_compute_pgm_rsrc_two32_t amd_kernel_code_s::compute_pgm_rsrc2
```

Definition at line 235 of file amd_hsa_kernel_code.h.

6.2.2.10 control_directives

```
amd_control_directives_t amd_kernel_code_s::control_directives
```

Definition at line 257 of file amd_hsa_kernel_code.h.

6.2.2.11 debug_private_segment_buffer_sgpr

```
uint16_t amd_kernel_code_s::debug_private_segment_buffer_sgpr
```

Definition at line 249 of file amd_hsa_kernel_code.h.

6.2.2.12 debug_wavefront_private_segment_offset_sgpr

```
uint16_t amd_kernel_code_s::debug_wavefront_private_segment_offset_sgpr
```

Definition at line 248 of file amd_hsa_kernel_code.h.

6.2.2.13 gds_segment_byte_size

```
uint32_t amd_kernel_code_s::gds_segment_byte_size
```

Definition at line 239 of file amd_hsa_kernel_code.h.

6.2.2.14 group_segment_alignment

```
amd_powertwo8_t amd_kernel_code_s::group_segment_alignment
```

Definition at line 251 of file amd_hsa_kernel_code.h.

6.2.2.15 kernarg_segment_alignment

```
amd_powertwo8_t amd_kernel_code_s::kernarg_segment_alignment
```

Definition at line 250 of file amd_hsa_kernel_code.h.

6.2.2.16 kernarg_segment_byte_size

```
uint64_t amd_kernel_code_s::kernarg_segment_byte_size
```

Definition at line 240 of file amd_hsa_kernel_code.h.

6.2.2.17 kernel_code_entry_byte_offset

```
int64_t amd_kernel_code_s::kernel_code_entry_byte_offset
```

Definition at line 230 of file amd_hsa_kernel_code.h.

6.2.2.18 kernel_code_prefetch_byte_offset

```
int64_t amd_kernel_code_s::kernel_code_prefetch_byte_offset
```

Definition at line 231 of file amd_hsa_kernel_code.h.

6.2.2.19 kernel_code_prefetch_byte_size

```
uint64_t amd_kernel_code_s::kernel_code_prefetch_byte_size
```

Definition at line 232 of file amd_hsa_kernel_code.h.

6.2.2.20 kernel_code_properties

```
amd_kernel_code_properties32_t amd_kernel_code_s::kernel_code_properties
```

Definition at line 236 of file amd_hsa_kernel_code.h.

6.2.2.21 max_scratch_backing_memory_byte_size

```
uint64_t amd_kernel_code_s::max_scratch_backing_memory_byte_size
```

Definition at line 233 of file amd_hsa_kernel_code.h.

6.2.2.22 private_segment_alignment

```
amd_powertwo8_t amd_kernel_code_s::private_segment_alignment
```

Definition at line 252 of file amd_hsa_kernel_code.h.

6.2.2.23 reserved1

```
uint8_t amd_kernel_code_s::reserved1[12]
```

Definition at line 255 of file amd_hsa_kernel_code.h.

6.2.2.24 reserved_sgpr_count

```
uint16_t amd_kernel_code_s::reserved_sgpr_count
```

Definition at line 247 of file amd_hsa_kernel_code.h.

6.2.2.25 reserved_sgpr_first

```
uint16_t amd_kernel_code_s::reserved_sgpr_first
```

Definition at line 246 of file amd_hsa_kernel_code.h.

6.2.2.26 reserved_vgpr_count

```
uint16_t amd_kernel_code_s::reserved_vgpr_count
```

Definition at line 245 of file amd_hsa_kernel_code.h.

6.2.2.27 reserved_vgpr_first

```
uint16_t amd_kernel_code_s::reserved_vgpr_first
```

Definition at line 244 of file amd_hsa_kernel_code.h.

6.2.2.28 runtime_loader_kernel_symbol

```
uint64_t amd_kernel_code_s::runtime_loader_kernel_symbol
```

Definition at line 256 of file amd_hsa_kernel_code.h.

6.2.2.29 wavefront_sgpr_count

```
uint16_t amd_kernel_code_s::wavefront_sgpr_count
```

Definition at line 242 of file amd_hsa_kernel_code.h.

6.2.2.30 wavefront_size

```
amd_powertwo8_t amd_kernel_code_s::wavefront_size
```

Definition at line 253 of file amd_hsa_kernel_code.h.

6.2.2.31 workgroup_fbarrier_count

```
uint32_t amd_kernel_code_s::workgroup_fbarrier_count
```

Definition at line 241 of file amd_hsa_kernel_code.h.

6.2.2.32 workgroup_group_segment_byte_size

```
uint32_t amd_kernel_code_s::workgroup_group_segment_byte_size
```

Definition at line 238 of file amd_hsa_kernel_code.h.

6.2.2.33 workitem_private_segment_byte_size

```
uint32_t amd_kernel_code_s::workitem_private_segment_byte_size
```

Definition at line 237 of file amd_hsa_kernel_code.h.

6.2.2.34 workitem_vgpr_count

```
uint16_t amd_kernel_code_s::workitem_vgpr_count
```

Definition at line 243 of file amd_hsa_kernel_code.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_kernel_code.h

6.3 amd_queue_s Struct Reference

Collaboration diagram for amd_queue_s:

Public Attributes

- hsa_queue_t hsa_queue
- uint32_t reserved1 [4]
- volatile uint64_t write_dispatch_id
- uint32_t group_segment_aperture_base_hi
- uint32_t private_segment_aperture_base_hi
- uint32_t max_cu_id
- uint32_t max_wave_id
- volatile uint64_t max_legacy_doorbell_dispatch_id_plus_1
- volatile uint32_t legacy_doorbell_lock
- uint32_t reserved2 [9]
- volatile uint64_t read_dispatch_id
- uint32_t read_dispatch_id_field_base_byte_offset
- uint32_t compute_tmpring_size
- uint32_t scratch_resource_descriptor [4]
- uint64_t scratch_backing_memory_location
- uint64_t scratch_backing_memory_byte_size
- uint32_t scratch_wave64_lane_byte_size
- amd_queue_properties32_t queue_properties
- uint32_t reserved3 [2]
- hsa_signal_t queue_inactive_signal
- uint32_t reserved4 [14]

6.3.1 Detailed Description

Definition at line 63 of file amd_hsa_queue.h.

6.3.2 Member Data Documentation

6.3.2.1 compute_tmpring_size

```
uint32_t amd_queue_s::compute_tmpring_size
```

Definition at line 76 of file amd_hsa_queue.h.

6.3.2.2 group_segment_aperture_base_hi

```
uint32_t amd_queue_s::group_segment_aperture_base_hi
```

Definition at line 67 of file amd_hsa_queue.h.

6.3.2.3 hsa_queue

```
hsa_queue_t amd_queue_s::hsa_queue
```

Definition at line 64 of file amd_hsa_queue.h.

6.3.2.4 legacy_doorbell_lock

```
volatile uint32_t amd_queue_s::legacy_doorbell_lock
```

Definition at line 72 of file amd_hsa_queue.h.

6.3.2.5 max_cu_id

```
uint32_t amd_queue_s::max_cu_id
```

Definition at line 69 of file amd_hsa_queue.h.

6.3.2.6 max_legacy_doorbell_dispatch_id_plus_1

```
volatile uint64_t amd_queue_s::max_legacy_doorbell_dispatch_id_plus_1
```

Definition at line 71 of file amd_hsa_queue.h.

6.3.2.7 max_wave_id

```
uint32_t amd_queue_s::max_wave_id
```

Definition at line 70 of file amd_hsa_queue.h.

6.3.2.8 private_segment_aperture_base_hi

```
uint32_t amd_queue_s::private_segment_aperture_base_hi
```

Definition at line 68 of file amd_hsa_queue.h.

6.3.2.9 queue_inactive_signal

```
hsa_signal_t amd_queue_s::queue_inactive_signal
```

Definition at line 83 of file amd_hsa_queue.h.

6.3.2.10 queue_properties

```
amd_queue_properties32_t amd_queue_s::queue_properties
```

Definition at line 81 of file amd_hsa_queue.h.

6.3.2.11 read_dispatch_id

```
volatile uint64_t amd_queue_s::read_dispatch_id
```

Definition at line 74 of file amd_hsa_queue.h.

6.3.2.12 read_dispatch_id_field_base_byte_offset

```
uint32_t amd_queue_s::read_dispatch_id_field_base_byte_offset
```

Definition at line 75 of file amd_hsa_queue.h.

6.3.2.13 reserved1

```
uint32_t amd_queue_s::reserved1[4]
```

Definition at line 65 of file amd_hsa_queue.h.

6.3.2.14 reserved2

```
uint32_t amd_queue_s::reserved2[9]
```

Definition at line 73 of file amd_hsa_queue.h.

6.3.2.15 reserved3

```
uint32_t amd_queue_s::reserved3[2]
```

Definition at line 82 of file amd_hsa_queue.h.

6.3.2.16 reserved4

```
uint32_t amd_queue_s::reserved4[14]
```

Definition at line 84 of file amd_hsa_queue.h.

6.3.2.17 scratch_backing_memory_byte_size

```
uint64_t amd_queue_s::scratch_backing_memory_byte_size
```

Definition at line 79 of file amd_hsa_queue.h.

6.3.2.18 scratch_backing_memory_location

```
uint64_t amd_queue_s::scratch_backing_memory_location
```

Definition at line 78 of file amd_hsa_queue.h.

6.3.2.19 scratch_resource_descriptor

```
uint32_t amd_queue_s::scratch_resource_descriptor[4]
```

Definition at line 77 of file amd_hsa_queue.h.

6.3.2.20 scratch_wave64_lane_byte_size

```
uint32_t amd_queue_s::scratch_wave64_lane_byte_size
```

Definition at line 80 of file amd_hsa_queue.h.

6.3.2.21 write_dispatch_id

```
volatile uint64_t amd_queue_s::write_dispatch_id
```

Definition at line 66 of file amd_hsa_queue.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_queue.h

6.4 amd_runtime_loader_debug_info_s Struct Reference

Public Attributes

- const void * elf_raw
- size_t elf_size
- const char * kernel_name
- const void * owning_segment

6.4.1 Detailed Description

Definition at line 262 of file amd_hsa_kernel_code.h.

6.4.2 Member Data Documentation

6.4.2.1 elf_raw

```
const void* amd_runtime_loader_debug_info_s::elf_raw
```

Definition at line 263 of file amd_hsa_kernel_code.h.

6.4.2.2 elf_size

```
size_t amd_runtime_loader_debug_info_s::elf_size
```

Definition at line 264 of file amd_hsa_kernel_code.h.

6.4.2.3 kernel_name

```
const char* amd_runtime_loader_debug_info_s::kernel_name
```

Definition at line 265 of file amd_hsa_kernel_code.h.

6.4.2.4 owning_segment

```
const void* amd_runtime_loader_debug_info_s::owning_segment
```

Definition at line 266 of file amd_hsa_kernel_code.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_kernel_code.h

6.5 amd_signal_s Struct Reference

Public Attributes

- amd_signal_kind64_t kind
- union {
  volatile int64_t value
  volatile uint32_t * legacy_hardware_doorbell_ptr
  volatile uint64_t * hardware_doorbell_ptr
  };
- uint64_t event_mailbox_ptr
- uint32_t event_id
- uint32_t reserved1
- uint64_t start_ts
- uint64_t end_ts
- union {
  amd_queue_t * queue_ptr
  uint64_t reserved2
  };
- uint32_t reserved3 [2]

6.5.1 Detailed Description

Definition at line 61 of file amd_hsa_signal.h.

6.5.2 Member Data Documentation

6.5.2.1 end_ts

```
uint64_t amd_signal_s::end_ts
```

Definition at line 72 of file amd_hsa_signal.h.

6.5.2.2 event_id

```
uint32_t amd_signal_s::event_id
```

Definition at line 69 of file amd_hsa_signal.h.

6.5.2.3 event_mailbox_ptr

```
uint64_t amd_signal_s::event_mailbox_ptr
```

Definition at line 68 of file amd_hsa_signal.h.

6.5.2.4 hardware_doorbell_ptr

```
volatile uint64_t* amd_signal_s::hardware_doorbell_ptr
```

Definition at line 66 of file amd_hsa_signal.h.

6.5.2.5 kind

```
amd_signal_kind64_t amd_signal_s::kind
```

Definition at line 62 of file amd_hsa_signal.h.

6.5.2.6 legacy_hardware_doorbell_ptr

```
volatile uint32_t* amd_signal_s::legacy_hardware_doorbell_ptr
```

Definition at line 65 of file amd_hsa_signal.h.

6.5.2.7 queue_ptr

```
amd_queue_t* amd_signal_s::queue_ptr
```

Definition at line 74 of file amd_hsa_signal.h.

6.5.2.8 reserved1

```
uint32_t amd_signal_s::reserved1
```

Definition at line 70 of file amd_hsa_signal.h.

6.5.2.9 reserved2

```
uint64_t amd_signal_s::reserved2
```

Definition at line 75 of file amd_hsa_signal.h.

6.5.2.10 reserved3

```
uint32_t amd_signal_s::reserved3[2]
```

Definition at line 77 of file amd_hsa_signal.h.

6.5.2.11 start_ts

```
uint64_t amd_signal_s::start_ts
```

Definition at line 71 of file amd_hsa_signal.h.

6.5.2.12 value

```
volatile int64_t amd_signal_s::value
```

Definition at line 64 of file amd_hsa_signal.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_signal.h

6.6 AmdExtTable Struct Reference

Collaboration diagram for AmdExtTable:

6.7 amdgpu_hsa_image_descriptor_s Struct Reference

Public Attributes

- uint16_t size
- amdgpu_hsa_metadata_kind16_t kind
- amdgpu_hsa_image_geometry8_t geometry
- amdgpu_hsa_image_channel_order8_t channel_order
- amdgpu_hsa_image_channel_type8_t channel_type
- uint8_t reserved1
- uint64_t width
- uint64_t height
- uint64_t depth
- uint64_t array

6.7.1 Detailed Description

Definition at line 367 of file amd_hsa_elf.h.

6.7.2 Member Data Documentation

6.7.2.1 array

```
uint64_t amdgpu_hsa_image_descriptor_s::array
```

Definition at line 377 of file amd_hsa_elf.h.

6.7.2.2 channel_order

```
amdgpu_hsa_image_channel_order8_t amdgpu_hsa_image_descriptor_s::channel_order
```

Definition at line 371 of file amd_hsa_elf.h.

6.7.2.3 channel_type

```
amdgpu_hsa_image_channel_type8_t amdgpu_hsa_image_descriptor_s::channel_type
```

Definition at line 372 of file amd_hsa_elf.h.

6.7.2.4 depth

```
uint64_t amdgpu_hsa_image_descriptor_s::depth
```

Definition at line 376 of file amd_hsa_elf.h.

6.7.2.5 geometry

```
amdgpu_hsa_image_geometry8_t amdgpu_hsa_image_descriptor_s::geometry
```

Definition at line 370 of file amd_hsa_elf.h.

6.7.2.6 height

```
uint64_t amdgpu_hsa_image_descriptor_s::height
```

Definition at line 375 of file amd_hsa_elf.h.

6.7.2.7 kind

```
amdgpu_hsa_metadata_kind16_t amdgpu_hsa_image_descriptor_s::kind
```

Definition at line 369 of file amd_hsa_elf.h.

6.7.2.8 reserved1

```
uint8_t amdgpu_hsa_image_descriptor_s::reserved1
```

Definition at line 373 of file amd_hsa_elf.h.

6.7.2.9 size

```
uint16_t amdgpu_hsa_image_descriptor_s::size
```

Definition at line 368 of file amd_hsa_elf.h.

6.7.2.10 width

```
uint64_t amdgpu_hsa_image_descriptor_s::width
```

Definition at line 374 of file amd_hsa_elf.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_elf.h

6.8 amdgpu_hsa_note_code_object_version_s Struct Reference

Public Attributes

- uint32_t major_version
- uint32_t minor_version

6.8.1 Detailed Description

Definition at line 380 of file amd_hsa_elf.h.

6.8.2 Member Data Documentation

6.8.2.1 major_version

```
uint32_t amdgpu_hsa_note_code_object_version_s::major_version
```

Definition at line 381 of file amd_hsa_elf.h.

6.8.2.2 minor_version

```
uint32_t amdgpu_hsa_note_code_object_version_s::minor_version
```

Definition at line 382 of file amd_hsa_elf.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_elf.h

6.9 amdgpu_hsa_note_hsail_s Struct Reference

Public Attributes

- uint32_t hsail_major_version
- uint32_t hsail_minor_version
- uint8_t profile
- uint8_t machine_model
- uint8_t default_float_round

6.9.1 Detailed Description

Definition at line 385 of file amd_hsa_elf.h.

6.9.2 Member Data Documentation

6.9.2.1 default_float_round

```
uint8_t amdgpu_hsa_note_hsail_s::default_float_round
```

Definition at line 390 of file amd_hsa_elf.h.

6.9.2.2 hsail_major_version

```
uint32_t amdgpu_hsa_note_hsail_s::hsail_major_version
```

Definition at line 386 of file amd_hsa_elf.h.

6.9.2.3 hsail_minor_version

```
uint32_t amdgpu_hsa_note_hsail_s::hsail_minor_version
```

Definition at line 387 of file amd_hsa_elf.h.

6.9.2.4 machine_model

```
uint8_t amdgpu_hsa_note_hsail_s::machine_model
```

Definition at line 389 of file amd_hsa_elf.h.

6.9.2.5 profile

```
uint8_t amdgpu_hsa_note_hsail_s::profile
```

Definition at line 388 of file amd_hsa_elf.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_elf.h

6.10 amdgpu_hsa_note_isa_s Struct Reference

Public Attributes

- uint16_t vendor_name_size
- uint16_t architecture_name_size
- uint32_t major
- uint32_t minor
- uint32_t stepping
- char vendor_and_architecture_name [1]

6.10.1 Detailed Description

Definition at line 393 of file amd_hsa_elf.h.

6.10.2 Member Data Documentation

6.10.2.1 architecture_name_size

```
uint16_t amdgpu_hsa_note_isa_s::architecture_name_size
```

Definition at line 395 of file amd_hsa_elf.h.

6.10.2.2 major

```
uint32_t amdgpu_hsa_note_isa_s::major
```

Definition at line 396 of file amd_hsa_elf.h.

6.10.2.3 minor

```
uint32_t amdgpu_hsa_note_isa_s::minor
```

Definition at line 397 of file amd_hsa_elf.h.

6.10.2.4 stepping

```
uint32_t amdgpu_hsa_note_isa_s::stepping
```

Definition at line 398 of file amd_hsa_elf.h.

6.10.2.5 vendor_and_architecture_name

```
char amdgpu_hsa_note_isa_s::vendor_and_architecture_name[1]
```

Definition at line 399 of file amd_hsa_elf.h.

6.10.2.6 vendor_name_size

```
uint16_t amdgpu_hsa_note_isa_s::vendor_name_size
```

Definition at line 394 of file amd_hsa_elf.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_elf.h

6.11 amdgpu_hsa_note_producer_options_s Struct Reference

Public Attributes

- uint16_t producer_options_size
- char producer_options [1]

6.11.1 Detailed Description

Definition at line 410 of file amd_hsa_elf.h.

6.11.2 Member Data Documentation

6.11.2.1 producer_options

```
char amdgpu_hsa_note_producer_options_s::producer_options[1]
```

Definition at line 412 of file amd_hsa_elf.h.

6.11.2.2 producer_options_size

```
uint16_t amdgpu_hsa_note_producer_options_s::producer_options_size
```

Definition at line 411 of file amd_hsa_elf.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_elf.h

6.12 amdgpu_hsa_note_producer_s Struct Reference

Public Attributes

- uint16_t producer_name_size
- uint16_t reserved
- uint32_t producer_major_version
- uint32_t producer_minor_version
- char producer_name [1]

6.12.1 Detailed Description

Definition at line 402 of file amd_hsa_elf.h.

6.12.2 Member Data Documentation

6.12.2.1 producer_major_version

```
uint32_t amdgpu_hsa_note_producer_s::producer_major_version
```

Definition at line 405 of file amd_hsa_elf.h.

6.12.2.2 producer_minor_version

```
uint32_t amdgpu_hsa_note_producer_s::producer_minor_version
```

Definition at line 406 of file amd_hsa_elf.h.

6.12.2.3 producer_name

```
char amdgpu_hsa_note_producer_s::producer_name[1]
```

Definition at line 407 of file amd_hsa_elf.h.

6.12.2.4 producer_name_size

```
uint16_t amdgpu_hsa_note_producer_s::producer_name_size
```

Definition at line 403 of file amd_hsa_elf.h.

6.12.2.5 reserved

```
uint16_t amdgpu_hsa_note_producer_s::reserved
```

Definition at line 404 of file amd_hsa_elf.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_elf.h

6.13 amdgpu_hsa_sampler_descriptor_s Struct Reference

Public Attributes

- uint16_t size
- amdgpu_hsa_metadata_kind16_t kind
- amdgpu_hsa_sampler_coord8_t coord
- amdgpu_hsa_sampler_filter8_t filter
- amdgpu_hsa_sampler_addressing8_t addressing
- uint8_t reserved1

6.13.1 Detailed Description

Definition at line 298 of file amd_hsa_elf.h.

6.13.2 Member Data Documentation

6.13.2.1 addressing

```
amdgpu_hsa_sampler_addressing8_t amdgpu_hsa_sampler_descriptor_s::addressing
```

Definition at line 303 of file amd_hsa_elf.h.

6.13.2.2 coord

```
amdgpu_hsa_sampler_coord8_t amdgpu_hsa_sampler_descriptor_s::coord
```

Definition at line 301 of file amd_hsa_elf.h.

6.13.2.3 filter

```
amdgpu_hsa_sampler_filter8_t amdgpu_hsa_sampler_descriptor_s::filter
```

Definition at line 302 of file amd_hsa_elf.h.

6.13.2.4 kind

```
amdgpu_hsa_metadata_kind16_t amdgpu_hsa_sampler_descriptor_s::kind
```

Definition at line 300 of file amd_hsa_elf.h.

6.13.2.5 reserved1

```
uint8_t amdgpu_hsa_sampler_descriptor_s::reserved1
```

Definition at line 304 of file amd_hsa_elf.h.

6.13.2.6 size

```
uint16_t amdgpu_hsa_sampler_descriptor_s::size
```

Definition at line 299 of file amd_hsa_elf.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/amd_hsa_elf.h

6.14 ApiTableVersion Struct Reference

Public Attributes

- uint32_t major_id
- uint32_t minor_id
- uint32_t step_id
- uint32_t reserved

6.14.1 Detailed Description

Definition at line 103 of file hsa_api_trace.h.

6.14.2 Member Data Documentation

6.14.2.1 major_id

```
uint32_t ApiTableVersion::major_id
```

Definition at line 104 of file hsa_api_trace.h.

6.14.2.2 minor_id

```
uint32_t ApiTableVersion::minor_id
```

Definition at line 105 of file hsa_api_trace.h.

6.14.2.3 reserved

```
uint32_t ApiTableVersion::reserved
```

Definition at line 107 of file hsa_api_trace.h.

6.14.2.4 step_id

```
uint32_t ApiTableVersion::step_id
```

Definition at line 106 of file hsa_api_trace.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_api_trace.h

6.15 BrigBase Struct Reference

Public Attributes

- uint16_t byteCount
- BrigKind16_t kind

6.15.1 Detailed Description

Definition at line 782 of file Brig.h.

6.15.2 Member Data Documentation

6.15.2.1 byteCount

```
uint16_t BrigBase::byteCount
```

Definition at line 783 of file Brig.h.

6.15.2.2 kind

```
BrigKind16_t BrigBase::kind
```

Definition at line 784 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.16 BrigData Struct Reference

Public Attributes

- uint32_t byteCount
- uint8_t bytes [1]

6.16.1 Detailed Description

Definition at line 787 of file Brig.h.

6.16.2 Member Data Documentation

6.16.2.1 byteCount

```
uint32_t BrigData::byteCount
```

Definition at line 788 of file Brig.h.

6.16.2.2 bytes

```
uint8_t BrigData::bytes[1]
```

Definition at line 789 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.17 BrigDirectiveArgBlock Struct Reference

Collaboration diagram for BrigDirectiveArgBlock:

Public Attributes

- BrigBase base

6.17.1 Detailed Description

Definition at line 792 of file Brig.h.

6.17.2 Member Data Documentation

6.17.2.1 base

```
BrigBase BrigDirectiveArgBlock::base
```

Definition at line 793 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.18 BrigDirectiveComment Struct Reference

Collaboration diagram for BrigDirectiveComment:

Public Attributes

- BrigBase base
- BrigDataOffsetString32_t name

6.18.1 Detailed Description

Definition at line 796 of file Brig.h.

6.18.2 Member Data Documentation

6.18.2.1 base

```
BrigBase BrigDirectiveComment::base
```

Definition at line 797 of file Brig.h.

6.18.2.2 name

```
BrigDataOffsetString32_t BrigDirectiveComment::name
```

Definition at line 798 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.19 BrigDirectiveControl Struct Reference

Collaboration diagram for BrigDirectiveControl:

Public Attributes

- BrigBase base
- BrigControlDirective16_t control
- uint16_t reserved
- BrigDataOffsetOperandList32_t operands

6.19.1 Detailed Description

Definition at line 801 of file Brig.h.

6.19.2 Member Data Documentation

6.19.2.1 base

```
BrigBase BrigDirectiveControl::base
```

Definition at line 802 of file Brig.h.

6.19.2.2 control

```
BrigControlDirective16_t BrigDirectiveControl::control
```

Definition at line 803 of file Brig.h.

6.19.2.3 operands

```
BrigDataOffsetOperandList32_t BrigDirectiveControl::operands
```

Definition at line 805 of file Brig.h.

6.19.2.4 reserved

```
uint16_t BrigDirectiveControl::reserved
```

Definition at line 804 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.20 BrigDirectiveExecutable Struct Reference

Collaboration diagram for BrigDirectiveExecutable:

Public Attributes

- BrigBase base
- BrigDataOffsetString32_t name
- uint16_t outArgCount
- uint16_t inArgCount
- BrigCodeOffset32_t firstInArg
- BrigCodeOffset32_t firstCodeBlockEntry
- BrigCodeOffset32_t nextModuleEntry
- BrigExecutableModifier8_t modifier
- BrigLinkage8_t linkage
- uint16_t reserved

6.20.1 Detailed Description

Definition at line 808 of file Brig.h.

6.20.2 Member Data Documentation

6.20.2.1 base

```
BrigBase BrigDirectiveExecutable::base
```

Definition at line 809 of file Brig.h.

6.20.2.2 firstCodeBlockEntry

```
BrigCodeOffset32_t BrigDirectiveExecutable::firstCodeBlockEntry
```

Definition at line 814 of file Brig.h.

6.20.2.3 firstInArg

```
BrigCodeOffset32_t BrigDirectiveExecutable::firstInArg
```

Definition at line 813 of file Brig.h.

6.20.2.4 inArgCount

```
uint16_t BrigDirectiveExecutable::inArgCount
```

Definition at line 812 of file Brig.h.

6.20.2.5 linkage

```
BrigLinkage8_t BrigDirectiveExecutable::linkage
```

Definition at line 817 of file Brig.h.

6.20.2.6 modifier

```
BrigExecutableModifier8_t BrigDirectiveExecutable::modifier
```

Definition at line 816 of file Brig.h.

6.20.2.7 name

```
BrigDataOffsetString32_t BrigDirectiveExecutable::name
```

Definition at line 810 of file Brig.h.

6.20.2.8 nextModuleEntry

```
BrigCodeOffset32_t BrigDirectiveExecutable::nextModuleEntry
```

Definition at line 815 of file Brig.h.

6.20.2.9 outArgCount

```
uint16_t BrigDirectiveExecutable::outArgCount
```

Definition at line 811 of file Brig.h.

6.20.2.10 reserved

```
uint16_t BrigDirectiveExecutable::reserved
```

Definition at line 818 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.21 BrigDirectiveExtension Struct Reference

Collaboration diagram for BrigDirectiveExtension:

Public Attributes

- BrigBase base
- BrigDataOffsetString32_t name

6.21.1 Detailed Description

Definition at line 821 of file Brig.h.

6.21.2 Member Data Documentation

6.21.2.1 base

```
BrigBase BrigDirectiveExtension::base
```

Definition at line 822 of file Brig.h.

6.21.2.2 name

```
BrigDataOffsetString32_t BrigDirectiveExtension::name
```

Definition at line 823 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.22 BrigDirectiveFbarrier Struct Reference

Collaboration diagram for BrigDirectiveFbarrier:

Public Attributes

- BrigBase base
- BrigDataOffsetString32_t name
- BrigVariableModifier8_t modifier
- BrigLinkage8_t linkage
- uint16_t reserved

6.22.1 Detailed Description

Definition at line 826 of file Brig.h.

6.22.2 Member Data Documentation

6.22.2.1 base

```
BrigBase BrigDirectiveFbarrier::base
```

Definition at line 827 of file Brig.h.

6.22.2.2 linkage

```
BrigLinkage8_t BrigDirectiveFbarrier::linkage
```

Definition at line 830 of file Brig.h.

6.22.2.3 modifier

```
BrigVariableModifier8_t BrigDirectiveFbarrier::modifier
```

Definition at line 829 of file Brig.h.

6.22.2.4 name

```
BrigDataOffsetString32_t BrigDirectiveFbarrier::name
```

Definition at line 828 of file Brig.h.

6.22.2.5 reserved

```
uint16_t BrigDirectiveFbarrier::reserved
```

Definition at line 831 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.23 BrigDirectiveLabel Struct Reference

Collaboration diagram for BrigDirectiveLabel:

Public Attributes

- BrigBase base
- BrigDataOffsetString32_t name

6.23.1 Detailed Description

Definition at line 834 of file Brig.h.

6.23.2 Member Data Documentation

6.23.2.1 base

```
BrigBase BrigDirectiveLabel::base
```

Definition at line 835 of file Brig.h.

6.23.2.2 name

```
BrigDataOffsetString32_t BrigDirectiveLabel::name
```

Definition at line 836 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.24 BrigDirectiveLoc Struct Reference

Collaboration diagram for BrigDirectiveLoc:

Public Attributes

- BrigBase base
- BrigDataOffsetString32_t filename
- uint32_t line
- uint32_t column

6.24.1 Detailed Description

Definition at line 839 of file Brig.h.

6.24.2 Member Data Documentation

6.24.2.1 base

```
BrigBase BrigDirectiveLoc::base
```

Definition at line 840 of file Brig.h.

6.24.2.2 column

```
uint32_t BrigDirectiveLoc::column
```

Definition at line 843 of file Brig.h.

6.24.2.3 filename

```
BrigDataOffsetString32_t BrigDirectiveLoc::filename
```

Definition at line 841 of file Brig.h.

6.24.2.4 line

```
uint32_t BrigDirectiveLoc::line
```

Definition at line 842 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.25 BrigDirectiveModule Struct Reference

Collaboration diagram for BrigDirectiveModule:

Public Attributes

- BrigBase base
- BrigDataOffsetString32_t name
- BrigVersion32_t hsailMajor
- BrigVersion32_t hsailMinor
- BrigProfile8_t profile
- BrigMachineModel8_t machineModel
- BrigRound8_t defaultFloatRound
- uint8_t reserved

6.25.1 Detailed Description

Definition at line 869 of file Brig.h.

6.25.2 Member Data Documentation

6.25.2.1 base

```
BrigBase BrigDirectiveModule::base
```

Definition at line 870 of file Brig.h.

6.25.2.2 defaultFloatRound

```
BrigRound8_t BrigDirectiveModule::defaultFloatRound
```

Definition at line 876 of file Brig.h.

6.25.2.3 hsailMajor

```
BrigVersion32_t BrigDirectiveModule::hsailMajor
```

Definition at line 872 of file Brig.h.

6.25.2.4 hsailMinor

```
BrigVersion32_t BrigDirectiveModule::hsailMinor
```

Definition at line 873 of file Brig.h.

6.25.2.5 machineModel

```
BrigMachineModel8_t BrigDirectiveModule::machineModel
```

Definition at line 875 of file Brig.h.

6.25.2.6 name

```
BrigDataOffsetString32_t BrigDirectiveModule::name
```

Definition at line 871 of file Brig.h.

6.25.2.7 profile

```
BrigProfile8_t BrigDirectiveModule::profile
```

Definition at line 874 of file Brig.h.

6.25.2.8 reserved

```
uint8_t BrigDirectiveModule::reserved
```

Definition at line 877 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.26 BrigDirectiveNone Struct Reference

Collaboration diagram for BrigDirectiveNone:

Public Attributes

- BrigBase base

6.26.1 Detailed Description

Definition at line 846 of file Brig.h.

6.26.2 Member Data Documentation

6.26.2.1 base

```
BrigBase BrigDirectiveNone::base
```

Definition at line 847 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.27 BrigDirectivePragma Struct Reference

Collaboration diagram for BrigDirectivePragma:

Public Attributes

- BrigBase base
- BrigDataOffsetOperandList32_t operands

6.27.1 Detailed Description

Definition at line 850 of file Brig.h.

6.27.2 Member Data Documentation

6.27.2.1 base

```
BrigBase BrigDirectivePragma::base
```

Definition at line 851 of file Brig.h.

6.27.2.2 operands

```
BrigDataOffsetOperandList32_t BrigDirectivePragma::operands
```

Definition at line 852 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.28 BrigDirectiveVariable Struct Reference

Collaboration diagram for BrigDirectiveVariable:

Public Attributes

- BrigBase base
- BrigDataOffsetString32_t name
- BrigOperandOffset32_t init
- BrigType16_t type
- BrigSegment8_t segment
- BrigAlignment8_t align
- BrigUInt64 dim
- BrigVariableModifier8_t modifier
- BrigLinkage8_t linkage
- BrigAllocation8_t allocation
- uint8_t reserved

6.28.1 Detailed Description

Definition at line 855 of file Brig.h.

6.28.2 Member Data Documentation

6.28.2.1 align

```
BrigAlignment8_t BrigDirectiveVariable::align
```

Definition at line 861 of file Brig.h.

6.28.2.2 allocation

```
BrigAllocation8_t BrigDirectiveVariable::allocation
```

Definition at line 865 of file Brig.h.

6.28.2.3 base

```
BrigBase BrigDirectiveVariable::base
```

Definition at line 856 of file Brig.h.

6.28.2.4 dim

```
BrigUInt64 BrigDirectiveVariable::dim
```

Definition at line 862 of file Brig.h.

6.28.2.5 init

```
BrigOperandOffset32_t BrigDirectiveVariable::init
```

Definition at line 858 of file Brig.h.

6.28.2.6 linkage

```
BrigLinkage8_t BrigDirectiveVariable::linkage
```

Definition at line 864 of file Brig.h.

6.28.2.7 modifier

```
BrigVariableModifier8_t BrigDirectiveVariable::modifier
```

Definition at line 863 of file Brig.h.

6.28.2.8 name

```
BrigDataOffsetString32_t BrigDirectiveVariable::name
```

Definition at line 857 of file Brig.h.

6.28.2.9 reserved

```
uint8_t BrigDirectiveVariable::reserved
```

Definition at line 866 of file Brig.h.

6.28.2.10 segment

```
BrigSegment8_t BrigDirectiveVariable::segment
```

Definition at line 860 of file Brig.h.

6.28.2.11 type

```
BrigType16_t BrigDirectiveVariable::type
```

Definition at line 859 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.29 BrigInstAddr Struct Reference

Collaboration diagram for BrigInstAddr:

Public Attributes

- BrigInstBase base
- BrigSegment8_t segment
- uint8_t reserved [3]

6.29.1 Detailed Description

Definition at line 887 of file Brig.h.

6.29.2 Member Data Documentation

6.29.2.1 base

```
BrigInstBase BrigInstAddr::base
```

Definition at line 888 of file Brig.h.

6.29.2.2 reserved

```
uint8_t BrigInstAddr::reserved[3]
```

Definition at line 890 of file Brig.h.

6.29.2.3 segment

```
BrigSegment8_t BrigInstAddr::segment
```

Definition at line 889 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.30 BrigInstAtomic Struct Reference

Collaboration diagram for BrigInstAtomic:

Public Attributes

- BrigInstBase base
- BrigSegment8_t segment
- BrigMemoryOrder8_t memoryOrder
- BrigMemoryScope8_t memoryScope
- BrigAtomicOperation8_t atomicOperation
- uint8_t equivClass
- uint8_t reserved [3]

6.30.1 Detailed Description

Definition at line 893 of file Brig.h.

6.30.2 Member Data Documentation

6.30.2.1 atomicOperation

```
BrigAtomicOperation8_t BrigInstAtomic::atomicOperation
```

Definition at line 898 of file Brig.h.

6.30.2.2 base

```
BrigInstBase BrigInstAtomic::base
```

Definition at line 894 of file Brig.h.

6.30.2.3 equivClass

```
uint8_t BrigInstAtomic::equivClass
```

Definition at line 899 of file Brig.h.

6.30.2.4 memoryOrder

```
BrigMemoryOrder8_t BrigInstAtomic::memoryOrder
```

Definition at line 896 of file Brig.h.

6.30.2.5 memoryScope

```
BrigMemoryScope8_t BrigInstAtomic::memoryScope
```

Definition at line 897 of file Brig.h.

6.30.2.6 reserved

```
uint8_t BrigInstAtomic::reserved[3]
```

Definition at line 900 of file Brig.h.

6.30.2.7 segment

```
BrigSegment8_t BrigInstAtomic::segment
```

Definition at line 895 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.31 BrigInstBase Struct Reference

Collaboration diagram for BrigInstBase:

Public Attributes

- BrigBase base
- BrigOpcode16_t opcode
- BrigType16_t type
- BrigDataOffsetOperandList32_t operands

6.31.1 Detailed Description

Definition at line 880 of file Brig.h.

6.31.2 Member Data Documentation

6.31.2.1 base

```
BrigBase BrigInstBase::base
```

Definition at line 881 of file Brig.h.

6.31.2.2 opcode

```
BrigOpcode16_t BrigInstBase::opcode
```

Definition at line 882 of file Brig.h.

6.31.2.3 operands

```
BrigDataOffsetOperandList32_t BrigInstBase::operands
```

Definition at line 884 of file Brig.h.

6.31.2.4 type

```
BrigType16_t BrigInstBase::type
```

Definition at line 883 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.32 BrigInstBasic Struct Reference

Collaboration diagram for BrigInstBasic:

Public Attributes

- BrigInstBase base

6.32.1 Detailed Description

Definition at line 903 of file Brig.h.

6.32.2 Member Data Documentation

6.32.2.1 base

```
BrigInstBase BrigInstBasic::base
```

Definition at line 904 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.33 BrigInstBr Struct Reference

Collaboration diagram for BrigInstBr:

Public Attributes

- BrigInstBase base
- BrigWidth8_t width
- uint8_t reserved [3]

6.33.1 Detailed Description

Definition at line 907 of file Brig.h.

6.33.2 Member Data Documentation

6.33.2.1 base

```
BrigInstBase BrigInstBr::base
```

Definition at line 908 of file Brig.h.

6.33.2.2 reserved

```
uint8_t BrigInstBr::reserved[3]
```

Definition at line 910 of file Brig.h.

6.33.2.3 width

```
BrigWidth8_t BrigInstBr::width
```

Definition at line 909 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.34 BrigInstCmp Struct Reference

Collaboration diagram for BrigInstCmp:

Public Attributes

- BrigInstBase base
- BrigType16_t sourceType
- BrigAluModifier8_t modifier
- BrigCompareOperation8_t compare
- BrigPack8_t pack
- uint8_t reserved [3]

6.34.1 Detailed Description

Definition at line 913 of file Brig.h.

6.34.2 Member Data Documentation

6.34.2.1 base

```
BrigInstBase BrigInstCmp::base
```

Definition at line 914 of file Brig.h.

6.34.2.2 compare

```
BrigCompareOperation8_t BrigInstCmp::compare
```

Definition at line 917 of file Brig.h.

6.34.2.3 modifier

```
BrigAluModifier8_t BrigInstCmp::modifier
```

Definition at line 916 of file Brig.h.

6.34.2.4 pack

```
BrigPack8_t BrigInstCmp::pack
```

Definition at line 918 of file Brig.h.

6.34.2.5 reserved

```
uint8_t BrigInstCmp::reserved[3]
```

Definition at line 919 of file Brig.h.

6.34.2.6 sourceType

```
BrigType16_t BrigInstCmp::sourceType
```

Definition at line 915 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.35 BrigInstCvt Struct Reference

Collaboration diagram for BrigInstCvt:

Public Attributes

- BrigInstBase base
- BrigType16_t sourceType
- BrigAluModifier8_t modifier
- BrigRound8_t round

6.35.1 Detailed Description

Definition at line 922 of file Brig.h.

6.35.2 Member Data Documentation

6.35.2.1 base

```
BrigInstBase BrigInstCvt::base
```

Definition at line 923 of file Brig.h.

6.35.2.2 modifier

```
BrigAluModifier8_t BrigInstCvt::modifier
```

Definition at line 925 of file Brig.h.

6.35.2.3 round

```
BrigRound8_t BrigInstCvt::round
```

Definition at line 926 of file Brig.h.

6.35.2.4 sourceType

```
BrigType16_t BrigInstCvt::sourceType
```

Definition at line 924 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.36 BrigInstImage Struct Reference

Collaboration diagram for BrigInstImage:

Public Attributes

- BrigInstBase base
- BrigType16_t imageType
- BrigType16_t coordType
- BrigImageGeometry8_t geometry
- uint8_t equivClass
- uint16_t reserved

6.36.1 Detailed Description

Definition at line 929 of file Brig.h.

6.36.2 Member Data Documentation

6.36.2.1 base

```
BrigInstBase BrigInstImage::base
```

Definition at line 930 of file Brig.h.

6.36.2.2 coordType

```
BrigType16_t BrigInstImage::coordType
```

Definition at line 932 of file Brig.h.

6.36.2.3 equivClass

```
uint8_t BrigInstImage::equivClass
```

Definition at line 934 of file Brig.h.

6.36.2.4 geometry

```
BrigImageGeometry8_t BrigInstImage::geometry
```

Definition at line 933 of file Brig.h.

6.36.2.5 imageType

```
BrigType16_t BrigInstImage::imageType
```

Definition at line 931 of file Brig.h.

6.36.2.6 reserved

```
uint16_t BrigInstImage::reserved
```

Definition at line 935 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.37 BrigInstLane Struct Reference

Collaboration diagram for BrigInstLane:

Public Attributes

- BrigInstBase base
- BrigType16_t sourceType
- BrigWidth8_t width
- uint8_t reserved

6.37.1 Detailed Description

Definition at line 938 of file Brig.h.

6.37.2 Member Data Documentation

6.37.2.1 base

```
BrigInstBase BrigInstLane::base
```

Definition at line 939 of file Brig.h.

6.37.2.2 reserved

```
uint8_t BrigInstLane::reserved
```

Definition at line 942 of file Brig.h.

6.37.2.3 sourceType

```
BrigType16_t BrigInstLane::sourceType
```

Definition at line 940 of file Brig.h.

6.37.2.4 width

```
BrigWidth8_t BrigInstLane::width
```

Definition at line 941 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.38 BrigInstMem Struct Reference

Collaboration diagram for BrigInstMem:

Public Attributes

- BrigInstBase base
- BrigSegment8_t segment
- BrigAlignment8_t align
- uint8_t equivClass
- BrigWidth8_t width
- BrigMemoryModifier8_t modifier
- uint8_t reserved [3]

6.38.1 Detailed Description

Definition at line 945 of file Brig.h.

6.38.2 Member Data Documentation

6.38.2.1 align

```
BrigAlignment8_t BrigInstMem::align
```

Definition at line 948 of file Brig.h.

6.38.2.2 base

```
BrigInstBase BrigInstMem::base
```

Definition at line 946 of file Brig.h.

6.38.2.3 equivClass

```
uint8_t BrigInstMem::equivClass
```

Definition at line 949 of file Brig.h.

6.38.2.4 modifier

```
BrigMemoryModifier8_t BrigInstMem::modifier
```

Definition at line 951 of file Brig.h.

6.38.2.5 reserved

```
uint8_t BrigInstMem::reserved[3]
```

Definition at line 952 of file Brig.h.

6.38.2.6 segment

```
BrigSegment8_t BrigInstMem::segment
```

Definition at line 947 of file Brig.h.

6.38.2.7 width

```
BrigWidth8_t BrigInstMem::width
```

Definition at line 950 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.39 BrigInstMemFence Struct Reference

Collaboration diagram for BrigInstMemFence:

Public Attributes

- BrigInstBase base
- BrigMemoryOrder8_t memoryOrder
- BrigMemoryScope8_t globalSegmentMemoryScope
- BrigMemoryScope8_t groupSegmentMemoryScope
- BrigMemoryScope8_t imageSegmentMemoryScope

6.39.1 Detailed Description

Definition at line 955 of file Brig.h.

6.39.2 Member Data Documentation

6.39.2.1 base

```
BrigInstBase BrigInstMemFence::base
```

Definition at line 956 of file Brig.h.

6.39.2.2 globalSegmentMemoryScope

```
BrigMemoryScope8_t BrigInstMemFence::globalSegmentMemoryScope
```

Definition at line 958 of file Brig.h.

6.39.2.3 groupSegmentMemoryScope

```
BrigMemoryScope8_t BrigInstMemFence::groupSegmentMemoryScope
```

Definition at line 959 of file Brig.h.

6.39.2.4 imageSegmentMemoryScope

```
BrigMemoryScope8_t BrigInstMemFence::imageSegmentMemoryScope
```

Definition at line 960 of file Brig.h.

6.39.2.5 memoryOrder

```
BrigMemoryOrder8_t BrigInstMemFence::memoryOrder
```

Definition at line 957 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.40 BrigInstMod Struct Reference

Collaboration diagram for BrigInstMod:

Public Attributes

- BrigInstBase base
- BrigAluModifier8_t modifier
- BrigRound8_t round
- BrigPack8_t pack
- uint8_t reserved

6.40.1 Detailed Description

Definition at line 963 of file Brig.h.

6.40.2 Member Data Documentation

6.40.2.1 base

```
BrigInstBase BrigInstMod::base
```

Definition at line 964 of file Brig.h.

6.40.2.2 modifier

```
BrigAluModifier8_t BrigInstMod::modifier
```

Definition at line 965 of file Brig.h.

6.40.2.3 pack

```
BrigPack8_t BrigInstMod::pack
```

Definition at line 967 of file Brig.h.

6.40.2.4 reserved

```
uint8_t BrigInstMod::reserved
```

Definition at line 968 of file Brig.h.

6.40.2.5 round

```
BrigRound8_t BrigInstMod::round
```

Definition at line 966 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.41 BrigInstQueryImage Struct Reference

Collaboration diagram for BrigInstQueryImage:

Public Attributes

- BrigInstBase base
- BrigType16_t imageType
- BrigImageGeometry8_t geometry
- BrigImageQuery8_t query

6.41.1 Detailed Description

Definition at line 971 of file Brig.h.

6.41.2 Member Data Documentation

6.41.2.1 base

```
BrigInstBase BrigInstQueryImage::base
```

Definition at line 972 of file Brig.h.

6.41.2.2 geometry

```
BrigImageGeometry8_t BrigInstQueryImage::geometry
```

Definition at line 974 of file Brig.h.

6.41.2.3 imageType

```
BrigType16_t BrigInstQueryImage::imageType
```

Definition at line 973 of file Brig.h.

6.41.2.4 query

```
BrigImageQuery8_t BrigInstQueryImage::query
```

Definition at line 975 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.42 BrigInstQuerySampler Struct Reference

Collaboration diagram for BrigInstQuerySampler:

Public Attributes

- BrigInstBase base
- BrigSamplerQuery8_t query
- uint8_t reserved [3]

6.42.1 Detailed Description

Definition at line 978 of file Brig.h.

6.42.2 Member Data Documentation

6.42.2.1 base

```
BrigInstBase BrigInstQuerySampler::base
```

Definition at line 979 of file Brig.h.

6.42.2.2 query

```
BrigSamplerQuery8_t BrigInstQuerySampler::query
```

Definition at line 980 of file Brig.h.

6.42.2.3 reserved

```
uint8_t BrigInstQuerySampler::reserved[3]
```

Definition at line 981 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.43 BrigInstQueue Struct Reference

Collaboration diagram for BrigInstQueue:

Public Attributes

- BrigInstBase base
- BrigSegment8_t segment
- BrigMemoryOrder8_t memoryOrder
- uint16_t reserved

6.43.1 Detailed Description

Definition at line 984 of file Brig.h.

6.43.2 Member Data Documentation

6.43.2.1 base

```
BrigInstBase BrigInstQueue::base
```

Definition at line 985 of file Brig.h.

6.43.2.2 memoryOrder

```
BrigMemoryOrder8_t BrigInstQueue::memoryOrder
```

Definition at line 987 of file Brig.h.

6.43.2.3 reserved

```
uint16_t BrigInstQueue::reserved
```

Definition at line 988 of file Brig.h.

6.43.2.4 segment

```
BrigSegment8_t BrigInstQueue::segment
```

Definition at line 986 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.44 BrigInstSeg Struct Reference

Collaboration diagram for BrigInstSeg:

Public Attributes

- BrigInstBase base
- BrigSegment8_t segment
- uint8_t reserved [3]

6.44.1 Detailed Description

Definition at line 991 of file Brig.h.

6.44.2 Member Data Documentation

6.44.2.1 base

```
BrigInstBase BrigInstSeg::base
```

Definition at line 992 of file Brig.h.

6.44.2.2 reserved

```
uint8_t BrigInstSeg::reserved[3]
```

Definition at line 994 of file Brig.h.

6.44.2.3 segment

```
BrigSegment8_t BrigInstSeg::segment
```

Definition at line 993 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.45 BrigInstSegCvt Struct Reference

Collaboration diagram for BrigInstSegCvt:

Public Attributes

- BrigInstBase base
- BrigType16_t sourceType
- BrigSegment8_t segment
- BrigSegCvtModifier8_t modifier

6.45.1 Detailed Description

Definition at line 997 of file Brig.h.

6.45.2 Member Data Documentation

6.45.2.1 base

```
BrigInstBase BrigInstSegCvt::base
```

Definition at line 998 of file Brig.h.

6.45.2.2 modifier

```
BrigSegCvtModifier8_t BrigInstSegCvt::modifier
```

Definition at line 1001 of file Brig.h.

6.45.2.3 segment

```
BrigSegment8_t BrigInstSegCvt::segment
```

Definition at line 1000 of file Brig.h.

6.45.2.4 sourceType

```
BrigType16_t BrigInstSegCvt::sourceType
```

Definition at line 999 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.46 BrigInstSignal Struct Reference

Collaboration diagram for BrigInstSignal:

Public Attributes

- BrigInstBase base
- BrigType16_t signalType
- BrigMemoryOrder8_t memoryOrder
- BrigAtomicOperation8_t signalOperation

6.46.1 Detailed Description

Definition at line 1004 of file Brig.h.

6.46.2 Member Data Documentation

6.46.2.1 base

```
BrigInstBase BrigInstSignal::base
```

Definition at line 1005 of file Brig.h.

6.46.2.2 memoryOrder

```
BrigMemoryOrder8_t BrigInstSignal::memoryOrder
```

Definition at line 1007 of file Brig.h.

6.46.2.3 signalOperation

```
BrigAtomicOperation8_t BrigInstSignal::signalOperation
```

Definition at line 1008 of file Brig.h.

6.46.2.4 signalType

```
BrigType16_t BrigInstSignal::signalType
```

Definition at line 1006 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.47 BrigInstSourceType Struct Reference

Collaboration diagram for BrigInstSourceType:

Public Attributes

- BrigInstBase base
- BrigType16_t sourceType
- uint16_t reserved

6.47.1 Detailed Description

Definition at line 1011 of file Brig.h.

6.47.2 Member Data Documentation

6.47.2.1 base

```
BrigInstBase BrigInstSourceType::base
```

Definition at line 1012 of file Brig.h.

6.47.2.2 reserved

```
uint16_t BrigInstSourceType::reserved
```

Definition at line 1014 of file Brig.h.

6.47.2.3 sourceType

```
BrigType16_t BrigInstSourceType::sourceType
```

Definition at line 1013 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.48 BrigModuleHeader Struct Reference

Public Attributes

- char identification [8]
- BrigVersion32_t brigMajor
- BrigVersion32_t brigMinor
- uint64_t byteCount
- uint8_t hash [64]
- uint32_t reserved
- uint32_t sectionCount
- uint64_t sectionIndex

6.48.1 Detailed Description

Definition at line 1114 of file Brig.h.

6.48.2 Member Data Documentation

6.48.2.1 brigMajor

```
BrigVersion32_t BrigModuleHeader::brigMajor
```

Definition at line 1116 of file Brig.h.

6.48.2.2 brigMinor

```
BrigVersion32_t BrigModuleHeader::brigMinor
```

Definition at line 1117 of file Brig.h.

6.48.2.3 byteCount

```
uint64_t BrigModuleHeader::byteCount
```

Definition at line 1118 of file Brig.h.

6.48.2.4 hash

```
uint8_t BrigModuleHeader::hash[64]
```

Definition at line 1119 of file Brig.h.

6.48.2.5 identification

```
char BrigModuleHeader::identification[8]
```

Definition at line 1115 of file Brig.h.

6.48.2.6 reserved

```
uint32_t BrigModuleHeader::reserved
```

Definition at line 1120 of file Brig.h.

6.48.2.7 sectionCount

```
uint32_t BrigModuleHeader::sectionCount
```

Definition at line 1121 of file Brig.h.

6.48.2.8 sectionIndex

```
uint64_t BrigModuleHeader::sectionIndex
```

Definition at line 1122 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.49 BrigOperandAddress Struct Reference

Collaboration diagram for BrigOperandAddress:

Public Attributes

- BrigBase base
- BrigCodeOffset32_t symbol
- BrigOperandOffset32_t reg
- BrigUInt64 offset

6.49.1 Detailed Description

Definition at line 1017 of file Brig.h.

6.49.2 Member Data Documentation

6.49.2.1 base

```
BrigBase BrigOperandAddress::base
```

Definition at line 1018 of file Brig.h.

6.49.2.2 offset

```
BrigUInt64 BrigOperandAddress::offset
```

Definition at line 1021 of file Brig.h.

6.49.2.3 reg

```
BrigOperandOffset32_t BrigOperandAddress::reg
```

Definition at line 1020 of file Brig.h.

6.49.2.4 symbol

```
BrigCodeOffset32_t BrigOperandAddress::symbol
```

Definition at line 1019 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.50 BrigOperandAlign Struct Reference

Collaboration diagram for BrigOperandAlign:

Public Attributes

- BrigBase base
- BrigAlignment8_t align
- uint8_t reserved [3]

6.50.1 Detailed Description

Definition at line 1024 of file Brig.h.

6.50.2 Member Data Documentation

6.50.2.1 align

```
BrigAlignment8_t BrigOperandAlign::align
```

Definition at line 1026 of file Brig.h.

6.50.2.2 base

```
BrigBase BrigOperandAlign::base
```

Definition at line 1025 of file Brig.h.

6.50.2.3 reserved

```
uint8_t BrigOperandAlign::reserved[3]
```

Definition at line 1027 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.51 BrigOperandCodeList Struct Reference

Collaboration diagram for BrigOperandCodeList:

Public Attributes

- BrigBase base
- BrigDataOffsetCodeList32_t elements

6.51.1 Detailed Description

Definition at line 1030 of file Brig.h.

6.51.2 Member Data Documentation

6.51.2.1 base

```
BrigBase BrigOperandCodeList::base
```

Definition at line 1031 of file Brig.h.

6.51.2.2 elements

```
BrigDataOffsetCodeList32_t BrigOperandCodeList::elements
```

Definition at line 1032 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.52 BrigOperandCodeRef Struct Reference

Collaboration diagram for BrigOperandCodeRef:

Public Attributes

- BrigBase base
- BrigCodeOffset32_t ref

6.52.1 Detailed Description

Definition at line 1035 of file Brig.h.

6.52.2 Member Data Documentation

6.52.2.1 base

```
BrigBase BrigOperandCodeRef::base
```

Definition at line 1036 of file Brig.h.

6.52.2.2 ref

```
BrigCodeOffset32_t BrigOperandCodeRef::ref
```

Definition at line 1037 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.53 BrigOperandConstantBytes Struct Reference

Collaboration diagram for BrigOperandConstantBytes:

Public Attributes

- BrigBase base
- BrigType16_t type
- uint16_t reserved
- BrigDataOffsetString32_t bytes

6.53.1 Detailed Description

Definition at line 1040 of file Brig.h.

6.53.2 Member Data Documentation

6.53.2.1 base

```
BrigBase BrigOperandConstantBytes::base
```

Definition at line 1041 of file Brig.h.

6.53.2.2 bytes

```
BrigDataOffsetString32_t BrigOperandConstantBytes::bytes
```

Definition at line 1044 of file Brig.h.

6.53.2.3 reserved

```
uint16_t BrigOperandConstantBytes::reserved
```

Definition at line 1043 of file Brig.h.

6.53.2.4 type

```
BrigType16_t BrigOperandConstantBytes::type
```

Definition at line 1042 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.54 BrigOperandConstantImage Struct Reference

Collaboration diagram for BrigOperandConstantImage:

Public Attributes

- BrigBase base
- BrigType16_t type
- BrigImageGeometry8_t geometry
- BrigImageChannelOrder8_t channelOrder
- BrigImageChannelType8_t channelType
- uint8_t reserved [3]
- BrigUInt64 width
- BrigUInt64 height
- BrigUInt64 depth
- BrigUInt64 array

6.54.1 Detailed Description

Definition at line 1054 of file Brig.h.

6.54.2 Member Data Documentation

6.54.2.1 array

```
BrigUInt64 BrigOperandConstantImage::array
```

Definition at line 1064 of file Brig.h.

6.54.2.2 base

```
BrigBase BrigOperandConstantImage::base
```

Definition at line 1055 of file Brig.h.

6.54.2.3 channelOrder

```
BrigImageChannelOrder8_t BrigOperandConstantImage::channelOrder
```

Definition at line 1058 of file Brig.h.

6.54.2.4 channelType

```
BrigImageChannelType8_t BrigOperandConstantImage::channelType
```

Definition at line 1059 of file Brig.h.

6.54.2.5 depth

```
BrigUInt64 BrigOperandConstantImage::depth
```

Definition at line 1063 of file Brig.h.

6.54.2.6 geometry

```
BrigImageGeometry8_t BrigOperandConstantImage::geometry
```

Definition at line 1057 of file Brig.h.

6.54.2.7 height

```
BrigUInt64 BrigOperandConstantImage::height
```

Definition at line 1062 of file Brig.h.

6.54.2.8 reserved

```
uint8_t BrigOperandConstantImage::reserved[3]
```

Definition at line 1060 of file Brig.h.

6.54.2.9 type

```
BrigType16_t BrigOperandConstantImage::type
```

Definition at line 1056 of file Brig.h.

6.54.2.10 width

```
BrigUInt64 BrigOperandConstantImage::width
```

Definition at line 1061 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.55 BrigOperandConstantOperandList Struct Reference

Collaboration diagram for BrigOperandConstantOperandList:

Public Attributes

- BrigBase base
- BrigType16_t type
- uint16_t reserved
- BrigDataOffsetOperandList32_t elements

6.55.1 Detailed Description

Definition at line 1047 of file Brig.h.

6.55.2 Member Data Documentation

6.55.2.1 base

```
BrigBase BrigOperandConstantOperandList::base
```

Definition at line 1048 of file Brig.h.

6.55.2.2 elements

```
BrigDataOffsetOperandList32_t BrigOperandConstantOperandList::elements
```

Definition at line 1051 of file Brig.h.

6.55.2.3 reserved

```
uint16_t BrigOperandConstantOperandList::reserved
```

Definition at line 1050 of file Brig.h.

6.55.2.4 type

```
BrigType16_t BrigOperandConstantOperandList::type
```

Definition at line 1049 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.56 BrigOperandConstantSampler Struct Reference

Collaboration diagram for BrigOperandConstantSampler:

Public Attributes

- BrigBase base
- BrigType16_t type
- BrigSamplerCoordNormalization8_t coord
- BrigSamplerFilter8_t filter
- BrigSamplerAddressing8_t addressing
- uint8_t reserved [3]

6.56.1 Detailed Description

Definition at line 1078 of file Brig.h.

6.56.2 Member Data Documentation

6.56.2.1 addressing

```
BrigSamplerAddressing8_t BrigOperandConstantSampler::addressing
```

Definition at line 1083 of file Brig.h.

6.56.2.2 base

```
BrigBase BrigOperandConstantSampler::base
```

Definition at line 1079 of file Brig.h.

6.56.2.3 coord

```
BrigSamplerCoordNormalization8_t BrigOperandConstantSampler::coord
```

Definition at line 1081 of file Brig.h.

6.56.2.4 filter

```
BrigSamplerFilter8_t BrigOperandConstantSampler::filter
```

Definition at line 1082 of file Brig.h.

6.56.2.5 reserved

```
uint8_t BrigOperandConstantSampler::reserved[3]
```

Definition at line 1084 of file Brig.h.

6.56.2.6 type

```
BrigType16_t BrigOperandConstantSampler::type
```

Definition at line 1080 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.57 BrigOperandOperandList Struct Reference

Collaboration diagram for BrigOperandOperandList:

Public Attributes

- BrigBase base
- BrigDataOffsetOperandList32_t elements

6.57.1 Detailed Description

Definition at line 1067 of file Brig.h.

6.57.2 Member Data Documentation

6.57.2.1 base

```
BrigBase BrigOperandOperandList::base
```

Definition at line 1068 of file Brig.h.

6.57.2.2 elements

```
BrigDataOffsetOperandList32_t BrigOperandOperandList::elements
```

Definition at line 1069 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.58 BrigOperandRegister Struct Reference

Collaboration diagram for BrigOperandRegister:

Public Attributes

- BrigBase base
- BrigRegisterKind16_t regKind
- uint16_t regNum

6.58.1 Detailed Description

Definition at line 1072 of file Brig.h.

6.58.2 Member Data Documentation

6.58.2.1 base

```
BrigBase BrigOperandRegister::base
```

Definition at line 1073 of file Brig.h.

6.58.2.2 regKind

```
BrigRegisterKind16_t BrigOperandRegister::regKind
```

Definition at line 1074 of file Brig.h.

6.58.2.3 regNum

```
uint16_t BrigOperandRegister::regNum
```

Definition at line 1075 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.59 BrigOperandString Struct Reference

Collaboration diagram for BrigOperandString:

Public Attributes

- BrigBase base
- BrigDataOffsetString32_t string

6.59.1 Detailed Description

Definition at line 1087 of file Brig.h.

6.59.2 Member Data Documentation

6.59.2.1 base

```
BrigBase BrigOperandString::base
```

Definition at line 1088 of file Brig.h.

6.59.2.2 string

```
BrigDataOffsetString32_t BrigOperandString::string
```

Definition at line 1089 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.60 BrigOperandWavesize Struct Reference

Collaboration diagram for BrigOperandWavesize:

Public Attributes

- BrigBase base

6.60.1 Detailed Description

Definition at line 1092 of file Brig.h.

6.60.2 Member Data Documentation

6.60.2.1 base

```
BrigBase BrigOperandWavesize::base
```

Definition at line 1093 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.61 BrigSectionHeader Struct Reference

Public Attributes

- uint64_t byteCount
- uint32_t headerByteCount
- uint32_t nameLength
- uint8_t name [1]

6.61.1 Detailed Description

Definition at line 1107 of file Brig.h.

6.61.2 Member Data Documentation

6.61.2.1 byteCount

```
uint64_t BrigSectionHeader::byteCount
```

Definition at line 1108 of file Brig.h.

6.61.2.2 headerByteCount

```
uint32_t BrigSectionHeader::headerByteCount
```

Definition at line 1109 of file Brig.h.

6.61.2.3 name

```
uint8_t BrigSectionHeader::name[1]
```

Definition at line 1111 of file Brig.h.

6.61.2.4 nameLength

```
uint32_t BrigSectionHeader::nameLength
```

Definition at line 1110 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.62 BrigUInt64 Struct Reference

Public Attributes

- uint32_t lo
- uint32_t hi

6.62.1 Detailed Description

Definition at line 777 of file Brig.h.

6.62.2 Member Data Documentation

6.62.2.1 hi

```
uint32_t BrigUInt64::hi
```

Definition at line 779 of file Brig.h.

6.62.2.2 lo

```
uint32_t BrigUInt64::lo
```

Definition at line 778 of file Brig.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/Brig.h

6.63 CoreApiTable Struct Reference

Collaboration diagram for CoreApiTable:

Public Attributes

- ApiTableVersion version
- decltype(hsa_init) * hsa_init_fn
- decltype(hsa_shut_down) * hsa_shut_down_fn
- decltype(hsa_system_get_info) * hsa_system_get_info_fn
- decltype(hsa_system_extension_supported) * hsa_system_extension_supported_fn
- decltype(hsa_system_get_extension_table) * hsa_system_get_extension_table_fn
- decltype(hsa_iterate_agents) * hsa_iterate_agents_fn
- decltype(hsa_agent_get_info) * hsa_agent_get_info_fn
- decltype(hsa_queue_create) * hsa_queue_create_fn
- decltype(hsa_soft_queue_create) * hsa_soft_queue_create_fn
- decltype(hsa_queue_destroy) * hsa_queue_destroy_fn
- decltype(hsa_queue_inactivate) * hsa_queue_inactivate_fn
- decltype(hsa_queue_load_read_index_scacquire) * hsa_queue_load_read_index_scacquire_fn
- decltype(hsa_queue_load_read_index_relaxed) * hsa_queue_load_read_index_relaxed_fn
- decltype(hsa_queue_load_write_index_scacquire) * hsa_queue_load_write_index_scacquire_fn

- decltype(hsa_queue_load_write_index_relaxed) * hsa_queue_load_write_index_relaxed_fn
- decltype(hsa_queue_store_write_index_relaxed) * hsa_queue_store_write_index_relaxed_fn
- decltype(hsa_queue_store_write_index_screlease) * hsa_queue_store_write_index_screlease_fn
- decltype(hsa_queue_cas_write_index_scacq_screl) * hsa_queue_cas_write_index_scacq_screl_fn
- decltype(hsa_queue_cas_write_index_scacquire) * hsa_queue_cas_write_index_scacquire_fn
- decltype(hsa_queue_cas_write_index_relaxed) * hsa_queue_cas_write_index_relaxed_fn
- decltype(hsa_queue_cas_write_index_screlease) * hsa_queue_cas_write_index_screlease_fn
- decltype(hsa_queue_add_write_index_scacq_screl) * hsa_queue_add_write_index_scacq_screl_fn
- decltype(hsa_queue_add_write_index_scacquire) * hsa_queue_add_write_index_scacquire_fn
- decltype(hsa_queue_add_write_index_relaxed) * hsa_queue_add_write_index_relaxed_fn
- decltype(hsa_queue_add_write_index_screlease) * hsa_queue_add_write_index_screlease_fn
- decltype(hsa_queue_store_read_index_relaxed) * hsa_queue_store_read_index_relaxed_fn
- decltype(hsa_queue_store_read_index_screlease) * hsa_queue_store_read_index_screlease_fn
- decltype(hsa_agent_iterate_regions) * hsa_agent_iterate_regions_fn
- decltype(hsa_region_get_info) * hsa_region_get_info_fn
- decltype(hsa_agent_get_exception_policies) * hsa_agent_get_exception_policies_fn
- decltype(hsa_agent_extension_supported) * hsa_agent_extension_supported_fn
- decltype(hsa_memory_register) * hsa_memory_register_fn
- decltype(hsa_memory_deregister) * hsa_memory_deregister_fn
- decltype(hsa_memory_allocate) * hsa_memory_allocate_fn
- decltype(hsa_memory_free) * hsa_memory_free_fn
- decltype(hsa_memory_copy) * hsa_memory_copy_fn
- decltype(hsa_memory_assign_agent) * hsa_memory_assign_agent_fn
- decltype(hsa_signal_create) * hsa_signal_create_fn
- decltype(hsa_signal_destroy) * hsa_signal_destroy_fn
- decltype(hsa_signal_load_relaxed) * hsa_signal_load_relaxed_fn
- decltype(hsa_signal_load_scacquire) * hsa_signal_load_scacquire_fn
- decltype(hsa_signal_store_relaxed) * hsa_signal_store_relaxed_fn
- decltype(hsa_signal_store_screlease) * hsa_signal_store_screlease_fn
- decltype(hsa_signal_wait_relaxed) * hsa_signal_wait_relaxed_fn
- decltype(hsa_signal_wait_scacquire) * hsa_signal_wait_scacquire_fn
- decltype(hsa_signal_and_relaxed) * hsa_signal_and_relaxed_fn
- decltype(hsa_signal_and_scacquire) * hsa_signal_and_scacquire_fn
- decltype(hsa_signal_and_screlease) * hsa_signal_and_screlease_fn
- decltype(hsa_signal_and_scacq_screl) * hsa_signal_and_scacq_screl_fn
- decltype(hsa_signal_or_relaxed) * hsa_signal_or_relaxed_fn
- decltype(hsa_signal_or_scacquire) * hsa_signal_or_scacquire_fn
- decltype(hsa_signal_or_screlease) * hsa_signal_or_screlease_fn
- decltype(hsa_signal_or_scacq_screl) * hsa_signal_or_scacq_screl_fn
- decltype(hsa_signal_xor_relaxed) * hsa_signal_xor_relaxed_fn
- decltype(hsa_signal_xor_scacquire) * hsa_signal_xor_scacquire_fn
- decltype(hsa_signal_xor_screlease) * hsa_signal_xor_screlease_fn
- decltype(hsa_signal_xor_scacq_screl) * hsa_signal_xor_scacq_screl_fn
- decltype(hsa_signal_exchange_relaxed) * hsa_signal_exchange_relaxed_fn
- decltype(hsa_signal_exchange_scacquire) * hsa_signal_exchange_scacquire_fn
- decltype(hsa_signal_exchange_screlease) * hsa_signal_exchange_screlease_fn
- decltype(hsa_signal_exchange_scacq_screl) * hsa_signal_exchange_scacq_screl_fn
- decltype(hsa_signal_add_relaxed) * hsa_signal_add_relaxed_fn
- decltype(hsa_signal_add_scacquire) * hsa_signal_add_scacquire_fn
- decltype(hsa_signal_add_screlease) * hsa_signal_add_screlease_fn
- decltype(hsa_signal_add_scacq_screl) * hsa_signal_add_scacq_screl_fn
- decltype(hsa_signal_subtract_relaxed) * hsa_signal_subtract_relaxed_fn
- decltype(hsa_signal_subtract_scacquire) * hsa_signal_subtract_scacquire_fn
- decltype(hsa_signal_subtract_screlease) * hsa_signal_subtract_screlease_fn
- decltype(hsa_signal_subtract_scacq_screl) * hsa_signal_subtract_scacq_screl_fn

- decltype(hsa_signal_cas_relaxed) * hsa_signal_cas_relaxed_fn
- decltype(hsa_signal_cas_scacquire) * hsa_signal_cas_scacquire_fn
- decltype(hsa_signal_cas_screlease) * hsa_signal_cas_screlease_fn
- decltype(hsa_signal_cas_scacq_screl) * hsa_signal_cas_scacq_screl_fn
- decltype(hsa_isa_from_name) * hsa_isa_from_name_fn
- decltype(hsa_isa_get_info) * hsa_isa_get_info_fn
- decltype(hsa_isa_compatible) * hsa_isa_compatible_fn
- decltype(hsa_code_object_serialize) * hsa_code_object_serialize_fn
- decltype(hsa_code_object_deserialize) * hsa_code_object_deserialize_fn
- decltype(hsa_code_object_destroy) * hsa_code_object_destroy_fn
- decltype(hsa_code_object_get_info) * hsa_code_object_get_info_fn
- decltype(hsa_code_object_get_symbol) * hsa_code_object_get_symbol_fn
- decltype(hsa_code_symbol_get_info) * hsa_code_symbol_get_info_fn
- decltype(hsa_code_object_iterate_symbols) * hsa_code_object_iterate_symbols_fn
- decltype(hsa_executable_create) * hsa_executable_create_fn
- decltype(hsa_executable_destroy) * hsa_executable_destroy_fn
- decltype(hsa_executable_load_code_object) * hsa_executable_load_code_object_fn
- decltype(hsa_executable_freeze) * hsa_executable_freeze_fn
- decltype(hsa_executable_get_info) * hsa_executable_get_info_fn
- decltype(hsa_executable_global_variable_define) * hsa_executable_global_variable_define_fn
- decltype(hsa_executable_agent_global_variable_define) * hsa_executable_agent_global_variable_define_fn
- decltype(hsa_executable_readonly_variable_define) * hsa_executable_readonly_variable_define_fn
- decltype(hsa_executable_validate) * hsa_executable_validate_fn
- decltype(hsa_executable_get_symbol) * hsa_executable_get_symbol_fn
- decltype(hsa_executable_symbol_get_info) * hsa_executable_symbol_get_info_fn
- decltype(hsa_executable_iterate_symbols) * hsa_executable_iterate_symbols_fn
- decltype(hsa_status_string) * hsa_status_string_fn
- decltype(hsa_extension_get_name) * hsa_extension_get_name_fn
- decltype(hsa_system_major_extension_supported) * hsa_system_major_extension_supported_fn
- decltype(hsa_system_get_major_extension_table) * hsa_system_get_major_extension_table_fn
- decltype(hsa_agent_major_extension_supported) * hsa_agent_major_extension_supported_fn
- decltype(hsa_cache_get_info) * hsa_cache_get_info_fn
- decltype(hsa_agent_iterate_caches) * hsa_agent_iterate_caches_fn
- decltype(hsa_signal_silent_store_relaxed) * hsa_signal_silent_store_relaxed_fn
- decltype(hsa_signal_silent_store_screlease) * hsa_signal_silent_store_screlease_fn
- decltype(hsa_signal_group_create) * hsa_signal_group_create_fn
- decltype(hsa_signal_group_destroy) * hsa_signal_group_destroy_fn
- decltype(hsa_signal_group_wait_any_scacquire) * hsa_signal_group_wait_any_scacquire_fn
- decltype(hsa_signal_group_wait_any_relaxed) * hsa_signal_group_wait_any_relaxed_fn
- decltype(hsa_agent_iterate_isas) * hsa_agent_iterate_isas_fn
- decltype(hsa_isa_get_info_alt) * hsa_isa_get_info_alt_fn
- decltype(hsa_isa_get_exception_policies) * hsa_isa_get_exception_policies_fn
- decltype(hsa_isa_get_round_method) * hsa_isa_get_round_method_fn
- decltype(hsa_wavefront_get_info) * hsa_wavefront_get_info_fn
- decltype(hsa_isa_iterate_wavefronts) * hsa_isa_iterate_wavefronts_fn
- decltype(hsa_code_object_get_symbol_from_name) * hsa_code_object_get_symbol_from_name_fn
- decltype(hsa_code_object_reader_create_from_file) * hsa_code_object_reader_create_from_file_fn
- decltype(hsa_code_object_reader_create_from_memory) * hsa_code_object_reader_create_from_memory_fn
- decltype(hsa_code_object_reader_destroy) * hsa_code_object_reader_destroy_fn
- decltype(hsa_executable_create_alt) * hsa_executable_create_alt_fn
- decltype(hsa_executable_load_program_code_object) * hsa_executable_load_program_code_object_fn
- decltype(hsa_executable_load_agent_code_object) * hsa_executable_load_agent_code_object_fn
- decltype(hsa_executable_validate_alt) * hsa_executable_validate_alt_fn
- decltype(hsa_executable_get_symbol_by_name) * hsa_executable_get_symbol_by_name_fn
- decltype(hsa_executable_iterate_agent_symbols) * hsa_executable_iterate_agent_symbols_fn
- decltype(hsa_executable_iterate_program_symbols) * hsa_executable_iterate_program_symbols_fn

6.63.1 Detailed Description

Definition at line 193 of file hsa_api_trace.h.

6.63.2 Member Data Documentation

6.63.2.1 hsa_agent_extension_supported_fn

```
decltype(hsa_agent_extension_supported) * CoreApiTable::hsa_agent_extension_supported_fn
```

Definition at line 225 of file hsa_api_trace.h.

6.63.2.2 hsa_agent_get_exception_policies_fn

```
decltype(hsa_agent_get_exception_policies) * CoreApiTable::hsa_agent_get_exception_policies_fn
```

Definition at line 224 of file hsa_api_trace.h.

6.63.2.3 hsa_agent_get_info_fn

```
decltype(hsa_agent_get_info) * CoreApiTable::hsa_agent_get_info_fn
```

Definition at line 201 of file hsa_api_trace.h.

6.63.2.4 hsa_agent_iterate_caches_fn

```
decltype(hsa_agent_iterate_caches) * CoreApiTable::hsa_agent_iterate_caches_fn
```

Definition at line 326 of file hsa_api_trace.h.

6.63.2.5 hsa_agent_iterate_isas_fn

```
decltype(hsa_agent_iterate_isas) * CoreApiTable::hsa_agent_iterate_isas_fn
```

Definition at line 336 of file hsa_api_trace.h.

6.63.2.6 hsa_agent_iterate_regions_fn

```
decltype(hsa_agent_iterate_regions) * CoreApiTable::hsa_agent_iterate_regions_fn
```

Definition at line 222 of file hsa_api_trace.h.

6.63.2.7 hsa_agent_major_extension_supported_fn

```
decltype(hsa_agent_major_extension_supported) * CoreApiTable::hsa_agent_major_extension_↩
supported_fn
```

Definition at line 324 of file hsa_api_trace.h.

6.63.2.8 hsa_cache_get_info_fn

```
decltype(hsa_cache_get_info) * CoreApiTable::hsa_cache_get_info_fn
```

Definition at line 325 of file hsa_api_trace.h.

6.63.2.9 hsa_code_object_deserialize_fn

```
decltype(hsa_code_object_deserialize) * CoreApiTable::hsa_code_object_deserialize_fn
```

Definition at line 282 of file hsa_api_trace.h.

6.63.2.10 hsa_code_object_destroy_fn

```
decltype(hsa_code_object_destroy) * CoreApiTable::hsa_code_object_destroy_fn
```

Definition at line 284 of file hsa_api_trace.h.

6.63.2.11 hsa_code_object_get_info_fn

```
decltype(hsa_code_object_get_info) * CoreApiTable::hsa_code_object_get_info_fn
```

Definition at line 286 of file hsa_api_trace.h.

6.63.2.12 hsa_code_object_get_symbol_fn

```
decltype(hsa_code_object_get_symbol) * CoreApiTable::hsa_code_object_get_symbol_fn
```

Definition at line 288 of file hsa_api_trace.h.

6.63.2.13 hsa_code_object_get_symbol_from_name_fn

```
decltype(hsa_code_object_get_symbol_from_name) * CoreApiTable::hsa_code_object_get_symbol_↩
from_name_fn
```

Definition at line 347 of file hsa_api_trace.h.

6.63.2.14 hsa_code_object_iterate_symbols_fn

```
decltype(hsa_code_object_iterate_symbols) * CoreApiTable::hsa_code_object_iterate_symbols_fn
```

Definition at line 292 of file hsa_api_trace.h.

6.63.2.15 hsa_code_object_reader_create_from_file_fn

```
decltype(hsa_code_object_reader_create_from_file) * CoreApiTable::hsa_code_object_reader_↩
create_from_file_fn
```

Definition at line 352 of file hsa_api_trace.h.

6.63.2.16 hsa_code_object_reader_create_from_memory_fn

```
decltype(hsa_code_object_reader_create_from_memory) * CoreApiTable::hsa_code_object_reader_↩
create_from_memory_fn
```

Definition at line 354 of file hsa_api_trace.h.

6.63.2.17 hsa_code_object_reader_destroy_fn

```
decltype(hsa_code_object_reader_destroy) * CoreApiTable::hsa_code_object_reader_destroy_fn
```

Definition at line 355 of file hsa_api_trace.h.

6.63.2.18 hsa_code_object_serialize_fn

```
decltype(hsa_code_object_serialize) * CoreApiTable::hsa_code_object_serialize_fn
```

Definition at line 280 of file hsa_api_trace.h.

6.63.2.19 hsa_code_symbol_get_info_fn

```
decltype(hsa_code_symbol_get_info) * CoreApiTable::hsa_code_symbol_get_info_fn
```

Definition at line 290 of file hsa_api_trace.h.

6.63.2.20 hsa_executable_agent_global_variable_define_fn

```
decltype(hsa_executable_agent_global_variable_define) * CoreApiTable::hsa_executable_agent_↩
global_variable_define_fn
```

Definition at line 306 of file hsa_api_trace.h.

6.63.2.21 hsa_executable_create_alt_fn

```
decltype(hsa_executable_create_alt) * CoreApiTable::hsa_executable_create_alt_fn
```

Definition at line 356 of file hsa_api_trace.h.

6.63.2.22 hsa_executable_create_fn

```
decltype(hsa_executable_create) * CoreApiTable::hsa_executable_create_fn
```

Definition at line 297 of file hsa_api_trace.h.

6.63.2.23 hsa_executable_destroy_fn

```
decltype(hsa_executable_destroy) * CoreApiTable::hsa_executable_destroy_fn
```

Definition at line 298 of file hsa_api_trace.h.

6.63.2.24 hsa_executable_freeze_fn

```
decltype(hsa_executable_freeze) * CoreApiTable::hsa_executable_freeze_fn
```

Definition at line 301 of file hsa_api_trace.h.

6.63.2.25 hsa_executable_get_info_fn

```
decltype(hsa_executable_get_info) * CoreApiTable::hsa_executable_get_info_fn
```

Definition at line 302 of file hsa_api_trace.h.

6.63.2.26 hsa_executable_get_symbol_by_name_fn

```
decltype(hsa_executable_get_symbol_by_name) * CoreApiTable::hsa_executable_get_symbol_by_name↩
_fn
```

Definition at line 363 of file hsa_api_trace.h.

6.63.2.27 hsa_executable_get_symbol_fn

```
decltype(hsa_executable_get_symbol) * CoreApiTable::hsa_executable_get_symbol_fn
```

Definition at line 311 of file hsa_api_trace.h.

6.63.2.28 hsa_executable_global_variable_define_fn

```
decltype(hsa_executable_global_variable_define) * CoreApiTable::hsa_executable_global_variable↩
_define_fn
```

Definition at line 304 of file hsa_api_trace.h.

6.63.2.29 hsa_executable_iterate_agent_symbols_fn

```
decltype(hsa_executable_iterate_agent_symbols) * CoreApiTable::hsa_executable_iterate_agent_↩
symbols_fn
```

Definition at line 365 of file hsa_api_trace.h.

6.63.2.30 hsa_executable_iterate_program_symbols_fn

```
decltype(hsa_executable_iterate_program_symbols) * CoreApiTable::hsa_executable_iterate_↩
program_symbols_fn
```

Definition at line 367 of file hsa_api_trace.h.

6.63.2.31 hsa_executable_iterate_symbols_fn

```
decltype(hsa_executable_iterate_symbols) * CoreApiTable::hsa_executable_iterate_symbols_fn
```

Definition at line 314 of file hsa_api_trace.h.

6.63.2.32 hsa_executable_load_agent_code_object_fn

```
decltype(hsa_executable_load_agent_code_object) * CoreApiTable::hsa_executable_load_agent_↩
code_object_fn
```

Definition at line 360 of file hsa_api_trace.h.

6.63.2.33 hsa_executable_load_code_object_fn

```
decltype(hsa_executable_load_code_object) * CoreApiTable::hsa_executable_load_code_object_fn
```

Definition at line 300 of file hsa_api_trace.h.

6.63.2.34 hsa_executable_load_program_code_object_fn

```
decltype(hsa_executable_load_program_code_object) * CoreApiTable::hsa_executable_load_program↩
_code_object_fn
```

Definition at line 358 of file hsa_api_trace.h.

6.63.2.35 hsa_executable_readonly_variable_define_fn

```
decltype(hsa_executable_readonly_variable_define) * CoreApiTable::hsa_executable_readonly_↩
variable_define_fn
```

Definition at line 308 of file hsa_api_trace.h.

6.63.2.36 hsa_executable_symbol_get_info_fn

```
decltype(hsa_executable_symbol_get_info) * CoreApiTable::hsa_executable_symbol_get_info_fn
```

Definition at line 312 of file hsa_api_trace.h.

6.63.2.37 hsa_executable_validate_alt_fn

```
decltype(hsa_executable_validate_alt) * CoreApiTable::hsa_executable_validate_alt_fn
```

Definition at line 361 of file hsa_api_trace.h.

6.63.2.38 hsa_executable_validate_fn

```
decltype(hsa_executable_validate) * CoreApiTable::hsa_executable_validate_fn
```

Definition at line 309 of file hsa_api_trace.h.

6.63.2.39 hsa_extension_get_name_fn

```
decltype(hsa_extension_get_name) * CoreApiTable::hsa_extension_get_name_fn
```

Definition at line 321 of file hsa_api_trace.h.

6.63.2.40 hsa_init_fn

```
decltype(hsa_init) * CoreApiTable::hsa_init_fn
```

Definition at line 195 of file hsa_api_trace.h.

6.63.2.41 hsa_isa_compatible_fn

```
decltype(hsa_isa_compatible) * CoreApiTable::hsa_isa_compatible_fn
```

Definition at line 275 of file hsa_api_trace.h.

6.63.2.42 hsa_isa_from_name_fn

```
decltype(hsa_isa_from_name) * CoreApiTable::hsa_isa_from_name_fn
```

Definition at line 271 of file hsa_api_trace.h.

6.63.2.43 hsa_isa_get_exception_policies_fn

```
decltype(hsa_isa_get_exception_policies) * CoreApiTable::hsa_isa_get_exception_policies_fn
```

Definition at line 338 of file hsa_api_trace.h.

6.63.2.44 hsa_isa_get_info_alt_fn

```
decltype(hsa_isa_get_info_alt) * CoreApiTable::hsa_isa_get_info_alt_fn
```

Definition at line 337 of file hsa_api_trace.h.

6.63.2.45 hsa_isa_get_info_fn

```
decltype(hsa_isa_get_info) * CoreApiTable::hsa_isa_get_info_fn
```

Definition at line 273 of file hsa_api_trace.h.

6.63.2.46 hsa_isa_get_round_method_fn

```
decltype(hsa_isa_get_round_method) * CoreApiTable::hsa_isa_get_round_method_fn
```

Definition at line 339 of file hsa_api_trace.h.

6.63.2.47 hsa_isa_iterate_wavefronts_fn

```
decltype(hsa_isa_iterate_wavefronts) * CoreApiTable::hsa_isa_iterate_wavefronts_fn
```

Definition at line 341 of file hsa_api_trace.h.

6.63.2.48 hsa_iterate_agents_fn

```
decltype(hsa_iterate_agents) * CoreApiTable::hsa_iterate_agents_fn
```

Definition at line 200 of file hsa_api_trace.h.

6.63.2.49 hsa_memory_allocate_fn

```
decltype(hsa_memory_allocate) * CoreApiTable::hsa_memory_allocate_fn
```

Definition at line 228 of file hsa_api_trace.h.

6.63.2.50 hsa_memory_assign_agent_fn

```
decltype(hsa_memory_assign_agent) * CoreApiTable::hsa_memory_assign_agent_fn
```

Definition at line 231 of file hsa_api_trace.h.

6.63.2.51 hsa_memory_copy_fn

```
decltype(hsa_memory_copy) * CoreApiTable::hsa_memory_copy_fn
```

Definition at line 230 of file hsa_api_trace.h.

6.63.2.52 hsa_memory_deregister_fn

```
decltype(hsa_memory_deregister) * CoreApiTable::hsa_memory_deregister_fn
```

Definition at line 227 of file hsa_api_trace.h.

6.63.2.53 hsa_memory_free_fn

```
decltype(hsa_memory_free) * CoreApiTable::hsa_memory_free_fn
```

Definition at line 229 of file hsa_api_trace.h.

6.63.2.54 hsa_memory_register_fn

```
decltype(hsa_memory_register) * CoreApiTable::hsa_memory_register_fn
```

Definition at line 226 of file hsa_api_trace.h.

6.63.2.55 hsa_queue_add_write_index_relaxed_fn

```
decltype(hsa_queue_add_write_index_relaxed) * CoreApiTable::hsa_queue_add_write_index_relaxed↩
_fn
```

Definition at line 218 of file hsa_api_trace.h.

6.63.2.56 hsa_queue_add_write_index_scacq_screl_fn

```
decltype(hsa_queue_add_write_index_scacq_screl) * CoreApiTable::hsa_queue_add_write_index_↩
scacq_screl_fn
```

Definition at line 216 of file hsa_api_trace.h.

6.63.2.57 hsa_queue_add_write_index_scacquire_fn

```
decltype(hsa_queue_add_write_index_scacquire) * CoreApiTable::hsa_queue_add_write_index_↩
scacquire_fn
```

Definition at line 217 of file hsa_api_trace.h.

6.63.2.58 hsa_queue_add_write_index_screlease_fn

```
decltype(hsa_queue_add_write_index_screlease) * CoreApiTable::hsa_queue_add_write_index_↩
screlease_fn
```

Definition at line 219 of file hsa_api_trace.h.

6.63.2.59 hsa_queue_cas_write_index_relaxed_fn

```
decltype(hsa_queue_cas_write_index_relaxed) * CoreApiTable::hsa_queue_cas_write_index_relaxed↩
_fn
```

Definition at line 214 of file hsa_api_trace.h.

6.63.2.60 hsa_queue_cas_write_index_scacq_screl_fn

```
decltype(hsa_queue_cas_write_index_scacq_screl) * CoreApiTable::hsa_queue_cas_write_index_↩
scacq_screl_fn
```

Definition at line 212 of file hsa_api_trace.h.

6.63.2.61 hsa_queue_cas_write_index_scacquire_fn

```
decltype(hsa_queue_cas_write_index_scacquire) * CoreApiTable::hsa_queue_cas_write_index_↩
scacquire_fn
```

Definition at line 213 of file hsa_api_trace.h.

6.63.2.62 hsa_queue_cas_write_index_screlease_fn

```
decltype(hsa_queue_cas_write_index_screlease) * CoreApiTable::hsa_queue_cas_write_index_↩
screlease_fn
```

Definition at line 215 of file hsa_api_trace.h.

6.63.2.63 hsa_queue_create_fn

```
decltype(hsa_queue_create) * CoreApiTable::hsa_queue_create_fn
```

Definition at line 202 of file hsa_api_trace.h.

6.63.2.64 hsa_queue_destroy_fn

```
decltype(hsa_queue_destroy) * CoreApiTable::hsa_queue_destroy_fn
```

Definition at line 204 of file hsa_api_trace.h.

6.63.2.65 hsa_queue_inactivate_fn

```
decltype(hsa_queue_inactivate) * CoreApiTable::hsa_queue_inactivate_fn
```

Definition at line 205 of file hsa_api_trace.h.

6.63.2.66 hsa_queue_load_read_index_relaxed_fn

```
decltype(hsa_queue_load_read_index_relaxed) * CoreApiTable::hsa_queue_load_read_index_relaxed↩
_fn
```

Definition at line 207 of file hsa_api_trace.h.

6.63.2.67 hsa_queue_load_read_index_scacquire_fn

```
decltype(hsa_queue_load_read_index_scacquire) * CoreApiTable::hsa_queue_load_read_index_↩
scacquire_fn
```

Definition at line 206 of file hsa_api_trace.h.

6.63.2.68 hsa_queue_load_write_index_relaxed_fn

```
decltype(hsa_queue_load_write_index_relaxed) * CoreApiTable::hsa_queue_load_write_index_↩
relaxed_fn
```

Definition at line 209 of file hsa_api_trace.h.

6.63.2.69 hsa_queue_load_write_index_scacquire_fn

```
decltype(hsa_queue_load_write_index_scacquire) * CoreApiTable::hsa_queue_load_write_index_↩
scacquire_fn
```

Definition at line 208 of file hsa_api_trace.h.

6.63.2.70 hsa_queue_store_read_index_relaxed_fn

```
decltype(hsa_queue_store_read_index_relaxed) * CoreApiTable::hsa_queue_store_read_index_↩
relaxed_fn
```

Definition at line 220 of file hsa_api_trace.h.

6.63.2.71 hsa_queue_store_read_index_screlease_fn

```
decltype(hsa_queue_store_read_index_screlease) * CoreApiTable::hsa_queue_store_read_index_↩
screlease_fn
```

Definition at line 221 of file hsa_api_trace.h.

6.63.2.72 hsa_queue_store_write_index_relaxed_fn

```
decltype(hsa_queue_store_write_index_relaxed) * CoreApiTable::hsa_queue_store_write_index_↩
relaxed_fn
```

Definition at line 210 of file hsa_api_trace.h.

6.63.2.73 hsa_queue_store_write_index_screlease_fn

```
decltype(hsa_queue_store_write_index_screlease) * CoreApiTable::hsa_queue_store_write_index_↩
screlease_fn
```

Definition at line 211 of file hsa_api_trace.h.

6.63.2.74 hsa_region_get_info_fn

```
decltype(hsa_region_get_info) * CoreApiTable::hsa_region_get_info_fn
```

Definition at line 223 of file hsa_api_trace.h.

6.63.2.75 hsa_shut_down_fn

```
decltype(hsa_shut_down) * CoreApiTable::hsa_shut_down_fn
```

Definition at line 196 of file hsa_api_trace.h.

6.63.2.76 hsa_signal_add_relaxed_fn

```
decltype(hsa_signal_add_relaxed) * CoreApiTable::hsa_signal_add_relaxed_fn
```

Definition at line 256 of file hsa_api_trace.h.

6.63.2.77 hsa_signal_add_scacq_screl_fn

```
decltype(hsa_signal_add_scacq_screl) * CoreApiTable::hsa_signal_add_scacq_screl_fn
```

Definition at line 259 of file hsa_api_trace.h.

6.63.2.78 hsa_signal_add_scacquire_fn

```
decltype(hsa_signal_add_scacquire) * CoreApiTable::hsa_signal_add_scacquire_fn
```

Definition at line 257 of file hsa_api_trace.h.

6.63.2.79 hsa_signal_add_screlease_fn

```
decltype(hsa_signal_add_screlease) * CoreApiTable::hsa_signal_add_screlease_fn
```

Definition at line 258 of file hsa_api_trace.h.

6.63.2.80 hsa_signal_and_relaxed_fn

```
decltype(hsa_signal_and_relaxed) * CoreApiTable::hsa_signal_and_relaxed_fn
```

Definition at line 240 of file hsa_api_trace.h.

6.63.2.81 hsa_signal_and_scacq_screl_fn

```
decltype(hsa_signal_and_scacq_screl) * CoreApiTable::hsa_signal_and_scacq_screl_fn
```

Definition at line 243 of file hsa_api_trace.h.

6.63.2.82 hsa_signal_and_scacquire_fn

```
decltype(hsa_signal_and_scacquire) * CoreApiTable::hsa_signal_and_scacquire_fn
```

Definition at line 241 of file hsa_api_trace.h.

6.63.2.83 hsa_signal_and_screlease_fn

```
decltype(hsa_signal_and_screlease) * CoreApiTable::hsa_signal_and_screlease_fn
```

Definition at line 242 of file hsa_api_trace.h.

6.63.2.84 hsa_signal_cas_relaxed_fn

```
decltype(hsa_signal_cas_relaxed) * CoreApiTable::hsa_signal_cas_relaxed_fn
```

Definition at line 264 of file hsa_api_trace.h.

6.63.2.85 hsa_signal_cas_scacq_screl_fn

```
decltype(hsa_signal_cas_scacq_screl) * CoreApiTable::hsa_signal_cas_scacq_screl_fn
```

Definition at line 267 of file hsa_api_trace.h.

6.63.2.86 hsa_signal_cas_scacquire_fn

```
decltype(hsa_signal_cas_scacquire) * CoreApiTable::hsa_signal_cas_scacquire_fn
```

Definition at line 265 of file hsa_api_trace.h.

6.63.2.87 hsa_signal_cas_screlease_fn

```
decltype(hsa_signal_cas_screlease) * CoreApiTable::hsa_signal_cas_screlease_fn
```

Definition at line 266 of file hsa_api_trace.h.

6.63.2.88 hsa_signal_create_fn

```
decltype(hsa_signal_create) * CoreApiTable::hsa_signal_create_fn
```

Definition at line 232 of file hsa_api_trace.h.

6.63.2.89 hsa_signal_destroy_fn

```
decltype(hsa_signal_destroy) * CoreApiTable::hsa_signal_destroy_fn
```

Definition at line 233 of file hsa_api_trace.h.

6.63.2.90 hsa_signal_exchange_relaxed_fn

```
decltype(hsa_signal_exchange_relaxed) * CoreApiTable::hsa_signal_exchange_relaxed_fn
```

Definition at line 252 of file hsa_api_trace.h.

6.63.2.91 hsa_signal_exchange_scacq_screl_fn

```
decltype(hsa_signal_exchange_scacq_screl) * CoreApiTable::hsa_signal_exchange_scacq_screl_fn
```

Definition at line 255 of file hsa_api_trace.h.

6.63.2.92 hsa_signal_exchange_scacquire_fn

```
decltype(hsa_signal_exchange_scacquire) * CoreApiTable::hsa_signal_exchange_scacquire_fn
```

Definition at line 253 of file hsa_api_trace.h.

6.63.2.93 hsa_signal_exchange_screlease_fn

```
decltype(hsa_signal_exchange_screlease) * CoreApiTable::hsa_signal_exchange_screlease_fn
```

Definition at line 254 of file hsa_api_trace.h.

6.63.2.94 hsa_signal_group_create_fn

```
decltype(hsa_signal_group_create) * CoreApiTable::hsa_signal_group_create_fn
```

Definition at line 329 of file hsa_api_trace.h.

6.63.2.95 hsa_signal_group_destroy_fn

```
decltype(hsa_signal_group_destroy) * CoreApiTable::hsa_signal_group_destroy_fn
```

Definition at line 330 of file hsa_api_trace.h.

6.63.2.96 hsa_signal_group_wait_any_relaxed_fn

```
decltype(hsa_signal_group_wait_any_relaxed) * CoreApiTable::hsa_signal_group_wait_any_relaxed↩
_fn
```

Definition at line 332 of file hsa_api_trace.h.

6.63.2.97 hsa_signal_group_wait_any_scacquire_fn

```
decltype(hsa_signal_group_wait_any_scacquire) * CoreApiTable::hsa_signal_group_wait_any_↩
scacquire_fn
```

Definition at line 331 of file hsa_api_trace.h.

6.63.2.98 hsa_signal_load_relaxed_fn

```
decltype(hsa_signal_load_relaxed) * CoreApiTable::hsa_signal_load_relaxed_fn
```

Definition at line 234 of file hsa_api_trace.h.

6.63.2.99 hsa_signal_load_scacquire_fn

```
decltype(hsa_signal_load_scacquire) * CoreApiTable::hsa_signal_load_scacquire_fn
```

Definition at line 235 of file hsa_api_trace.h.

6.63.2.100 hsa_signal_or_relaxed_fn

```
decltype(hsa_signal_or_relaxed) * CoreApiTable::hsa_signal_or_relaxed_fn
```

Definition at line 244 of file hsa_api_trace.h.

6.63.2.101 hsa_signal_or_scacq_screl_fn

```
decltype(hsa_signal_or_scacq_screl) * CoreApiTable::hsa_signal_or_scacq_screl_fn
```

Definition at line 247 of file hsa_api_trace.h.

6.63.2.102 hsa_signal_or_scacquire_fn

```
decltype(hsa_signal_or_scacquire) * CoreApiTable::hsa_signal_or_scacquire_fn
```

Definition at line 245 of file hsa_api_trace.h.

6.63.2.103 hsa_signal_or_screlease_fn

```
decltype(hsa_signal_or_screlease) * CoreApiTable::hsa_signal_or_screlease_fn
```

Definition at line 246 of file hsa_api_trace.h.

6.63.2.104 hsa_signal_silent_store_relaxed_fn

```
decltype(hsa_signal_silent_store_relaxed) * CoreApiTable::hsa_signal_silent_store_relaxed_fn
```

Definition at line 327 of file hsa_api_trace.h.

6.63.2.105 hsa_signal_silent_store_screlease_fn

```
decltype(hsa_signal_silent_store_screlease) * CoreApiTable::hsa_signal_silent_store_screlease↩
_fn
```

Definition at line 328 of file hsa_api_trace.h.

6.63.2.106 hsa_signal_store_relaxed_fn

```
decltype(hsa_signal_store_relaxed) * CoreApiTable::hsa_signal_store_relaxed_fn
```

Definition at line 236 of file hsa_api_trace.h.

6.63.2.107 hsa_signal_store_screlease_fn

```
decltype(hsa_signal_store_screlease) * CoreApiTable::hsa_signal_store_screlease_fn
```

Definition at line 237 of file hsa_api_trace.h.

6.63.2.108 hsa_signal_subtract_relaxed_fn

```
decltype(hsa_signal_subtract_relaxed) * CoreApiTable::hsa_signal_subtract_relaxed_fn
```

Definition at line 260 of file hsa_api_trace.h.

6.63.2.109 hsa_signal_subtract_scacq_screl_fn

```
decltype(hsa_signal_subtract_scacq_screl) * CoreApiTable::hsa_signal_subtract_scacq_screl_fn
```

Definition at line 263 of file hsa_api_trace.h.

6.63.2.110 hsa_signal_subtract_scacquire_fn

```
decltype(hsa_signal_subtract_scacquire) * CoreApiTable::hsa_signal_subtract_scacquire_fn
```

Definition at line 261 of file hsa_api_trace.h.

6.63.2.111 hsa_signal_subtract_screlease_fn

```
decltype(hsa_signal_subtract_screlease) * CoreApiTable::hsa_signal_subtract_screlease_fn
```

Definition at line 262 of file hsa_api_trace.h.

6.63.2.112 hsa_signal_wait_relaxed_fn

```
decltype(hsa_signal_wait_relaxed) * CoreApiTable::hsa_signal_wait_relaxed_fn
```

Definition at line 238 of file hsa_api_trace.h.

6.63.2.113 hsa_signal_wait_scacquire_fn

```
decltype(hsa_signal_wait_scacquire) * CoreApiTable::hsa_signal_wait_scacquire_fn
```

Definition at line 239 of file hsa_api_trace.h.

6.63.2.114 hsa_signal_xor_relaxed_fn

```
decltype(hsa_signal_xor_relaxed) * CoreApiTable::hsa_signal_xor_relaxed_fn
```

Definition at line 248 of file hsa_api_trace.h.

6.63.2.115 hsa_signal_xor_scacq_screl_fn

```
decltype(hsa_signal_xor_scacq_screl) * CoreApiTable::hsa_signal_xor_scacq_screl_fn
```

Definition at line 251 of file hsa_api_trace.h.

6.63.2.116 hsa_signal_xor_scacquire_fn

```
decltype(hsa_signal_xor_scacquire) * CoreApiTable::hsa_signal_xor_scacquire_fn
```

Definition at line 249 of file hsa_api_trace.h.

6.63.2.117 hsa_signal_xor_screlease_fn

```
decltype(hsa_signal_xor_screlease) * CoreApiTable::hsa_signal_xor_screlease_fn
```

Definition at line 250 of file hsa_api_trace.h.

6.63.2.118 hsa_soft_queue_create_fn

```
decltype(hsa_soft_queue_create) * CoreApiTable::hsa_soft_queue_create_fn
```

Definition at line 203 of file hsa_api_trace.h.

6.63.2.119 hsa_status_string_fn

```
decltype(hsa_status_string) * CoreApiTable::hsa_status_string_fn
```

Definition at line 318 of file hsa_api_trace.h.

6.63.2.120 hsa_system_extension_supported_fn

```
decltype(hsa_system_extension_supported) * CoreApiTable::hsa_system_extension_supported_fn
```

Definition at line 198 of file hsa_api_trace.h.

6.63.2.121 hsa_system_get_extension_table_fn

```
decltype(hsa_system_get_extension_table) * CoreApiTable::hsa_system_get_extension_table_fn
```

Definition at line 199 of file hsa_api_trace.h.

6.63.2.122 hsa_system_get_info_fn

```
decltype(hsa_system_get_info) * CoreApiTable::hsa_system_get_info_fn
```

Definition at line 197 of file hsa_api_trace.h.

6.63.2.123 hsa_system_get_major_extension_table_fn

```
decltype(hsa_system_get_major_extension_table) * CoreApiTable::hsa_system_get_major_extension↩
_table_fn
```

Definition at line 323 of file hsa_api_trace.h.

6.63.2.124 hsa_system_major_extension_supported_fn

```
decltype(hsa_system_major_extension_supported) * CoreApiTable::hsa_system_major_extension_↩
supported_fn
```

Definition at line 322 of file hsa_api_trace.h.

6.63.2.125 hsa_wavefront_get_info_fn

```
decltype(hsa_wavefront_get_info) * CoreApiTable::hsa_wavefront_get_info_fn
```

Definition at line 340 of file hsa_api_trace.h.

6.63.2.126 version

```
ApiTableVersion CoreApiTable::version
```

Definition at line 194 of file hsa_api_trace.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_api_trace.h

6.64 FinalizerExtTable Struct Reference

Collaboration diagram for FinalizerExtTable:

Public Attributes

- ApiTableVersion version
- decltype(hsa_ext_program_create) * hsa_ext_program_create_fn
- decltype(hsa_ext_program_destroy) * hsa_ext_program_destroy_fn
- decltype(hsa_ext_program_add_module) * hsa_ext_program_add_module_fn
- decltype(hsa_ext_program_iterate_modules) * hsa_ext_program_iterate_modules_fn
- decltype(hsa_ext_program_get_info) * hsa_ext_program_get_info_fn
- decltype(hsa_ext_program_finalize) * hsa_ext_program_finalize_fn

6.64.1 Detailed Description

Definition at line 111 of file hsa_api_trace.h.

6.64.2 Member Data Documentation

6.64.2.1 hsa_ext_program_add_module_fn

```
decltype(hsa_ext_program_add_module) * FinalizerExtTable::hsa_ext_program_add_module_fn
```

Definition at line 115 of file hsa_api_trace.h.

6.64.2.2 hsa_ext_program_create_fn

```
decltype(hsa_ext_program_create) * FinalizerExtTable::hsa_ext_program_create_fn
```

Definition at line 113 of file hsa_api_trace.h.

6.64.2.3 hsa_ext_program_destroy_fn

```
decltype(hsa_ext_program_destroy) * FinalizerExtTable::hsa_ext_program_destroy_fn
```

Definition at line 114 of file hsa_api_trace.h.

6.64.2.4 hsa_ext_program_finalize_fn

```
decltype(hsa_ext_program_finalize) * FinalizerExtTable::hsa_ext_program_finalize_fn
```

Definition at line 118 of file hsa_api_trace.h.

6.64.2.5 hsa_ext_program_get_info_fn

```
decltype(hsa_ext_program_get_info) * FinalizerExtTable::hsa_ext_program_get_info_fn
```

Definition at line 117 of file hsa_api_trace.h.

6.64.2.6 hsa_ext_program_iterate_modules_fn

```
decltype(hsa_ext_program_iterate_modules) * FinalizerExtTable::hsa_ext_program_iterate_↩
modules_fn
```

Definition at line 116 of file hsa_api_trace.h.

6.64.2.7 version

```
ApiTableVersion FinalizerExtTable::version
```

Definition at line 112 of file hsa_api_trace.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_api_trace.h

6.65 hsa_agent_dispatch_packet_s Struct Reference

Agent dispatch packet.

```
#include <hsa.h>
```

Collaboration diagram for hsa_agent_dispatch_packet_s:

Public Attributes

- uint16_t header
- uint16_t type
- uint32_t reserved0
- uint32_t reserved1
- void * return_address
- uint64_t arg [4]
- uint64_t reserved2
- hsa_signal_t completion_signal

6.65.1 Detailed Description

Agent dispatch packet.

Definition at line 3029 of file hsa.h.

6.65.2 Member Data Documentation

6.65.2.1 arg

```
uint64_t hsa_agent_dispatch_packet_s::arg[4]
```

Function arguments.

Definition at line 3065 of file hsa.h.

6.65.2.2 completion_signal

```
hsa_signal_t hsa_agent_dispatch_packet_s::completion_signal
```

Signal used to indicate completion of the job. The application can use the special signal handle 0 to indicate that no signal is used.

Definition at line 3076 of file hsa.h.

6.65.2.3 header

```
uint16_t hsa_agent_dispatch_packet_s::header
```

Packet header. Used to configure multiple packet parameters such as the packet type. The parameters are described by hsa_packet_header_t.

Definition at line 3034 of file hsa.h.

6.65.2.4 reserved0

```
uint32_t hsa_agent_dispatch_packet_s::reserved0
```

Reserved. Must be 0.

Definition at line 3044 of file hsa.h.

6.65.2.5 reserved1

```
uint32_t hsa_agent_dispatch_packet_s::reserved1
```

Definition at line 3058 of file hsa.h.

6.65.2.6 reserved2

```
uint64_t hsa_agent_dispatch_packet_s::reserved2
```

Reserved. Must be 0.

Definition at line 3070 of file hsa.h.

6.65.2.7 return_address

```
void* hsa_agent_dispatch_packet_s::return_address
```

Definition at line 3059 of file hsa.h.

6.65.2.8 type

```
uint16_t hsa_agent_dispatch_packet_s::type
```

Application-defined function to be performed by the destination agent.

Definition at line 3039 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.66 hsa_agent_s Struct Reference

Struct containing an opaque handle to an agent, a device that participates in the HSA memory model. An agent can submit AQL packets for execution, and may also accept AQL packets for execution (agent dispatch packets or kernel dispatch packets launching HSAIL-derived binaries).

`#include <hsa.h>`

Public Attributes

- uint64_t handle

6.66.1 Detailed Description

Struct containing an opaque handle to an agent, a device that participates in the HSA memory model. An agent can submit AQL packets for execution, and may also accept AQL packets for execution (agent dispatch packets or kernel dispatch packets launching HSAIL-derived binaries).

Definition at line 741 of file hsa.h.

6.66.2 Member Data Documentation

6.66.2.1 handle

`uint64_t hsa_agent_s::handle`

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 746 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.67 hsa_amd_barrier_value_packet_s Struct Reference

AMD barrier value packet. Halts packet processing and waits for (signal_value & mask) cond value to be satisfied, where signal_value is the value of the signal signal.

`#include <hsa_ext_amd.h>`

Collaboration diagram for hsa_amd_barrier_value_packet_s:

Public Attributes

- hsa_amd_vendor_packet_header_t header
- uint32_t reserved0
- hsa_signal_t signal
- hsa_signal_value_t value
- hsa_signal_value_t mask
- hsa_signal_condition32_t cond
- uint32_t reserved1
- uint64_t reserved2
- uint64_t reserved3
- hsa_signal_t completion_signal

6.67.1 Detailed Description

AMD barrier value packet. Halts packet processing and waits for (signal_value & mask) cond value to be satisfied, where signal_value is the value of the signal signal.

Definition at line 110 of file hsa_ext_amd.h.

6.67.2 Member Data Documentation

6.67.2.1 completion_signal

```
hsa_signal_t hsa_amd_barrier_value_packet_s::completion_signal
```

Signal used to indicate completion of the job. The application can use the special signal handle 0 to indicate that no signal is used.

Definition at line 162 of file hsa_ext_amd.h.

6.67.2.2 cond

```
hsa_signal_condition32_t hsa_amd_barrier_value_packet_s::cond
```

Comparison operation. See hsa_signal_condition_t.

Definition at line 141 of file hsa_ext_amd.h.

6.67.2.3 header

`hsa_amd_vendor_packet_header_t hsa_amd_barrier_value_packet_s::header`

AMD vendor specific packet header.

Definition at line 114 of file hsa_ext_amd.h.

6.67.2.4 mask

`hsa_signal_value_t hsa_amd_barrier_value_packet_s::mask`

Bit mask to be combined by bitwise AND with signal's value.

Definition at line 136 of file hsa_ext_amd.h.

6.67.2.5 reserved0

`uint32_t hsa_amd_barrier_value_packet_s::reserved0`

Reserved. Must be 0.

Definition at line 119 of file hsa_ext_amd.h.

6.67.2.6 reserved1

`uint32_t hsa_amd_barrier_value_packet_s::reserved1`

Reserved. Must be 0.

Definition at line 146 of file hsa_ext_amd.h.

6.67.2.7 reserved2

`uint64_t hsa_amd_barrier_value_packet_s::reserved2`

Reserved. Must be 0.

Definition at line 151 of file hsa_ext_amd.h.

6.67.2.8 reserved3

```
uint64_t hsa_amd_barrier_value_packet_s::reserved3
```

Reserved. Must be 0.

Definition at line 156 of file hsa_ext_amd.h.

6.67.2.9 signal

```
hsa_signal_t hsa_amd_barrier_value_packet_s::signal
```

Dependent signal object. A signal with a handle value of 0 is allowed and is interpreted by the packet processor a satisfied dependency.

Definition at line 126 of file hsa_ext_amd.h.

6.67.2.10 value

```
hsa_signal_value_t hsa_amd_barrier_value_packet_s::value
```

Value to compare against.

Definition at line 131 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.68 hsa_amd_event_s Struct Reference

AMD GPU event data passed to event handler.

```
#include <hsa_ext_amd.h>
```

Collaboration diagram for hsa_amd_event_s:

Public Attributes

- hsa_amd_event_type_t event_type
- union {
  hsa_amd_gpu_memory_fault_info_t memory_fault
  };

6.68.1 Detailed Description

AMD GPU event data passed to event handler.

Definition at line 2099 of file hsa_ext_amd.h.

6.68.2 Member Data Documentation

6.68.2.1 event_type

```
hsa_amd_event_type_t hsa_amd_event_s::event_type
```

Definition at line 2103 of file hsa_ext_amd.h.

6.68.2.2 memory_fault

```
hsa_amd_gpu_memory_fault_info_t hsa_amd_event_s::memory_fault
```

Definition at line 2108 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.69 hsa_amd_gpu_memory_fault_info_s Struct Reference

AMD GPU memory fault event data.

```
#include <hsa_ext_amd.h>
```

Collaboration diagram for hsa_amd_gpu_memory_fault_info_s:

Public Attributes

- hsa_agent_t agent
- uint64_t virtual_address
- uint32_t fault_reason_mask

6.69.1 Detailed Description

AMD GPU memory fault event data.

Definition at line 2080 of file hsa_ext_amd.h.

6.69.2 Member Data Documentation

6.69.2.1 agent

```
hsa_agent_t hsa_amd_gpu_memory_fault_info_s::agent
```

Definition at line 2084 of file hsa_ext_amd.h.

6.69.2.2 fault_reason_mask

```
uint32_t hsa_amd_gpu_memory_fault_info_s::fault_reason_mask
```

Definition at line 2093 of file hsa_ext_amd.h.

6.69.2.3 virtual_address

```
uint64_t hsa_amd_gpu_memory_fault_info_s::virtual_address
```

Definition at line 2088 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.70 hsa_amd_hdp_flush_s Struct Reference

Public Attributes

- uint32_t * HDP_MEM_FLUSH_CNTL
- uint32_t * HDP_REG_FLUSH_CNTL

6.70.1 Detailed Description

Definition at line 342 of file hsa_ext_amd.h.

6.70.2 Member Data Documentation

6.70.2.1 HDP_MEM_FLUSH_CNTL

```
uint32_t* hsa_amd_hdp_flush_s::HDP_MEM_FLUSH_CNTL
```

Definition at line 343 of file hsa_ext_amd.h.

6.70.2.2 HDP_REG_FLUSH_CNTL

```
uint32_t* hsa_amd_hdp_flush_s::HDP_REG_FLUSH_CNTL
```

Definition at line 344 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.71 hsa_amd_image_descriptor_s Struct Reference

Encodes an opaque vendor specific image format. The length of data depends on the underlying format. This structure must not be copied as its true length can not be determined.

```
#include <hsa_ext_amd.h>
```

Public Attributes

- uint32_t version
- uint32_t deviceID
- uint32_t data [1]

6.71.1 Detailed Description

Encodes an opaque vendor specific image format. The length of data depends on the underlying format. This structure must not be copied as its true length can not be determined.

Definition at line 1709 of file hsa_ext_amd.h.

6.71.2 Member Data Documentation

6.71.2.1 data

```
uint32_t hsa_amd_image_descriptor_s::data[1]
```

Definition at line 1723 of file hsa_ext_amd.h.

6.71.2.2 deviceID

```
uint32_t hsa_amd_image_descriptor_s::deviceID
```

Definition at line 1718 of file hsa_ext_amd.h.

6.71.2.3 version

```
uint32_t hsa_amd_image_descriptor_s::version
```

Definition at line 1713 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.72 hsa_amd_ipc_memory_s Struct Reference

256-bit process independent identifier for a ROCr shared memory allocation.

```
#include <hsa_ext_amd.h>
```

Public Attributes

- uint32_t handle [8]

6.72.1 Detailed Description

256-bit process independent identifier for a ROCr shared memory allocation.

Definition at line 1901 of file hsa_ext_amd.h.

6.72.2 Member Data Documentation

6.72.2.1 handle

```
uint32_t hsa_amd_ipc_memory_s::handle[8]
```

Definition at line 1902 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.73 hsa_amd_memory_pool_link_info_s Struct Reference

Link properties when accessing the memory pool from the specified agent.

`#include <hsa_ext_amd.h>`

Public Attributes

- uint32_t min_latency
- uint32_t max_latency
- uint32_t min_bandwidth
- uint32_t max_bandwidth
- bool atomic_support_32bit
- bool atomic_support_64bit
- bool coherent_support
- hsa_amd_link_info_type_t link_type
- uint32_t numa_distance

6.73.1 Detailed Description

Link properties when accessing the memory pool from the specified agent.

Definition at line 1297 of file hsa_ext_amd.h.

6.73.2 Member Data Documentation

6.73.2.1 atomic_support_32bit

`bool hsa_amd_memory_pool_link_info_s::atomic_support_32bit`

Support for 32-bit atomic transactions.

Definition at line 1321 of file hsa_ext_amd.h.

6.73.2.2 atomic_support_64bit

`bool hsa_amd_memory_pool_link_info_s::atomic_support_64bit`

Support for 64-bit atomic transactions.

Definition at line 1326 of file hsa_ext_amd.h.

6.73.2.3 coherent_support

```
bool hsa_amd_memory_pool_link_info_s::coherent_support
```

Support for cache coherent transactions.

Definition at line 1331 of file hsa_ext_amd.h.

6.73.2.4 link_type

```
hsa_amd_link_info_type_t hsa_amd_memory_pool_link_info_s::link_type
```

The type of bus/link.

Definition at line 1336 of file hsa_ext_amd.h.

6.73.2.5 max_bandwidth

```
uint32_t hsa_amd_memory_pool_link_info_s::max_bandwidth
```

Maximum link interface bandwidth in MB/s.

Definition at line 1316 of file hsa_ext_amd.h.

6.73.2.6 max_latency

```
uint32_t hsa_amd_memory_pool_link_info_s::max_latency
```

Maximum transfer latency (rounded to ns).

Definition at line 1306 of file hsa_ext_amd.h.

6.73.2.7 min_bandwidth

```
uint32_t hsa_amd_memory_pool_link_info_s::min_bandwidth
```

Minimum link interface bandwidth in MB/s.

Definition at line 1311 of file hsa_ext_amd.h.

6.73.2.8 min_latency

```
uint32_t hsa_amd_memory_pool_link_info_s::min_latency
```

Minimum transfer latency (rounded to ns).

Definition at line 1301 of file hsa_ext_amd.h.

6.73.2.9 numa_distance

```
uint32_t hsa_amd_memory_pool_link_info_s::numa_distance
```

NUMA distance of memory pool relative to querying agent

Definition at line 1341 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.74 hsa_amd_memory_pool_s Struct Reference

A memory pool encapsulates physical storage on an agent along with a memory access model.

```
#include <hsa_ext_amd.h>
```

Public Attributes

- uint64_t handle

6.74.1 Detailed Description

A memory pool encapsulates physical storage on an agent along with a memory access model.

A memory pool encapsulates a physical partition of an agent's memory system along with a memory access model. Division of a single memory system into separate pools allows querying each partition's access path properties (see hsa_amd_agent_memory_pool_get_info). Allocations from a pool are preferentially bound to that pool's physical partition. Binding to the pool's preferential physical partition may not be possible or persistent depending on the system's memory policy and/or state which is beyond the scope of HSA APIs.

For example, a multi-node NUMA memory system may be represented by multiple pool's with each pool providing size and access path information for the partition it represents. Allocations from a pool are preferentially bound to the pool's partition (which in this example is a NUMA node) while following its memory access model. The actual placement may vary or migrate due to the system's NUMA policy and state, which is beyond the scope of HSA APIs.

Definition at line 923 of file hsa_ext_amd.h.

6.74.2 Member Data Documentation

6.74.2.1 handle

```
uint64_t hsa_amd_memory_pool_s::handle
```

Opaque handle.

Definition at line 927 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.75 hsa_amd_packet_header_s Struct Reference

AMD vendor specific AQL packet header.

```
#include <hsa_ext_amd.h>
```

Public Attributes

- uint16_t header
- hsa_amd_packet_type8_t AmdFormat
- uint8_t reserved

6.75.1 Detailed Description

AMD vendor specific AQL packet header.

Definition at line 87 of file hsa_ext_amd.h.

6.75.2 Member Data Documentation

6.75.2.1 AmdFormat

```
hsa_amd_packet_type8_t hsa_amd_packet_header_s::AmdFormat
```

Format of the vendor specific packet.

Definition at line 97 of file hsa_ext_amd.h.

6.75.2.2 header

```
uint16_t hsa_amd_packet_header_s::header
```

Packet header. Used to configure multiple packet parameters such as the packet type. The parameters are described by hsa_packet_header_t.

Definition at line 92 of file hsa_ext_amd.h.

6.75.2.3 reserved

```
uint8_t hsa_amd_packet_header_s::reserved
```

Reserved. Must be 0.

Definition at line 102 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.76 hsa_amd_pointer_info_s Struct Reference

Describes a memory allocation known to ROCr. Within a ROCr major version this structure can only grow.

```
#include <hsa_ext_amd.h>
```

Collaboration diagram for hsa_amd_pointer_info_s:

Public Attributes

- uint32_t size
- hsa_amd_pointer_type_t type
- void * agentBaseAddress
- void * hostBaseAddress
- size_t sizeInBytes
- void * userData
- hsa_agent_t agentOwner
- uint32_t global_flags

6.76.1 Detailed Description

Describes a memory allocation known to ROCr. Within a ROCr major version this structure can only grow.

Definition at line 1794 of file hsa_ext_amd.h.

6.76.2 Member Data Documentation

6.76.2.1 agentBaseAddress

```
void* hsa_amd_pointer_info_s::agentBaseAddress
```

Definition at line 1810 of file hsa_ext_amd.h.

6.76.2.2 agentOwner

```
hsa_agent_t hsa_amd_pointer_info_s::agentOwner
```

Definition at line 1828 of file hsa_ext_amd.h.

6.76.2.3 global_flags

```
uint32_t hsa_amd_pointer_info_s::global_flags
```

Definition at line 1834 of file hsa_ext_amd.h.

6.76.2.4 hostBaseAddress

```
void* hsa_amd_pointer_info_s::hostBaseAddress
```

Definition at line 1814 of file hsa_ext_amd.h.

6.76.2.5 size

```
uint32_t hsa_amd_pointer_info_s::size
```

Definition at line 1802 of file hsa_ext_amd.h.

6.76.2.6 sizeInBytes

```
size_t hsa_amd_pointer_info_s::sizeInBytes
```

Definition at line 1818 of file hsa_ext_amd.h.

6.76.2.7 type

```
hsa_amd_pointer_type_t hsa_amd_pointer_info_s::type
```

Definition at line 1806 of file hsa_ext_amd.h.

6.76.2.8 userData

```
void* hsa_amd_pointer_info_s::userData
```

Definition at line 1822 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.77 hsa_amd_profiling_async_copy_time_s Struct Reference

Structure containing profiling async copy time information.

```
#include <hsa_ext_amd.h>
```

Public Attributes

- uint64_t start
- uint64_t end

6.77.1 Detailed Description

Structure containing profiling async copy time information.

Times are reported as ticks in the domain of the HSA system clock. The HSA system clock tick and frequency is obtained via hsa_system_get_info.

Definition at line 450 of file hsa_ext_amd.h.

6.77.2 Member Data Documentation

6.77.2.1 end

```
uint64_t hsa_amd_profiling_async_copy_time_s::end
```

Async copy completion time.

Definition at line 458 of file hsa_ext_amd.h.

6.77.2.2 start

```
uint64_t hsa_amd_profiling_async_copy_time_s::start
```

Async copy processing start time.

Definition at line 454 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.78 hsa_amd_profiling_dispatch_time_s Struct Reference

Structure containing profiling dispatch time information.

```
#include <hsa_ext_amd.h>
```

Public Attributes

- uint64_t start
- uint64_t end

6.78.1 Detailed Description

Structure containing profiling dispatch time information.

Times are reported as ticks in the domain of the HSA system clock. The HSA system clock tick and frequency is obtained via hsa_system_get_info.

Definition at line 433 of file hsa_ext_amd.h.

6.78.2 Member Data Documentation

6.78.2.1 end

`uint64_t hsa_amd_profiling_dispatch_time_s::end`

Dispatch packet completion time.

Definition at line 441 of file hsa_ext_amd.h.

6.78.2.2 start

`uint64_t hsa_amd_profiling_dispatch_time_s::start`

Dispatch packet processing start time.

Definition at line 437 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.79 hsa_amd_svm_attribute_pair_s Struct Reference

Public Attributes

- uint64_t attribute
- uint64_t value

6.79.1 Detailed Description

Definition at line 2323 of file hsa_ext_amd.h.

6.79.2 Member Data Documentation

6.79.2.1 attribute

`uint64_t hsa_amd_svm_attribute_pair_s::attribute`

Definition at line 2325 of file hsa_ext_amd.h.

6.79.2.2 value

```
uint64_t hsa_amd_svm_attribute_pair_s::value
```

Definition at line 2328 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.80 hsa_barrier_and_packet_s Struct Reference

Barrier-AND packet.

```
#include <hsa.h>
```

Collaboration diagram for hsa_barrier_and_packet_s:

Public Attributes

- uint16_t header
- uint16_t reserved0
- uint32_t reserved1
- hsa_signal_t dep_signal [5]
- uint64_t reserved2
- hsa_signal_t completion_signal

6.80.1 Detailed Description

Barrier-AND packet.

Definition at line 3083 of file hsa.h.

6.80.2 Member Data Documentation

6.80.2.1 completion_signal

```
hsa_signal_t hsa_barrier_and_packet_s::completion_signal
```

Signal used to indicate completion of the job. The application can use the special signal handle 0 to indicate that no signal is used.

Definition at line 3116 of file hsa.h.

6.80.2.2 dep_signal

```
hsa_signal_t hsa_barrier_and_packet_s::dep_signal[5]
```

Array of dependent signal objects. Signals with a handle value of 0 are allowed and are interpreted by the packet processor as satisfied dependencies.

Definition at line 3105 of file hsa.h.

6.80.2.3 header

```
uint16_t hsa_barrier_and_packet_s::header
```

Packet header. Used to configure multiple packet parameters such as the packet type. The parameters are described by hsa_packet_header_t.

Definition at line 3088 of file hsa.h.

6.80.2.4 reserved0

```
uint16_t hsa_barrier_and_packet_s::reserved0
```

Reserved. Must be 0.

Definition at line 3093 of file hsa.h.

6.80.2.5 reserved1

```
uint32_t hsa_barrier_and_packet_s::reserved1
```

Reserved. Must be 0.

Definition at line 3098 of file hsa.h.

6.80.2.6 reserved2

```
uint64_t hsa_barrier_and_packet_s::reserved2
```

Reserved. Must be 0.

Definition at line 3110 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.81 hsa_barrier_or_packet_s Struct Reference

Barrier-OR packet.

`#include <hsa.h>`

Collaboration diagram for hsa_barrier_or_packet_s:

Public Attributes

- uint16_t header
- uint16_t reserved0
- uint32_t reserved1
- hsa_signal_t dep_signal [5]
- uint64_t reserved2
- hsa_signal_t completion_signal

6.81.1 Detailed Description

Barrier-OR packet.

Definition at line 3123 of file hsa.h.

6.81.2 Member Data Documentation

6.81.2.1 completion_signal

`hsa_signal_t hsa_barrier_or_packet_s::completion_signal`

Signal used to indicate completion of the job. The application can use the special signal handle 0 to indicate that no signal is used.

Definition at line 3156 of file hsa.h.

6.81.2.2 dep_signal

`hsa_signal_t hsa_barrier_or_packet_s::dep_signal[5]`

Array of dependent signal objects. Signals with a handle value of 0 are allowed and are interpreted by the packet processor as dependencies not satisfied.

Definition at line 3145 of file hsa.h.

6.81.2.3 header

```
uint16_t hsa_barrier_or_packet_s::header
```

Packet header. Used to configure multiple packet parameters such as the packet type. The parameters are described by hsa_packet_header_t.

Definition at line 3128 of file hsa.h.

6.81.2.4 reserved0

```
uint16_t hsa_barrier_or_packet_s::reserved0
```

Reserved. Must be 0.

Definition at line 3133 of file hsa.h.

6.81.2.5 reserved1

```
uint32_t hsa_barrier_or_packet_s::reserved1
```

Reserved. Must be 0.

Definition at line 3138 of file hsa.h.

6.81.2.6 reserved2

```
uint64_t hsa_barrier_or_packet_s::reserved2
```

Reserved. Must be 0.

Definition at line 3150 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.82 hsa_cache_s Struct Reference

Cache handle.

```
#include <hsa.h>
```

Public Attributes

- uint64_t handle

6.82.1 Detailed Description

Cache handle.

Definition at line 1149 of file hsa.h.

6.82.2 Member Data Documentation

6.82.2.1 handle

```
uint64_t hsa_cache_s::handle
```

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 1154 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.83 hsa_callback_data_s Struct Reference

Application data handle that is passed to the serialization and deserialization functions.

```
#include <hsa.h>
```

Public Attributes

- uint64_t handle

6.83.1 Detailed Description

Application data handle that is passed to the serialization and deserialization functions.

Deprecated

Definition at line 5143 of file hsa.h.

6.83.2 Member Data Documentation

6.83.2.1 handle

`uint64_t hsa_callback_data_s::handle`

Opaque handle.

Definition at line 5147 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.84 hsa_code_object_reader_s Struct Reference

Code object reader handle. A code object reader is used to load a code object from file (when created using hsa_code_object_reader_create_from_file), or from memory (if created using hsa_code_object_reader_create_from_memory).

`#include <hsa.h>`

Public Attributes

- uint64_t handle

6.84.1 Detailed Description

Code object reader handle. A code object reader is used to load a code object from file (when created using hsa_code_object_reader_create_from_file), or from memory (if created using hsa_code_object_reader_create_from_memory).

Definition at line 4019 of file hsa.h.

6.84.2 Member Data Documentation

6.84.2.1 handle

`uint64_t hsa_code_object_reader_s::handle`

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 4024 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.85 hsa_code_object_s Struct Reference

Struct containing an opaque handle to a code object, which contains ISA for finalized kernels and indirect functions together with information about the global or readonly segment variables they reference.

`#include <hsa.h>`

Public Attributes

- uint64_t handle

6.85.1 Detailed Description

Struct containing an opaque handle to a code object, which contains ISA for finalized kernels and indirect functions together with information about the global or readonly segment variables they reference.

Deprecated

Definition at line 5129 of file hsa.h.

6.85.2 Member Data Documentation

6.85.2.1 handle

`uint64_t hsa_code_object_s::handle`

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 5134 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.86 hsa_code_symbol_s Struct Reference

Code object symbol handle.

`#include <hsa.h>`

Public Attributes

- uint64_t handle

6.86.1 Detailed Description

Code object symbol handle.

Deprecated

The lifetime of a code object symbol matches that of the code object associated with it. An operation on a symbol whose associated code object has been destroyed results in undefined behavior.

Definition at line 5412 of file hsa.h.

6.86.2 Member Data Documentation

6.86.2.1 handle

```
uint64_t hsa_code_symbol_s::handle
```

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 5417 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.87 hsa_dim3_s Struct Reference

Three-dimensional coordinate.

```
#include <hsa.h>
```

Public Attributes

- uint32_t x
- uint32_t y
- uint32_t z

6.87.1 Detailed Description

Three-dimensional coordinate.

Definition at line 298 of file hsa.h.

6.87.2 Member Data Documentation

6.87.2.1 x

```
uint32_t hsa_dim3_s::x
```

X dimension.

Definition at line 302 of file hsa.h.

6.87.2.2 y

```
uint32_t hsa_dim3_s::y
```

Y dimension.

Definition at line 307 of file hsa.h.

6.87.2.3 z

```
uint32_t hsa_dim3_s::z
```

Z dimension.

Definition at line 312 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.88 hsa_executable_s Struct Reference

Struct containing an opaque handle to an executable, which contains ISA for finalized kernels and indirect functions together with the allocated global or readonly segment variables they reference.

```
#include <hsa.h>
```

Public Attributes

- uint64_t handle

6.88.1 Detailed Description

Struct containing an opaque handle to an executable, which contains ISA for finalized kernels and indirect functions together with the allocated global or readonly segment variables they reference.

Definition at line 4110 of file hsa.h.

6.88.2 Member Data Documentation

6.88.2.1 handle

```
uint64_t hsa_executable_s::handle
```

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 4115 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.89 hsa_executable_symbol_s Struct Reference

Executable symbol handle.

```
#include <hsa.h>
```

Public Attributes

- uint64_t handle

6.89.1 Detailed Description

Executable symbol handle.

The lifetime of an executable object symbol matches that of the executable associated with it. An operation on a symbol whose associated executable has been destroyed results in undefined behavior.

Definition at line 4652 of file hsa.h.

6.89.2 Member Data Documentation

6.89.2.1 handle

```
uint64_t hsa_executable_symbol_s::handle
```

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 4657 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.90 hsa_ext_amd_aql_pm4_packet_t Struct Reference

Collaboration diagram for hsa_ext_amd_aql_pm4_packet_t:

Public Attributes

- uint16_t header
- uint16_t pm4_command [27]
- hsa_signal_t completion_signal

6.90.1 Detailed Description

Definition at line 202 of file hsa_ven_amd_aqlprofile.h.

6.90.2 Member Data Documentation

6.90.2.1 completion_signal

```
hsa_signal_t hsa_ext_amd_aql_pm4_packet_t::completion_signal
```

Definition at line 205 of file hsa_ven_amd_aqlprofile.h.

6.90.2.2 header

```
uint16_t hsa_ext_amd_aql_pm4_packet_t::header
```

Definition at line 203 of file hsa_ven_amd_aqlprofile.h.

6.90.2.3 pm4_command

```
uint16_t hsa_ext_amd_aql_pm4_packet_t::pm4_command[27]
```

Definition at line 204 of file hsa_ven_amd_aqlprofile.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_aqlprofile.h

6.91 hsa_ext_control_directives_s Struct Reference

Control directives specify low-level information about the finalization process.

```
#include <hsa_ext_finalize.h>
```

Collaboration diagram for hsa_ext_control_directives_s:

Public Attributes

- uint64_t control_directives_mask
- uint16_t break_exceptions_mask
- uint16_t detect_exceptions_mask
- uint32_t max_dynamic_group_size
- uint64_t max_flat_grid_size
- uint32_t max_flat_workgroup_size
- uint32_t reserved1
- uint64_t required_grid_size [3]
- hsa_dim3_t required_workgroup_size
- uint8_t required_dim
- uint8_t reserved2 [75]

6.91.1 Detailed Description

Control directives specify low-level information about the finalization process.

Definition at line 322 of file hsa_ext_finalize.h.

6.91.2 Member Data Documentation

6.91.2.1 break_exceptions_mask

```
uint16_t hsa_ext_control_directives_s::break_exceptions_mask
```

Bitset of HSAIL exceptions that must have the BREAK policy enabled. The bit assigned to an HSAIL exception is determined by the corresponding value in BrigExceptionsMask. If the kernel contains a enablebreakexceptions control directive, the finalizer uses the union of the two masks.

Definition at line 340 of file hsa_ext_finalize.h.

6.91.2.2 control_directives_mask

```
uint64_t hsa_ext_control_directives_s::control_directives_mask
```

Bitset indicating which control directives are enabled. The bit assigned to a control directive is determined by the corresponding value in BrigControlDirective.

If a control directive is disabled, its corresponding field value (if any) must be 0. Control directives that are only present or absent (such as partial workgroups) have no corresponding field as the presence of the bit in this mask is sufficient.

Definition at line 333 of file hsa_ext_finalize.h.

6.91.2.3 detect_exceptions_mask

```
uint16_t hsa_ext_control_directives_s::detect_exceptions_mask
```

Bitset of HSAIL exceptions that must have the DETECT policy enabled. The bit assigned to an HSAIL exception is determined by the corresponding value in BrigExceptionsMask. If the kernel contains a enabledetectexceptions control directive, the finalizer uses the union of the two masks.

Definition at line 347 of file hsa_ext_finalize.h.

6.91.2.4 max_dynamic_group_size

```
uint32_t hsa_ext_control_directives_s::max_dynamic_group_size
```

Maximum size (in bytes) of dynamic group memory that will be allocated by the application for any dispatch of the kernel. If the kernel contains a maxdynamicsize control directive, the two values should match.

Definition at line 353 of file hsa_ext_finalize.h.

6.91.2.5 max_flat_grid_size

```
uint64_t hsa_ext_control_directives_s::max_flat_grid_size
```

Maximum number of grid work-items that will be used by the application to launch the kernel. If the kernel contains a maxflatgridsize control directive, the value of *max_flat_grid_size* must not be greater than the value of the directive, and takes precedence.

The value specified for maximum absolute grid size must be greater than or equal to the product of the values specified by *required_grid_size*.

If the bit at position BRIG_CONTROL_MAXFLATGRIDSIZE is set in *control_directives_mask*, this field must be greater than 0.

Definition at line 366 of file hsa_ext_finalize.h.

6.91.2.6 max_flat_workgroup_size

```
uint32_t hsa_ext_control_directives_s::max_flat_workgroup_size
```

Maximum number of work-group work-items that will be used by the application to launch the kernel. If the kernel contains a maxflatworkgroupsize control directive, the value of *max_flat_workgroup_size* must not be greater than the value of the directive, and takes precedence.

The value specified for maximum absolute grid size must be greater than or equal to the product of the values specified by *required_workgroup_size*.

If the bit at position BRIG_CONTROL_MAXFLATWORKGROUPSIZE is set in *control_directives_mask*, this field must be greater than 0.

Definition at line 380 of file hsa_ext_finalize.h.

6.91.2.7 required_dim

```
uint8_t hsa_ext_control_directives_s::required_dim
```

Number of dimensions that will be used by the application to launch the kernel. If the kernel contains a requireddim control directive, the two values should match.

The specified dimensions must be consistent with *required_grid_size* and *required_workgroup_size*. This invariant must hold only if all the corresponding control directives are enabled.

If the bit at position BRIG_CONTROL_REQUIREDDIM is set in *control_directives_mask*, this field must be 1, 2, or 3.

Definition at line 425 of file hsa_ext_finalize.h.

6.91.2.8 required_grid_size

```
uint64_t hsa_ext_control_directives_s::required_grid_size[3]
```

Grid size that will be used by the application in any dispatch of the kernel. If the kernel contains a requiredgridsize control directive, the dimensions should match.

The specified grid size must be consistent with *required_workgroup_size* and *required_dim*. Also, the product of the three dimensions must not exceed *max_flat_grid_size*. Note that the listed invariants must hold only if all the corresponding control directives are enabled.

If the bit at position BRIG_CONTROL_REQUIREDGRIDSIZE is set in *control_directives_mask*, the three dimension values must be greater than 0.

Definition at line 398 of file hsa_ext_finalize.h.

6.91.2.9 required_workgroup_size

```
hsa_dim3_t hsa_ext_control_directives_s::required_workgroup_size
```

Work-group size that will be used by the application in any dispatch of the kernel. If the kernel contains a required-workgroupsize control directive, the dimensions should match.

The specified work-group size must be consistent with *required_grid_size* and *required_dim*. Also, the product of the three dimensions must not exceed *max_flat_workgroup_size*. Note that the listed invariants must hold only if all the corresponding control directives are enabled.

If the bit at position BRIG_CONTROL_REQUIREDWORKGROUPSIZE is set in *control_directives_mask*, the three dimension values must be greater than 0.

Definition at line 412 of file hsa_ext_finalize.h.

6.91.2.10 reserved1

```
uint32_t hsa_ext_control_directives_s::reserved1
```

Reserved. Must be 0.

Definition at line 384 of file hsa_ext_finalize.h.

6.91.2.11 reserved2

```
uint8_t hsa_ext_control_directives_s::reserved2[75]
```

Reserved. Must be 0.

Definition at line 429 of file hsa_ext_finalize.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_finalize.h

6.92 hsa_ext_finalizer_1_00_pfn_s Struct Reference

Public Attributes

- hsa_status_t(∗ hsa_ext_program_create)(hsa_machine_model_t machine_model, hsa_profile_t profile, hsa_default_float_rounding_mode_t default_float_rounding_mode, const char ∗options, hsa_ext_program_t ∗program)
- hsa_status_t(∗ hsa_ext_program_destroy)(hsa_ext_program_t program)
- hsa_status_t(∗ hsa_ext_program_add_module)(hsa_ext_program_t program, hsa_ext_module_t module)
- hsa_status_t(∗ hsa_ext_program_iterate_modules)(hsa_ext_program_t program, hsa_status_t(∗callback)(hsa_ext_program_t program, hsa_ext_module_t module, void ∗data), void ∗data)
- hsa_status_t(∗ hsa_ext_program_get_info)(hsa_ext_program_t program, hsa_ext_program_info_t attribute, void ∗value)
- hsa_status_t(∗ hsa_ext_program_finalize)(hsa_ext_program_t program, hsa_isa_t isa, int32_t call_convention, hsa_ext_control_directives_t control_directives, const char ∗options, hsa_code_object_type_t code_object_type, hsa_code_object_t ∗code_object)

6.92.1 Detailed Description

Definition at line 500 of file hsa_ext_finalize.h.

6.92.2 Member Data Documentation

6.92.2.1 hsa_ext_program_add_module

```
hsa_status_t(* hsa_ext_finalizer_1_00_pfn_s::hsa_ext_program_add_module) (hsa_ext_program_t
program, hsa_ext_module_t module)
```

Definition at line 508 of file hsa_ext_finalize.h.

6.92.2.2 hsa_ext_program_create

```
hsa_status_t(* hsa_ext_finalizer_1_00_pfn_s::hsa_ext_program_create) (hsa_machine_model_t
machine_model, hsa_profile_t profile, hsa_default_float_rounding_mode_t default_float_rounding
_mode, const char *options, hsa_ext_program_t *program)
```

Definition at line 501 of file hsa_ext_finalize.h.

6.92.2.3 hsa_ext_program_destroy

```
hsa_status_t(* hsa_ext_finalizer_1_00_pfn_s::hsa_ext_program_destroy) (hsa_ext_program_t program)
```

Definition at line 506 of file hsa_ext_finalize.h.

6.92.2.4 hsa_ext_program_finalize

```
hsa_status_t(* hsa_ext_finalizer_1_00_pfn_s::hsa_ext_program_finalize) (hsa_ext_program_t
program, hsa_isa_t isa, int32_t call_convention, hsa_ext_control_directives_t control_directives,
const char *options, hsa_code_object_type_t code_object_type, hsa_code_object_t *code_object)
```

Definition at line 521 of file hsa_ext_finalize.h.

6.92.2.5 hsa_ext_program_get_info

```
hsa_status_t(* hsa_ext_finalizer_1_00_pfn_s::hsa_ext_program_get_info) (hsa_ext_program_t
program, hsa_ext_program_info_t attribute, void *value)
```

Definition at line 517 of file hsa_ext_finalize.h.

6.92.2.6 hsa_ext_program_iterate_modules

```
hsa_status_t(* hsa_ext_finalizer_1_00_pfn_s::hsa_ext_program_iterate_modules) (hsa_ext_program_t
program, hsa_status_t(*callback)(hsa_ext_program_t program, hsa_ext_module_t module, void
*data), void *data)
```

Definition at line 511 of file hsa_ext_finalize.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_finalize.h

6.93 hsa_ext_image_data_info_s Struct Reference

Agent specific image size and alignment requirements, populated by hsa_ext_image_data_get_info and hsa_ext_image_data_get_info_with_layout.

```
#include <hsa_ext_image.h>
```

Public Attributes

- size_t size
- size_t alignment

6.93.1 Detailed Description

Agent specific image size and alignment requirements, populated by hsa_ext_image_data_get_info and hsa_ext_image_data_get_info_with_layout.

Definition at line 509 of file hsa_ext_image.h.

6.93.2 Member Data Documentation

6.93.2.1 alignment

```
size_t hsa_ext_image_data_info_s::alignment
```

Image data alignment, in bytes. Must always be a power of 2.

Definition at line 518 of file hsa_ext_image.h.

6.93.2.2 size

```
size_t hsa_ext_image_data_info_s::size
```

Image data size, in bytes.

Definition at line 513 of file hsa_ext_image.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_image.h

6.94 hsa_ext_image_descriptor_s Struct Reference

Implementation independent image descriptor.

```
#include <hsa_ext_image.h>
```

Collaboration diagram for hsa_ext_image_descriptor_s:

Public Attributes

- hsa_ext_image_geometry_t geometry
- size_t width
- size_t height
- size_t depth
- size_t array_size
- hsa_ext_image_format_t format

6.94.1 Detailed Description

Implementation independent image descriptor.

Definition at line 326 of file hsa_ext_image.h.

6.94.2 Member Data Documentation

6.94.2.1 array_size

```
size_t hsa_ext_image_descriptor_s::array_size
```

Number of image layers in the image array. Only used if the geometry is HSA_EXT_IMAGE_GEOMETRY_1DA, HSA_EXT_IMAGE_GEOMETRY_2DA, or HSA_EXT_IMAGE_GEOMETRY_2DADEPTH, otherwise must be 0.

Definition at line 352 of file hsa_ext_image.h.

6.94.2.2 depth

```
size_t hsa_ext_image_descriptor_s::depth
```

Depth of the image, in components. Only used if the geometry is HSA_EXT_IMAGE_GEOMETRY_3D, otherwise must be 0.

Definition at line 346 of file hsa_ext_image.h.

6.94.2.3 format

```
hsa_ext_image_format_t hsa_ext_image_descriptor_s::format
```

Image format.

Definition at line 356 of file hsa_ext_image.h.

6.94.2.4 geometry

```
hsa_ext_image_geometry_t hsa_ext_image_descriptor_s::geometry
```

Image geometry.

Definition at line 330 of file hsa_ext_image.h.

6.94.2.5 height

```
size_t hsa_ext_image_descriptor_s::height
```

Height of the image, in components. Only used if the geometry is HSA_EXT_IMAGE_GEOMETRY_2D, HSA_EXT_IMAGE_GEOMETRY_3D, HSA_EXT_IMAGE_GEOMETRY_2DA, HSA_EXT_IMAGE_GEOMETRY_↩ 2DDEPTH, or HSA_EXT_IMAGE_GEOMETRY_2DADEPTH, otherwise must be 0.

Definition at line 341 of file hsa_ext_image.h.

6.94.2.6 width

```
size_t hsa_ext_image_descriptor_s::width
```

Width of the image, in components.

Definition at line 334 of file hsa_ext_image.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_image.h

6.95 hsa_ext_image_format_s Struct Reference

Image format.

```
#include <hsa_ext_image.h>
```

Public Attributes

- hsa_ext_image_channel_type32_t channel_type
- hsa_ext_image_channel_order32_t channel_order

6.95.1 Detailed Description

Image format.

Definition at line 311 of file hsa_ext_image.h.

6.95.2 Member Data Documentation

6.95.2.1 channel_order

`hsa_ext_image_channel_order32_t hsa_ext_image_format_s::channel_order`

Channel order.

Definition at line 320 of file hsa_ext_image.h.

6.95.2.2 channel_type

`hsa_ext_image_channel_type32_t hsa_ext_image_format_s::channel_type`

Channel type.

Definition at line 315 of file hsa_ext_image.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_image.h

6.96 hsa_ext_image_region_s Struct Reference

Image region.

`#include <hsa_ext_image.h>`

Collaboration diagram for hsa_ext_image_region_s:

Public Attributes

- hsa_dim3_t offset
- hsa_dim3_t range

6.96.1 Detailed Description

Image region.

Definition at line 927 of file hsa_ext_image.h.

6.96.2 Member Data Documentation

6.96.2.1 offset

```
hsa_dim3_t hsa_ext_image_region_s::offset
```

Offset within an image (in coordinates).

Definition at line 931 of file hsa_ext_image.h.

6.96.2.2 range

```
hsa_dim3_t hsa_ext_image_region_s::range
```

Dimension size of the image range (in coordinates). The x, y, and z dimensions correspond to width, height, and depth or index respectively.

Definition at line 937 of file hsa_ext_image.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_image.h

6.97 hsa_ext_image_s Struct Reference

Image handle, populated by hsa_ext_image_create or hsa_ext_image_create_with_layout. Image handles are only unique within an agent, not across agents.

```
#include <hsa_ext_image.h>
```

Public Attributes

- uint64_t handle

6.97.1 Detailed Description

Image handle, populated by hsa_ext_image_create or hsa_ext_image_create_with_layout. Image handles are only unique within an agent, not across agents.

Definition at line 174 of file hsa_ext_image.h.

6.97.2 Member Data Documentation

6.97.2.1 handle

```
uint64_t hsa_ext_image_s::handle
```

Opaque handle. For a given agent, two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 179 of file hsa_ext_image.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_image.h

6.98 hsa_ext_images_1_00_pfn_s Struct Reference

The function pointer table for the images v1.00 extension. Can be returned by hsa_system_get_extension_table or hsa_system_get_major_extension_table.

```
#include <hsa_ext_image.h>
```

Public Attributes

- hsa_status_t(∗ hsa_ext_image_get_capability)(hsa_agent_t agent, hsa_ext_image_geometry_t geometry, const hsa_ext_image_format_t ∗image_format, uint32_t ∗capability_mask)
- hsa_status_t(∗ hsa_ext_image_data_get_info)(hsa_agent_t agent, const hsa_ext_image_descriptor_t ∗image_descriptor, hsa_access_permission_t access_permission, hsa_ext_image_data_info_t ∗image↩ _data_info)
- hsa_status_t(∗ hsa_ext_image_create)(hsa_agent_t agent, const hsa_ext_image_descriptor_t ∗image↩ _descriptor, const void ∗image_data, hsa_access_permission_t access_permission, hsa_ext_image_t ∗image)
- hsa_status_t(∗ hsa_ext_image_destroy)(hsa_agent_t agent, hsa_ext_image_t image)
- hsa_status_t(∗ hsa_ext_image_copy)(hsa_agent_t agent, hsa_ext_image_t src_image, const hsa_dim3_t ∗src_offset, hsa_ext_image_t dst_image, const hsa_dim3_t ∗dst_offset, const hsa_dim3_t ∗range)
- hsa_status_t(∗ hsa_ext_image_import)(hsa_agent_t agent, const void ∗src_memory, size_t src_row_pitch, size_t src_slice_pitch, hsa_ext_image_t dst_image, const hsa_ext_image_region_t ∗image_region)
- hsa_status_t(∗ hsa_ext_image_export)(hsa_agent_t agent, hsa_ext_image_t src_image, void ∗dst_memory, size_t dst_row_pitch, size_t dst_slice_pitch, const hsa_ext_image_region_t ∗image_region)
- hsa_status_t(∗ hsa_ext_image_clear)(hsa_agent_t agent, hsa_ext_image_t image, const void ∗data, const hsa_ext_image_region_t ∗image_region)
- hsa_status_t(∗ hsa_ext_sampler_create)(hsa_agent_t agent, const hsa_ext_sampler_descriptor_t ∗sampler_descriptor, hsa_ext_sampler_t ∗sampler)
- hsa_status_t(∗ hsa_ext_sampler_destroy)(hsa_agent_t agent, hsa_ext_sampler_t sampler)

6.98.1 Detailed Description

The function pointer table for the images v1.00 extension. Can be returned by hsa_system_get_extension_table or hsa_system_get_major_extension_table.

Definition at line 1286 of file hsa_ext_image.h.

6.98.2 Member Data Documentation

6.98.2.1 hsa_ext_image_clear

```
hsa_status_t(* hsa_ext_images_1_00_pfn_s::hsa_ext_image_clear) (hsa_agent_t agent, hsa_ext_image_t
image, const void *data, const hsa_ext_image_region_t *image_region)
```

Definition at line 1335 of file hsa_ext_image.h.

6.98.2.2 hsa_ext_image_copy

```
hsa_status_t(* hsa_ext_images_1_00_pfn_s::hsa_ext_image_copy) (hsa_agent_t agent, hsa_ext_image_t
src_image, const hsa_dim3_t *src_offset, hsa_ext_image_t dst_image, const hsa_dim3_t *dst_↩
offset, const hsa_dim3_t *range)
```

Definition at line 1311 of file hsa_ext_image.h.

6.98.2.3 hsa_ext_image_create

```
hsa_status_t(* hsa_ext_images_1_00_pfn_s::hsa_ext_image_create) (hsa_agent_t agent, const
hsa_ext_image_descriptor_t *image_descriptor, const void *image_data, hsa_access_permission_t
access_permission, hsa_ext_image_t *image)
```

Definition at line 1300 of file hsa_ext_image.h.

6.98.2.4 hsa_ext_image_data_get_info

```
hsa_status_t(* hsa_ext_images_1_00_pfn_s::hsa_ext_image_data_get_info) (hsa_agent_t agent,
const hsa_ext_image_descriptor_t *image_descriptor, hsa_access_permission_t access_permission,
hsa_ext_image_data_info_t *image_data_info)
```

Definition at line 1294 of file hsa_ext_image.h.

6.98.2.5 hsa_ext_image_destroy

```
hsa_status_t(* hsa_ext_images_1_00_pfn_s::hsa_ext_image_destroy) (hsa_agent_t agent, hsa_ext_image_t
image)
```

Definition at line 1307 of file hsa_ext_image.h.

6.98.2.6 hsa_ext_image_export

```
hsa_status_t(* hsa_ext_images_1_00_pfn_s::hsa_ext_image_export) (hsa_agent_t agent, hsa_ext_image_t
src_image, void *dst_memory, size_t dst_row_pitch, size_t dst_slice_pitch, const hsa_ext_image_region_t
*image_region)
```

Definition at line 1327 of file hsa_ext_image.h.

6.98.2.7 hsa_ext_image_get_capability

```
hsa_status_t(* hsa_ext_images_1_00_pfn_s::hsa_ext_image_get_capability) (hsa_agent_t agent,
hsa_ext_image_geometry_t geometry, const hsa_ext_image_format_t *image_format, uint32_t *capability↩
_mask)
```

Definition at line 1288 of file hsa_ext_image.h.

6.98.2.8 hsa_ext_image_import

```
hsa_status_t(* hsa_ext_images_1_00_pfn_s::hsa_ext_image_import) (hsa_agent_t agent, const void
*src_memory, size_t src_row_pitch, size_t src_slice_pitch, hsa_ext_image_t dst_image, const
hsa_ext_image_region_t *image_region)
```

Definition at line 1319 of file hsa_ext_image.h.

6.98.2.9 hsa_ext_sampler_create

```
hsa_status_t(* hsa_ext_images_1_00_pfn_s::hsa_ext_sampler_create) (hsa_agent_t agent, const
hsa_ext_sampler_descriptor_t *sampler_descriptor, hsa_ext_sampler_t *sampler)
```

Definition at line 1341 of file hsa_ext_image.h.

6.98.2.10 hsa_ext_sampler_destroy

```
hsa_status_t(* hsa_ext_images_1_00_pfn_s::hsa_ext_sampler_destroy) (hsa_agent_t agent, hsa_ext_sampler_t
sampler)
```

Definition at line 1346 of file hsa_ext_image.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_image.h

6.99 hsa_ext_images_1_pfn_s Struct Reference

The function pointer table for the images v1 extension. Can be returned by hsa_system_get_extension_table or hsa_system_get_major_extension_table.

`#include <hsa_ext_image.h>`

Public Attributes

- hsa_status_t(∗ hsa_ext_image_get_capability)(hsa_agent_t agent, hsa_ext_image_geometry_t geometry, const hsa_ext_image_format_t ∗image_format, uint32_t ∗capability_mask)
- hsa_status_t(∗ hsa_ext_image_data_get_info)(hsa_agent_t agent, const hsa_ext_image_descriptor_t ∗image_descriptor, hsa_access_permission_t access_permission, hsa_ext_image_data_info_t ∗image_data_info)
- hsa_status_t(∗ hsa_ext_image_create)(hsa_agent_t agent, const hsa_ext_image_descriptor_t ∗image_descriptor, const void ∗image_data, hsa_access_permission_t access_permission, hsa_ext_image_t ∗image)
- hsa_status_t(∗ hsa_ext_image_destroy)(hsa_agent_t agent, hsa_ext_image_t image)
- hsa_status_t(∗ hsa_ext_image_copy)(hsa_agent_t agent, hsa_ext_image_t src_image, const hsa_dim3_t ∗src_offset, hsa_ext_image_t dst_image, const hsa_dim3_t ∗dst_offset, const hsa_dim3_t ∗range)
- hsa_status_t(∗ hsa_ext_image_import)(hsa_agent_t agent, const void ∗src_memory, size_t src_row_pitch, size_t src_slice_pitch, hsa_ext_image_t dst_image, const hsa_ext_image_region_t ∗image_region)
- hsa_status_t(∗ hsa_ext_image_export)(hsa_agent_t agent, hsa_ext_image_t src_image, void ∗dst_memory, size_t dst_row_pitch, size_t dst_slice_pitch, const hsa_ext_image_region_t ∗image_region)
- hsa_status_t(∗ hsa_ext_image_clear)(hsa_agent_t agent, hsa_ext_image_t image, const void ∗data, const hsa_ext_image_region_t ∗image_region)
- hsa_status_t(∗ hsa_ext_sampler_create)(hsa_agent_t agent, const hsa_ext_sampler_descriptor_t ∗sampler_descriptor, hsa_ext_sampler_t ∗sampler)
- hsa_status_t(∗ hsa_ext_sampler_destroy)(hsa_agent_t agent, hsa_ext_sampler_t sampler)
- hsa_status_t(∗ hsa_ext_image_get_capability_with_layout)(hsa_agent_t agent, hsa_ext_image_geometry_t geometry, const hsa_ext_image_format_t ∗image_format, hsa_ext_image_data_layout_t image_data_layout, uint32_t ∗capability_mask)
- hsa_status_t(∗ hsa_ext_image_data_get_info_with_layout)(hsa_agent_t agent, const hsa_ext_image_descriptor_t ∗image_descriptor, hsa_access_permission_t access_permission, hsa_ext_image_data_layout_t image_data_layout, size_t image_data_row_pitch, size_t image_data_slice_pitch, hsa_ext_image_data_info_t ∗image_data_info)
- hsa_status_t(∗ hsa_ext_image_create_with_layout)(hsa_agent_t agent, const hsa_ext_image_descriptor_t ∗image_descriptor, const void ∗image_data, hsa_access_permission_t access_permission, hsa_ext_image_data_layout_t image_data_layout, size_t image_data_row_pitch, size_t image_data_slice_pitch, hsa_ext_image_t ∗image)

6.99.1 Detailed Description

The function pointer table for the images v1 extension. Can be returned by hsa_system_get_extension_table or hsa_system_get_major_extension_table.

Definition at line 1357 of file hsa_ext_image.h.

6.99.2 Member Data Documentation

6.99.2.1 hsa_ext_image_clear

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_clear) (hsa_agent_t agent, hsa_ext_image_t
image, const void *data, const hsa_ext_image_region_t *image_region)
```

Definition at line 1406 of file hsa_ext_image.h.

6.99.2.2 hsa_ext_image_copy

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_copy) (hsa_agent_t agent, hsa_ext_image_t
src_image, const hsa_dim3_t *src_offset, hsa_ext_image_t dst_image, const hsa_dim3_t *dst_↩
offset, const hsa_dim3_t *range)
```

Definition at line 1382 of file hsa_ext_image.h.

6.99.2.3 hsa_ext_image_create

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_create) (hsa_agent_t agent, const hsa_ext_image_descriptc
*image_descriptor, const void *image_data, hsa_access_permission_t access_permission, hsa_ext_image_t
*image)
```

Definition at line 1371 of file hsa_ext_image.h.

6.99.2.4 hsa_ext_image_create_with_layout

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_create_with_layout) (hsa_agent_t agent,
const hsa_ext_image_descriptor_t *image_descriptor, const void *image_data, hsa_access_permission_t
access_permission, hsa_ext_image_data_layout_t image_data_layout, size_t image_data_row_pitch,
size_t image_data_slice_pitch, hsa_ext_image_t *image)
```

Definition at line 1437 of file hsa_ext_image.h.

6.99.2.5 hsa_ext_image_data_get_info

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_data_get_info) (hsa_agent_t agent, const
hsa_ext_image_descriptor_t *image_descriptor, hsa_access_permission_t access_permission, hsa_ext_image_data_in
*image_data_info)
```

Definition at line 1365 of file hsa_ext_image.h.

6.99.2.6 hsa_ext_image_data_get_info_with_layout

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_data_get_info_with_layout) (hsa_agent_t
agent, const hsa_ext_image_descriptor_t *image_descriptor, hsa_access_permission_t access_↩
permission, hsa_ext_image_data_layout_t image_data_layout, size_t image_data_row_pitch, size↩
_t image_data_slice_pitch, hsa_ext_image_data_info_t *image_data_info)
```

Definition at line 1428 of file hsa_ext_image.h.

6.99.2.7 hsa_ext_image_destroy

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_destroy) (hsa_agent_t agent, hsa_ext_image_t
image)
```

Definition at line 1378 of file hsa_ext_image.h.

6.99.2.8 hsa_ext_image_export

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_export) (hsa_agent_t agent, hsa_ext_image_t
src_image, void *dst_memory, size_t dst_row_pitch, size_t dst_slice_pitch, const hsa_ext_image_region_t
*image_region)
```

Definition at line 1398 of file hsa_ext_image.h.

6.99.2.9 hsa_ext_image_get_capability

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_get_capability) (hsa_agent_t agent, hsa_ext_image_geometr
geometry, const hsa_ext_image_format_t *image_format, uint32_t *capability_mask)
```

Definition at line 1359 of file hsa_ext_image.h.

6.99.2.10 hsa_ext_image_get_capability_with_layout

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_get_capability_with_layout) (hsa_agent_t
agent, hsa_ext_image_geometry_t geometry, const hsa_ext_image_format_t *image_format, hsa_ext_image_data_layou
image_data_layout, uint32_t *capability_mask)
```

Definition at line 1421 of file hsa_ext_image.h.

6.99.2.11 hsa_ext_image_import

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_image_import) (hsa_agent_t agent, const void
*src_memory, size_t src_row_pitch, size_t src_slice_pitch, hsa_ext_image_t dst_image, const
hsa_ext_image_region_t *image_region)
```

Definition at line 1390 of file hsa_ext_image.h.

6.99.2.12 hsa_ext_sampler_create

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_sampler_create) (hsa_agent_t agent, const hsa_ext_sampler_descr
*sampler_descriptor, hsa_ext_sampler_t *sampler)
```

Definition at line 1412 of file hsa_ext_image.h.

6.99.2.13 hsa_ext_sampler_destroy

```
hsa_status_t(* hsa_ext_images_1_pfn_s::hsa_ext_sampler_destroy) (hsa_agent_t agent, hsa_ext_sampler_t
sampler)
```

Definition at line 1417 of file hsa_ext_image.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_image.h

6.100 hsa_ext_program_s Struct Reference

An opaque handle to a HSAIL program, which groups a set of HSAIL modules that collectively define functions and variables used by kernels and indirect functions.

```
#include <hsa_ext_finalize.h>
```

Public Attributes

- uint64_t handle

6.100.1 Detailed Description

An opaque handle to a HSAIL program, which groups a set of HSAIL modules that collectively define functions and variables used by kernels and indirect functions.

Definition at line 120 of file hsa_ext_finalize.h.

6.100.2 Member Data Documentation

6.100.2.1 handle

```
uint64_t hsa_ext_program_s::handle
```

Opaque handle.

Definition at line 124 of file hsa_ext_finalize.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_finalize.h

6.101 hsa_ext_sampler_descriptor_s Struct Reference

Implementation independent sampler descriptor.

```
#include <hsa_ext_image.h>
```

Public Attributes

- hsa_ext_sampler_coordinate_mode32_t coordinate_mode
- hsa_ext_sampler_filter_mode32_t filter_mode
- hsa_ext_sampler_addressing_mode32_t address_mode

6.101.1 Detailed Description

Implementation independent sampler descriptor.

Definition at line 1205 of file hsa_ext_image.h.

6.101.2 Member Data Documentation

6.101.2.1 address_mode

```
hsa_ext_sampler_addressing_mode32_t hsa_ext_sampler_descriptor_s::address_mode
```

Sampler address mode describes the processing of out-of-range image coordinates.

Definition at line 1220 of file hsa_ext_image.h.

6.101.2.2 coordinate_mode

`hsa_ext_sampler_coordinate_mode32_t hsa_ext_sampler_descriptor_s::coordinate_mode`

Sampler coordinate mode describes the normalization of image coordinates.

Definition at line 1209 of file hsa_ext_image.h.

6.101.2.3 filter_mode

`hsa_ext_sampler_filter_mode32_t hsa_ext_sampler_descriptor_s::filter_mode`

Sampler filter type describes the type of sampling performed.

Definition at line 1214 of file hsa_ext_image.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_image.h

6.102 hsa_ext_sampler_s Struct Reference

Sampler handle. Samplers are populated by hsa_ext_sampler_create. Sampler handles are only unique within an agent, not across agents.

`#include <hsa_ext_image.h>`

Public Attributes

- uint64_t handle

6.102.1 Detailed Description

Sampler handle. Samplers are populated by hsa_ext_sampler_create. Sampler handles are only unique within an agent, not across agents.

Definition at line 1096 of file hsa_ext_image.h.

6.102.2 Member Data Documentation

6.102.2.1 handle

```
uint64_t hsa_ext_sampler_s::handle
```

Opaque handle. For a given agent, two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 1101 of file hsa_ext_image.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_image.h

6.103 hsa_isa_s Struct Reference

Instruction set architecture.

```
#include <hsa.h>
```

Public Attributes

- uint64_t handle

6.103.1 Detailed Description

Instruction set architecture.

Definition at line 3539 of file hsa.h.

6.103.2 Member Data Documentation

6.103.2.1 handle

```
uint64_t hsa_isa_s::handle
```

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 3544 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.104 hsa_kernel_dispatch_packet_s Struct Reference

AQL kernel dispatch packet.

`#include <hsa.h>`

Collaboration diagram for hsa_kernel_dispatch_packet_s:

Public Attributes

- uint16_t header
- uint16_t setup
- uint16_t workgroup_size_x
- uint16_t workgroup_size_y
- uint16_t workgroup_size_z
- uint16_t reserved0
- uint32_t grid_size_x
- uint32_t grid_size_y
- uint32_t grid_size_z
- uint32_t private_segment_size
- uint32_t group_segment_size
- uint64_t kernel_object
- uint32_t reserved1
- void * kernarg_address
- uint64_t reserved2
- hsa_signal_t completion_signal

6.104.1 Detailed Description

AQL kernel dispatch packet.

Definition at line 2918 of file hsa.h.

6.104.2 Member Data Documentation

6.104.2.1 completion_signal

`hsa_signal_t hsa_kernel_dispatch_packet_s::completion_signal`

Signal used to indicate completion of the job. The application can use the special signal handle 0 to indicate that no signal is used.

Definition at line 3022 of file hsa.h.

6.104.2.2 grid_size_x

```
uint32_t hsa_kernel_dispatch_packet_s::grid_size_x
```

X dimension of grid, in work-items. Must be greater than 0. Must not be smaller than *workgroup_size_x*.

Definition at line 2958 of file hsa.h.

6.104.2.3 grid_size_y

```
uint32_t hsa_kernel_dispatch_packet_s::grid_size_y
```

Y dimension of grid, in work-items. Must be greater than 0. If the grid has 1 dimension, the only valid value is 1. Must not be smaller than *workgroup_size_y*.

Definition at line 2965 of file hsa.h.

6.104.2.4 grid_size_z

```
uint32_t hsa_kernel_dispatch_packet_s::grid_size_z
```

Z dimension of grid, in work-items. Must be greater than 0. If the grid has 1 or 2 dimensions, the only valid value is 1. Must not be smaller than *workgroup_size_z*.

Definition at line 2972 of file hsa.h.

6.104.2.5 group_segment_size

```
uint32_t hsa_kernel_dispatch_packet_s::group_segment_size
```

Size in bytes of group memory allocation request (per work-group). Must not be less than the sum of the group memory used by the kernel (and the functions it calls directly or indirectly) and the dynamically allocated group segment variables.

Definition at line 2985 of file hsa.h.

6.104.2.6 header

```
uint16_t hsa_kernel_dispatch_packet_s::header
```

Packet header. Used to configure multiple packet parameters such as the packet type. The parameters are described by hsa_packet_header_t.

Definition at line 2923 of file hsa.h.

6.104.2.7 kernarg_address

```
void* hsa_kernel_dispatch_packet_s::kernarg_address
```

Definition at line 3010 of file hsa.h.

6.104.2.8 kernel_object

```
uint64_t hsa_kernel_dispatch_packet_s::kernel_object
```

Opaque handle to a code object that includes an implementation-defined executable code for the kernel.

Definition at line 2991 of file hsa.h.

6.104.2.9 private_segment_size

```
uint32_t hsa_kernel_dispatch_packet_s::private_segment_size
```

Size in bytes of private memory allocation request (per work-item).

Definition at line 2977 of file hsa.h.

6.104.2.10 reserved0

```
uint16_t hsa_kernel_dispatch_packet_s::reserved0
```

Reserved. Must be 0.

Definition at line 2952 of file hsa.h.

6.104.2.11 reserved1

```
uint32_t hsa_kernel_dispatch_packet_s::reserved1
```

Definition at line 3009 of file hsa.h.

6.104.2.12 reserved2

```
uint64_t hsa_kernel_dispatch_packet_s::reserved2
```

Reserved. Must be 0.

Definition at line 3016 of file hsa.h.

6.104.2.13 setup

```
uint16_t hsa_kernel_dispatch_packet_s::setup
```

Dispatch setup parameters. Used to configure kernel dispatch parameters such as the number of dimensions in the grid. The parameters are described by hsa_kernel_dispatch_packet_setup_t.

Definition at line 2930 of file hsa.h.

6.104.2.14 workgroup_size_x

```
uint16_t hsa_kernel_dispatch_packet_s::workgroup_size_x
```

X dimension of work-group, in work-items. Must be greater than 0.

Definition at line 2935 of file hsa.h.

6.104.2.15 workgroup_size_y

```
uint16_t hsa_kernel_dispatch_packet_s::workgroup_size_y
```

Y dimension of work-group, in work-items. Must be greater than 0. If the grid has 1 dimension, the only valid value is 1.

Definition at line 2941 of file hsa.h.

6.104.2.16 workgroup_size_z

```
uint16_t hsa_kernel_dispatch_packet_s::workgroup_size_z
```

Z dimension of work-group, in work-items. Must be greater than 0. If the grid has 1 or 2 dimensions, the only valid value is 1.

Definition at line 2947 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.105 hsa_loaded_code_object_s Struct Reference

Loaded code object handle.

`#include <hsa.h>`

Public Attributes

- uint64_t handle

6.105.1 Detailed Description

Loaded code object handle.

Definition at line 4240 of file hsa.h.

6.105.2 Member Data Documentation

6.105.2.1 handle

`uint64_t hsa_loaded_code_object_s::handle`

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 4245 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.106 hsa_pitched_ptr_s Struct Reference

Public Attributes

- void * base
- size_t pitch
- size_t slice

6.106.1 Detailed Description

Definition at line 1209 of file hsa_ext_amd.h.

6.106.2 Member Data Documentation

6.106.2.1 base

```
void* hsa_pitched_ptr_s::base
```

Definition at line 1210 of file hsa_ext_amd.h.

6.106.2.2 pitch

```
size_t hsa_pitched_ptr_s::pitch
```

Definition at line 1211 of file hsa_ext_amd.h.

6.106.2.3 slice

```
size_t hsa_pitched_ptr_s::slice
```

Definition at line 1212 of file hsa_ext_amd.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ext_amd.h

6.107 hsa_queue_s Struct Reference

User mode queue.

```
#include <hsa.h>
```

Collaboration diagram for hsa_queue_s:

Public Attributes

- hsa_queue_type32_t type
- uint32_t features
- uint32_t reserved0
- void * base_address
- hsa_signal_t doorbell_signal
- uint32_t size
- uint32_t reserved1
- uint64_t id

6.107.1 Detailed Description

User mode queue.

The queue structure is read-only and allocated by the HSA runtime, but agents can directly modify the contents of the buffer pointed by *base_address*, or use HSA runtime APIs to access the doorbell signal.

Definition at line 2267 of file hsa.h.

6.107.2 Member Data Documentation

6.107.2.1 base_address

```
void* hsa_queue_s::base_address
```

Definition at line 2293 of file hsa.h.

6.107.2.2 doorbell_signal

```
hsa_signal_t hsa_queue_s::doorbell_signal
```

Signal object used by the application to indicate the ID of a packet that is ready to be processed. The HSA runtime manages the doorbell signal. If the application tries to replace or destroy this signal, the behavior is undefined.

If *type* is HSA_QUEUE_TYPE_SINGLE, the doorbell signal value must be updated in a monotonically increasing fashion. If *type* is HSA_QUEUE_TYPE_MULTI, the doorbell signal value can be updated with any value.

Definition at line 2307 of file hsa.h.

6.107.2.3 features

```
uint32_t hsa_queue_s::features
```

Queue features mask. This is a bit-field of hsa_queue_feature_t values. Applications should ignore any unknown set bits.

Definition at line 2277 of file hsa.h.

6.107.2.4 id

```
uint64_t hsa_queue_s::id
```

Queue identifier, which is unique over the lifetime of the application.

Definition at line 2320 of file hsa.h.

6.107.2.5 reserved0

```
uint32_t hsa_queue_s::reserved0
```

Definition at line 2292 of file hsa.h.

6.107.2.6 reserved1

```
uint32_t hsa_queue_s::reserved1
```

Reserved. Must be 0.

Definition at line 2316 of file hsa.h.

6.107.2.7 size

```
uint32_t hsa_queue_s::size
```

Maximum number of packets the queue can hold. Must be a power of 2.

Definition at line 2312 of file hsa.h.

6.107.2.8 type

```
hsa_queue_type32_t hsa_queue_s::type
```

Queue type.

Definition at line 2271 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.108 hsa_region_s Struct Reference

A memory region represents a block of virtual memory with certain properties. For example, the HSA runtime represents fine-grained memory in the global segment using a region. A region might be associated with more than one agent.

`#include <hsa.h>`

Public Attributes

- uint64_t handle

6.108.1 Detailed Description

A memory region represents a block of virtual memory with certain properties. For example, the HSA runtime represents fine-grained memory in the global segment using a region. A region might be associated with more than one agent.

Definition at line 2193 of file hsa.h.

6.108.2 Member Data Documentation

6.108.2.1 handle

`uint64_t hsa_region_s::handle`

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 2198 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.109 hsa_signal_group_s Struct Reference

Group of signals.

`#include <hsa.h>`

Public Attributes

- uint64_t handle

6.109.1 Detailed Description

Group of signals.

Definition at line 2053 of file hsa.h.

6.109.2 Member Data Documentation

6.109.2.1 handle

```
uint64_t hsa_signal_group_s::handle
```

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 2058 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.110 hsa_signal_s Struct Reference

Signal handle.

```
#include <hsa.h>
```

Public Attributes

- uint64_t handle

6.110.1 Detailed Description

Signal handle.

Definition at line 1325 of file hsa.h.

6.110.2 Member Data Documentation

6.110.2.1 handle

```
uint64_t hsa_signal_s::handle
```

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal. The value 0 is reserved.

Definition at line 1330 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.111 hsa_ven_amd_aqlprofile_1_00_pfn_s Struct Reference

Extension function table.

```
#include <hsa_ven_amd_aqlprofile.h>
```

Public Attributes

- uint32_t(∗ hsa_ven_amd_aqlprofile_version_major)()
- uint32_t(∗ hsa_ven_amd_aqlprofile_version_minor)()
- hsa_status_t(∗ hsa_ven_amd_aqlprofile_error_string)(const char ∗∗str)
- hsa_status_t(∗ hsa_ven_amd_aqlprofile_validate_event)(hsa_agent_t agent, const hsa_ven_amd_aqlprofile_event_t ∗event, bool ∗result)
- hsa_status_t(∗ hsa_ven_amd_aqlprofile_start)(hsa_ven_amd_aqlprofile_profile_t ∗profile, hsa_ext_amd_aql_pm4_packet_t ∗aql_start_packet)
- hsa_status_t(∗ hsa_ven_amd_aqlprofile_stop)(const hsa_ven_amd_aqlprofile_profile_t ∗profile, hsa_ext_amd_aql_pm4_packe ∗aql_stop_packet)
- hsa_status_t(∗ hsa_ven_amd_aqlprofile_read)(const hsa_ven_amd_aqlprofile_profile_t ∗profile, hsa_ext_amd_aql_pm4_pack ∗aql_read_packet)
- hsa_status_t(∗ hsa_ven_amd_aqlprofile_legacy_get_pm4)(const hsa_ext_amd_aql_pm4_packet_t ∗aql_↩ packet, void ∗data)
- hsa_status_t(∗ hsa_ven_amd_aqlprofile_get_info)(const hsa_ven_amd_aqlprofile_profile_t ∗profile, hsa_↩ ven_amd_aqlprofile_info_type_t attribute, void ∗value)
- hsa_status_t(∗ hsa_ven_amd_aqlprofile_iterate_data)(const hsa_ven_amd_aqlprofile_profile_t ∗profile, hsa_ven_amd_aqlprofile_data_callback_t callback, void ∗data)

6.111.1 Detailed Description

Extension function table.

Definition at line 317 of file hsa_ven_amd_aqlprofile.h.

6.111.2 Member Data Documentation

6.111.2.1 hsa_ven_amd_aqlprofile_error_string

```
hsa_status_t(* hsa_ven_amd_aqlprofile_1_00_pfn_s::hsa_ven_amd_aqlprofile_error_string) (const
char **str)
```

Definition at line 321 of file hsa_ven_amd_aqlprofile.h.

6.111.2.2 hsa_ven_amd_aqlprofile_get_info

```
hsa_status_t(* hsa_ven_amd_aqlprofile_1_00_pfn_s::hsa_ven_amd_aqlprofile_get_info) (const
hsa_ven_amd_aqlprofile_profile_t *profile, hsa_ven_amd_aqlprofile_info_type_t attribute, void
*value)
```

Definition at line 345 of file hsa_ven_amd_aqlprofile.h.

6.111.2.3 hsa_ven_amd_aqlprofile_iterate_data

```
hsa_status_t(* hsa_ven_amd_aqlprofile_1_00_pfn_s::hsa_ven_amd_aqlprofile_iterate_data) (const
hsa_ven_amd_aqlprofile_profile_t *profile, hsa_ven_amd_aqlprofile_data_callback_t callback,
void *data)
```

Definition at line 350 of file hsa_ven_amd_aqlprofile.h.

6.111.2.4 hsa_ven_amd_aqlprofile_legacy_get_pm4

```
hsa_status_t(* hsa_ven_amd_aqlprofile_1_00_pfn_s::hsa_ven_amd_aqlprofile_legacy_get_pm4) (const
hsa_ext_amd_aql_pm4_packet_t *aql_packet, void *data)
```

Definition at line 341 of file hsa_ven_amd_aqlprofile.h.

6.111.2.5 hsa_ven_amd_aqlprofile_read

```
hsa_status_t(* hsa_ven_amd_aqlprofile_1_00_pfn_s::hsa_ven_amd_aqlprofile_read) (const hsa_ven_amd_aqlprofile_p
*profile, hsa_ext_amd_aql_pm4_packet_t *aql_read_packet)
```

Definition at line 337 of file hsa_ven_amd_aqlprofile.h.

6.111.2.6 hsa_ven_amd_aqlprofile_start

```
hsa_status_t(* hsa_ven_amd_aqlprofile_1_00_pfn_s::hsa_ven_amd_aqlprofile_start) (hsa_ven_amd_aqlprofile_profil
*profile, hsa_ext_amd_aql_pm4_packet_t *aql_start_packet)
```

Definition at line 329 of file hsa_ven_amd_aqlprofile.h.

6.111.2.7 hsa_ven_amd_aqlprofile_stop

```
hsa_status_t(* hsa_ven_amd_aqlprofile_1_00_pfn_s::hsa_ven_amd_aqlprofile_stop) (const hsa_ven_amd_aqlprofile_p
*profile, hsa_ext_amd_aql_pm4_packet_t *aql_stop_packet)
```

Definition at line 333 of file hsa_ven_amd_aqlprofile.h.

6.111.2.8 hsa_ven_amd_aqlprofile_validate_event

```
hsa_status_t(* hsa_ven_amd_aqlprofile_1_00_pfn_s::hsa_ven_amd_aqlprofile_validate_event) (hsa_agent_t
agent, const hsa_ven_amd_aqlprofile_event_t *event, bool *result)
```

Definition at line 324 of file hsa_ven_amd_aqlprofile.h.

6.111.2.9 hsa_ven_amd_aqlprofile_version_major

```
uint32_t(* hsa_ven_amd_aqlprofile_1_00_pfn_s::hsa_ven_amd_aqlprofile_version_major) ()
```

Definition at line 318 of file hsa_ven_amd_aqlprofile.h.

6.111.2.10 hsa_ven_amd_aqlprofile_version_minor

```
uint32_t(* hsa_ven_amd_aqlprofile_1_00_pfn_s::hsa_ven_amd_aqlprofile_version_minor) ()
```

Definition at line 319 of file hsa_ven_amd_aqlprofile.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_aqlprofile.h

6.112 hsa_ven_amd_aqlprofile_descriptor_t Struct Reference

Public Attributes

- void * ptr
- uint32_t size

6.112.1 Detailed Description

Definition at line 176 of file hsa_ven_amd_aqlprofile.h.

6.112.2 Member Data Documentation

6.112.2.1 ptr

```
void* hsa_ven_amd_aqlprofile_descriptor_t::ptr
```

Definition at line 177 of file hsa_ven_amd_aqlprofile.h.

6.112.2.2 size

```
uint32_t hsa_ven_amd_aqlprofile_descriptor_t::size
```

Definition at line 178 of file hsa_ven_amd_aqlprofile.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_aqlprofile.h

6.113 hsa_ven_amd_aqlprofile_event_t Struct Reference

Public Attributes

- hsa_ven_amd_aqlprofile_block_name_t block_name
- uint32_t block_index
- uint32_t counter_id

6.113.1 Detailed Description

Definition at line 132 of file hsa_ven_amd_aqlprofile.h.

6.113.2 Member Data Documentation

6.113.2.1 block_index

```
uint32_t hsa_ven_amd_aqlprofile_event_t::block_index
```

Definition at line 134 of file hsa_ven_amd_aqlprofile.h.

6.113.2.2 block_name

```
hsa_ven_amd_aqlprofile_block_name_t hsa_ven_amd_aqlprofile_event_t::block_name
```

Definition at line 133 of file hsa_ven_amd_aqlprofile.h.

6.113.2.3 counter_id

```
uint32_t hsa_ven_amd_aqlprofile_event_t::counter_id
```

Definition at line 135 of file hsa_ven_amd_aqlprofile.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_aqlprofile.h

6.114 hsa_ven_amd_aqlprofile_id_query_t Struct Reference

Public Attributes

- const char * name
- uint32_t id
- uint32_t instance_count

6.114.1 Detailed Description

Definition at line 255 of file hsa_ven_amd_aqlprofile.h.

6.114.2 Member Data Documentation

6.114.2.1 id

```
uint32_t hsa_ven_amd_aqlprofile_id_query_t::id
```

Definition at line 257 of file hsa_ven_amd_aqlprofile.h.

6.114.2.2 instance_count

```
uint32_t hsa_ven_amd_aqlprofile_id_query_t::instance_count
```

Definition at line 258 of file hsa_ven_amd_aqlprofile.h.

6.114.2.3 name

```
const char* hsa_ven_amd_aqlprofile_id_query_t::name
```

Definition at line 256 of file hsa_ven_amd_aqlprofile.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_aqlprofile.h

6.115 hsa_ven_amd_aqlprofile_info_data_t Struct Reference

Collaboration diagram for hsa_ven_amd_aqlprofile_info_data_t:

Public Attributes

- uint32_t sample_id
-

  union {
  struct {
  hsa_ven_amd_aqlprofile_event_t event
  uint64_t result
  } **pmc_data**
  hsa_ven_amd_aqlprofile_descriptor_t trace_data
  };

6.115.1 Detailed Description

Definition at line 243 of file hsa_ven_amd_aqlprofile.h.

6.115.2 Member Data Documentation

6.115.2.1 event

`hsa_ven_amd_aqlprofile_event_t hsa_ven_amd_aqlprofile_info_data_t::event`

Definition at line 247 of file hsa_ven_amd_aqlprofile.h.

6.115.2.2 result

`uint64_t hsa_ven_amd_aqlprofile_info_data_t::result`

Definition at line 248 of file hsa_ven_amd_aqlprofile.h.

6.115.2.3 sample_id

`uint32_t hsa_ven_amd_aqlprofile_info_data_t::sample_id`

Definition at line 244 of file hsa_ven_amd_aqlprofile.h.

6.115.2.4 trace_data

`hsa_ven_amd_aqlprofile_descriptor_t hsa_ven_amd_aqlprofile_info_data_t::trace_data`

Definition at line 250 of file hsa_ven_amd_aqlprofile.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_aqlprofile.h

6.116 hsa_ven_amd_aqlprofile_parameter_t Struct Reference

Public Attributes

- hsa_ven_amd_aqlprofile_parameter_name_t parameter_name
- uint32_t value

6.116.1 Detailed Description

Definition at line 160 of file hsa_ven_amd_aqlprofile.h.

6.116.2 Member Data Documentation

6.116.2.1 parameter_name

```
hsa_ven_amd_aqlprofile_parameter_name_t hsa_ven_amd_aqlprofile_parameter_t::parameter_name
```

Definition at line 161 of file hsa_ven_amd_aqlprofile.h.

6.116.2.2 value

```
uint32_t hsa_ven_amd_aqlprofile_parameter_t::value
```

Definition at line 162 of file hsa_ven_amd_aqlprofile.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_aqlprofile.h

6.117 hsa_ven_amd_aqlprofile_profile_t Struct Reference

Collaboration diagram for hsa_ven_amd_aqlprofile_profile_t:

Public Attributes

- hsa_agent_t agent
- hsa_ven_amd_aqlprofile_event_type_t type
- const hsa_ven_amd_aqlprofile_event_t * events
- uint32_t event_count
- const hsa_ven_amd_aqlprofile_parameter_t * parameters
- uint32_t parameter_count
- hsa_ven_amd_aqlprofile_descriptor_t output_buffer
- hsa_ven_amd_aqlprofile_descriptor_t command_buffer

6.117.1 Detailed Description

Definition at line 184 of file hsa_ven_amd_aqlprofile.h.

6.117.2 Member Data Documentation

6.117.2.1 agent

```
hsa_agent_t hsa_ven_amd_aqlprofile_profile_t::agent
```

Definition at line 185 of file hsa_ven_amd_aqlprofile.h.

6.117.2.2 command_buffer

```
hsa_ven_amd_aqlprofile_descriptor_t hsa_ven_amd_aqlprofile_profile_t::command_buffer
```

Definition at line 192 of file hsa_ven_amd_aqlprofile.h.

6.117.2.3 event_count

```
uint32_t hsa_ven_amd_aqlprofile_profile_t::event_count
```

Definition at line 188 of file hsa_ven_amd_aqlprofile.h.

6.117.2.4 events

```
const hsa_ven_amd_aqlprofile_event_t* hsa_ven_amd_aqlprofile_profile_t::events
```

Definition at line 187 of file hsa_ven_amd_aqlprofile.h.

6.117.2.5 output_buffer

```
hsa_ven_amd_aqlprofile_descriptor_t hsa_ven_amd_aqlprofile_profile_t::output_buffer
```

Definition at line 191 of file hsa_ven_amd_aqlprofile.h.

6.117.2.6 parameter_count

```
uint32_t hsa_ven_amd_aqlprofile_profile_t::parameter_count
```

Definition at line 190 of file hsa_ven_amd_aqlprofile.h.

6.117.2.7 parameters

```
const hsa_ven_amd_aqlprofile_parameter_t* hsa_ven_amd_aqlprofile_profile_t::parameters
```

Definition at line 189 of file hsa_ven_amd_aqlprofile.h.

6.117.2.8 type

```
hsa_ven_amd_aqlprofile_event_type_t hsa_ven_amd_aqlprofile_profile_t::type
```

Definition at line 186 of file hsa_ven_amd_aqlprofile.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_aqlprofile.h

6.118 hsa_ven_amd_loader_1_00_pfn_s Struct Reference

Extension function table version 1.00.

```
#include <hsa_ven_amd_loader.h>
```

Public Attributes

- hsa_status_t(* hsa_ven_amd_loader_query_host_address)(const void *device_address, const void **host_address)
- hsa_status_t(* hsa_ven_amd_loader_query_segment_descriptors)(hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors, size_t *num_segment_descriptors)
- hsa_status_t(* hsa_ven_amd_loader_query_executable)(const void *device_address, hsa_executable_t *executable)

6.118.1 Detailed Description

Extension function table version 1.00.

Definition at line 538 of file hsa_ven_amd_loader.h.

6.118.2 Member Data Documentation

6.118.2.1 hsa_ven_amd_loader_query_executable

```
hsa_status_t(* hsa_ven_amd_loader_1_00_pfn_s::hsa_ven_amd_loader_query_executable) (const void
*device_address, hsa_executable_t *executable)
```

Definition at line 547 of file hsa_ven_amd_loader.h.

6.118.2.2 hsa_ven_amd_loader_query_host_address

```
hsa_status_t(* hsa_ven_amd_loader_1_00_pfn_s::hsa_ven_amd_loader_query_host_address) (const
void *device_address, const void **host_address)
```

Definition at line 539 of file hsa_ven_amd_loader.h.

6.118.2.3 hsa_ven_amd_loader_query_segment_descriptors

```
hsa_status_t(* hsa_ven_amd_loader_1_00_pfn_s::hsa_ven_amd_loader_query_segment_descriptors)
(hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors, size_t *num_segment_descriptors)
```

Definition at line 543 of file hsa_ven_amd_loader.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_loader.h

6.119 hsa_ven_amd_loader_1_01_pfn_s Struct Reference

Extension function table version 1.01.

```
#include <hsa_ven_amd_loader.h>
```

Public Attributes

- hsa_status_t(* hsa_ven_amd_loader_query_host_address)(const void *device_address, const void **host_address)
- hsa_status_t(* hsa_ven_amd_loader_query_segment_descriptors)(hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors, size_t *num_segment_descriptors)
- hsa_status_t(* hsa_ven_amd_loader_query_executable)(const void *device_address, hsa_executable_t *executable)
- hsa_status_t(* hsa_ven_amd_loader_executable_iterate_loaded_code_objects)(hsa_executable_t executable, hsa_status_t(*callback)(hsa_executable_t executable, hsa_loaded_code_object_t loaded_code_object, void *data), void *data)
- hsa_status_t(* hsa_ven_amd_loader_loaded_code_object_get_info)(hsa_loaded_code_object_t loaded_code_object, hsa_ven_amd_loader_loaded_code_object_info_t attribute, void *value)

6.119.1 Detailed Description

Extension function table version 1.01.

Definition at line 555 of file hsa_ven_amd_loader.h.

6.119.2 Member Data Documentation

6.119.2.1 hsa_ven_amd_loader_executable_iterate_loaded_code_objects

```
hsa_status_t(* hsa_ven_amd_loader_1_01_pfn_s::hsa_ven_amd_loader_executable_iterate_loaded_↩
code_objects) (hsa_executable_t executable, hsa_status_t(*callback)( hsa_executable_t executable,
hsa_loaded_code_object_t loaded_code_object, void *data), void *data)
```

Definition at line 568 of file hsa_ven_amd_loader.h.

6.119.2.2 hsa_ven_amd_loader_loaded_code_object_get_info

```
hsa_status_t(* hsa_ven_amd_loader_1_01_pfn_s::hsa_ven_amd_loader_loaded_code_object_get_info)
(hsa_loaded_code_object_t loaded_code_object, hsa_ven_amd_loader_loaded_code_object_info_↩
t attribute, void *value)
```

Definition at line 576 of file hsa_ven_amd_loader.h.

6.119.2.3 hsa_ven_amd_loader_query_executable

```
hsa_status_t(* hsa_ven_amd_loader_1_01_pfn_s::hsa_ven_amd_loader_query_executable) (const void
*device_address, hsa_executable_t *executable)
```

Definition at line 564 of file hsa_ven_amd_loader.h.

6.119.2.4 hsa_ven_amd_loader_query_host_address

```
hsa_status_t(* hsa_ven_amd_loader_1_01_pfn_s::hsa_ven_amd_loader_query_host_address) (const
void *device_address, const void **host_address)
```

Definition at line 556 of file hsa_ven_amd_loader.h.

6.119.2.5 hsa_ven_amd_loader_query_segment_descriptors

```
hsa_status_t(* hsa_ven_amd_loader_1_01_pfn_s::hsa_ven_amd_loader_query_segment_descriptors)
(hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors, size_t *num_segment_descriptors)
```

Definition at line 560 of file hsa_ven_amd_loader.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_loader.h

6.120 hsa_ven_amd_loader_1_02_pfn_s Struct Reference

Extension function table version 1.02.

```
#include <hsa_ven_amd_loader.h>
```

Public Attributes

- hsa_status_t(* hsa_ven_amd_loader_query_host_address)(const void *device_address, const void **host_address)
- hsa_status_t(* hsa_ven_amd_loader_query_segment_descriptors)(hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors, size_t *num_segment_descriptors)
- hsa_status_t(* hsa_ven_amd_loader_query_executable)(const void *device_address, hsa_executable_t *executable)
- hsa_status_t(* hsa_ven_amd_loader_executable_iterate_loaded_code_objects)(hsa_executable_t executable, hsa_status_t(*callback)(hsa_executable_t executable, hsa_loaded_code_object_t loaded_code_object, void *data), void *data)
- hsa_status_t(* hsa_ven_amd_loader_loaded_code_object_get_info)(hsa_loaded_code_object_t loaded_code_object, hsa_ven_amd_loader_loaded_code_object_info_t attribute, void *value)
- hsa_status_t(* hsa_ven_amd_loader_code_object_reader_create_from_file_with_offset_size)(hsa_file_t file, size_t offset, size_t size, hsa_code_object_reader_t *code_object_reader)

6.120.1 Detailed Description

Extension function table version 1.02.

Definition at line 585 of file hsa_ven_amd_loader.h.

6.120.2 Member Data Documentation

6.120.2.1 hsa_ven_amd_loader_code_object_reader_create_from_file_with_offset_size

```
hsa_status_t(* hsa_ven_amd_loader_1_02_pfn_s::hsa_ven_amd_loader_code_object_reader_create_↩
from_file_with_offset_size) (hsa_file_t file, size_t offset, size_t size, hsa_code_object_reader_t
*code_object_reader)
```

Definition at line 611 of file hsa_ven_amd_loader.h.

6.120.2.2 hsa_ven_amd_loader_executable_iterate_loaded_code_objects

```
hsa_status_t(* hsa_ven_amd_loader_1_02_pfn_s::hsa_ven_amd_loader_executable_iterate_loaded_↩
code_objects) (hsa_executable_t executable, hsa_status_t(*callback)( hsa_executable_t executable,
hsa_loaded_code_object_t loaded_code_object, void *data), void *data)
```

Definition at line 598 of file hsa_ven_amd_loader.h.

6.120.2.3 hsa_ven_amd_loader_loaded_code_object_get_info

```
hsa_status_t(* hsa_ven_amd_loader_1_02_pfn_s::hsa_ven_amd_loader_loaded_code_object_get_info)
(hsa_loaded_code_object_t loaded_code_object, hsa_ven_amd_loader_loaded_code_object_info_↩
t attribute, void *value)
```

Definition at line 606 of file hsa_ven_amd_loader.h.

6.120.2.4 hsa_ven_amd_loader_query_executable

```
hsa_status_t(* hsa_ven_amd_loader_1_02_pfn_s::hsa_ven_amd_loader_query_executable) (const void
*device_address, hsa_executable_t *executable)
```

Definition at line 594 of file hsa_ven_amd_loader.h.

6.120.2.5 hsa_ven_amd_loader_query_host_address

```
hsa_status_t(* hsa_ven_amd_loader_1_02_pfn_s::hsa_ven_amd_loader_query_host_address) (const
void *device_address, const void **host_address)
```

Definition at line 586 of file hsa_ven_amd_loader.h.

6.120.2.6 hsa_ven_amd_loader_query_segment_descriptors

```
hsa_status_t(* hsa_ven_amd_loader_1_02_pfn_s::hsa_ven_amd_loader_query_segment_descriptors)
(hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors, size_t *num_segment_descriptors)
```

Definition at line 590 of file hsa_ven_amd_loader.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_loader.h

6.121 hsa_ven_amd_loader_1_03_pfn_s Struct Reference

Extension function table version 1.03.

```
#include <hsa_ven_amd_loader.h>
```

Public Attributes

- hsa_status_t(* hsa_ven_amd_loader_query_host_address)(const void *device_address, const void **host_address)
- hsa_status_t(* hsa_ven_amd_loader_query_segment_descriptors)(hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors, size_t *num_segment_descriptors)
- hsa_status_t(* hsa_ven_amd_loader_query_executable)(const void *device_address, hsa_executable_t *executable)
- hsa_status_t(* hsa_ven_amd_loader_executable_iterate_loaded_code_objects)(hsa_executable_t executable, hsa_status_t(*callback)(hsa_executable_t executable, hsa_loaded_code_object_t loaded_code_object, void *data), void *data)
- hsa_status_t(* hsa_ven_amd_loader_loaded_code_object_get_info)(hsa_loaded_code_object_t loaded_code_object, hsa_ven_amd_loader_loaded_code_object_info_t attribute, void *value)
- hsa_status_t(* hsa_ven_amd_loader_code_object_reader_create_from_file_with_offset_size)(hsa_file_t file, size_t offset, size_t size, hsa_code_object_reader_t *code_object_reader)
- hsa_status_t(* hsa_ven_amd_loader_iterate_executables)(hsa_status_t(*callback)(hsa_executable_t executable, void *data), void *data)

6.121.1 Detailed Description

Extension function table version 1.03.

Definition at line 622 of file hsa_ven_amd_loader.h.

6.121.2 Member Data Documentation

6.121.2.1 hsa_ven_amd_loader_code_object_reader_create_from_file_with_offset_size

```
hsa_status_t(* hsa_ven_amd_loader_1_03_pfn_s::hsa_ven_amd_loader_code_object_reader_create_↩
from_file_with_offset_size) (hsa_file_t file, size_t offset, size_t size, hsa_code_object_reader_t
*code_object_reader)
```

Definition at line 648 of file hsa_ven_amd_loader.h.

6.121.2.2 hsa_ven_amd_loader_executable_iterate_loaded_code_objects

```
hsa_status_t(* hsa_ven_amd_loader_1_03_pfn_s::hsa_ven_amd_loader_executable_iterate_loaded_↩
code_objects) (hsa_executable_t executable, hsa_status_t(*callback)( hsa_executable_t executable,
hsa_loaded_code_object_t loaded_code_object, void *data), void *data)
```

Definition at line 635 of file hsa_ven_amd_loader.h.

6.121.2.3 hsa_ven_amd_loader_iterate_executables

```
hsa_status_t(* hsa_ven_amd_loader_1_03_pfn_s::hsa_ven_amd_loader_iterate_executables) (hsa_status_t(*callback)
hsa_executable_t executable, void *data), void *data)
```

Definition at line 655 of file hsa_ven_amd_loader.h.

6.121.2.4 hsa_ven_amd_loader_loaded_code_object_get_info

```
hsa_status_t(* hsa_ven_amd_loader_1_03_pfn_s::hsa_ven_amd_loader_loaded_code_object_get_info)
(hsa_loaded_code_object_t loaded_code_object, hsa_ven_amd_loader_loaded_code_object_info_↩
t attribute, void *value)
```

Definition at line 643 of file hsa_ven_amd_loader.h.

6.121.2.5 hsa_ven_amd_loader_query_executable

```
hsa_status_t(* hsa_ven_amd_loader_1_03_pfn_s::hsa_ven_amd_loader_query_executable) (const void
*device_address, hsa_executable_t *executable)
```

Definition at line 631 of file hsa_ven_amd_loader.h.

6.121.2.6 hsa_ven_amd_loader_query_host_address

```
hsa_status_t(* hsa_ven_amd_loader_1_03_pfn_s::hsa_ven_amd_loader_query_host_address) (const
void *device_address, const void **host_address)
```

Definition at line 623 of file hsa_ven_amd_loader.h.

6.121.2.7 hsa_ven_amd_loader_query_segment_descriptors

```
hsa_status_t(* hsa_ven_amd_loader_1_03_pfn_s::hsa_ven_amd_loader_query_segment_descriptors)
(hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors, size_t *num_segment_descriptors)
```

Definition at line 627 of file hsa_ven_amd_loader.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_loader.h

6.122 hsa_ven_amd_loader_segment_descriptor_s Struct Reference

Loaded memory segment descriptor.

```
#include <hsa_ven_amd_loader.h>
```

Collaboration diagram for hsa_ven_amd_loader_segment_descriptor_s:

Public Attributes

- hsa_agent_t agent
- hsa_executable_t executable
- hsa_ven_amd_loader_code_object_storage_type_t code_object_storage_type
- const void * code_object_storage_base
- size_t code_object_storage_size
- size_t code_object_storage_offset
- const void * segment_base
- size_t segment_size

6.122.1 Detailed Description

Loaded memory segment descriptor.

Loaded memory segment descriptor describes underlying loaded memory segment. Loaded memory segment is created/allocated by the executable during the loading of the code object that is backing underlying memory segment.

The lifetime of underlying memory segment is limited by the lifetime of the executable that is managing underlying memory segment.

Definition at line 123 of file hsa_ven_amd_loader.h.

6.122.2 Member Data Documentation

6.122.2.1 agent

```
hsa_agent_t hsa_ven_amd_loader_segment_descriptor_s::agent
```

Agent underlying memory segment is allocated on. If the code object that is backing underlying memory segment is program code object, then 0.

Definition at line 128 of file hsa_ven_amd_loader.h.

6.122.2.2 code_object_storage_base

```
const void* hsa_ven_amd_loader_segment_descriptor_s::code_object_storage_base
```

If the storage type of the code object that is backing underlying memory segment is:

- HSA_VEN_AMD_LOADER_CODE_OBJECT_STORAGE_TYPE_NONE, then null;
- HSA_VEN_AMD_LOADER_CODE_OBJECT_STORAGE_TYPE_FILE, then null-terminated filepath to the code object;
- HSA_VEN_AMD_LOADER_CODE_OBJECT_STORAGE_TYPE_MEMORY, then host accessible pointer to the first byte of the code object.

Definition at line 146 of file hsa_ven_amd_loader.h.

6.122.2.3 code_object_storage_offset

```
size_t hsa_ven_amd_loader_segment_descriptor_s::code_object_storage_offset
```

If the storage type of the code object that is backing underlying memory segment is:

- HSA_VEN_AMD_LOADER_CODE_OBJECT_STORAGE_TYPE_NONE, then 0;
- other, then offset, in bytes, from the beginning of the code object to the first byte in the code object data is copied from.

Definition at line 164 of file hsa_ven_amd_loader.h.

6.122.2.4 code_object_storage_size

```
size_t hsa_ven_amd_loader_segment_descriptor_s::code_object_storage_size
```

If the storage type of the code object that is backing underlying memory segment is:

- HSA_VEN_AMD_LOADER_CODE_OBJECT_STORAGE_TYPE_NONE, then 0;
- HSA_VEN_AMD_LOADER_CODE_OBJECT_STORAGE_TYPE_FILE, then the length of the filepath to the code object (including null-terminating character);
- HSA_VEN_AMD_LOADER_CODE_OBJECT_STORAGE_TYPE_MEMORY, then the size, in bytes, of the memory occupied by the code object.

Definition at line 156 of file hsa_ven_amd_loader.h.

6.122.2.5 code_object_storage_type

```
hsa_ven_amd_loader_code_object_storage_type_t hsa_ven_amd_loader_segment_descriptor_s::code_↩
object_storage_type
```

Storage type of the code object that is backing underlying memory segment.

Definition at line 136 of file hsa_ven_amd_loader.h.

6.122.2.6 executable

```
hsa_executable_t hsa_ven_amd_loader_segment_descriptor_s::executable
```

Executable that is managing this underlying memory segment.

Definition at line 132 of file hsa_ven_amd_loader.h.

6.122.2.7 segment_base

```
const void* hsa_ven_amd_loader_segment_descriptor_s::segment_base
```

Starting address of the underlying memory segment.

Definition at line 168 of file hsa_ven_amd_loader.h.

6.122.2.8 segment_size

```
size_t hsa_ven_amd_loader_segment_descriptor_s::segment_size
```

Size, in bytes, of the underlying memory segment.

Definition at line 172 of file hsa_ven_amd_loader.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_ven_amd_loader.h

6.123 hsa_wavefront_s Struct Reference

Wavefront handle.

```
#include <hsa.h>
```

Public Attributes

- uint64_t handle

6.123.1 Detailed Description

Wavefront handle.

Definition at line 3900 of file hsa.h.

6.123.2 Member Data Documentation

6.123.2.1 handle

```
uint64_t hsa_wavefront_s::handle
```

Opaque handle. Two handles reference the same object of the enclosing type if and only if they are equal.

Definition at line 3905 of file hsa.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa.h

6.124 HsaApiTable Struct Reference

Collaboration diagram for HsaApiTable:

Public Attributes

- ApiTableVersion version
- CoreApiTable * core_
- AmdExtTable * amd_ext_
- FinalizerExtTable * finalizer_ext_
- ImageExtTable * image_ext_

6.124.1 Detailed Description

Definition at line 372 of file hsa_api_trace.h.

6.124.2 Member Data Documentation

6.124.2.1 amd_ext_

```
AmdExtTable* HsaApiTable::amd_ext_
```

Definition at line 381 of file hsa_api_trace.h.

6.124.2.2 core_

```
CoreApiTable* HsaApiTable::core_
```

Definition at line 378 of file hsa_api_trace.h.

6.124.2.3 finalizer_ext_

```
FinalizerExtTable* HsaApiTable::finalizer_ext_
```

Definition at line 384 of file hsa_api_trace.h.

6.124.2.4 image_ext_

```
ImageExtTable* HsaApiTable::image_ext_
```

Definition at line 387 of file hsa_api_trace.h.

6.124.2.5 version

```
ApiTableVersion HsaApiTable::version
```

Definition at line 375 of file hsa_api_trace.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_api_trace.h

6.125 HsaApiTableContainer Struct Reference

Collaboration diagram for HsaApiTableContainer:

Public Attributes

- HsaApiTable root
- CoreApiTable core
- AmdExtTable amd_ext
- FinalizerExtTable finalizer_ext
- ImageExtTable image_ext

6.125.1 Detailed Description

Definition at line 391 of file hsa_api_trace.h.

6.125.2 Constructor & Destructor Documentation

6.125.2.1 HsaApiTableContainer()

```
HsaApiTableContainer::HsaApiTableContainer ( )  [inline]
```

Definition at line 399 of file hsa_api_trace.h.

6.125.3 Member Data Documentation

6.125.3.1 amd_ext

```
AmdExtTable HsaApiTableContainer::amd_ext
```

Definition at line 394 of file hsa_api_trace.h.

6.125.3.2 core

```
CoreApiTable HsaApiTableContainer::core
```

Definition at line 393 of file hsa_api_trace.h.

6.125.3.3 finalizer_ext

```
FinalizerExtTable HsaApiTableContainer::finalizer_ext
```

Definition at line 395 of file hsa_api_trace.h.

6.125.3.4 image_ext

```
ImageExtTable HsaApiTableContainer::image_ext
```

Definition at line 396 of file hsa_api_trace.h.

6.125.3.5 root

```
HsaApiTable HsaApiTableContainer::root
```

Definition at line 392 of file hsa_api_trace.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_api_trace.h

6.126 ImageExtTable Struct Reference

Collaboration diagram for ImageExtTable:

Public Attributes

- ApiTableVersion version
- decltype(hsa_ext_image_get_capability) * hsa_ext_image_get_capability_fn
- decltype(hsa_ext_image_data_get_info) * hsa_ext_image_data_get_info_fn
- decltype(hsa_ext_image_create) * hsa_ext_image_create_fn
- decltype(hsa_ext_image_import) * hsa_ext_image_import_fn
- decltype(hsa_ext_image_export) * hsa_ext_image_export_fn
- decltype(hsa_ext_image_copy) * hsa_ext_image_copy_fn
- decltype(hsa_ext_image_clear) * hsa_ext_image_clear_fn
- decltype(hsa_ext_image_destroy) * hsa_ext_image_destroy_fn
- decltype(hsa_ext_sampler_create) * hsa_ext_sampler_create_fn
- decltype(hsa_ext_sampler_destroy) * hsa_ext_sampler_destroy_fn
- decltype(hsa_ext_image_get_capability_with_layout) * hsa_ext_image_get_capability_with_layout_fn
- decltype(hsa_ext_image_data_get_info_with_layout) * hsa_ext_image_data_get_info_with_layout_fn
- decltype(hsa_ext_image_create_with_layout) * hsa_ext_image_create_with_layout_fn

6.126.1 Detailed Description

Definition at line 122 of file hsa_api_trace.h.

6.126.2 Member Data Documentation

6.126.2.1 hsa_ext_image_clear_fn

```
decltype(hsa_ext_image_clear) * ImageExtTable::hsa_ext_image_clear_fn
```

Definition at line 130 of file hsa_api_trace.h.

6.126.2.2 hsa_ext_image_copy_fn

```
decltype(hsa_ext_image_copy) * ImageExtTable::hsa_ext_image_copy_fn
```

Definition at line 129 of file hsa_api_trace.h.

6.126.2.3 hsa_ext_image_create_fn

```
decltype(hsa_ext_image_create) * ImageExtTable::hsa_ext_image_create_fn
```

Definition at line 126 of file hsa_api_trace.h.

6.126.2.4 hsa_ext_image_create_with_layout_fn

```
decltype(hsa_ext_image_create_with_layout) * ImageExtTable::hsa_ext_image_create_with_layout_↩
fn
```

Definition at line 136 of file hsa_api_trace.h.

6.126.2.5 hsa_ext_image_data_get_info_fn

```
decltype(hsa_ext_image_data_get_info) * ImageExtTable::hsa_ext_image_data_get_info_fn
```

Definition at line 125 of file hsa_api_trace.h.

6.126.2.6 hsa_ext_image_data_get_info_with_layout_fn

```
decltype(hsa_ext_image_data_get_info_with_layout) * ImageExtTable::hsa_ext_image_data_get_↩
info_with_layout_fn
```

Definition at line 135 of file hsa_api_trace.h.

6.126.2.7 hsa_ext_image_destroy_fn

```
decltype(hsa_ext_image_destroy) * ImageExtTable::hsa_ext_image_destroy_fn
```

Definition at line 131 of file hsa_api_trace.h.

6.126.2.8 hsa_ext_image_export_fn

```
decltype(hsa_ext_image_export) * ImageExtTable::hsa_ext_image_export_fn
```

Definition at line 128 of file hsa_api_trace.h.

6.126.2.9 hsa_ext_image_get_capability_fn

```
decltype(hsa_ext_image_get_capability) * ImageExtTable::hsa_ext_image_get_capability_fn
```

Definition at line 124 of file hsa_api_trace.h.

6.126.2.10 hsa_ext_image_get_capability_with_layout_fn

```
decltype(hsa_ext_image_get_capability_with_layout) * ImageExtTable::hsa_ext_image_get_capability↩
_with_layout_fn
```

Definition at line 134 of file hsa_api_trace.h.

6.126.2.11 hsa_ext_image_import_fn

```
decltype(hsa_ext_image_import) * ImageExtTable::hsa_ext_image_import_fn
```

Definition at line 127 of file hsa_api_trace.h.

6.126.2.12 hsa_ext_sampler_create_fn

```
decltype(hsa_ext_sampler_create) * ImageExtTable::hsa_ext_sampler_create_fn
```

Definition at line 132 of file hsa_api_trace.h.

6.126.2.13 hsa_ext_sampler_destroy_fn

```
decltype(hsa_ext_sampler_destroy) * ImageExtTable::hsa_ext_sampler_destroy_fn
```

Definition at line 133 of file hsa_api_trace.h.

6.126.2.14 version

```
ApiTableVersion ImageExtTable::version
```

Definition at line 123 of file hsa_api_trace.h.

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

- /home/alexv/Programming/ROCR-Runtime/include/hsa_api_trace.h

Chapter 7

File Documentation

7.1 amd_hsa_common.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 // The following set of header files provides definitions for AMD GPU
00044 // Architecture:
00045 //   - amd_hsa_common.h
00046 //   - amd_hsa_elf.h
00047 //   - amd_hsa_kernel_code.h
00048 //   - amd_hsa_queue.h
00049 //   - amd_hsa_signal.h
00050 //
00051 // Refer to "HSA Application Binary Interface: AMD GPU Architecture" for more
00052 // information.
00053
00054 #ifndef AMD_HSA_COMMON_H
00055 #define AMD_HSA_COMMON_H
00056
00057 #include <stddef.h>
00058 #include <stdint.h>
00059
```

```
00060 // Descriptive version of the HSA Application Binary Interface.
00061 #define AMD_HSA_ABI_VERSION "AMD GPU Architecture v0.35 (June 25, 2015)"
00062
00063 // Alignment attribute that specifies a minimum alignment (in bytes) for
00064 // variables of the specified type.
00065 #if defined(__GNUC__)
00066 #  define __ALIGNED__(x) __attribute__((aligned(x)))
00067 #elif defined(_MSC_VER)
00068 #  define __ALIGNED__(x) __declspec(align(x))
00069 #elif defined(RC_INVOKED)
00070 #  define __ALIGNED__(x)
00071 #else
00072 #  error
00073 #endif
00074
00075 // Creates enumeration entries for packed types. Enumeration entries include
00076 // bit shift amount, bit width, and bit mask.
00077 #define AMD_HSA_BITS_CREATE_ENUM_ENTRIES(name, shift, width)                   \
00078   name##_SHIFT = (shift),                                                      \
00079   name##_WIDTH = (width),                                                      \
00080   name = (((1 << (width)) - 1) << (shift))                                   \
00081
00082 // Gets bits for specified mask from specified src packed instance.
00083 #define AMD_HSA_BITS_GET(src, mask)                                            \
00084   ((src & mask) >> mask ## _SHIFT)                                            \
00085
00086 // Sets val bits for specified mask in specified dst packed instance.
00087 #define AMD_HSA_BITS_SET(dst, mask, val)                                       \
00088   dst &= (~(1 << mask##_SHIFT) & ~mask);                                       \
00089   dst |= (((val) << mask##_SHIFT) & mask)                                      \
00090
00091 #endif // AMD_HSA_COMMON_H
```

7.2 amd_hsa_elf.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 // Undefine the macro in case it is defined in the system elf.h.
00044 #undef EM_AMDGPU
00045
00046 #ifndef AMD_HSA_ELF_H
00047 #define AMD_HSA_ELF_H
00048
00049 // AMD GPU Specific ELF Header Enumeration Values.
```

```
00050 //
00051 // Values are copied from LLVM BinaryFormat/ELF.h . This file also contains
00052 // code object V1 defintions which are not part of the LLVM header. Code object
00053 // V1 was only supported by the Finalizer which is now deprecated and removed.
00054 //
00055 // TODO: Deprecate and remove V1 support and replace this header with using the
00056 // LLVM header.
00057 namespace ELF {
00058
00059 // Machine architectures
00060 // See current registered ELF machine architectures at:
00061 //    http://www.uxsglobal.com/developers/gabi/latest/ch4.eheader.html
00062 enum {
00063   EM_AMDGPU = 224,        // AMD GPU architecture
00064 };
00065
00066 // OS ABI identification.
00067 enum {
00068   ELFOSABI_AMDGPU_HSA = 64,    // AMD HSA runtime
00069 };
00070
00071 // AMDGPU OS ABI Version identification.
00072 enum {
00073   // ELFABIVERSION_AMDGPU_HSA_V1 does not exist because OS ABI identification
00074   // was never defined for V1.
00075   ELFABIVERSION_AMDGPU_HSA_V2 = 0,
00076   ELFABIVERSION_AMDGPU_HSA_V3 = 1,
00077   ELFABIVERSION_AMDGPU_HSA_V4 = 2,
00078   ELFABIVERSION_AMDGPU_HSA_V5 = 3
00079 };
00080
00081 // AMDGPU specific e_flags.
00082 enum : unsigned {
00083   // Processor selection mask for EF_AMDGPU_MACH_* values.
00084   EF_AMDGPU_MACH = 0x0ff,
00085
00086   // Not specified processor.
00087   EF_AMDGPU_MACH_NONE = 0x000,
00088
00089   // AMDGCN-based processors.
00090   EF_AMDGPU_MACH_AMDGCN_GFX600        = 0x020,
00091   EF_AMDGPU_MACH_AMDGCN_GFX601        = 0x021,
00092   EF_AMDGPU_MACH_AMDGCN_GFX700        = 0x022,
00093   EF_AMDGPU_MACH_AMDGCN_GFX701        = 0x023,
00094   EF_AMDGPU_MACH_AMDGCN_GFX702        = 0x024,
00095   EF_AMDGPU_MACH_AMDGCN_GFX703        = 0x025,
00096   EF_AMDGPU_MACH_AMDGCN_GFX704        = 0x026,
00097   EF_AMDGPU_MACH_AMDGCN_RESERVED_0X27 = 0x027,
00098   EF_AMDGPU_MACH_AMDGCN_GFX801        = 0x028,
00099   EF_AMDGPU_MACH_AMDGCN_GFX802        = 0x029,
00100   EF_AMDGPU_MACH_AMDGCN_GFX803        = 0x02a,
00101   EF_AMDGPU_MACH_AMDGCN_GFX810        = 0x02b,
00102   EF_AMDGPU_MACH_AMDGCN_GFX900        = 0x02c,
00103   EF_AMDGPU_MACH_AMDGCN_GFX902        = 0x02d,
00104   EF_AMDGPU_MACH_AMDGCN_GFX904        = 0x02e,
00105   EF_AMDGPU_MACH_AMDGCN_GFX906        = 0x02f,
00106   EF_AMDGPU_MACH_AMDGCN_GFX908        = 0x030,
00107   EF_AMDGPU_MACH_AMDGCN_GFX909        = 0x031,
00108   EF_AMDGPU_MACH_AMDGCN_GFX90C        = 0x032,
00109   EF_AMDGPU_MACH_AMDGCN_GFX1010       = 0x033,
00110   EF_AMDGPU_MACH_AMDGCN_GFX1011       = 0x034,
00111   EF_AMDGPU_MACH_AMDGCN_GFX1012       = 0x035,
00112   EF_AMDGPU_MACH_AMDGCN_GFX1030       = 0x036,
00113   EF_AMDGPU_MACH_AMDGCN_GFX1031       = 0x037,
00114   EF_AMDGPU_MACH_AMDGCN_GFX1032       = 0x038,
00115   EF_AMDGPU_MACH_AMDGCN_GFX1033       = 0x039,
00116   EF_AMDGPU_MACH_AMDGCN_GFX602        = 0x03a,
00117   EF_AMDGPU_MACH_AMDGCN_GFX705        = 0x03b,
00118   EF_AMDGPU_MACH_AMDGCN_GFX805        = 0x03c,
00119   EF_AMDGPU_MACH_AMDGCN_GFX1035       = 0x03d,
00120   EF_AMDGPU_MACH_AMDGCN_GFX1034       = 0x03e,
00121   EF_AMDGPU_MACH_AMDGCN_GFX90A        = 0x03f,
00122   EF_AMDGPU_MACH_AMDGCN_RESERVED_0X40 = 0x040,
00123   EF_AMDGPU_MACH_AMDGCN_GFX1100       = 0x041,
00124   EF_AMDGPU_MACH_AMDGCN_GFX1013       = 0x042,
00125   EF_AMDGPU_MACH_AMDGCN_GFX1103       = 0x044,
00126   EF_AMDGPU_MACH_AMDGCN_GFX1036       = 0x045,
00127   EF_AMDGPU_MACH_AMDGCN_GFX1102       = 0x047,
00128
00129   // First/last AMDGCN-based processors.
00130   EF_AMDGPU_MACH_AMDGCN_FIRST = EF_AMDGPU_MACH_AMDGCN_GFX600,
00131   EF_AMDGPU_MACH_AMDGCN_LAST = EF_AMDGPU_MACH_AMDGCN_GFX1102,
00132
00133   // Indicates if the "xnack" target feature is enabled for all code contained
00134   // in the object.
00135   //
00136   // Only valid for ELFOSABI_AMDGPU_HSA and ELFABIVERSION_AMDGPU_HSA_V2.
```

```
00137   EF_AMDGPU_FEATURE_XNACK_V2 = 0x01,
00138   // Indicates if the trap handler is enabled for all code contained
00139   // in the object.
00140   //
00141   // Only valid for ELFOSABI_AMDGPU_HSA and ELFABIVERSION_AMDGPU_HSA_V2.
00142   EF_AMDGPU_FEATURE_TRAP_HANDLER_V2 = 0x02,
00143
00144   // Indicates if the "xnack" target feature is enabled for all code contained
00145   // in the object.
00146   //
00147   // Only valid for ELFOSABI_AMDGPU_HSA and ELFABIVERSION_AMDGPU_HSA_V3.
00148   EF_AMDGPU_FEATURE_XNACK_V3 = 0x100,
00149   // Indicates if the "sramecc" target feature is enabled for all code
00150   // contained in the object.
00151   //
00152   // Only valid for ELFOSABI_AMDGPU_HSA and ELFABIVERSION_AMDGPU_HSA_V3.
00153   EF_AMDGPU_FEATURE_SRAMECC_V3 = 0x200,
00154
00155   // XNACK selection mask for EF_AMDGPU_FEATURE_XNACK_* values.
00156   //
00157   // Only valid for ELFOSABI_AMDGPU_HSA and ELFABIVERSION_AMDGPU_HSA_V4,
00158   // ELFABIVERSION_AMDGPU_HSA_V5.
00159   EF_AMDGPU_FEATURE_XNACK_V4 = 0x300,
00160   // XNACK is not supported.
00161   EF_AMDGPU_FEATURE_XNACK_UNSUPPORTED_V4 = 0x000,
00162   // XNACK is any/default/unspecified.
00163   EF_AMDGPU_FEATURE_XNACK_ANY_V4 = 0x100,
00164   // XNACK is off.
00165   EF_AMDGPU_FEATURE_XNACK_OFF_V4 = 0x200,
00166   // XNACK is on.
00167   EF_AMDGPU_FEATURE_XNACK_ON_V4 = 0x300,
00168
00169   // SRAMECC selection mask for EF_AMDGPU_FEATURE_SRAMECC_* values.
00170   //
00171   // Only valid for ELFOSABI_AMDGPU_HSA and ELFABIVERSION_AMDGPU_HSA_V4,
00172   // ELFABIVERSION_AMDGPU_HSA_V5.
00173   EF_AMDGPU_FEATURE_SRAMECC_V4 = 0xc00,
00174   // SRAMECC is not supported.
00175   EF_AMDGPU_FEATURE_SRAMECC_UNSUPPORTED_V4 = 0x000,
00176   // SRAMECC is any/default/unspecified.
00177   EF_AMDGPU_FEATURE_SRAMECC_ANY_V4 = 0x400,
00178   // SRAMECC is off.
00179   EF_AMDGPU_FEATURE_SRAMECC_OFF_V4 = 0x800,
00180   // SRAMECC is on.
00181   EF_AMDGPU_FEATURE_SRAMECC_ON_V4 = 0xc00,
00182 };
00183
00184 } // end namespace ELF
00185
00186 // ELF Section Header Flag Enumeration Values.
00187 #define SHF_AMDGPU_HSA_GLOBAL   (0x00100000 & SHF_MASKOS)
00188 #define SHF_AMDGPU_HSA_READONLY (0x00200000 & SHF_MASKOS)
00189 #define SHF_AMDGPU_HSA_CODE     (0x00400000 & SHF_MASKOS)
00190 #define SHF_AMDGPU_HSA_AGENT    (0x00800000 & SHF_MASKOS)
00191
00192 //
00193 typedef enum {
00194   AMDGPU_HSA_SEGMENT_GLOBAL_PROGRAM = 0,
00195   AMDGPU_HSA_SEGMENT_GLOBAL_AGENT = 1,
00196   AMDGPU_HSA_SEGMENT_READONLY_AGENT = 2,
00197   AMDGPU_HSA_SEGMENT_CODE_AGENT = 3,
00198   AMDGPU_HSA_SEGMENT_LAST,
00199 } amdgpu_hsa_elf_segment_t;
00200
00201 // ELF Program Header Type Enumeration Values.
00202 #define PT_AMDGPU_HSA_LOAD_GLOBAL_PROGRAM (PT_LOOS + AMDGPU_HSA_SEGMENT_GLOBAL_PROGRAM)
00203 #define PT_AMDGPU_HSA_LOAD_GLOBAL_AGENT   (PT_LOOS + AMDGPU_HSA_SEGMENT_GLOBAL_AGENT)
00204 #define PT_AMDGPU_HSA_LOAD_READONLY_AGENT (PT_LOOS + AMDGPU_HSA_SEGMENT_READONLY_AGENT)
00205 #define PT_AMDGPU_HSA_LOAD_CODE_AGENT     (PT_LOOS + AMDGPU_HSA_SEGMENT_CODE_AGENT)
00206
00207 // ELF Symbol Type Enumeration Values.
00208 #define STT_AMDGPU_HSA_KERNEL            (STT_LOOS + 0)
00209 #define STT_AMDGPU_HSA_INDIRECT_FUNCTION (STT_LOOS + 1)
00210 #define STT_AMDGPU_HSA_METADATA          (STT_LOOS + 2)
00211
00212 // ELF Symbol Binding Enumeration Values.
00213 #define STB_AMDGPU_HSA_EXTERNAL (STB_LOOS + 0)
00214
00215 // ELF Symbol Other Information Creation/Retrieval.
00216 #define ELF64_ST_AMDGPU_ALLOCATION(o)  (((o) » 2) & 0x3)
00217 #define ELF64_ST_AMDGPU_FLAGS(o)       ((o) » 4)
00218 #define ELF64_ST_AMDGPU_OTHER(f, a, v) (((f) « 4) + (((a) & 0x3) « 2) + ((v) & 0x3))
00219
00220 typedef enum {
00221   AMDGPU_HSA_SYMBOL_ALLOCATION_DEFAULT = 0,
00222   AMDGPU_HSA_SYMBOL_ALLOCATION_GLOBAL_PROGRAM = 1,
00223   AMDGPU_HSA_SYMBOL_ALLOCATION_GLOBAL_AGENT = 2,
```

```
00224   AMDGPU_HSA_SYMBOL_ALLOCATION_READONLY_AGENT = 3,
00225   AMDGPU_HSA_SYMBOL_ALLOCATION_LAST,
00226 } amdgpu_hsa_symbol_allocation_t;
00227
00228 // ELF Symbol Allocation Enumeration Values.
00229 #define STA_AMDGPU_HSA_DEFAULT        AMDGPU_HSA_SYMBOL_ALLOCATION_DEFAULT
00230 #define STA_AMDGPU_HSA_GLOBAL_PROGRAM AMDGPU_HSA_SYMBOL_ALLOCATION_GLOBAL_PROGRAM
00231 #define STA_AMDGPU_HSA_GLOBAL_AGENT   AMDGPU_HSA_SYMBOL_ALLOCATION_GLOBAL_AGENT
00232 #define STA_AMDGPU_HSA_READONLY_AGENT AMDGPU_HSA_SYMBOL_ALLOCATION_READONLY_AGENT
00233
00234 typedef enum {
00235   AMDGPU_HSA_SYMBOL_FLAG_DEFAULT = 0,
00236   AMDGPU_HSA_SYMBOL_FLAG_CONST = 1,
00237   AMDGPU_HSA_SYMBOL_FLAG_LAST,
00238 } amdgpu_hsa_symbol_flag_t;
00239
00240 // ELF Symbol Flag Enumeration Values.
00241 #define STF_AMDGPU_HSA_CONST AMDGPU_HSA_SYMBOL_FLAG_CONST
00242
00243 // AMD GPU Relocation Type Enumeration Values.
00244 #define R_AMDGPU_NONE         0
00245 #define R_AMDGPU_32_LOW       1
00246 #define R_AMDGPU_32_HIGH      2
00247 #define R_AMDGPU_64           3
00248 #define R_AMDGPU_INIT_SAMPLER 4
00249 #define R_AMDGPU_INIT_IMAGE   5
00250 #define R_AMDGPU_RELATIVE64   13
00251
00252 // AMD GPU Note Type Enumeration Values.
00253 #define NT_AMD_HSA_CODE_OBJECT_VERSION 1
00254 #define NT_AMD_HSA_HSAIL               2
00255 #define NT_AMD_HSA_ISA_VERSION         3
00256 #define NT_AMD_HSA_PRODUCER            4
00257 #define NT_AMD_HSA_PRODUCER_OPTIONS    5
00258 #define NT_AMD_HSA_EXTENSION           6
00259 #define NT_AMD_HSA_ISA_NAME            11
00260 #define NT_AMD_HSA_HLDEBUG_DEBUG       101
00261 #define NT_AMD_HSA_HLDEBUG_TARGET      102
00262
00263 // AMD GPU Metadata Kind Enumeration Values.
00264 typedef uint16_t amdgpu_hsa_metadata_kind16_t;
00265 typedef enum {
00266   AMDGPU_HSA_METADATA_KIND_NONE = 0,
00267   AMDGPU_HSA_METADATA_KIND_INIT_SAMP = 1,
00268   AMDGPU_HSA_METADATA_KIND_INIT_ROIMG = 2,
00269   AMDGPU_HSA_METADATA_KIND_INIT_WOIMG = 3,
00270   AMDGPU_HSA_METADATA_KIND_INIT_RWIMG = 4
00271 } amdgpu_hsa_metadata_kind_t;
00272
00273 // AMD GPU Sampler Coordinate Normalization Enumeration Values.
00274 typedef uint8_t amdgpu_hsa_sampler_coord8_t;
00275 typedef enum {
00276   AMDGPU_HSA_SAMPLER_COORD_UNNORMALIZED = 0,
00277   AMDGPU_HSA_SAMPLER_COORD_NORMALIZED = 1
00278 } amdgpu_hsa_sampler_coord_t;
00279
00280 // AMD GPU Sampler Filter Enumeration Values.
00281 typedef uint8_t amdgpu_hsa_sampler_filter8_t;
00282 typedef enum {
00283   AMDGPU_HSA_SAMPLER_FILTER_NEAREST = 0,
00284   AMDGPU_HSA_SAMPLER_FILTER_LINEAR = 1
00285 } amdgpu_hsa_sampler_filter_t;
00286
00287 // AMD GPU Sampler Addressing Enumeration Values.
00288 typedef uint8_t amdgpu_hsa_sampler_addressing8_t;
00289 typedef enum {
00290   AMDGPU_HSA_SAMPLER_ADDRESSING_UNDEFINED = 0,
00291   AMDGPU_HSA_SAMPLER_ADDRESSING_CLAMP_TO_EDGE = 1,
00292   AMDGPU_HSA_SAMPLER_ADDRESSING_CLAMP_TO_BORDER = 2,
00293   AMDGPU_HSA_SAMPLER_ADDRESSING_REPEAT = 3,
00294   AMDGPU_HSA_SAMPLER_ADDRESSING_MIRRORED_REPEAT = 4
00295 } amdgpu_hsa_sampler_addressing_t;
00296
00297 // AMD GPU Sampler Descriptor.
00298 typedef struct amdgpu_hsa_sampler_descriptor_s {
00299   uint16_t size;
00300   amdgpu_hsa_metadata_kind16_t kind;
00301   amdgpu_hsa_sampler_coord8_t coord;
00302   amdgpu_hsa_sampler_filter8_t filter;
00303   amdgpu_hsa_sampler_addressing8_t addressing;
00304   uint8_t reserved1;
00305 } amdgpu_hsa_sampler_descriptor_t;
00306
00307 // AMD GPU Image Geometry Enumeration Values.
00308 typedef uint8_t amdgpu_hsa_image_geometry8_t;
00309 typedef enum {
00310   AMDGPU_HSA_IMAGE_GEOMETRY_1D = 0,
```

```
00311   AMDGPU_HSA_IMAGE_GEOMETRY_2D = 1,
00312   AMDGPU_HSA_IMAGE_GEOMETRY_3D = 2,
00313   AMDGPU_HSA_IMAGE_GEOMETRY_1DA = 3,
00314   AMDGPU_HSA_IMAGE_GEOMETRY_2DA = 4,
00315   AMDGPU_HSA_IMAGE_GEOMETRY_1DB = 5,
00316   AMDGPU_HSA_IMAGE_GEOMETRY_2DDEPTH = 6,
00317   AMDGPU_HSA_IMAGE_GEOMETRY_2DADEPTH = 7
00318 } amdgpu_hsa_image_geometry_t;
00319
00320 // AMD GPU Image Channel Order Enumeration Values.
00321 typedef uint8_t amdgpu_hsa_image_channel_order8_t;
00322 typedef enum {
00323   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_A = 0,
00324   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_R = 1,
00325   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_RX = 2,
00326   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_RG = 3,
00327   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_RGX = 4,
00328   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_RA = 5,
00329   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_RGB = 6,
00330   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_RGBX = 7,
00331   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_RGBA = 8,
00332   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_BGRA = 9,
00333   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_ARGB = 10,
00334   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_ABGR = 11,
00335   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_SRGB = 12,
00336   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_SRGBX = 13,
00337   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_SRGBA = 14,
00338   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_SBGRA = 15,
00339   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_INTENSITY = 16,
00340   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_LUMINANCE = 17,
00341   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_DEPTH = 18,
00342   AMDGPU_HSA_IMAGE_CHANNEL_ORDER_DEPTH_STENCIL = 19
00343 } amdgpu_hsa_image_channel_order_t;
00344
00345 // AMD GPU Image Channel Type Enumeration Values.
00346 typedef uint8_t amdgpu_hsa_image_channel_type8_t;
00347 typedef enum {
00348   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_SNORM_INT8 = 0,
00349   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_SNORM_INT16 = 1,
00350   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_UNORM_INT8 = 2,
00351   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_UNORM_INT16 = 3,
00352   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_UNORM_INT24 = 4,
00353   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_SHORT_555 = 5,
00354   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_SHORT_565 = 6,
00355   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_INT_101010 = 7,
00356   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_SIGNED_INT8 = 8,
00357   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_SIGNED_INT16 = 9,
00358   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_SIGNED_INT32 = 10,
00359   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_UNSIGNED_INT8 = 11,
00360   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_UNSIGNED_INT16 = 12,
00361   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_UNSIGNED_INT32 = 13,
00362   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_HALF_FLOAT = 14,
00363   AMDGPU_HSA_IMAGE_CHANNEL_TYPE_FLOAT = 15
00364 } amdgpu_hsa_image_channel_type_t;
00365
00366 // AMD GPU Image Descriptor.
00367 typedef struct amdgpu_hsa_image_descriptor_s {
00368   uint16_t size;
00369   amdgpu_hsa_metadata_kind16_t kind;
00370   amdgpu_hsa_image_geometry8_t geometry;
00371   amdgpu_hsa_image_channel_order8_t channel_order;
00372   amdgpu_hsa_image_channel_type8_t channel_type;
00373   uint8_t reserved1;
00374   uint64_t width;
00375   uint64_t height;
00376   uint64_t depth;
00377   uint64_t array;
00378 } amdgpu_hsa_image_descriptor_t;
00379
00380 typedef struct amdgpu_hsa_note_code_object_version_s {
00381   uint32_t major_version;
00382   uint32_t minor_version;
00383 } amdgpu_hsa_note_code_object_version_t;
00384
00385 typedef struct amdgpu_hsa_note_hsail_s {
00386   uint32_t hsail_major_version;
00387   uint32_t hsail_minor_version;
00388   uint8_t profile;
00389   uint8_t machine_model;
00390   uint8_t default_float_round;
00391 } amdgpu_hsa_note_hsail_t;
00392
00393 typedef struct amdgpu_hsa_note_isa_s {
00394   uint16_t vendor_name_size;
00395   uint16_t architecture_name_size;
00396   uint32_t major;
00397   uint32_t minor;
```

```
00398   uint32_t stepping;
00399   char vendor_and_architecture_name[1];
00400 } amdgpu_hsa_note_isa_t;
00401
00402 typedef struct amdgpu_hsa_note_producer_s {
00403   uint16_t producer_name_size;
00404   uint16_t reserved;
00405   uint32_t producer_major_version;
00406   uint32_t producer_minor_version;
00407   char producer_name[1];
00408 } amdgpu_hsa_note_producer_t;
00409
00410 typedef struct amdgpu_hsa_note_producer_options_s {
00411   uint16_t producer_options_size;
00412   char producer_options[1];
00413 } amdgpu_hsa_note_producer_options_t;
00414
00415 typedef enum {
00416   AMDGPU_HSA_RODATA_GLOBAL_PROGRAM = 0,
00417   AMDGPU_HSA_RODATA_GLOBAL_AGENT,
00418   AMDGPU_HSA_RODATA_READONLY_AGENT,
00419   AMDGPU_HSA_DATA_GLOBAL_PROGRAM,
00420   AMDGPU_HSA_DATA_GLOBAL_AGENT,
00421   AMDGPU_HSA_DATA_READONLY_AGENT,
00422   AMDGPU_HSA_BSS_GLOBAL_PROGRAM,
00423   AMDGPU_HSA_BSS_GLOBAL_AGENT,
00424   AMDGPU_HSA_BSS_READONLY_AGENT,
00425   AMDGPU_HSA_SECTION_LAST,
00426 } amdgpu_hsa_elf_section_t;
00427
00428 #endif // AMD_HSA_ELF_H
```

7.3 amd_hsa_kernel_code.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 #ifndef AMD_HSA_KERNEL_CODE_H
00044 #define AMD_HSA_KERNEL_CODE_H
00045
00046 #include "amd_hsa_common.h"
00047 #include "hsa.h"
00048
00049 // AMD Kernel Code Version Enumeration Values.
00050 typedef uint32_t amd_kernel_code_version32_t;
```

```
00051 enum amd_kernel_code_version_t {
00052   AMD_KERNEL_CODE_VERSION_MAJOR = 1,
00053   AMD_KERNEL_CODE_VERSION_MINOR = 1
00054 };
00055
00056 // AMD Machine Kind Enumeration Values.
00057 typedef uint16_t amd_machine_kind16_t;
00058 enum amd_machine_kind_t {
00059   AMD_MACHINE_KIND_UNDEFINED = 0,
00060   AMD_MACHINE_KIND_AMDGPU = 1
00061 };
00062
00063 // AMD Machine Version.
00064 typedef uint16_t amd_machine_version16_t;
00065
00066 // AMD Float Round Mode Enumeration Values.
00067 enum amd_float_round_mode_t {
00068   AMD_FLOAT_ROUND_MODE_NEAREST_EVEN = 0,
00069   AMD_FLOAT_ROUND_MODE_PLUS_INFINITY = 1,
00070   AMD_FLOAT_ROUND_MODE_MINUS_INFINITY = 2,
00071   AMD_FLOAT_ROUND_MODE_ZERO = 3
00072 };
00073
00074 // AMD Float Denorm Mode Enumeration Values.
00075 enum amd_float_denorm_mode_t {
00076   AMD_FLOAT_DENORM_MODE_FLUSH_SOURCE_OUTPUT = 0,
00077   AMD_FLOAT_DENORM_MODE_FLUSH_OUTPUT = 1,
00078   AMD_FLOAT_DENORM_MODE_FLUSH_SOURCE = 2,
00079   AMD_FLOAT_DENORM_MODE_NO_FLUSH = 3
00080 };
00081
00082 // AMD Compute Program Resource Register One.
00083 typedef uint32_t amd_compute_pgm_rsrc_one32_t;
00084 enum amd_compute_pgm_rsrc_one_t {
00085   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_GRANULATED_WORKITEM_VGPR_COUNT, 0, 6),
00086   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_GRANULATED_WAVEFRONT_SGPR_COUNT, 6, 4),
00087   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_PRIORITY, 10, 2),
00088   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_FLOAT_ROUND_MODE_32, 12, 2),
00089   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_FLOAT_ROUND_MODE_16_64, 14, 2),
00090   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_FLOAT_DENORM_MODE_32, 16, 2),
00091   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_FLOAT_DENORM_MODE_16_64, 18, 2),
00092   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_PRIV, 20, 1),
00093   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_ENABLE_DX10_CLAMP, 21, 1),
00094   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_DEBUG_MODE, 22, 1),
00095   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_ENABLE_IEEE_MODE, 23, 1),
00096   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_BULKY, 24, 1),
00097   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_CDBG_USER, 25, 1),
00098   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_ONE_RESERVED1, 26, 6)
00099 };
00100
00101 // AMD System VGPR Workitem ID Enumeration Values.
00102 enum amd_system_vgpr_workitem_id_t {
00103   AMD_SYSTEM_VGPR_WORKITEM_ID_X = 0,
00104   AMD_SYSTEM_VGPR_WORKITEM_ID_X_Y = 1,
00105   AMD_SYSTEM_VGPR_WORKITEM_ID_X_Y_Z = 2,
00106   AMD_SYSTEM_VGPR_WORKITEM_ID_UNDEFINED = 3
00107 };
00108
00109 // AMD Compute Program Resource Register Two.
00110 typedef uint32_t amd_compute_pgm_rsrc_two32_t;
00111 enum amd_compute_pgm_rsrc_two_t {
00112
       AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_SGPR_PRIVATE_SEGMENT_WAVE_BYTE_OFFSET,
       0, 1),
00113   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_USER_SGPR_COUNT, 1, 5),
00114   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_TRAP_HANDLER, 6, 1),
00115   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_SGPR_WORKGROUP_ID_X, 7, 1),
00116   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_SGPR_WORKGROUP_ID_Y, 8, 1),
00117   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_SGPR_WORKGROUP_ID_Z, 9, 1),
00118   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_SGPR_WORKGROUP_INFO, 10, 1),
00119   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_VGPR_WORKITEM_ID, 11, 2),
00120   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_EXCEPTION_ADDRESS_WATCH, 13, 1),
00121   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_EXCEPTION_MEMORY_VIOLATION, 14, 1),
00122   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_GRANULATED_LDS_SIZE, 15, 9),
00123
       AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_EXCEPTION_IEEE_754_FP_INVALID_OPERATION,
       24, 1),
00124   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_EXCEPTION_FP_DENORMAL_SOURCE, 25,
       1),
00125
       AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_EXCEPTION_IEEE_754_FP_DIVISION_BY_ZERO,
       26, 1),
00126   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_EXCEPTION_IEEE_754_FP_OVERFLOW, 27,
       1),
00127   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_EXCEPTION_IEEE_754_FP_UNDERFLOW,
       28, 1),
00128   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_EXCEPTION_IEEE_754_FP_INEXACT, 29,
```

```
      1),
00129   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_ENABLE_EXCEPTION_INT_DIVISION_BY_ZERO, 30,
      1),
00130   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_COMPUTE_PGM_RSRC_TWO_RESERVED1, 31, 1)
00131 };
00132
00133 // AMD Element Byte Size Enumeration Values.
00134 enum amd_element_byte_size_t {
00135   AMD_ELEMENT_BYTE_SIZE_2 = 0,
00136   AMD_ELEMENT_BYTE_SIZE_4 = 1,
00137   AMD_ELEMENT_BYTE_SIZE_8 = 2,
00138   AMD_ELEMENT_BYTE_SIZE_16 = 3
00139 };
00140
00141 // AMD Kernel Code Properties.
00142 typedef uint32_t amd_kernel_code_properties32_t;
00143 enum amd_kernel_code_properties_t {
00144   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_SGPR_PRIVATE_SEGMENT_BUFFER, 0,
      1),
00145   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_SGPR_DISPATCH_PTR, 1, 1),
00146   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_SGPR_QUEUE_PTR, 2, 1),
00147   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_SGPR_KERNARG_SEGMENT_PTR, 3, 1),
00148   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_SGPR_DISPATCH_ID, 4, 1),
00149   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_SGPR_FLAT_SCRATCH_INIT, 5, 1),
00150   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_SGPR_PRIVATE_SEGMENT_SIZE, 6, 1),
00151   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_SGPR_GRID_WORKGROUP_COUNT_X, 7,
      1),
00152   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_SGPR_GRID_WORKGROUP_COUNT_Y, 8,
      1),
00153   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_SGPR_GRID_WORKGROUP_COUNT_Z, 9,
      1),
00154   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_RESERVED1, 10, 6),
00155   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_ENABLE_ORDERED_APPEND_GDS, 16, 1),
00156   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_PRIVATE_ELEMENT_SIZE, 17, 2),
00157   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_IS_PTR64, 19, 1),
00158   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_IS_DYNAMIC_CALLSTACK, 20, 1),
00159   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_IS_DEBUG_ENABLED, 21, 1),
00160   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_IS_XNACK_ENABLED, 22, 1),
00161   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_KERNEL_CODE_PROPERTIES_RESERVED2, 23, 9)
00162 };
00163
00164 // AMD Power Of Two Enumeration Values.
00165 typedef uint8_t amd_powertwo8_t;
00166 enum amd_powertwo_t {
00167   AMD_POWERTWO_1 = 0,
00168   AMD_POWERTWO_2 = 1,
00169   AMD_POWERTWO_4 = 2,
00170   AMD_POWERTWO_8 = 3,
00171   AMD_POWERTWO_16 = 4,
00172   AMD_POWERTWO_32 = 5,
00173   AMD_POWERTWO_64 = 6,
00174   AMD_POWERTWO_128 = 7,
00175   AMD_POWERTWO_256 = 8
00176 };
00177
00178 // AMD Enabled Control Directive Enumeration Values.
00179 typedef uint64_t amd_enabled_control_directive64_t;
00180 enum amd_enabled_control_directive_t {
00181   AMD_ENABLED_CONTROL_DIRECTIVE_ENABLE_BREAK_EXCEPTIONS = 1,
00182   AMD_ENABLED_CONTROL_DIRECTIVE_ENABLE_DETECT_EXCEPTIONS = 2,
00183   AMD_ENABLED_CONTROL_DIRECTIVE_MAX_DYNAMIC_GROUP_SIZE = 4,
00184   AMD_ENABLED_CONTROL_DIRECTIVE_MAX_FLAT_GRID_SIZE = 8,
00185   AMD_ENABLED_CONTROL_DIRECTIVE_MAX_FLAT_WORKGROUP_SIZE = 16,
00186   AMD_ENABLED_CONTROL_DIRECTIVE_REQUIRED_DIM = 32,
00187   AMD_ENABLED_CONTROL_DIRECTIVE_REQUIRED_GRID_SIZE = 64,
00188   AMD_ENABLED_CONTROL_DIRECTIVE_REQUIRED_WORKGROUP_SIZE = 128,
00189   AMD_ENABLED_CONTROL_DIRECTIVE_REQUIRE_NO_PARTIAL_WORKGROUPS = 256
00190 };
00191
00192 // AMD Exception Kind Enumeration Values.
00193 typedef uint16_t amd_exception_kind16_t;
00194 enum amd_exception_kind_t {
00195   AMD_EXCEPTION_KIND_INVALID_OPERATION = 1,
00196   AMD_EXCEPTION_KIND_DIVISION_BY_ZERO = 2,
00197   AMD_EXCEPTION_KIND_OVERFLOW = 4,
00198   AMD_EXCEPTION_KIND_UNDERFLOW = 8,
00199   AMD_EXCEPTION_KIND_INEXACT = 16
00200 };
00201
00202 // AMD Control Directives.
00203 #define AMD_CONTROL_DIRECTIVES_ALIGN_BYTES 64
00204 #define AMD_CONTROL_DIRECTIVES_ALIGN __ALIGNED__(AMD_CONTROL_DIRECTIVES_ALIGN_BYTES)
00205 typedef AMD_CONTROL_DIRECTIVES_ALIGN struct amd_control_directives_s {
00206   amd_enabled_control_directive64_t enabled_control_directives;
00207   uint16_t enable_break_exceptions;
00208   uint16_t enable_detect_exceptions;
00209   uint32_t max_dynamic_group_size;
```

```
00210   uint64_t max_flat_grid_size;
00211   uint32_t max_flat_workgroup_size;
00212   uint8_t required_dim;
00213   uint8_t reserved1[3];
00214   uint64_t required_grid_size[3];
00215   uint32_t required_workgroup_size[3];
00216   uint8_t reserved2[60];
00217 } amd_control_directives_t;
00218
00219 // AMD Kernel Code.
00220 #define AMD_ISA_ALIGN_BYTES 256
00221 #define AMD_KERNEL_CODE_ALIGN_BYTES 64
00222 #define AMD_KERNEL_CODE_ALIGN __ALIGNED__(AMD_KERNEL_CODE_ALIGN_BYTES)
00223 typedef AMD_KERNEL_CODE_ALIGN struct amd_kernel_code_s {
00224   amd_kernel_code_version32_t amd_kernel_code_version_major;
00225   amd_kernel_code_version32_t amd_kernel_code_version_minor;
00226   amd_machine_kind16_t amd_machine_kind;
00227   amd_machine_version16_t amd_machine_version_major;
00228   amd_machine_version16_t amd_machine_version_minor;
00229   amd_machine_version16_t amd_machine_version_stepping;
00230   int64_t kernel_code_entry_byte_offset;
00231   int64_t kernel_code_prefetch_byte_offset;
00232   uint64_t kernel_code_prefetch_byte_size;
00233   uint64_t max_scratch_backing_memory_byte_size;
00234   amd_compute_pgm_rsrc_one32_t compute_pgm_rsrc1;
00235   amd_compute_pgm_rsrc_two32_t compute_pgm_rsrc2;
00236   amd_kernel_code_properties32_t kernel_code_properties;
00237   uint32_t workitem_private_segment_byte_size;
00238   uint32_t workgroup_group_segment_byte_size;
00239   uint32_t gds_segment_byte_size;
00240   uint64_t kernarg_segment_byte_size;
00241   uint32_t workgroup_fbarrier_count;
00242   uint16_t wavefront_sgpr_count;
00243   uint16_t workitem_vgpr_count;
00244   uint16_t reserved_vgpr_first;
00245   uint16_t reserved_vgpr_count;
00246   uint16_t reserved_sgpr_first;
00247   uint16_t reserved_sgpr_count;
00248   uint16_t debug_wavefront_private_segment_offset_sgpr;
00249   uint16_t debug_private_segment_buffer_sgpr;
00250   amd_powertwo8_t kernarg_segment_alignment;
00251   amd_powertwo8_t group_segment_alignment;
00252   amd_powertwo8_t private_segment_alignment;
00253   amd_powertwo8_t wavefront_size;
00254   int32_t call_convention;
00255   uint8_t reserved1[12];
00256   uint64_t runtime_loader_kernel_symbol;
00257   amd_control_directives_t control_directives;
00258 } amd_kernel_code_t;
00259
00260 // TODO: this struct should be completely gone once debugger designs/implements
00261 // Debugger APIs.
00262 typedef struct amd_runtime_loader_debug_info_s {
00263   const void* elf_raw;
00264   size_t elf_size;
00265   const char *kernel_name;
00266   const void *owning_segment;
00267 } amd_runtime_loader_debug_info_t;
00268
00269 #endif // AMD_HSA_KERNEL_CODE_H
```

7.4 amd_hsa_queue.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
```

```
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 #ifndef AMD_HSA_QUEUE_H
00044 #define AMD_HSA_QUEUE_H
00045
00046 #include "amd_hsa_common.h"
00047 #include "hsa.h"
00048
00049 // AMD Queue Properties.
00050 typedef uint32_t amd_queue_properties32_t;
00051 enum amd_queue_properties_t {
00052   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_QUEUE_PROPERTIES_ENABLE_TRAP_HANDLER, 0, 1),
00053   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_QUEUE_PROPERTIES_IS_PTR64, 1, 1),
00054   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_QUEUE_PROPERTIES_ENABLE_TRAP_HANDLER_DEBUG_SGPRS, 2, 1),
00055   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_QUEUE_PROPERTIES_ENABLE_PROFILING, 3, 1),
00056   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_QUEUE_PROPERTIES_USE_SCRATCH_ONCE, 4, 1),
00057   AMD_HSA_BITS_CREATE_ENUM_ENTRIES(AMD_QUEUE_PROPERTIES_RESERVED1, 5, 27)
00058 };
00059
00060 // AMD Queue.
00061 #define AMD_QUEUE_ALIGN_BYTES 64
00062 #define AMD_QUEUE_ALIGN __ALIGNED__(AMD_QUEUE_ALIGN_BYTES)
00063 typedef struct AMD_QUEUE_ALIGN amd_queue_s {
00064   hsa_queue_t hsa_queue;
00065   uint32_t reserved1[4];
00066   volatile uint64_t write_dispatch_id;
00067   uint32_t group_segment_aperture_base_hi;
00068   uint32_t private_segment_aperture_base_hi;
00069   uint32_t max_cu_id;
00070   uint32_t max_wave_id;
00071   volatile uint64_t max_legacy_doorbell_dispatch_id_plus_1;
00072   volatile uint32_t legacy_doorbell_lock;
00073   uint32_t reserved2[9];
00074   volatile uint64_t read_dispatch_id;
00075   uint32_t read_dispatch_id_field_base_byte_offset;
00076   uint32_t compute_tmpring_size;
00077   uint32_t scratch_resource_descriptor[4];
00078   uint64_t scratch_backing_memory_location;
00079   uint64_t scratch_backing_memory_byte_size;
00080   uint32_t scratch_wave64_lane_byte_size;
00081   amd_queue_properties32_t queue_properties;
00082   uint32_t reserved3[2];
00083   hsa_signal_t queue_inactive_signal;
00084   uint32_t reserved4[14];
00085 } amd_queue_t;
00086
00087 #endif // AMD_HSA_QUEUE_H
```

7.5 amd_hsa_signal.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
```

```
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 #ifndef AMD_HSA_SIGNAL_H
00044 #define AMD_HSA_SIGNAL_H
00045
00046 #include "amd_hsa_common.h"
00047 #include "amd_hsa_queue.h"
00048
00049 // AMD Signal Kind Enumeration Values.
00050 typedef int64_t amd_signal_kind64_t;
00051 enum amd_signal_kind_t {
00052   AMD_SIGNAL_KIND_INVALID = 0,
00053   AMD_SIGNAL_KIND_USER = 1,
00054   AMD_SIGNAL_KIND_DOORBELL = -1,
00055   AMD_SIGNAL_KIND_LEGACY_DOORBELL = -2
00056 };
00057
00058 // AMD Signal.
00059 #define AMD_SIGNAL_ALIGN_BYTES 64
00060 #define AMD_SIGNAL_ALIGN __ALIGNED__(AMD_SIGNAL_ALIGN_BYTES)
00061 typedef struct AMD_SIGNAL_ALIGN amd_signal_s {
00062   amd_signal_kind64_t kind;
00063   union {
00064     volatile int64_t value;
00065     volatile uint32_t* legacy_hardware_doorbell_ptr;
00066     volatile uint64_t* hardware_doorbell_ptr;
00067   };
00068   uint64_t event_mailbox_ptr;
00069   uint32_t event_id;
00070   uint32_t reserved1;
00071   uint64_t start_ts;
00072   uint64_t end_ts;
00073   union {
00074     amd_queue_t* queue_ptr;
00075     uint64_t reserved2;
00076   };
00077   uint32_t reserved3[2];
00078 } amd_signal_t;
00079
00080 #endif // AMD_HSA_SIGNAL_H
```

7.6 Brig.h

```
00001 // University of Illinois/NCSA
00002 // Open Source License
00003 //
00004 // Copyright (c) 2013-2015, Advanced Micro Devices, Inc.
00005 // All rights reserved.
00006 //
00007 // Developed by:
00008 //
00009 //     HSA Team
00010 //
00011 //     Advanced Micro Devices, Inc
00012 //
00013 //     www.amd.com
00014 //
00015 // Permission is hereby granted, free of charge, to any person obtaining a copy of
```

```
00016 // this software and associated documentation files (the "Software"), to deal with
00017 // the Software without restriction, including without limitation the rights to
00018 // use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
00019 // of the Software, and to permit persons to whom the Software is furnished to do
00020 // so, subject to the following conditions:
00021 //
00022 //     * Redistributions of source code must retain the above copyright notice,
00023 //       this list of conditions and the following disclaimers.
00024 //
00025 //     * Redistributions in binary form must reproduce the above copyright notice,
00026 //       this list of conditions and the following disclaimers in the
00027 //       documentation and/or other materials provided with the distribution.
00028 //
00029 //     * Neither the names of the LLVM Team, University of Illinois at
00030 //       Urbana-Champaign, nor the names of its contributors may be used to
00031 //       endorse or promote products derived from this Software without specific
00032 //       prior written permission.
00033 //
00034 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00035 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
00036 // FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL THE
00037 // CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00038 // LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00039 // OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS WITH THE
00040 // SOFTWARE.
00041
00042 #ifndef INCLUDED_BRIG_H
00043 #define INCLUDED_BRIG_H
00044
00045 #include <stddef.h>   /* size_t */
00046 #include <stdint.h>   /* uintXX_t */
00047
00048 #ifdef __cplusplus
00049 extern "C" {
00050 #endif  /* __cplusplus */
00051
00052 /*============================================================================*/
00053 /* ============================================================================*/
00054 /* ============================================================================*/
00055 /* ============================================================================*/
00056
00057 typedef uint32_t BrigCodeOffset32_t;
00058 typedef uint32_t BrigOperandOffset32_t;
00059 typedef uint32_t BrigDataOffset32_t;
00060
00061 typedef BrigDataOffset32_t BrigDataOffsetCodeList32_t;
00062 typedef BrigDataOffset32_t BrigDataOffsetOperandList32_t;
00063 typedef BrigDataOffset32_t BrigDataOffsetString32_t;
00064
00065 typedef uint32_t BrigVersion32_t;
00066 enum BrigVersion {
00067     BRIG_VERSION_HSAIL_MAJOR = 1,
00068     BRIG_VERSION_HSAIL_MINOR = 0,
00069     BRIG_VERSION_BRIG_MAJOR  = 1,
00070     BRIG_VERSION_BRIG_MINOR  = 0
00071 };
00072
00073 typedef uint16_t BrigKind16_t;
00074 enum BrigKind {
00075     BRIG_KIND_NONE = 0x0000,
00076
00077     BRIG_KIND_DIRECTIVE_BEGIN = 0x1000,
00078         BRIG_KIND_DIRECTIVE_ARG_BLOCK_END = 0x1000,
00079         BRIG_KIND_DIRECTIVE_ARG_BLOCK_START = 0x1001,
00080         BRIG_KIND_DIRECTIVE_COMMENT = 0x1002,
00081         BRIG_KIND_DIRECTIVE_CONTROL = 0x1003,
00082         BRIG_KIND_DIRECTIVE_EXTENSION = 0x1004,
00083         BRIG_KIND_DIRECTIVE_FBARRIER = 0x1005,
00084         BRIG_KIND_DIRECTIVE_FUNCTION = 0x1006,
00085         BRIG_KIND_DIRECTIVE_INDIRECT_FUNCTION = 0x1007,
00086         BRIG_KIND_DIRECTIVE_KERNEL = 0x1008,
00087         BRIG_KIND_DIRECTIVE_LABEL = 0x1009,
00088         BRIG_KIND_DIRECTIVE_LOC = 0x100a,
00089         BRIG_KIND_DIRECTIVE_MODULE = 0x100b,
00090         BRIG_KIND_DIRECTIVE_PRAGMA = 0x100c,
00091         BRIG_KIND_DIRECTIVE_SIGNATURE = 0x100d,
00092         BRIG_KIND_DIRECTIVE_VARIABLE = 0x100e,
00093     BRIG_KIND_DIRECTIVE_END = 0x100f,
00094
00095     BRIG_KIND_INST_BEGIN = 0x2000,
00096         BRIG_KIND_INST_ADDR = 0x2000,
00097         BRIG_KIND_INST_ATOMIC = 0x2001,
00098         BRIG_KIND_INST_BASIC = 0x2002,
00099         BRIG_KIND_INST_BR = 0x2003,
00100         BRIG_KIND_INST_CMP = 0x2004,
00101         BRIG_KIND_INST_CVT = 0x2005,
00102         BRIG_KIND_INST_IMAGE = 0x2006,
```

```
00103         BRIG_KIND_INST_LANE = 0x2007,
00104         BRIG_KIND_INST_MEM = 0x2008,
00105         BRIG_KIND_INST_MEM_FENCE = 0x2009,
00106         BRIG_KIND_INST_MOD = 0x200a,
00107         BRIG_KIND_INST_QUERY_IMAGE = 0x200b,
00108         BRIG_KIND_INST_QUERY_SAMPLER = 0x200c,
00109         BRIG_KIND_INST_QUEUE = 0x200d,
00110         BRIG_KIND_INST_SEG = 0x200e,
00111         BRIG_KIND_INST_SEG_CVT = 0x200f,
00112         BRIG_KIND_INST_SIGNAL = 0x2010,
00113         BRIG_KIND_INST_SOURCE_TYPE = 0x2011,
00114     BRIG_KIND_INST_END = 0x2012,
00115
00116     BRIG_KIND_OPERAND_BEGIN = 0x3000,
00117         BRIG_KIND_OPERAND_ADDRESS = 0x3000,
00118         BRIG_KIND_OPERAND_ALIGN = 0x3001,
00119         BRIG_KIND_OPERAND_CODE_LIST = 0x3002,
00120         BRIG_KIND_OPERAND_CODE_REF = 0x3003,
00121         BRIG_KIND_OPERAND_CONSTANT_BYTES = 0x3004,
00122         BRIG_KIND_OPERAND_RESERVED = 0x3005,
00123         BRIG_KIND_OPERAND_CONSTANT_IMAGE = 0x3006,
00124         BRIG_KIND_OPERAND_CONSTANT_OPERAND_LIST = 0x3007,
00125         BRIG_KIND_OPERAND_CONSTANT_SAMPLER = 0x3008,
00126         BRIG_KIND_OPERAND_OPERAND_LIST = 0x3009,
00127         BRIG_KIND_OPERAND_REGISTER = 0x300a,
00128         BRIG_KIND_OPERAND_STRING = 0x300b,
00129         BRIG_KIND_OPERAND_WAVESIZE = 0x300c,
00130     BRIG_KIND_OPERAND_END = 0x300d
00131 };
00132
00133 typedef uint8_t BrigAlignment8_t;
00134 enum BrigAlignment {
00135     BRIG_ALIGNMENT_NONE = 0,
00136     BRIG_ALIGNMENT_1 = 1,
00137     BRIG_ALIGNMENT_2 = 2,
00138     BRIG_ALIGNMENT_4 = 3,
00139     BRIG_ALIGNMENT_8 = 4,
00140     BRIG_ALIGNMENT_16 = 5,
00141     BRIG_ALIGNMENT_32 = 6,
00142     BRIG_ALIGNMENT_64 = 7,
00143     BRIG_ALIGNMENT_128 = 8,
00144     BRIG_ALIGNMENT_256 = 9,
00145     BRIG_ALIGNMENT_MAX = BRIG_ALIGNMENT_256
00146 };
00147
00148 typedef uint8_t BrigAllocation8_t;
00149 enum BrigAllocation {
00150     BRIG_ALLOCATION_NONE = 0,
00151     BRIG_ALLOCATION_PROGRAM = 1,
00152     BRIG_ALLOCATION_AGENT = 2,
00153     BRIG_ALLOCATION_AUTOMATIC = 3
00154 };
00155
00156 typedef uint8_t BrigAluModifier8_t;
00157 enum BrigAluModifierMask {
00158     BRIG_ALU_FTZ = 1
00159 };
00160
00161 typedef uint8_t BrigAtomicOperation8_t;
00162 enum BrigAtomicOperation {
00163     BRIG_ATOMIC_ADD = 0,
00164     BRIG_ATOMIC_AND = 1,
00165     BRIG_ATOMIC_CAS = 2,
00166     BRIG_ATOMIC_EXCH = 3,
00167     BRIG_ATOMIC_LD = 4,
00168     BRIG_ATOMIC_MAX = 5,
00169     BRIG_ATOMIC_MIN = 6,
00170     BRIG_ATOMIC_OR = 7,
00171     BRIG_ATOMIC_ST = 8,
00172     BRIG_ATOMIC_SUB = 9,
00173     BRIG_ATOMIC_WRAPDEC = 10,
00174     BRIG_ATOMIC_WRAPINC = 11,
00175     BRIG_ATOMIC_XOR = 12,
00176     BRIG_ATOMIC_WAIT_EQ = 13,
00177     BRIG_ATOMIC_WAIT_NE = 14,
00178     BRIG_ATOMIC_WAIT_LT = 15,
00179     BRIG_ATOMIC_WAIT_GTE = 16,
00180     BRIG_ATOMIC_WAITTIMEOUT_EQ = 17,
00181     BRIG_ATOMIC_WAITTIMEOUT_NE = 18,
00182     BRIG_ATOMIC_WAITTIMEOUT_LT = 19,
00183     BRIG_ATOMIC_WAITTIMEOUT_GTE = 20
00184 };
00185
00186 typedef uint8_t BrigCompareOperation8_t;
00187 enum BrigCompareOperation {
00188     BRIG_COMPARE_EQ = 0,
00189     BRIG_COMPARE_NE = 1,
```

```
00190     BRIG_COMPARE_LT = 2,
00191     BRIG_COMPARE_LE = 3,
00192     BRIG_COMPARE_GT = 4,
00193     BRIG_COMPARE_GE = 5,
00194     BRIG_COMPARE_EQU = 6,
00195     BRIG_COMPARE_NEU = 7,
00196     BRIG_COMPARE_LTU = 8,
00197     BRIG_COMPARE_LEU = 9,
00198     BRIG_COMPARE_GTU = 10,
00199     BRIG_COMPARE_GEU = 11,
00200     BRIG_COMPARE_NUM = 12,
00201     BRIG_COMPARE_NAN = 13,
00202     BRIG_COMPARE_SEQ = 14,
00203     BRIG_COMPARE_SNE = 15,
00204     BRIG_COMPARE_SLT = 16,
00205     BRIG_COMPARE_SLE = 17,
00206     BRIG_COMPARE_SGT = 18,
00207     BRIG_COMPARE_SGE = 19,
00208     BRIG_COMPARE_SGEU = 20,
00209     BRIG_COMPARE_SEQU = 21,
00210     BRIG_COMPARE_SNEU = 22,
00211     BRIG_COMPARE_SLTU = 23,
00212     BRIG_COMPARE_SLEU = 24,
00213     BRIG_COMPARE_SNUM = 25,
00214     BRIG_COMPARE_SNAN = 26,
00215     BRIG_COMPARE_SGTU = 27
00216 };
00217
00218 typedef uint16_t BrigControlDirective16_t;
00219 enum BrigControlDirective {
00220     BRIG_CONTROL_NONE = 0,
00221     BRIG_CONTROL_ENABLEBREAKEXCEPTIONS = 1,
00222     BRIG_CONTROL_ENABLEDETECTEXCEPTIONS = 2,
00223     BRIG_CONTROL_MAXDYNAMICGROUPSIZE = 3,
00224     BRIG_CONTROL_MAXFLATGRIDSIZE = 4,
00225     BRIG_CONTROL_MAXFLATWORKGROUPSIZE = 5,
00226     BRIG_CONTROL_REQUIREDDIM = 6,
00227     BRIG_CONTROL_REQUIREDGRIDSIZE = 7,
00228     BRIG_CONTROL_REQUIREDWORKGROUPSIZE = 8,
00229     BRIG_CONTROL_REQUIRENOPARTIALWORKGROUPS = 9
00230 };
00231
00232 typedef uint8_t BrigExecutableModifier8_t;
00233 enum BrigExecutableModifierMask {
00234     BRIG_EXECUTABLE_DEFINITION = 1
00235 };
00236
00237 typedef uint8_t BrigImageChannelOrder8_t;
00238 enum BrigImageChannelOrder {
00239     BRIG_CHANNEL_ORDER_A = 0,
00240     BRIG_CHANNEL_ORDER_R = 1,
00241     BRIG_CHANNEL_ORDER_RX = 2,
00242     BRIG_CHANNEL_ORDER_RG = 3,
00243     BRIG_CHANNEL_ORDER_RGX = 4,
00244     BRIG_CHANNEL_ORDER_RA = 5,
00245     BRIG_CHANNEL_ORDER_RGB = 6,
00246     BRIG_CHANNEL_ORDER_RGBX = 7,
00247     BRIG_CHANNEL_ORDER_RGBA = 8,
00248     BRIG_CHANNEL_ORDER_BGRA = 9,
00249     BRIG_CHANNEL_ORDER_ARGB = 10,
00250     BRIG_CHANNEL_ORDER_ABGR = 11,
00251     BRIG_CHANNEL_ORDER_SRGB = 12,
00252     BRIG_CHANNEL_ORDER_SRGBX = 13,
00253     BRIG_CHANNEL_ORDER_SRGBA = 14,
00254     BRIG_CHANNEL_ORDER_SBGRA = 15,
00255     BRIG_CHANNEL_ORDER_INTENSITY = 16,
00256     BRIG_CHANNEL_ORDER_LUMINANCE = 17,
00257     BRIG_CHANNEL_ORDER_DEPTH = 18,
00258     BRIG_CHANNEL_ORDER_DEPTH_STENCIL = 19,
00259
00260     BRIG_CHANNEL_ORDER_FIRST_USER_DEFINED = 128
00261 };
00262
00263 typedef uint8_t BrigImageChannelType8_t;
00264 enum BrigImageChannelType {
00265     BRIG_CHANNEL_TYPE_SNORM_INT8 = 0,
00266     BRIG_CHANNEL_TYPE_SNORM_INT16 = 1,
00267     BRIG_CHANNEL_TYPE_UNORM_INT8 = 2,
00268     BRIG_CHANNEL_TYPE_UNORM_INT16 = 3,
00269     BRIG_CHANNEL_TYPE_UNORM_INT24 = 4,
00270     BRIG_CHANNEL_TYPE_UNORM_SHORT_555 = 5,
00271     BRIG_CHANNEL_TYPE_UNORM_SHORT_565 = 6,
00272     BRIG_CHANNEL_TYPE_UNORM_INT_101010 = 7,
00273     BRIG_CHANNEL_TYPE_SIGNED_INT8 = 8,
00274     BRIG_CHANNEL_TYPE_SIGNED_INT16 = 9,
00275     BRIG_CHANNEL_TYPE_SIGNED_INT32 = 10,
00276     BRIG_CHANNEL_TYPE_UNSIGNED_INT8 = 11,
```

```
00277     BRIG_CHANNEL_TYPE_UNSIGNED_INT16 = 12,
00278     BRIG_CHANNEL_TYPE_UNSIGNED_INT32 = 13,
00279     BRIG_CHANNEL_TYPE_HALF_FLOAT = 14,
00280     BRIG_CHANNEL_TYPE_FLOAT = 15,
00281
00282     BRIG_CHANNEL_TYPE_FIRST_USER_DEFINED = 128
00283 };
00284
00285 typedef uint8_t BrigImageGeometry8_t;
00286 enum BrigImageGeometry {
00287     BRIG_GEOMETRY_1D = 0,
00288     BRIG_GEOMETRY_2D = 1,
00289     BRIG_GEOMETRY_3D = 2,
00290     BRIG_GEOMETRY_1DA = 3,
00291     BRIG_GEOMETRY_2DA = 4,
00292     BRIG_GEOMETRY_1DB = 5,
00293     BRIG_GEOMETRY_2DDEPTH = 6,
00294     BRIG_GEOMETRY_2DADEPTH = 7,
00295
00296     BRIG_GEOMETRY_FIRST_USER_DEFINED = 128
00297 };
00298
00299 typedef uint8_t BrigImageQuery8_t;
00300 enum BrigImageQuery {
00301     BRIG_IMAGE_QUERY_WIDTH = 0,
00302     BRIG_IMAGE_QUERY_HEIGHT = 1,
00303     BRIG_IMAGE_QUERY_DEPTH = 2,
00304     BRIG_IMAGE_QUERY_ARRAY = 3,
00305     BRIG_IMAGE_QUERY_CHANNELORDER = 4,
00306     BRIG_IMAGE_QUERY_CHANNELTYPE = 5,
00307
00308     BRIG_IMAGE_QUERY_FIRST_USER_DEFINED = 6
00309 };
00310
00311 typedef uint8_t BrigLinkage8_t;
00312 enum BrigLinkage {
00313     BRIG_LINKAGE_NONE = 0,
00314     BRIG_LINKAGE_PROGRAM = 1,
00315     BRIG_LINKAGE_MODULE = 2,
00316     BRIG_LINKAGE_FUNCTION = 3,
00317     BRIG_LINKAGE_ARG = 4
00318 };
00319
00320 typedef uint8_t BrigMachineModel8_t;
00321 enum BrigMachineModel {
00322     BRIG_MACHINE_SMALL = 0,
00323     BRIG_MACHINE_LARGE = 1,
00324 };
00325
00326 typedef uint8_t BrigMemoryModifier8_t;
00327 enum BrigMemoryModifierMask {
00328     BRIG_MEMORY_CONST = 1
00329 };
00330
00331 typedef uint8_t BrigMemoryOrder8_t;
00332 enum BrigMemoryOrder {
00333     BRIG_MEMORY_ORDER_NONE = 0,
00334     BRIG_MEMORY_ORDER_RELAXED = 1,
00335     BRIG_MEMORY_ORDER_SC_ACQUIRE = 2,
00336     BRIG_MEMORY_ORDER_SC_RELEASE = 3,
00337     BRIG_MEMORY_ORDER_SC_ACQUIRE_RELEASE = 4,
00338 };
00339
00340 typedef uint8_t BrigMemoryScope8_t;
00341 enum BrigMemoryScope {
00342     BRIG_MEMORY_SCOPE_NONE = 0,
00343     BRIG_MEMORY_SCOPE_WORKITEM = 1,
00344     BRIG_MEMORY_SCOPE_WAVEFRONT = 2,
00345     BRIG_MEMORY_SCOPE_WORKGROUP = 3,
00346     BRIG_MEMORY_SCOPE_AGENT = 4,
00347     BRIG_MEMORY_SCOPE_SYSTEM = 5,
00348 };
00349
00350 typedef uint16_t BrigOpcode16_t;
00351 enum BrigOpcode {
00352     BRIG_OPCODE_NOP = 0,
00353     BRIG_OPCODE_ABS = 1,
00354     BRIG_OPCODE_ADD = 2,
00355     BRIG_OPCODE_BORROW = 3,
00356     BRIG_OPCODE_CARRY = 4,
00357     BRIG_OPCODE_CEIL = 5,
00358     BRIG_OPCODE_COPYSIGN = 6,
00359     BRIG_OPCODE_DIV = 7,
00360     BRIG_OPCODE_FLOOR = 8,
00361     BRIG_OPCODE_FMA = 9,
00362     BRIG_OPCODE_FRACT = 10,
00363     BRIG_OPCODE_MAD = 11,
```

```
00364     BRIG_OPCODE_MAX = 12,
00365     BRIG_OPCODE_MIN = 13,
00366     BRIG_OPCODE_MUL = 14,
00367     BRIG_OPCODE_MULHI = 15,
00368     BRIG_OPCODE_NEG = 16,
00369     BRIG_OPCODE_REM = 17,
00370     BRIG_OPCODE_RINT = 18,
00371     BRIG_OPCODE_SQRT = 19,
00372     BRIG_OPCODE_SUB = 20,
00373     BRIG_OPCODE_TRUNC = 21,
00374     BRIG_OPCODE_MAD24 = 22,
00375     BRIG_OPCODE_MAD24HI = 23,
00376     BRIG_OPCODE_MUL24 = 24,
00377     BRIG_OPCODE_MUL24HI = 25,
00378     BRIG_OPCODE_SHL = 26,
00379     BRIG_OPCODE_SHR = 27,
00380     BRIG_OPCODE_AND = 28,
00381     BRIG_OPCODE_NOT = 29,
00382     BRIG_OPCODE_OR = 30,
00383     BRIG_OPCODE_POPCOUNT = 31,
00384     BRIG_OPCODE_XOR = 32,
00385     BRIG_OPCODE_BITEXTRACT = 33,
00386     BRIG_OPCODE_BITINSERT = 34,
00387     BRIG_OPCODE_BITMASK = 35,
00388     BRIG_OPCODE_BITREV = 36,
00389     BRIG_OPCODE_BITSELECT = 37,
00390     BRIG_OPCODE_FIRSTBIT = 38,
00391     BRIG_OPCODE_LASTBIT = 39,
00392     BRIG_OPCODE_COMBINE = 40,
00393     BRIG_OPCODE_EXPAND = 41,
00394     BRIG_OPCODE_LDA = 42,
00395     BRIG_OPCODE_MOV = 43,
00396     BRIG_OPCODE_SHUFFLE = 44,
00397     BRIG_OPCODE_UNPACKHI = 45,
00398     BRIG_OPCODE_UNPACKLO = 46,
00399     BRIG_OPCODE_PACK = 47,
00400     BRIG_OPCODE_UNPACK = 48,
00401     BRIG_OPCODE_CMOV = 49,
00402     BRIG_OPCODE_CLASS = 50,
00403     BRIG_OPCODE_NCOS = 51,
00404     BRIG_OPCODE_NEXP2 = 52,
00405     BRIG_OPCODE_NFMA = 53,
00406     BRIG_OPCODE_NLOG2 = 54,
00407     BRIG_OPCODE_NRCP = 55,
00408     BRIG_OPCODE_NRSQRT = 56,
00409     BRIG_OPCODE_NSIN = 57,
00410     BRIG_OPCODE_NSQRT = 58,
00411     BRIG_OPCODE_BITALIGN = 59,
00412     BRIG_OPCODE_BYTEALIGN = 60,
00413     BRIG_OPCODE_PACKCVT = 61,
00414     BRIG_OPCODE_UNPACKCVT = 62,
00415     BRIG_OPCODE_LERP = 63,
00416     BRIG_OPCODE_SAD = 64,
00417     BRIG_OPCODE_SADHI = 65,
00418     BRIG_OPCODE_SEGMENTP = 66,
00419     BRIG_OPCODE_FTOS = 67,
00420     BRIG_OPCODE_STOF = 68,
00421     BRIG_OPCODE_CMP = 69,
00422     BRIG_OPCODE_CVT = 70,
00423     BRIG_OPCODE_LD = 71,
00424     BRIG_OPCODE_ST = 72,
00425     BRIG_OPCODE_ATOMIC = 73,
00426     BRIG_OPCODE_ATOMICNORET = 74,
00427     BRIG_OPCODE_SIGNAL = 75,
00428     BRIG_OPCODE_SIGNALNORET = 76,
00429     BRIG_OPCODE_MEMFENCE = 77,
00430     BRIG_OPCODE_RDIMAGE = 78,
00431     BRIG_OPCODE_LDIMAGE = 79,
00432     BRIG_OPCODE_STIMAGE = 80,
00433     BRIG_OPCODE_IMAGEFENCE = 81,
00434     BRIG_OPCODE_QUERYIMAGE = 82,
00435     BRIG_OPCODE_QUERYSAMPLER = 83,
00436     BRIG_OPCODE_CBR = 84,
00437     BRIG_OPCODE_BR = 85,
00438     BRIG_OPCODE_SBR = 86,
00439     BRIG_OPCODE_BARRIER = 87,
00440     BRIG_OPCODE_WAVEBARRIER = 88,
00441     BRIG_OPCODE_ARRIVEFBAR = 89,
00442     BRIG_OPCODE_INITFBAR = 90,
00443     BRIG_OPCODE_JOINFBAR = 91,
00444     BRIG_OPCODE_LEAVEFBAR = 92,
00445     BRIG_OPCODE_RELEASEFBAR = 93,
00446     BRIG_OPCODE_WAITFBAR = 94,
00447     BRIG_OPCODE_LDF = 95,
00448     BRIG_OPCODE_ACTIVELANECOUNT = 96,
00449     BRIG_OPCODE_ACTIVELANEID = 97,
00450     BRIG_OPCODE_ACTIVELANEMASK = 98,
```

```
00451     BRIG_OPCODE_ACTIVELANEPERMUTE = 99,
00452     BRIG_OPCODE_CALL = 100,
00453     BRIG_OPCODE_SCALL = 101,
00454     BRIG_OPCODE_ICALL = 102,
00455     BRIG_OPCODE_RET = 103,
00456     BRIG_OPCODE_ALLOCA = 104,
00457     BRIG_OPCODE_CURRENTWORKGROUPSIZE = 105,
00458     BRIG_OPCODE_CURRENTWORKITEMFLATID = 106,
00459     BRIG_OPCODE_DIM = 107,
00460     BRIG_OPCODE_GRIDGROUPS = 108,
00461     BRIG_OPCODE_GRIDSIZE = 109,
00462     BRIG_OPCODE_PACKETCOMPLETIONSIG = 110,
00463     BRIG_OPCODE_PACKETID = 111,
00464     BRIG_OPCODE_WORKGROUPID = 112,
00465     BRIG_OPCODE_WORKGROUPSIZE = 113,
00466     BRIG_OPCODE_WORKITEMABSID = 114,
00467     BRIG_OPCODE_WORKITEMFLATABSID = 115,
00468     BRIG_OPCODE_WORKITEMFLATID = 116,
00469     BRIG_OPCODE_WORKITEMID = 117,
00470     BRIG_OPCODE_CLEARDETECTEXCEPT = 118,
00471     BRIG_OPCODE_GETDETECTEXCEPT = 119,
00472     BRIG_OPCODE_SETDETECTEXCEPT = 120,
00473     BRIG_OPCODE_ADDQUEUEWRITEINDEX = 121,
00474     BRIG_OPCODE_CASQUEUEWRITEINDEX = 122,
00475     BRIG_OPCODE_LDQUEUEREADINDEX = 123,
00476     BRIG_OPCODE_LDQUEUEWRITEINDEX = 124,
00477     BRIG_OPCODE_STQUEUEREADINDEX = 125,
00478     BRIG_OPCODE_STQUEUEWRITEINDEX = 126,
00479     BRIG_OPCODE_CLOCK = 127,
00480     BRIG_OPCODE_CUID = 128,
00481     BRIG_OPCODE_DEBUGTRAP = 129,
00482     BRIG_OPCODE_GROUPBASEPTR = 130,
00483     BRIG_OPCODE_KERNARGBASEPTR = 131,
00484     BRIG_OPCODE_LANEID = 132,
00485     BRIG_OPCODE_MAXCUID = 133,
00486     BRIG_OPCODE_MAXWAVEID = 134,
00487     BRIG_OPCODE_NULLPTR = 135,
00488     BRIG_OPCODE_WAVEID = 136,
00489
00490     BRIG_OPCODE_FIRST_USER_DEFINED = 32768,
00491 };
00492
00493 typedef uint8_t BrigPack8_t;
00494 enum BrigPack {
00495     BRIG_PACK_NONE = 0,
00496     BRIG_PACK_PP = 1,
00497     BRIG_PACK_PS = 2,
00498     BRIG_PACK_SP = 3,
00499     BRIG_PACK_SS = 4,
00500     BRIG_PACK_S = 5,
00501     BRIG_PACK_P = 6,
00502     BRIG_PACK_PPSAT = 7,
00503     BRIG_PACK_PSSAT = 8,
00504     BRIG_PACK_SPSAT = 9,
00505     BRIG_PACK_SSSAT = 10,
00506     BRIG_PACK_SSAT = 11,
00507     BRIG_PACK_PSAT = 12
00508 };
00509
00510 typedef uint8_t BrigProfile8_t;
00511 enum BrigProfile {
00512     BRIG_PROFILE_BASE = 0,
00513     BRIG_PROFILE_FULL = 1,
00514 };
00515
00516 typedef uint16_t BrigRegisterKind16_t;
00517 enum BrigRegisterKind {
00518     BRIG_REGISTER_KIND_CONTROL = 0,
00519     BRIG_REGISTER_KIND_SINGLE = 1,
00520     BRIG_REGISTER_KIND_DOUBLE = 2,
00521     BRIG_REGISTER_KIND_QUAD = 3
00522 };
00523
00524 typedef uint8_t BrigRound8_t;
00525 enum BrigRound {
00526     BRIG_ROUND_NONE = 0,
00527     BRIG_ROUND_FLOAT_DEFAULT = 1,
00528     BRIG_ROUND_FLOAT_NEAR_EVEN = 2,
00529     BRIG_ROUND_FLOAT_ZERO = 3,
00530     BRIG_ROUND_FLOAT_PLUS_INFINITY = 4,
00531     BRIG_ROUND_FLOAT_MINUS_INFINITY = 5,
00532     BRIG_ROUND_INTEGER_NEAR_EVEN = 6,
00533     BRIG_ROUND_INTEGER_ZERO = 7,
00534     BRIG_ROUND_INTEGER_PLUS_INFINITY = 8,
00535     BRIG_ROUND_INTEGER_MINUS_INFINITY = 9,
00536     BRIG_ROUND_INTEGER_NEAR_EVEN_SAT = 10,
00537     BRIG_ROUND_INTEGER_ZERO_SAT = 11,
```

```
00538     BRIG_ROUND_INTEGER_PLUS_INFINITY_SAT = 12,
00539     BRIG_ROUND_INTEGER_MINUS_INFINITY_SAT = 13,
00540     BRIG_ROUND_INTEGER_SIGNALING_NEAR_EVEN = 14,
00541     BRIG_ROUND_INTEGER_SIGNALING_ZERO = 15,
00542     BRIG_ROUND_INTEGER_SIGNALING_PLUS_INFINITY = 16,
00543     BRIG_ROUND_INTEGER_SIGNALING_MINUS_INFINITY = 17,
00544     BRIG_ROUND_INTEGER_SIGNALING_NEAR_EVEN_SAT = 18,
00545     BRIG_ROUND_INTEGER_SIGNALING_ZERO_SAT = 19,
00546     BRIG_ROUND_INTEGER_SIGNALING_PLUS_INFINITY_SAT = 20,
00547     BRIG_ROUND_INTEGER_SIGNALING_MINUS_INFINITY_SAT = 21
00548 };
00549
00550 typedef uint8_t BrigSamplerAddressing8_t;
00551 enum BrigSamplerAddressing {
00552     BRIG_ADDRESSING_UNDEFINED = 0,
00553     BRIG_ADDRESSING_CLAMP_TO_EDGE = 1,
00554     BRIG_ADDRESSING_CLAMP_TO_BORDER = 2,
00555     BRIG_ADDRESSING_REPEAT = 3,
00556     BRIG_ADDRESSING_MIRRORED_REPEAT = 4,
00557
00558     BRIG_ADDRESSING_FIRST_USER_DEFINED = 128
00559 };
00560
00561 typedef uint8_t BrigSamplerCoordNormalization8_t;
00562 enum BrigSamplerCoordNormalization {
00563     BRIG_COORD_UNNORMALIZED = 0,
00564     BRIG_COORD_NORMALIZED = 1
00565 };
00566
00567 typedef uint8_t BrigSamplerFilter8_t;
00568 enum BrigSamplerFilter {
00569     BRIG_FILTER_NEAREST = 0,
00570     BRIG_FILTER_LINEAR = 1,
00571
00572     BRIG_FILTER_FIRST_USER_DEFINED = 128
00573 };
00574
00575 typedef uint8_t BrigSamplerQuery8_t;
00576 enum BrigSamplerQuery {
00577     BRIG_SAMPLER_QUERY_ADDRESSING = 0,
00578     BRIG_SAMPLER_QUERY_COORD = 1,
00579     BRIG_SAMPLER_QUERY_FILTER = 2
00580 };
00581
00582 typedef uint32_t BrigSectionIndex32_t;
00583 enum BrigSectionIndex {
00584     BRIG_SECTION_INDEX_DATA = 0,
00585     BRIG_SECTION_INDEX_CODE = 1,
00586     BRIG_SECTION_INDEX_OPERAND = 2,
00587
00588     BRIG_SECTION_INDEX_BEGIN_IMPLEMENTATION_DEFINED = 3,
00589 };
00590
00591 typedef uint8_t BrigSegCvtModifier8_t;
00592 enum BrigSegCvtModifierMask {
00593     BRIG_SEG_CVT_NONULL = 1
00594 };
00595
00596 typedef uint8_t BrigSegment8_t;
00597 enum BrigSegment {
00598     BRIG_SEGMENT_NONE = 0,
00599     BRIG_SEGMENT_FLAT = 1,
00600     BRIG_SEGMENT_GLOBAL = 2,
00601     BRIG_SEGMENT_READONLY = 3,
00602     BRIG_SEGMENT_KERNARG = 4,
00603     BRIG_SEGMENT_GROUP = 5,
00604     BRIG_SEGMENT_PRIVATE = 6,
00605     BRIG_SEGMENT_SPILL = 7,
00606     BRIG_SEGMENT_ARG = 8,
00607
00608     BRIG_SEGMENT_FIRST_USER_DEFINED = 128
00609 };
00610
00611 enum {
00612     BRIG_TYPE_BASE_SIZE  = 5,
00613     BRIG_TYPE_PACK_SIZE  = 2,
00614     BRIG_TYPE_ARRAY_SIZE = 1,
00615
00616     BRIG_TYPE_BASE_SHIFT  = 0,
00617     BRIG_TYPE_PACK_SHIFT  = BRIG_TYPE_BASE_SHIFT + BRIG_TYPE_BASE_SIZE,
00618     BRIG_TYPE_ARRAY_SHIFT = BRIG_TYPE_PACK_SHIFT + BRIG_TYPE_PACK_SIZE,
00619
00620     BRIG_TYPE_BASE_MASK  = ((1 « BRIG_TYPE_BASE_SIZE)  - 1) « BRIG_TYPE_BASE_SHIFT,
00621     BRIG_TYPE_PACK_MASK  = ((1 « BRIG_TYPE_PACK_SIZE)  - 1) « BRIG_TYPE_PACK_SHIFT,
00622     BRIG_TYPE_ARRAY_MASK = ((1 « BRIG_TYPE_ARRAY_SIZE) - 1) « BRIG_TYPE_ARRAY_SHIFT,
00623
00624     BRIG_TYPE_PACK_NONE = 0 « BRIG_TYPE_PACK_SHIFT,
```

```
00625     BRIG_TYPE_PACK_32   = 1 << BRIG_TYPE_PACK_SHIFT,
00626     BRIG_TYPE_PACK_64   = 2 << BRIG_TYPE_PACK_SHIFT,
00627     BRIG_TYPE_PACK_128  = 3 << BRIG_TYPE_PACK_SHIFT,
00628
00629     BRIG_TYPE_ARRAY     = 1 << BRIG_TYPE_ARRAY_SHIFT
00630 };
00631
00632 typedef uint16_t BrigType16_t;
00633 enum BrigType {
00634     BRIG_TYPE_NONE  = 0,
00635     BRIG_TYPE_U8    = 1,
00636     BRIG_TYPE_U16   = 2,
00637     BRIG_TYPE_U32   = 3,
00638     BRIG_TYPE_U64   = 4,
00639     BRIG_TYPE_S8    = 5,
00640     BRIG_TYPE_S16   = 6,
00641     BRIG_TYPE_S32   = 7,
00642     BRIG_TYPE_S64   = 8,
00643     BRIG_TYPE_F16   = 9,
00644     BRIG_TYPE_F32   = 10,
00645     BRIG_TYPE_F64   = 11,
00646     BRIG_TYPE_B1    = 12,
00647     BRIG_TYPE_B8    = 13,
00648     BRIG_TYPE_B16   = 14,
00649     BRIG_TYPE_B32   = 15,
00650     BRIG_TYPE_B64   = 16,
00651     BRIG_TYPE_B128  = 17,
00652     BRIG_TYPE_SAMP  = 18,
00653     BRIG_TYPE_ROIMG = 19,
00654     BRIG_TYPE_WOIMG = 20,
00655     BRIG_TYPE_RWIMG = 21,
00656     BRIG_TYPE_SIG32 = 22,
00657     BRIG_TYPE_SIG64 = 23,
00658
00659     BRIG_TYPE_U8X4  = BRIG_TYPE_U8  | BRIG_TYPE_PACK_32,
00660     BRIG_TYPE_U8X8  = BRIG_TYPE_U8  | BRIG_TYPE_PACK_64,
00661     BRIG_TYPE_U8X16 = BRIG_TYPE_U8  | BRIG_TYPE_PACK_128,
00662     BRIG_TYPE_U16X2 = BRIG_TYPE_U16 | BRIG_TYPE_PACK_32,
00663     BRIG_TYPE_U16X4 = BRIG_TYPE_U16 | BRIG_TYPE_PACK_64,
00664     BRIG_TYPE_U16X8 = BRIG_TYPE_U16 | BRIG_TYPE_PACK_128,
00665     BRIG_TYPE_U32X2 = BRIG_TYPE_U32 | BRIG_TYPE_PACK_64,
00666     BRIG_TYPE_U32X4 = BRIG_TYPE_U32 | BRIG_TYPE_PACK_128,
00667     BRIG_TYPE_U64X2 = BRIG_TYPE_U64 | BRIG_TYPE_PACK_128,
00668     BRIG_TYPE_S8X4  = BRIG_TYPE_S8  | BRIG_TYPE_PACK_32,
00669     BRIG_TYPE_S8X8  = BRIG_TYPE_S8  | BRIG_TYPE_PACK_64,
00670     BRIG_TYPE_S8X16 = BRIG_TYPE_S8  | BRIG_TYPE_PACK_128,
00671     BRIG_TYPE_S16X2 = BRIG_TYPE_S16 | BRIG_TYPE_PACK_32,
00672     BRIG_TYPE_S16X4 = BRIG_TYPE_S16 | BRIG_TYPE_PACK_64,
00673     BRIG_TYPE_S16X8 = BRIG_TYPE_S16 | BRIG_TYPE_PACK_128,
00674     BRIG_TYPE_S32X2 = BRIG_TYPE_S32 | BRIG_TYPE_PACK_64,
00675     BRIG_TYPE_S32X4 = BRIG_TYPE_S32 | BRIG_TYPE_PACK_128,
00676     BRIG_TYPE_S64X2 = BRIG_TYPE_S64 | BRIG_TYPE_PACK_128,
00677     BRIG_TYPE_F16X2 = BRIG_TYPE_F16 | BRIG_TYPE_PACK_32,
00678     BRIG_TYPE_F16X4 = BRIG_TYPE_F16 | BRIG_TYPE_PACK_64,
00679     BRIG_TYPE_F16X8 = BRIG_TYPE_F16 | BRIG_TYPE_PACK_128,
00680     BRIG_TYPE_F32X2 = BRIG_TYPE_F32 | BRIG_TYPE_PACK_64,
00681     BRIG_TYPE_F32X4 = BRIG_TYPE_F32 | BRIG_TYPE_PACK_128,
00682     BRIG_TYPE_F64X2 = BRIG_TYPE_F64 | BRIG_TYPE_PACK_128,
00683
00684     BRIG_TYPE_U8_ARRAY    = BRIG_TYPE_U8    | BRIG_TYPE_ARRAY,
00685     BRIG_TYPE_U16_ARRAY   = BRIG_TYPE_U16   | BRIG_TYPE_ARRAY,
00686     BRIG_TYPE_U32_ARRAY   = BRIG_TYPE_U32   | BRIG_TYPE_ARRAY,
00687     BRIG_TYPE_U64_ARRAY   = BRIG_TYPE_U64   | BRIG_TYPE_ARRAY,
00688     BRIG_TYPE_S8_ARRAY    = BRIG_TYPE_S8    | BRIG_TYPE_ARRAY,
00689     BRIG_TYPE_S16_ARRAY   = BRIG_TYPE_S16   | BRIG_TYPE_ARRAY,
00690     BRIG_TYPE_S32_ARRAY   = BRIG_TYPE_S32   | BRIG_TYPE_ARRAY,
00691     BRIG_TYPE_S64_ARRAY   = BRIG_TYPE_S64   | BRIG_TYPE_ARRAY,
00692     BRIG_TYPE_F16_ARRAY   = BRIG_TYPE_F16   | BRIG_TYPE_ARRAY,
00693     BRIG_TYPE_F32_ARRAY   = BRIG_TYPE_F32   | BRIG_TYPE_ARRAY,
00694     BRIG_TYPE_F64_ARRAY   = BRIG_TYPE_F64   | BRIG_TYPE_ARRAY,
00695     BRIG_TYPE_B8_ARRAY    = BRIG_TYPE_B8    | BRIG_TYPE_ARRAY,
00696     BRIG_TYPE_B16_ARRAY   = BRIG_TYPE_B16   | BRIG_TYPE_ARRAY,
00697     BRIG_TYPE_B32_ARRAY   = BRIG_TYPE_B32   | BRIG_TYPE_ARRAY,
00698     BRIG_TYPE_B64_ARRAY   = BRIG_TYPE_B64   | BRIG_TYPE_ARRAY,
00699     BRIG_TYPE_B128_ARRAY  = BRIG_TYPE_B128  | BRIG_TYPE_ARRAY,
00700     BRIG_TYPE_SAMP_ARRAY  = BRIG_TYPE_SAMP  | BRIG_TYPE_ARRAY,
00701     BRIG_TYPE_ROIMG_ARRAY = BRIG_TYPE_ROIMG | BRIG_TYPE_ARRAY,
00702     BRIG_TYPE_WOIMG_ARRAY = BRIG_TYPE_WOIMG | BRIG_TYPE_ARRAY,
00703     BRIG_TYPE_RWIMG_ARRAY = BRIG_TYPE_RWIMG | BRIG_TYPE_ARRAY,
00704     BRIG_TYPE_SIG32_ARRAY = BRIG_TYPE_SIG32 | BRIG_TYPE_ARRAY,
00705     BRIG_TYPE_SIG64_ARRAY = BRIG_TYPE_SIG64 | BRIG_TYPE_ARRAY,
00706     BRIG_TYPE_U8X4_ARRAY  = BRIG_TYPE_U8X4  | BRIG_TYPE_ARRAY,
00707     BRIG_TYPE_U8X8_ARRAY  = BRIG_TYPE_U8X8  | BRIG_TYPE_ARRAY,
00708     BRIG_TYPE_U8X16_ARRAY = BRIG_TYPE_U8X16 | BRIG_TYPE_ARRAY,
00709     BRIG_TYPE_U16X2_ARRAY = BRIG_TYPE_U16X2 | BRIG_TYPE_ARRAY,
00710     BRIG_TYPE_U16X4_ARRAY = BRIG_TYPE_U16X4 | BRIG_TYPE_ARRAY,
00711     BRIG_TYPE_U16X8_ARRAY = BRIG_TYPE_U16X8 | BRIG_TYPE_ARRAY,
```

```
00712     BRIG_TYPE_U32X2_ARRAY = BRIG_TYPE_U32X2 | BRIG_TYPE_ARRAY,
00713     BRIG_TYPE_U32X4_ARRAY = BRIG_TYPE_U32X4 | BRIG_TYPE_ARRAY,
00714     BRIG_TYPE_U64X2_ARRAY = BRIG_TYPE_U64X2 | BRIG_TYPE_ARRAY,
00715     BRIG_TYPE_S8X4_ARRAY  = BRIG_TYPE_S8X4  | BRIG_TYPE_ARRAY,
00716     BRIG_TYPE_S8X8_ARRAY  = BRIG_TYPE_S8X8  | BRIG_TYPE_ARRAY,
00717     BRIG_TYPE_S8X16_ARRAY = BRIG_TYPE_S8X16 | BRIG_TYPE_ARRAY,
00718     BRIG_TYPE_S16X2_ARRAY = BRIG_TYPE_S16X2 | BRIG_TYPE_ARRAY,
00719     BRIG_TYPE_S16X4_ARRAY = BRIG_TYPE_S16X4 | BRIG_TYPE_ARRAY,
00720     BRIG_TYPE_S16X8_ARRAY = BRIG_TYPE_S16X8 | BRIG_TYPE_ARRAY,
00721     BRIG_TYPE_S32X2_ARRAY = BRIG_TYPE_S32X2 | BRIG_TYPE_ARRAY,
00722     BRIG_TYPE_S32X4_ARRAY = BRIG_TYPE_S32X4 | BRIG_TYPE_ARRAY,
00723     BRIG_TYPE_S64X2_ARRAY = BRIG_TYPE_S64X2 | BRIG_TYPE_ARRAY,
00724     BRIG_TYPE_F16X2_ARRAY = BRIG_TYPE_F16X2 | BRIG_TYPE_ARRAY,
00725     BRIG_TYPE_F16X4_ARRAY = BRIG_TYPE_F16X4 | BRIG_TYPE_ARRAY,
00726     BRIG_TYPE_F16X8_ARRAY = BRIG_TYPE_F16X8 | BRIG_TYPE_ARRAY,
00727     BRIG_TYPE_F32X2_ARRAY = BRIG_TYPE_F32X2 | BRIG_TYPE_ARRAY,
00728     BRIG_TYPE_F32X4_ARRAY = BRIG_TYPE_F32X4 | BRIG_TYPE_ARRAY,
00729     BRIG_TYPE_F64X2_ARRAY = BRIG_TYPE_F64X2 | BRIG_TYPE_ARRAY,
00730 };
00731
00732 typedef uint8_t BrigVariableModifier8_t;
00733 enum BrigVariableModifierMask {
00734     BRIG_VARIABLE_DEFINITION = 1,
00735     BRIG_VARIABLE_CONST = 2
00736 };
00737
00738 typedef uint8_t BrigWidth8_t;
00739 enum BrigWidth {
00740     BRIG_WIDTH_NONE = 0,
00741     BRIG_WIDTH_1 = 1,
00742     BRIG_WIDTH_2 = 2,
00743     BRIG_WIDTH_4 = 3,
00744     BRIG_WIDTH_8 = 4,
00745     BRIG_WIDTH_16 = 5,
00746     BRIG_WIDTH_32 = 6,
00747     BRIG_WIDTH_64 = 7,
00748     BRIG_WIDTH_128 = 8,
00749     BRIG_WIDTH_256 = 9,
00750     BRIG_WIDTH_512 = 10,
00751     BRIG_WIDTH_1024 = 11,
00752     BRIG_WIDTH_2048 = 12,
00753     BRIG_WIDTH_4096 = 13,
00754     BRIG_WIDTH_8192 = 14,
00755     BRIG_WIDTH_16384 = 15,
00756     BRIG_WIDTH_32768 = 16,
00757     BRIG_WIDTH_65536 = 17,
00758     BRIG_WIDTH_131072 = 18,
00759     BRIG_WIDTH_262144 = 19,
00760     BRIG_WIDTH_524288 = 20,
00761     BRIG_WIDTH_1048576 = 21,
00762     BRIG_WIDTH_2097152 = 22,
00763     BRIG_WIDTH_4194304 = 23,
00764     BRIG_WIDTH_8388608 = 24,
00765     BRIG_WIDTH_16777216 = 25,
00766     BRIG_WIDTH_33554432 = 26,
00767     BRIG_WIDTH_67108864 = 27,
00768     BRIG_WIDTH_134217728 = 28,
00769     BRIG_WIDTH_268435456 = 29,
00770     BRIG_WIDTH_536870912 = 30,
00771     BRIG_WIDTH_1073741824 = 31,
00772     BRIG_WIDTH_2147483648 = 32,
00773     BRIG_WIDTH_WAVESIZE = 33,
00774     BRIG_WIDTH_ALL = 34,
00775 };
00776
00777 struct BrigUInt64 {
00778     uint32_t lo;
00779     uint32_t hi;
00780 };
00781
00782 struct BrigBase {
00783     uint16_t byteCount;
00784     BrigKind16_t kind;
00785 };
00786
00787 struct BrigData {
00788     uint32_t byteCount;
00789     uint8_t bytes[1];
00790 };
00791
00792 struct BrigDirectiveArgBlock {
00793     BrigBase base;
00794 };
00795
00796 struct BrigDirectiveComment {
00797     BrigBase base;
00798     BrigDataOffsetString32_t name;
```

```
00799 };
00800
00801 struct BrigDirectiveControl {
00802     BrigBase base;
00803     BrigControlDirective16_t control;
00804     uint16_t reserved;
00805     BrigDataOffsetOperandList32_t operands;
00806 };
00807
00808 struct BrigDirectiveExecutable {
00809     BrigBase base;
00810     BrigDataOffsetString32_t name;
00811     uint16_t outArgCount;
00812     uint16_t inArgCount;
00813     BrigCodeOffset32_t firstInArg;
00814     BrigCodeOffset32_t firstCodeBlockEntry;
00815     BrigCodeOffset32_t nextModuleEntry;
00816     BrigExecutableModifier8_t modifier;
00817     BrigLinkage8_t linkage;
00818     uint16_t reserved;
00819 };
00820
00821 struct BrigDirectiveExtension {
00822     BrigBase base;
00823     BrigDataOffsetString32_t name;
00824 };
00825
00826 struct BrigDirectiveFbarrier {
00827     BrigBase base;
00828     BrigDataOffsetString32_t name;
00829     BrigVariableModifier8_t modifier;
00830     BrigLinkage8_t linkage;
00831     uint16_t reserved;
00832 };
00833
00834 struct BrigDirectiveLabel {
00835     BrigBase base;
00836     BrigDataOffsetString32_t name;
00837 };
00838
00839 struct BrigDirectiveLoc {
00840     BrigBase base;
00841     BrigDataOffsetString32_t filename;
00842     uint32_t line;
00843     uint32_t column;
00844 };
00845
00846 struct BrigDirectiveNone {
00847     BrigBase base;
00848 };
00849
00850 struct BrigDirectivePragma {
00851     BrigBase base;
00852     BrigDataOffsetOperandList32_t operands;
00853 };
00854
00855 struct BrigDirectiveVariable {
00856     BrigBase base;
00857     BrigDataOffsetString32_t name;
00858     BrigOperandOffset32_t init;
00859     BrigType16_t type;
00860     BrigSegment8_t segment;
00861     BrigAlignment8_t align;
00862     BrigUInt64 dim;
00863     BrigVariableModifier8_t modifier;
00864     BrigLinkage8_t linkage;
00865     BrigAllocation8_t allocation;
00866     uint8_t reserved;
00867 };
00868
00869 struct BrigDirectiveModule {
00870     BrigBase base;
00871     BrigDataOffsetString32_t name;
00872     BrigVersion32_t hsailMajor;
00873     BrigVersion32_t hsailMinor;
00874     BrigProfile8_t profile;
00875     BrigMachineModel8_t machineModel;
00876     BrigRound8_t defaultFloatRound;
00877     uint8_t reserved;
00878 };
00879
00880 struct BrigInstBase {
00881     BrigBase base;
00882     BrigOpcode16_t opcode;
00883     BrigType16_t type;
00884     BrigDataOffsetOperandList32_t operands;
00885 };
```

```
00886
00887 struct BrigInstAddr {
00888     BrigInstBase base;
00889     BrigSegment8_t segment;
00890     uint8_t reserved[3];
00891 };
00892
00893 struct BrigInstAtomic {
00894     BrigInstBase base;
00895     BrigSegment8_t segment;
00896     BrigMemoryOrder8_t memoryOrder;
00897     BrigMemoryScope8_t memoryScope;
00898     BrigAtomicOperation8_t atomicOperation;
00899     uint8_t equivClass;
00900     uint8_t reserved[3];
00901 };
00902
00903 struct BrigInstBasic {
00904     BrigInstBase base;
00905 };
00906
00907 struct BrigInstBr {
00908     BrigInstBase base;
00909     BrigWidth8_t width;
00910     uint8_t reserved[3];
00911 };
00912
00913 struct BrigInstCmp {
00914     BrigInstBase base;
00915     BrigType16_t sourceType;
00916     BrigAluModifier8_t modifier;
00917     BrigCompareOperation8_t compare;
00918     BrigPack8_t pack;
00919     uint8_t reserved[3];
00920 };
00921
00922 struct BrigInstCvt {
00923     BrigInstBase base;
00924     BrigType16_t sourceType;
00925     BrigAluModifier8_t modifier;
00926     BrigRound8_t round;
00927 };
00928
00929 struct BrigInstImage {
00930     BrigInstBase base;
00931     BrigType16_t imageType;
00932     BrigType16_t coordType;
00933     BrigImageGeometry8_t geometry;
00934     uint8_t equivClass;
00935     uint16_t reserved;
00936 };
00937
00938 struct BrigInstLane {
00939     BrigInstBase base;
00940     BrigType16_t sourceType;
00941     BrigWidth8_t width;
00942     uint8_t reserved;
00943 };
00944
00945 struct BrigInstMem {
00946     BrigInstBase base;
00947     BrigSegment8_t segment;
00948     BrigAlignment8_t align;
00949     uint8_t equivClass;
00950     BrigWidth8_t width;
00951     BrigMemoryModifier8_t modifier;
00952     uint8_t reserved[3];
00953 };
00954
00955 struct BrigInstMemFence {
00956     BrigInstBase base;
00957     BrigMemoryOrder8_t memoryOrder;
00958     BrigMemoryScope8_t globalSegmentMemoryScope;
00959     BrigMemoryScope8_t groupSegmentMemoryScope;
00960     BrigMemoryScope8_t imageSegmentMemoryScope;
00961 };
00962
00963 struct BrigInstMod {
00964     BrigInstBase base;
00965     BrigAluModifier8_t modifier;
00966     BrigRound8_t round;
00967     BrigPack8_t pack;
00968     uint8_t reserved;
00969 };
00970
00971 struct BrigInstQueryImage {
00972     BrigInstBase base;
```

```
00973     BrigType16_t imageType;
00974     BrigImageGeometry8_t geometry;
00975     BrigImageQuery8_t query;
00976 };
00977
00978 struct BrigInstQuerySampler {
00979     BrigInstBase base;
00980     BrigSamplerQuery8_t query;
00981     uint8_t reserved[3];
00982 };
00983
00984 struct BrigInstQueue {
00985     BrigInstBase base;
00986     BrigSegment8_t segment;
00987     BrigMemoryOrder8_t memoryOrder;
00988     uint16_t reserved;
00989 };
00990
00991 struct BrigInstSeg {
00992     BrigInstBase base;
00993     BrigSegment8_t segment;
00994     uint8_t reserved[3];
00995 };
00996
00997 struct BrigInstSegCvt {
00998     BrigInstBase base;
00999     BrigType16_t sourceType;
01000     BrigSegment8_t segment;
01001     BrigSegCvtModifier8_t modifier;
01002 };
01003
01004 struct BrigInstSignal {
01005     BrigInstBase base;
01006     BrigType16_t signalType;
01007     BrigMemoryOrder8_t memoryOrder;
01008     BrigAtomicOperation8_t signalOperation;
01009 };
01010
01011 struct BrigInstSourceType {
01012     BrigInstBase base;
01013     BrigType16_t sourceType;
01014     uint16_t reserved;
01015 };
01016
01017 struct BrigOperandAddress {
01018     BrigBase base;
01019     BrigCodeOffset32_t symbol;
01020     BrigOperandOffset32_t reg;
01021     BrigUInt64 offset;
01022 };
01023
01024 struct BrigOperandAlign {
01025     BrigBase base;
01026     BrigAlignment8_t align;
01027     uint8_t reserved[3];
01028 };
01029
01030 struct BrigOperandCodeList {
01031     BrigBase base;
01032     BrigDataOffsetCodeList32_t elements;
01033 };
01034
01035 struct BrigOperandCodeRef {
01036     BrigBase base;
01037     BrigCodeOffset32_t ref;
01038 };
01039
01040 struct BrigOperandConstantBytes {
01041     BrigBase base;
01042     BrigType16_t type;
01043     uint16_t reserved;
01044     BrigDataOffsetString32_t bytes;
01045 };
01046
01047 struct BrigOperandConstantOperandList {
01048     BrigBase base;
01049     BrigType16_t type;
01050     uint16_t reserved;
01051     BrigDataOffsetOperandList32_t elements;
01052 };
01053
01054 struct BrigOperandConstantImage {
01055     BrigBase base;
01056     BrigType16_t type;
01057     BrigImageGeometry8_t geometry;
01058     BrigImageChannelOrder8_t channelOrder;
01059     BrigImageChannelType8_t channelType;
```

```
01060     uint8_t reserved[3];
01061     BrigUInt64 width;
01062     BrigUInt64 height;
01063     BrigUInt64 depth;
01064     BrigUInt64 array;
01065 };
01066
01067 struct BrigOperandOperandList {
01068     BrigBase base;
01069     BrigDataOffsetOperandList32_t elements;
01070 };
01071
01072 struct BrigOperandRegister {
01073     BrigBase base;
01074     BrigRegisterKind16_t regKind;
01075     uint16_t regNum;
01076 };
01077
01078 struct BrigOperandConstantSampler {
01079     BrigBase base;
01080     BrigType16_t type;
01081     BrigSamplerCoordNormalization8_t coord;
01082     BrigSamplerFilter8_t filter;
01083     BrigSamplerAddressing8_t addressing;
01084     uint8_t reserved[3];
01085 };
01086
01087 struct BrigOperandString {
01088     BrigBase base;
01089     BrigDataOffsetString32_t string;
01090 };
01091
01092 struct BrigOperandWavesize {
01093     BrigBase base;
01094 };
01095
01096 typedef uint32_t BrigExceptions32_t;
01097 enum BrigExceptionsMask {
01098     BRIG_EXCEPTIONS_INVALID_OPERATION = 1 << 0,
01099     BRIG_EXCEPTIONS_DIVIDE_BY_ZERO = 1 << 1,
01100     BRIG_EXCEPTIONS_OVERFLOW = 1 << 2,
01101     BRIG_EXCEPTIONS_UNDERFLOW = 1 << 3,
01102     BRIG_EXCEPTIONS_INEXACT = 1 << 4,
01103
01104     BRIG_EXCEPTIONS_FIRST_USER_DEFINED = 1 << 16
01105 };
01106
01107 struct BrigSectionHeader {
01108     uint64_t byteCount;
01109     uint32_t headerByteCount;
01110     uint32_t nameLength;
01111     uint8_t name[1];
01112 };
01113
01114 struct BrigModuleHeader {
01115     char identification[8];
01116     BrigVersion32_t brigMajor;
01117     BrigVersion32_t brigMinor;
01118     uint64_t byteCount;
01119     uint8_t hash[64];
01120     uint32_t reserved;
01121     uint32_t sectionCount;
01122     uint64_t sectionIndex;
01123 };
01124
01125 typedef BrigModuleHeader* BrigModule_t;
01126
01127 #ifdef __cplusplus
01128 }
01129 #endif  /*__cplusplus*/
01130
01131 #endif // defined(INCLUDED_BRIG_H)
```

7.7 hsa.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
```

```
00009 //
00010 //                  AMD Research and AMD HSA Software Development
00011 //
00012 //                  Advanced Micro Devices, Inc.
00013 //
00014 //                  www.amd.com
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 #ifndef HSA_RUNTIME_INC_HSA_H_
00044 #define HSA_RUNTIME_INC_HSA_H_
00045
00046 #include <stddef.h>   /* size_t */
00047 #include <stdint.h>   /* uintXX_t */
00048
00049 #ifndef __cplusplus
00050 #include <stdbool.h>  /* bool */
00051 #endif /* __cplusplus */
00052
00053 // Placeholder for calling convention and import/export macros
00054 #ifndef HSA_CALL
00055 #define HSA_CALL
00056 #endif
00057
00058 #ifndef HSA_EXPORT_DECORATOR
00059 #ifdef __GNUC__
00060 #define HSA_EXPORT_DECORATOR __attribute__ ((visibility ("default")))
00061 #else
00062 #define HSA_EXPORT_DECORATOR
00063 #endif
00064 #endif
00065 #define HSA_API_EXPORT HSA_EXPORT_DECORATOR HSA_CALL
00066 #define HSA_API_IMPORT HSA_CALL
00067
00068 #if !defined(HSA_API) && defined(HSA_EXPORT)
00069 #define HSA_API HSA_API_EXPORT
00070 #else
00071 #define HSA_API HSA_API_IMPORT
00072 #endif
00073
00074 // Detect and set large model builds.
00075 #undef HSA_LARGE_MODEL
00076 #if defined(__LP64__) || defined(_M_X64)
00077 #define HSA_LARGE_MODEL
00078 #endif
00079
00080 // Try to detect CPU endianness
00081 #if !defined(LITTLEENDIAN_CPU) && !defined(BIGENDIAN_CPU)
00082 #if defined(__i386__) || defined(__x86_64__) || defined(_M_IX86) || \
00083     defined(_M_X64)
00084 #define LITTLEENDIAN_CPU
00085 #endif
00086 #endif
00087
00088 #undef HSA_LITTLE_ENDIAN
00089 #if defined(LITTLEENDIAN_CPU)
00090 #define HSA_LITTLE_ENDIAN
00091 #elif defined(BIGENDIAN_CPU)
00092 #else
00093 #error "BIGENDIAN_CPU or LITTLEENDIAN_CPU must be defined"
00094 #endif
00095
00096 #ifndef HSA_DEPRECATED
```

```
00097 #define HSA_DEPRECATED
00098 //#ifdef __GNUC__
00099 //#define HSA_DEPRECATED __attribute__((deprecated))
00100 //#else
00101 //#define HSA_DEPRECATED __declspec(deprecated)
00102 //#endif
00103 #endif
00104
00105 #define HSA_VERSION_1_0                                1
00106
00107 #ifdef __cplusplus
00108 extern "C" {
00109 #endif  /* __cplusplus */
00110
00118 typedef enum {
00122   HSA_STATUS_SUCCESS = 0x0,
00127   HSA_STATUS_INFO_BREAK = 0x1,
00131   HSA_STATUS_ERROR = 0x1000,
00136   HSA_STATUS_ERROR_INVALID_ARGUMENT = 0x1001,
00140   HSA_STATUS_ERROR_INVALID_QUEUE_CREATION = 0x1002,
00144   HSA_STATUS_ERROR_INVALID_ALLOCATION = 0x1003,
00148   HSA_STATUS_ERROR_INVALID_AGENT = 0x1004,
00152   HSA_STATUS_ERROR_INVALID_REGION = 0x1005,
00156   HSA_STATUS_ERROR_INVALID_SIGNAL = 0x1006,
00160   HSA_STATUS_ERROR_INVALID_QUEUE = 0x1007,
00166   HSA_STATUS_ERROR_OUT_OF_RESOURCES = 0x1008,
00170   HSA_STATUS_ERROR_INVALID_PACKET_FORMAT = 0x1009,
00174   HSA_STATUS_ERROR_RESOURCE_FREE = 0x100A,
00179   HSA_STATUS_ERROR_NOT_INITIALIZED = 0x100B,
00183   HSA_STATUS_ERROR_REFCOUNT_OVERFLOW = 0x100C,
00187   HSA_STATUS_ERROR_INCOMPATIBLE_ARGUMENTS = 0x100D,
00191   HSA_STATUS_ERROR_INVALID_INDEX = 0x100E,
00195   HSA_STATUS_ERROR_INVALID_ISA = 0x100F,
00199   HSA_STATUS_ERROR_INVALID_ISA_NAME = 0x1017,
00203   HSA_STATUS_ERROR_INVALID_CODE_OBJECT = 0x1010,
00207   HSA_STATUS_ERROR_INVALID_EXECUTABLE = 0x1011,
00211   HSA_STATUS_ERROR_FROZEN_EXECUTABLE = 0x1012,
00215   HSA_STATUS_ERROR_INVALID_SYMBOL_NAME = 0x1013,
00219   HSA_STATUS_ERROR_VARIABLE_ALREADY_DEFINED = 0x1014,
00223   HSA_STATUS_ERROR_VARIABLE_UNDEFINED = 0x1015,
00227   HSA_STATUS_ERROR_EXCEPTION = 0x1016,
00231   HSA_STATUS_ERROR_INVALID_CODE_SYMBOL = 0x1018,
00235   HSA_STATUS_ERROR_INVALID_EXECUTABLE_SYMBOL = 0x1019,
00239   HSA_STATUS_ERROR_INVALID_FILE = 0x1020,
00243   HSA_STATUS_ERROR_INVALID_CODE_OBJECT_READER = 0x1021,
00247   HSA_STATUS_ERROR_INVALID_CACHE = 0x1022,
00251   HSA_STATUS_ERROR_INVALID_WAVEFRONT = 0x1023,
00255   HSA_STATUS_ERROR_INVALID_SIGNAL_GROUP = 0x1024,
00259   HSA_STATUS_ERROR_INVALID_RUNTIME_STATE = 0x1025,
00263   HSA_STATUS_ERROR_FATAL = 0x1026
00264 } hsa_status_t;
00265
00282 hsa_status_t HSA_API hsa_status_string(
00283     hsa_status_t status,
00284     const char ** status_string);
00285
00286 /*
00287  * @}
00288  */
00289
00290 /*
00291  * \defgroup common Common Definitions
00292  *  @{
00293  */
00294
00298 typedef struct hsa_dim3_s {
00302   uint32_t x;
00303
00307   uint32_t y;
00308
00312   uint32_t z;
00313 } hsa_dim3_t;
00314
00318 typedef enum {
00322   HSA_ACCESS_PERMISSION_RO = 1,
00326   HSA_ACCESS_PERMISSION_WO = 2,
00330   HSA_ACCESS_PERMISSION_RW = 3
00331 } hsa_access_permission_t;
00332
00336 typedef int hsa_file_t;
00337
00338 /*
00339  * @} **/
00340
00341
00342 /*
00343  * \defgroup initshutdown Initialization and Shut Down
```

```
00344  *  @{
00345  */
00346
00364 hsa_status_t HSA_API hsa_init();
00365
00386 hsa_status_t HSA_API hsa_shut_down();
00387
00398 typedef enum {
00402     HSA_ENDIANNESS_LITTLE = 0,
00406     HSA_ENDIANNESS_BIG = 1
00407 } hsa_endianness_t;
00408
00413 typedef enum {
00417     HSA_MACHINE_MODEL_SMALL = 0,
00421     HSA_MACHINE_MODEL_LARGE = 1
00422 } hsa_machine_model_t;
00423
00430 typedef enum {
00434     HSA_PROFILE_BASE = 0,
00438     HSA_PROFILE_FULL = 1
00439 } hsa_profile_t;
00440
00444 typedef enum {
00449   HSA_SYSTEM_INFO_VERSION_MAJOR = 0,
00454   HSA_SYSTEM_INFO_VERSION_MINOR = 1,
00459   HSA_SYSTEM_INFO_TIMESTAMP = 2,
00464   HSA_SYSTEM_INFO_TIMESTAMP_FREQUENCY = 3,
00469   HSA_SYSTEM_INFO_SIGNAL_MAX_WAIT = 4,
00473   HSA_SYSTEM_INFO_ENDIANNESS = 5,
00478   HSA_SYSTEM_INFO_MACHINE_MODEL = 6,
00484   HSA_SYSTEM_INFO_EXTENSIONS = 7,
00488   HSA_AMD_SYSTEM_INFO_BUILD_VERSION = 0x200,
00493   HSA_AMD_SYSTEM_INFO_SVM_SUPPORTED = 0x201,
00494   // TODO: Should this be per Agent?
00502   HSA_AMD_SYSTEM_INFO_SVM_ACCESSIBLE_BY_DEFAULT = 0x202
00503 } hsa_system_info_t;
00504
00522 hsa_status_t HSA_API hsa_system_get_info(
00523     hsa_system_info_t attribute,
00524     void* value);
00525
00529 typedef enum {
00533   HSA_EXTENSION_FINALIZER = 0,
00537   HSA_EXTENSION_IMAGES = 1,
00538
00542   HSA_EXTENSION_PERFORMANCE_COUNTERS = 2,
00543
00547   HSA_EXTENSION_PROFILING_EVENTS = 3,
00551   HSA_EXTENSION_STD_LAST = 3,
00555   HSA_AMD_FIRST_EXTENSION = 0x200,
00559   HSA_EXTENSION_AMD_PROFILER = 0x200,
00563   HSA_EXTENSION_AMD_LOADER = 0x201,
00567   HSA_EXTENSION_AMD_AQLPROFILE = 0x202,
00571   HSA_AMD_LAST_EXTENSION = 0x202
00572 } hsa_extension_t;
00573
00592 hsa_status_t HSA_API hsa_extension_get_name(
00593     uint16_t extension,
00594     const char **name);
00595
00620 hsa_status_t HSA_API HSA_DEPRECATED hsa_system_extension_supported(
00621     uint16_t extension,
00622     uint16_t version_major,
00623     uint16_t version_minor,
00624     bool* result);
00625
00649 hsa_status_t HSA_API hsa_system_major_extension_supported(
00650     uint16_t extension,
00651     uint16_t version_major,
00652     uint16_t *version_minor,
00653     bool* result);
00654
00655
00689 hsa_status_t HSA_API HSA_DEPRECATED hsa_system_get_extension_table(
00690     uint16_t extension,
00691     uint16_t version_major,
00692     uint16_t version_minor,
00693     void *table);
00694
00729 hsa_status_t HSA_API hsa_system_get_major_extension_table(
00730     uint16_t extension,
00731     uint16_t version_major,
00732     size_t table_length,
00733     void *table);
00734
00741 typedef struct hsa_agent_s {
00746   uint64_t handle;
```

```
00747 } hsa_agent_t;
00748
00752 typedef enum {
00757     HSA_AGENT_FEATURE_KERNEL_DISPATCH = 1,
00761     HSA_AGENT_FEATURE_AGENT_DISPATCH = 2
00762 } hsa_agent_feature_t;
00763
00767 typedef enum {
00771     HSA_DEVICE_TYPE_CPU = 0,
00775     HSA_DEVICE_TYPE_GPU = 1,
00779     HSA_DEVICE_TYPE_DSP = 2
00780 } hsa_device_type_t;
00781
00785 typedef enum {
00789   HSA_DEFAULT_FLOAT_ROUNDING_MODE_DEFAULT = 0,
00794   HSA_DEFAULT_FLOAT_ROUNDING_MODE_ZERO = 1,
00800   HSA_DEFAULT_FLOAT_ROUNDING_MODE_NEAR = 2
00801 } hsa_default_float_rounding_mode_t;
00802
00806 typedef enum {
00812   HSA_AGENT_INFO_NAME = 0,
00818   HSA_AGENT_INFO_VENDOR_NAME = 1,
00822   HSA_AGENT_INFO_FEATURE = 2,
00832   HSA_AGENT_INFO_MACHINE_MODEL = 3,
00842   HSA_AGENT_INFO_PROFILE = 4,
00853   HSA_AGENT_INFO_DEFAULT_FLOAT_ROUNDING_MODE = 5,
00866   HSA_AGENT_INFO_BASE_PROFILE_DEFAULT_FLOAT_ROUNDING_MODES = 23,
00878   HSA_AGENT_INFO_FAST_F16_OPERATION = 24,
00890   HSA_AGENT_INFO_WAVEFRONT_SIZE = 6,
00903   HSA_AGENT_INFO_WORKGROUP_MAX_DIM = 7,
00914   HSA_AGENT_INFO_WORKGROUP_MAX_SIZE = 8,
00926   HSA_AGENT_INFO_GRID_MAX_DIM = 9,
00937   HSA_AGENT_INFO_GRID_MAX_SIZE = 10,
00948   HSA_AGENT_INFO_FBARRIER_MAX_SIZE = 11,
00955   HSA_AGENT_INFO_QUEUES_MAX = 12,
00962   HSA_AGENT_INFO_QUEUE_MIN_SIZE = 13,
00968   HSA_AGENT_INFO_QUEUE_MAX_SIZE = 14,
00973   HSA_AGENT_INFO_QUEUE_TYPE = 15,
00980   HSA_AGENT_INFO_NODE = 16,
00985   HSA_AGENT_INFO_DEVICE = 17,
00994   HSA_AGENT_INFO_CACHE_SIZE = 18,
01004   HSA_AGENT_INFO_ISA = 19,
01010   HSA_AGENT_INFO_EXTENSIONS = 20,
01015   HSA_AGENT_INFO_VERSION_MAJOR = 21,
01020   HSA_AGENT_INFO_VERSION_MINOR = 22,
01029   HSA_AGENT_INFO_LAST = INT32_MAX
01030 } hsa_agent_info_t;
01031
01053 hsa_status_t HSA_API hsa_agent_get_info(
01054     hsa_agent_t agent,
01055     hsa_agent_info_t attribute,
01056     void* value);
01057
01078 hsa_status_t HSA_API hsa_iterate_agents(
01079     hsa_status_t (*callback)(hsa_agent_t agent, void* data),
01080     void* data);
01081
01082 /*
01083
01084 // If we do not know the size of an attribute, we need to query it first
01085 // Note: this API will not be in the spec unless needed
01086 hsa_status_t HSA_API hsa_agent_get_info_size(
01087     hsa_agent_t agent,
01088     hsa_agent_info_t attribute,
01089     size_t* size);
01090
01091 // Set the value of an agents attribute
01092 // Note: this API will not be in the spec unless needed
01093 hsa_status_t HSA_API hsa_agent_set_info(
01094     hsa_agent_t agent,
01095     hsa_agent_info_t attribute,
01096     void* value);
01097
01098 */
01099
01103 typedef enum {
01107     HSA_EXCEPTION_POLICY_BREAK = 1,
01111     HSA_EXCEPTION_POLICY_DETECT = 2
01112 } hsa_exception_policy_t;
01113
01141 hsa_status_t HSA_API HSA_DEPRECATED hsa_agent_get_exception_policies(
01142     hsa_agent_t agent,
01143     hsa_profile_t profile,
01144     uint16_t *mask);
01145
01149 typedef struct hsa_cache_s {
01154   uint64_t handle;
```

```
01155 } hsa_cache_t;
01156
01160 typedef enum {
01165   HSA_CACHE_INFO_NAME_LENGTH = 0,
01171   HSA_CACHE_INFO_NAME = 1,
01176   HSA_CACHE_INFO_LEVEL = 2,
01181   HSA_CACHE_INFO_SIZE = 3
01182 } hsa_cache_info_t;
01183
01206 hsa_status_t HSA_API hsa_cache_get_info(
01207     hsa_cache_t cache,
01208     hsa_cache_info_t attribute,
01209     void* value);
01210
01238 hsa_status_t HSA_API hsa_agent_iterate_caches(
01239     hsa_agent_t agent,
01240     hsa_status_t (*callback)(hsa_cache_t cache, void* data),
01241     void* data);
01242
01272 hsa_status_t HSA_API HSA_DEPRECATED hsa_agent_extension_supported(
01273     uint16_t extension,
01274     hsa_agent_t agent,
01275     uint16_t version_major,
01276     uint16_t version_minor,
01277     bool* result);
01278
01307 hsa_status_t HSA_API hsa_agent_major_extension_supported(
01308     uint16_t extension,
01309     hsa_agent_t agent,
01310     uint16_t version_major,
01311     uint16_t *version_minor,
01312     bool* result);
01313
01314
01325 typedef struct hsa_signal_s {
01330   uint64_t handle;
01331 } hsa_signal_t;
01332
01337 #ifdef HSA_LARGE_MODEL
01338   typedef int64_t hsa_signal_value_t;
01339 #else
01340   typedef int32_t hsa_signal_value_t;
01341 #endif
01342
01373 hsa_status_t HSA_API hsa_signal_create(
01374     hsa_signal_value_t initial_value,
01375     uint32_t num_consumers,
01376     const hsa_agent_t *consumers,
01377     hsa_signal_t *signal);
01378
01393 hsa_status_t HSA_API hsa_signal_destroy(
01394     hsa_signal_t signal);
01395
01403 hsa_signal_value_t HSA_API hsa_signal_load_scacquire(
01404     hsa_signal_t signal);
01405
01409 hsa_signal_value_t HSA_API hsa_signal_load_relaxed(
01410     hsa_signal_t signal);
01411
01417 hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_load_acquire(
01418     hsa_signal_t signal);
01419
01430 void HSA_API hsa_signal_store_relaxed(
01431     hsa_signal_t signal,
01432     hsa_signal_value_t value);
01433
01437 void HSA_API hsa_signal_store_screlease(
01438     hsa_signal_t signal,
01439     hsa_signal_value_t value);
01440
01446 void HSA_API HSA_DEPRECATED hsa_signal_store_release(
01447     hsa_signal_t signal,
01448     hsa_signal_value_t value);
01449
01463 void HSA_API hsa_signal_silent_store_relaxed(
01464     hsa_signal_t signal,
01465     hsa_signal_value_t value);
01466
01470 void HSA_API hsa_signal_silent_store_screlease(
01471     hsa_signal_t signal,
01472     hsa_signal_value_t value);
01473
01488 hsa_signal_value_t HSA_API hsa_signal_exchange_scacq_screl(
01489     hsa_signal_t signal,
01490     hsa_signal_value_t value);
01491
01497 hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_exchange_acq_rel(
```

```
01498     hsa_signal_t signal,
01499     hsa_signal_value_t value);
01500
01504 hsa_signal_value_t HSA_API hsa_signal_exchange_scacquire(
01505     hsa_signal_t signal,
01506     hsa_signal_value_t value);
01507
01513 hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_exchange_acquire(
01514     hsa_signal_t signal,
01515     hsa_signal_value_t value);
01516
01520 hsa_signal_value_t HSA_API hsa_signal_exchange_relaxed(
01521     hsa_signal_t signal,
01522     hsa_signal_value_t value);
01526 hsa_signal_value_t HSA_API hsa_signal_exchange_screlease(
01527     hsa_signal_t signal,
01528     hsa_signal_value_t value);
01529
01535 hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_exchange_release(
01536     hsa_signal_t signal,
01537     hsa_signal_value_t value);
01538
01557 hsa_signal_value_t HSA_API hsa_signal_cas_scacq_screl(
01558     hsa_signal_t signal,
01559     hsa_signal_value_t expected,
01560     hsa_signal_value_t value);
01561
01562
01568 hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_cas_acq_rel(
01569     hsa_signal_t signal,
01570     hsa_signal_value_t expected,
01571     hsa_signal_value_t value);
01572
01576 hsa_signal_value_t HSA_API hsa_signal_cas_scacquire(
01577     hsa_signal_t signal,
01578     hsa_signal_value_t expected,
01579     hsa_signal_value_t value);
01580
01586 hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_cas_acquire(
01587     hsa_signal_t signal,
01588     hsa_signal_value_t expected,
01589     hsa_signal_value_t value);
01590
01594 hsa_signal_value_t HSA_API hsa_signal_cas_relaxed(
01595     hsa_signal_t signal,
01596     hsa_signal_value_t expected,
01597     hsa_signal_value_t value);
01598
01602 hsa_signal_value_t HSA_API hsa_signal_cas_screlease(
01603     hsa_signal_t signal,
01604     hsa_signal_value_t expected,
01605     hsa_signal_value_t value);
01606
01612 hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_cas_release(
01613     hsa_signal_t signal,
01614     hsa_signal_value_t expected,
01615     hsa_signal_value_t value);
01616
01629 void HSA_API hsa_signal_add_scacq_screl(
01630     hsa_signal_t signal,
01631     hsa_signal_value_t value);
01632
01638 void HSA_API HSA_DEPRECATED hsa_signal_add_acq_rel(
01639     hsa_signal_t signal,
01640     hsa_signal_value_t value);
01641
01645 void HSA_API hsa_signal_add_scacquire(
01646     hsa_signal_t signal,
01647     hsa_signal_value_t value);
01648
01654 void HSA_API HSA_DEPRECATED hsa_signal_add_acquire(
01655     hsa_signal_t signal,
01656     hsa_signal_value_t value);
01657
01661 void HSA_API hsa_signal_add_relaxed(
01662     hsa_signal_t signal,
01663     hsa_signal_value_t value);
01664
01668 void HSA_API hsa_signal_add_screlease(
01669     hsa_signal_t signal,
01670     hsa_signal_value_t value);
01671
01672
01678 void HSA_API HSA_DEPRECATED hsa_signal_add_release(
01679     hsa_signal_t signal,
01680     hsa_signal_value_t value);
01681
```

```
01694 void HSA_API hsa_signal_subtract_scacq_screl(
01695     hsa_signal_t signal,
01696     hsa_signal_value_t value);
01697
01698
01704 void HSA_API HSA_DEPRECATED hsa_signal_subtract_acq_rel(
01705     hsa_signal_t signal,
01706     hsa_signal_value_t value);
01707
01711 void HSA_API hsa_signal_subtract_scacquire(
01712     hsa_signal_t signal,
01713     hsa_signal_value_t value);
01714
01720 void HSA_API HSA_DEPRECATED hsa_signal_subtract_acquire(
01721     hsa_signal_t signal,
01722     hsa_signal_value_t value);
01723
01727 void HSA_API hsa_signal_subtract_relaxed(
01728     hsa_signal_t signal,
01729     hsa_signal_value_t value);
01730
01734 void HSA_API hsa_signal_subtract_screlease(
01735     hsa_signal_t signal,
01736     hsa_signal_value_t value);
01737
01738
01744 void HSA_API HSA_DEPRECATED hsa_signal_subtract_release(
01745     hsa_signal_t signal,
01746     hsa_signal_value_t value);
01747
01761 void HSA_API hsa_signal_and_scacq_screl(
01762     hsa_signal_t signal,
01763     hsa_signal_value_t value);
01764
01770 void HSA_API HSA_DEPRECATED hsa_signal_and_acq_rel(
01771     hsa_signal_t signal,
01772     hsa_signal_value_t value);
01773
01777 void HSA_API hsa_signal_and_scacquire(
01778     hsa_signal_t signal,
01779     hsa_signal_value_t value);
01780
01786 void HSA_API HSA_DEPRECATED hsa_signal_and_acquire(
01787     hsa_signal_t signal,
01788     hsa_signal_value_t value);
01789
01793 void HSA_API hsa_signal_and_relaxed(
01794     hsa_signal_t signal,
01795     hsa_signal_value_t value);
01796
01800 void HSA_API hsa_signal_and_screlease(
01801     hsa_signal_t signal,
01802     hsa_signal_value_t value);
01803
01804
01810 void HSA_API HSA_DEPRECATED hsa_signal_and_release(
01811     hsa_signal_t signal,
01812     hsa_signal_value_t value);
01813
01826 void HSA_API hsa_signal_or_scacq_screl(
01827     hsa_signal_t signal,
01828     hsa_signal_value_t value);
01829
01830
01836 void HSA_API HSA_DEPRECATED hsa_signal_or_acq_rel(
01837     hsa_signal_t signal,
01838     hsa_signal_value_t value);
01839
01843 void HSA_API hsa_signal_or_scacquire(
01844     hsa_signal_t signal,
01845     hsa_signal_value_t value);
01846
01852 void HSA_API HSA_DEPRECATED hsa_signal_or_acquire(
01853     hsa_signal_t signal,
01854     hsa_signal_value_t value);
01855
01859 void HSA_API hsa_signal_or_relaxed(
01860     hsa_signal_t signal,
01861     hsa_signal_value_t value);
01862
01866 void HSA_API hsa_signal_or_screlease(
01867     hsa_signal_t signal,
01868     hsa_signal_value_t value);
01869
01875 void HSA_API HSA_DEPRECATED hsa_signal_or_release(
01876     hsa_signal_t signal,
01877     hsa_signal_value_t value);
```

```
01878
01892 void HSA_API hsa_signal_xor_scacq_screl(
01893     hsa_signal_t signal,
01894     hsa_signal_value_t value);
01895
01896
01902 void HSA_API HSA_DEPRECATED hsa_signal_xor_acq_rel(
01903     hsa_signal_t signal,
01904     hsa_signal_value_t value);
01905
01909 void HSA_API hsa_signal_xor_scacquire(
01910     hsa_signal_t signal,
01911     hsa_signal_value_t value);
01912
01918 void HSA_API HSA_DEPRECATED hsa_signal_xor_acquire(
01919     hsa_signal_t signal,
01920     hsa_signal_value_t value);
01921
01925 void HSA_API hsa_signal_xor_relaxed(
01926     hsa_signal_t signal,
01927     hsa_signal_value_t value);
01928
01932 void HSA_API hsa_signal_xor_screlease(
01933     hsa_signal_t signal,
01934     hsa_signal_value_t value);
01935
01941 void HSA_API HSA_DEPRECATED hsa_signal_xor_release(
01942     hsa_signal_t signal,
01943     hsa_signal_value_t value);
01944
01948 typedef enum {
01952     HSA_SIGNAL_CONDITION_EQ = 0,
01956     HSA_SIGNAL_CONDITION_NE = 1,
01960     HSA_SIGNAL_CONDITION_LT = 2,
01964     HSA_SIGNAL_CONDITION_GTE = 3
01965 } hsa_signal_condition_t;
01966
01970 typedef enum {
01974     HSA_WAIT_STATE_BLOCKED = 0,
01978     HSA_WAIT_STATE_ACTIVE = 1
01979 } hsa_wait_state_t;
01980
01981
02021 hsa_signal_value_t HSA_API hsa_signal_wait_scacquire(
02022     hsa_signal_t signal,
02023     hsa_signal_condition_t condition,
02024     hsa_signal_value_t compare_value,
02025     uint64_t timeout_hint,
02026     hsa_wait_state_t wait_state_hint);
02027
02031 hsa_signal_value_t HSA_API hsa_signal_wait_relaxed(
02032     hsa_signal_t signal,
02033     hsa_signal_condition_t condition,
02034     hsa_signal_value_t compare_value,
02035     uint64_t timeout_hint,
02036     hsa_wait_state_t wait_state_hint);
02037
02043 hsa_signal_value_t HSA_API HSA_DEPRECATED hsa_signal_wait_acquire(
02044     hsa_signal_t signal,
02045     hsa_signal_condition_t condition,
02046     hsa_signal_value_t compare_value,
02047     uint64_t timeout_hint,
02048     hsa_wait_state_t wait_state_hint);
02049
02053 typedef struct hsa_signal_group_s {
02058   uint64_t handle;
02059 } hsa_signal_group_t;
02060
02093 hsa_status_t HSA_API hsa_signal_group_create(
02094     uint32_t num_signals,
02095     const hsa_signal_t *signals,
02096     uint32_t num_consumers,
02097     const hsa_agent_t *consumers,
02098     hsa_signal_group_t *signal_group);
02099
02112 hsa_status_t HSA_API hsa_signal_group_destroy(
02113     hsa_signal_group_t signal_group);
02114
02162 hsa_status_t HSA_API hsa_signal_group_wait_any_scacquire(
02163     hsa_signal_group_t signal_group,
02164     const hsa_signal_condition_t *conditions,
02165     const hsa_signal_value_t *compare_values,
02166     hsa_wait_state_t wait_state_hint,
02167     hsa_signal_t *signal,
02168     hsa_signal_value_t *value);
02169
02173 hsa_status_t HSA_API hsa_signal_group_wait_any_relaxed(
```

```
02174     hsa_signal_group_t signal_group,
02175     const hsa_signal_condition_t *conditions,
02176     const hsa_signal_value_t *compare_values,
02177     hsa_wait_state_t wait_state_hint,
02178     hsa_signal_t *signal,
02179     hsa_signal_value_t *value);
02180
02193 typedef struct hsa_region_s {
02198   uint64_t handle;
02199 } hsa_region_t;
02200
02212 typedef enum {
02217   HSA_QUEUE_TYPE_MULTI = 0,
02224   HSA_QUEUE_TYPE_SINGLE = 1,
02236   HSA_QUEUE_TYPE_COOPERATIVE = 2
02237 } hsa_queue_type_t;
02238
02242 typedef uint32_t hsa_queue_type32_t;
02243
02247 typedef enum {
02251   HSA_QUEUE_FEATURE_KERNEL_DISPATCH = 1,
02252
02256   HSA_QUEUE_FEATURE_AGENT_DISPATCH = 2
02257 } hsa_queue_feature_t;
02258
02267 typedef struct hsa_queue_s {
02271   hsa_queue_type32_t type;
02272
02277   uint32_t features;
02278
02279 #ifdef HSA_LARGE_MODEL
02280   void* base_address;
02281 #elif defined HSA_LITTLE_ENDIAN
02286   void* base_address;
02290   uint32_t reserved0;
02291 #else
02292   uint32_t reserved0;
02293   void* base_address;
02294 #endif
02295
02307   hsa_signal_t doorbell_signal;
02308
02312   uint32_t size;
02316   uint32_t reserved1;
02320   uint64_t id;
02321
02322 } hsa_queue_t;
02323
02394 hsa_status_t HSA_API hsa_queue_create(
02395     hsa_agent_t agent,
02396     uint32_t size,
02397     hsa_queue_type32_t type,
02398     void (*callback)(hsa_status_t status, hsa_queue_t *source, void *data),
02399     void *data,
02400     uint32_t private_segment_size,
02401     uint32_t group_segment_size,
02402     hsa_queue_t **queue);
02403
02458 hsa_status_t HSA_API hsa_soft_queue_create(
02459     hsa_region_t region,
02460     uint32_t size,
02461     hsa_queue_type32_t type,
02462     uint32_t features,
02463     hsa_signal_t doorbell_signal,
02464     hsa_queue_t **queue);
02465
02489 hsa_status_t HSA_API hsa_queue_destroy(
02490     hsa_queue_t *queue);
02491
02510 hsa_status_t HSA_API hsa_queue_inactivate(
02511     hsa_queue_t *queue);
02512
02518 uint64_t HSA_API HSA_DEPRECATED hsa_queue_load_read_index_acquire(
02519     const hsa_queue_t *queue);
02520
02528 uint64_t HSA_API hsa_queue_load_read_index_scacquire(
02529     const hsa_queue_t *queue);
02530
02534 uint64_t HSA_API hsa_queue_load_read_index_relaxed(
02535     const hsa_queue_t *queue);
02536
02542 uint64_t HSA_API HSA_DEPRECATED hsa_queue_load_write_index_acquire(
02543     const hsa_queue_t *queue);
02544
02552 uint64_t HSA_API hsa_queue_load_write_index_scacquire(
02553     const hsa_queue_t *queue);
02554
```

```
02558 uint64_t HSA_API hsa_queue_load_write_index_relaxed(
02559     const hsa_queue_t *queue);
02560
02573 void HSA_API hsa_queue_store_write_index_relaxed(
02574     const hsa_queue_t *queue,
02575     uint64_t value);
02576
02582 void HSA_API HSA_DEPRECATED hsa_queue_store_write_index_release(
02583     const hsa_queue_t *queue,
02584     uint64_t value);
02585
02589 void HSA_API hsa_queue_store_write_index_screlease(
02590     const hsa_queue_t *queue,
02591     uint64_t value);
02592
02598 uint64_t HSA_API HSA_DEPRECATED hsa_queue_cas_write_index_acq_rel(
02599     const hsa_queue_t *queue,
02600     uint64_t expected,
02601     uint64_t value);
02602
02617 uint64_t HSA_API hsa_queue_cas_write_index_scacq_screl(
02618     const hsa_queue_t *queue,
02619     uint64_t expected,
02620     uint64_t value);
02621
02627 uint64_t HSA_API HSA_DEPRECATED hsa_queue_cas_write_index_acquire(
02628     const hsa_queue_t *queue,
02629     uint64_t expected,
02630     uint64_t value);
02631
02635 uint64_t HSA_API hsa_queue_cas_write_index_scacquire(
02636     const hsa_queue_t *queue,
02637     uint64_t expected,
02638     uint64_t value);
02639
02643 uint64_t HSA_API hsa_queue_cas_write_index_relaxed(
02644     const hsa_queue_t *queue,
02645     uint64_t expected,
02646     uint64_t value);
02647
02653 uint64_t HSA_API HSA_DEPRECATED hsa_queue_cas_write_index_release(
02654     const hsa_queue_t *queue,
02655     uint64_t expected,
02656     uint64_t value);
02657
02661 uint64_t HSA_API hsa_queue_cas_write_index_screlease(
02662     const hsa_queue_t *queue,
02663     uint64_t expected,
02664     uint64_t value);
02665
02671 uint64_t HSA_API HSA_DEPRECATED hsa_queue_add_write_index_acq_rel(
02672     const hsa_queue_t *queue,
02673     uint64_t value);
02674
02684 uint64_t HSA_API hsa_queue_add_write_index_scacq_screl(
02685     const hsa_queue_t *queue,
02686     uint64_t value);
02687
02693 uint64_t HSA_API HSA_DEPRECATED hsa_queue_add_write_index_acquire(
02694     const hsa_queue_t *queue,
02695     uint64_t value);
02696
02700 uint64_t HSA_API hsa_queue_add_write_index_scacquire(
02701     const hsa_queue_t *queue,
02702     uint64_t value);
02703
02707 uint64_t HSA_API hsa_queue_add_write_index_relaxed(
02708     const hsa_queue_t *queue,
02709     uint64_t value);
02710
02716 uint64_t HSA_API HSA_DEPRECATED hsa_queue_add_write_index_release(
02717     const hsa_queue_t *queue,
02718     uint64_t value);
02719
02723 uint64_t HSA_API hsa_queue_add_write_index_screlease(
02724     const hsa_queue_t *queue,
02725     uint64_t value);
02726
02740 void HSA_API hsa_queue_store_read_index_relaxed(
02741     const hsa_queue_t *queue,
02742     uint64_t value);
02743
02749 void HSA_API HSA_DEPRECATED hsa_queue_store_read_index_release(
02750     const hsa_queue_t *queue,
02751     uint64_t value);
02752
02756 void HSA_API hsa_queue_store_read_index_screlease(
```

```
02757     const hsa_queue_t *queue,
02758     uint64_t value);
02769 typedef enum {
02773   HSA_PACKET_TYPE_VENDOR_SPECIFIC = 0,
02779   HSA_PACKET_TYPE_INVALID = 1,
02784   HSA_PACKET_TYPE_KERNEL_DISPATCH = 2,
02790   HSA_PACKET_TYPE_BARRIER_AND = 3,
02795   HSA_PACKET_TYPE_AGENT_DISPATCH = 4,
02801   HSA_PACKET_TYPE_BARRIER_OR = 5
02802 } hsa_packet_type_t;
02803
02807 typedef enum {
02812   HSA_FENCE_SCOPE_NONE = 0,
02816   HSA_FENCE_SCOPE_AGENT = 1,
02821   HSA_FENCE_SCOPE_SYSTEM = 2
02822 } hsa_fence_scope_t;
02823
02831  typedef enum {
02837    HSA_PACKET_HEADER_TYPE = 0,
02843    HSA_PACKET_HEADER_BARRIER = 8,
02853    HSA_PACKET_HEADER_SCACQUIRE_FENCE_SCOPE = 9,
02857    HSA_PACKET_HEADER_ACQUIRE_FENCE_SCOPE = 9,
02867    HSA_PACKET_HEADER_SCRELEASE_FENCE_SCOPE = 11,
02871    HSA_PACKET_HEADER_RELEASE_FENCE_SCOPE = 11
02872  } hsa_packet_header_t;
02873
02877  typedef enum {
02878    HSA_PACKET_HEADER_WIDTH_TYPE = 8,
02879    HSA_PACKET_HEADER_WIDTH_BARRIER = 1,
02880    HSA_PACKET_HEADER_WIDTH_SCACQUIRE_FENCE_SCOPE = 2,
02884    HSA_PACKET_HEADER_WIDTH_ACQUIRE_FENCE_SCOPE = 2,
02885    HSA_PACKET_HEADER_WIDTH_SCRELEASE_FENCE_SCOPE = 2,
02889    HSA_PACKET_HEADER_WIDTH_RELEASE_FENCE_SCOPE = 2
02890  } hsa_packet_header_width_t;
02891
02899  typedef enum {
02904    HSA_KERNEL_DISPATCH_PACKET_SETUP_DIMENSIONS = 0
02905  } hsa_kernel_dispatch_packet_setup_t;
02906
02911  typedef enum {
02912    HSA_KERNEL_DISPATCH_PACKET_SETUP_WIDTH_DIMENSIONS = 2
02913  } hsa_kernel_dispatch_packet_setup_width_t;
02914
02918 typedef struct hsa_kernel_dispatch_packet_s {
02923   uint16_t header;
02924
02930   uint16_t setup;
02931
02935   uint16_t workgroup_size_x;
02936
02941   uint16_t workgroup_size_y;
02942
02947   uint16_t workgroup_size_z;
02948
02952   uint16_t reserved0;
02953
02958   uint32_t grid_size_x;
02959
02965   uint32_t grid_size_y;
02966
02972   uint32_t grid_size_z;
02973
02977   uint32_t private_segment_size;
02978
02985   uint32_t group_segment_size;
02986
02991   uint64_t kernel_object;
02992
02993 #ifdef HSA_LARGE_MODEL
02994   void* kernarg_address;
02995 #elif defined HSA_LITTLE_ENDIAN
03003   void* kernarg_address;
03007   uint32_t reserved1;
03008 #else
03009   uint32_t reserved1;
03010   void* kernarg_address;
03011 #endif
03012
03016   uint64_t reserved2;
03017
03022   hsa_signal_t completion_signal;
03023
03024 } hsa_kernel_dispatch_packet_t;
03025
03029 typedef struct hsa_agent_dispatch_packet_s {
03034   uint16_t header;
03035
```

```
03039   uint16_t type;
03040
03044   uint32_t reserved0;
03045
03046 #ifdef HSA_LARGE_MODEL
03047   void* return_address;
03048 #elif defined HSA_LITTLE_ENDIAN
03052   void* return_address;
03056   uint32_t reserved1;
03057 #else
03058   uint32_t reserved1;
03059   void* return_address;
03060 #endif
03061
03065   uint64_t arg[4];
03066
03070   uint64_t reserved2;
03071
03076   hsa_signal_t completion_signal;
03077
03078 } hsa_agent_dispatch_packet_t;
03079
03083 typedef struct hsa_barrier_and_packet_s {
03088   uint16_t header;
03089
03093   uint16_t reserved0;
03094
03098   uint32_t reserved1;
03099
03105   hsa_signal_t dep_signal[5];
03106
03110   uint64_t reserved2;
03111
03116   hsa_signal_t completion_signal;
03117
03118 } hsa_barrier_and_packet_t;
03119
03123 typedef struct hsa_barrier_or_packet_s {
03128   uint16_t header;
03129
03133   uint16_t reserved0;
03134
03138   uint32_t reserved1;
03139
03145   hsa_signal_t dep_signal[5];
03146
03150   uint64_t reserved2;
03151
03156   hsa_signal_t completion_signal;
03157
03158 } hsa_barrier_or_packet_t;
03159
03169 typedef enum {
03173   HSA_REGION_SEGMENT_GLOBAL = 0,
03178   HSA_REGION_SEGMENT_READONLY = 1,
03182   HSA_REGION_SEGMENT_PRIVATE = 2,
03187   HSA_REGION_SEGMENT_GROUP = 3,
03191   HSA_REGION_SEGMENT_KERNARG = 4
03192 } hsa_region_segment_t;
03193
03197 typedef enum {
03203   HSA_REGION_GLOBAL_FLAG_KERNARG = 1,
03209   HSA_REGION_GLOBAL_FLAG_FINE_GRAINED = 2,
03216   HSA_REGION_GLOBAL_FLAG_COARSE_GRAINED = 4
03217 } hsa_region_global_flag_t;
03218
03222 typedef enum {
03227   HSA_REGION_INFO_SEGMENT = 0,
03234   HSA_REGION_INFO_GLOBAL_FLAGS = 1,
03238   HSA_REGION_INFO_SIZE = 2,
03252   HSA_REGION_INFO_ALLOC_MAX_SIZE = 4,
03259   HSA_REGION_INFO_ALLOC_MAX_PRIVATE_WORKGROUP_SIZE = 8,
03267   HSA_REGION_INFO_RUNTIME_ALLOC_ALLOWED = 5,
03275   HSA_REGION_INFO_RUNTIME_ALLOC_GRANULE = 6,
03282   HSA_REGION_INFO_RUNTIME_ALLOC_ALIGNMENT = 7
03283 } hsa_region_info_t;
03284
03306 hsa_status_t HSA_API hsa_region_get_info(
03307     hsa_region_t region,
03308     hsa_region_info_t attribute,
03309     void* value);
03310
03335 hsa_status_t HSA_API hsa_agent_iterate_regions(
03336     hsa_agent_t agent,
03337     hsa_status_t (*callback)(hsa_region_t region, void* data),
03338     void* data);
03339
```

```
03371 hsa_status_t HSA_API hsa_memory_allocate(hsa_region_t region,
03372     size_t size,
03373     void** ptr);
03374
03387 hsa_status_t HSA_API hsa_memory_free(void* ptr);
03388
03415 hsa_status_t HSA_API hsa_memory_copy(
03416     void *dst,
03417     const void *src,
03418     size_t size);
03419
03462 hsa_status_t HSA_API hsa_memory_assign_agent(
03463     void *ptr,
03464     hsa_agent_t agent,
03465     hsa_access_permission_t access);
03466
03503 hsa_status_t HSA_API hsa_memory_register(
03504     void *ptr,
03505     size_t size);
03506
03525 hsa_status_t HSA_API hsa_memory_deregister(
03526     void *ptr,
03527     size_t size);
03528
03539 typedef struct hsa_isa_s {
03544   uint64_t handle;
03545 } hsa_isa_t;
03546
03573 hsa_status_t HSA_API hsa_isa_from_name(
03574     const char *name,
03575     hsa_isa_t *isa);
03576
03603 hsa_status_t HSA_API hsa_agent_iterate_isas(
03604     hsa_agent_t agent,
03605     hsa_status_t (*callback)(hsa_isa_t isa, void *data),
03606     void *data);
03607
03611 typedef enum {
03616   HSA_ISA_INFO_NAME_LENGTH = 0,
03621   HSA_ISA_INFO_NAME = 1,
03628   HSA_ISA_INFO_CALL_CONVENTION_COUNT = 2,
03635   HSA_ISA_INFO_CALL_CONVENTION_INFO_WAVEFRONT_SIZE = 3,
03644   HSA_ISA_INFO_CALL_CONVENTION_INFO_WAVEFRONTS_PER_COMPUTE_UNIT = 4,
03651   HSA_ISA_INFO_MACHINE_MODELS = 5,
03658   HSA_ISA_INFO_PROFILES = 6,
03670   HSA_ISA_INFO_DEFAULT_FLOAT_ROUNDING_MODES = 7,
03680   HSA_ISA_INFO_BASE_PROFILE_DEFAULT_FLOAT_ROUNDING_MODES = 8,
03686   HSA_ISA_INFO_FAST_F16_OPERATION = 9,
03693   HSA_ISA_INFO_WORKGROUP_MAX_DIM = 12,
03698   HSA_ISA_INFO_WORKGROUP_MAX_SIZE = 13,
03706   HSA_ISA_INFO_GRID_MAX_DIM = 14,
03711   HSA_ISA_INFO_GRID_MAX_SIZE = 16,
03716   HSA_ISA_INFO_FBARRIER_MAX_SIZE = 17
03717 } hsa_isa_info_t;
03718
03756 hsa_status_t HSA_API HSA_DEPRECATED hsa_isa_get_info(
03757     hsa_isa_t isa,
03758     hsa_isa_info_t attribute,
03759     uint32_t index,
03760     void *value);
03761
03786 hsa_status_t HSA_API hsa_isa_get_info_alt(
03787     hsa_isa_t isa,
03788     hsa_isa_info_t attribute,
03789     void *value);
03790
03813 hsa_status_t HSA_API hsa_isa_get_exception_policies(
03814     hsa_isa_t isa,
03815     hsa_profile_t profile,
03816     uint16_t *mask);
03817
03821 typedef enum {
03825   HSA_FP_TYPE_16 = 1,
03829   HSA_FP_TYPE_32 = 2,
03833   HSA_FP_TYPE_64 = 4
03834 } hsa_fp_type_t;
03835
03839 typedef enum {
03843   HSA_FLUSH_MODE_FTZ = 1,
03847   HSA_FLUSH_MODE_NON_FTZ = 2
03848 } hsa_flush_mode_t;
03849
03853 typedef enum {
03857   HSA_ROUND_METHOD_SINGLE = 1,
03861   HSA_ROUND_METHOD_DOUBLE = 2
03862 } hsa_round_method_t;
03863
```

```
03891 hsa_status_t HSA_API hsa_isa_get_round_method(
03892     hsa_isa_t isa,
03893     hsa_fp_type_t fp_type,
03894     hsa_flush_mode_t flush_mode,
03895     hsa_round_method_t *round_method);
03896
03900 typedef struct hsa_wavefront_s {
03905   uint64_t handle;
03906 } hsa_wavefront_t;
03907
03911 typedef enum {
03916   HSA_WAVEFRONT_INFO_SIZE = 0
03917 } hsa_wavefront_info_t;
03918
03940 hsa_status_t HSA_API hsa_wavefront_get_info(
03941     hsa_wavefront_t wavefront,
03942     hsa_wavefront_info_t attribute,
03943     void *value);
03944
03970 hsa_status_t HSA_API hsa_isa_iterate_wavefronts(
03971     hsa_isa_t isa,
03972     hsa_status_t (*callback)(hsa_wavefront_t wavefront, void *data),
03973     void *data);
03974
04001 hsa_status_t HSA_API HSA_DEPRECATED hsa_isa_compatible(
04002     hsa_isa_t code_object_isa,
04003     hsa_isa_t agent_isa,
04004     bool *result);
04005
04019 typedef struct hsa_code_object_reader_s {
04024   uint64_t handle;
04025 } hsa_code_object_reader_t;
04026
04052 hsa_status_t HSA_API hsa_code_object_reader_create_from_file(
04053     hsa_file_t file,
04054     hsa_code_object_reader_t *code_object_reader);
04055
04080 hsa_status_t HSA_API hsa_code_object_reader_create_from_memory(
04081     const void *code_object,
04082     size_t size,
04083     hsa_code_object_reader_t *code_object_reader);
04084
04102 hsa_status_t HSA_API hsa_code_object_reader_destroy(
04103     hsa_code_object_reader_t code_object_reader);
04104
04110 typedef struct hsa_executable_s {
04115   uint64_t handle;
04116 } hsa_executable_t;
04117
04121 typedef enum {
04128   HSA_EXECUTABLE_STATE_UNFROZEN = 0,
04135   HSA_EXECUTABLE_STATE_FROZEN = 1
04136 } hsa_executable_state_t;
04137
04171 hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_create(
04172     hsa_profile_t profile,
04173     hsa_executable_state_t executable_state,
04174     const char *options,
04175     hsa_executable_t *executable);
04176
04207 hsa_status_t HSA_API hsa_executable_create_alt(
04208     hsa_profile_t profile,
04209     hsa_default_float_rounding_mode_t default_float_rounding_mode,
04210     const char *options,
04211     hsa_executable_t *executable);
04212
04234 hsa_status_t HSA_API hsa_executable_destroy(
04235     hsa_executable_t executable);
04236
04240 typedef struct hsa_loaded_code_object_s {
04245   uint64_t handle;
04246 } hsa_loaded_code_object_t;
04247
04292 hsa_status_t HSA_API hsa_executable_load_program_code_object(
04293     hsa_executable_t executable,
04294     hsa_code_object_reader_t code_object_reader,
04295     const char *options,
04296     hsa_loaded_code_object_t *loaded_code_object);
04297
04358 hsa_status_t HSA_API hsa_executable_load_agent_code_object(
04359     hsa_executable_t executable,
04360     hsa_agent_t agent,
04361     hsa_code_object_reader_t code_object_reader,
04362     const char *options,
04363     hsa_loaded_code_object_t *loaded_code_object);
04364
04394 hsa_status_t HSA_API hsa_executable_freeze(
```

```
04395     hsa_executable_t executable,
04396     const char *options);
04397
04401 typedef enum {
04406   HSA_EXECUTABLE_INFO_PROFILE = 1,
04410   HSA_EXECUTABLE_INFO_STATE = 2,
04415   HSA_EXECUTABLE_INFO_DEFAULT_FLOAT_ROUNDING_MODE = 3
04416 } hsa_executable_info_t;
04417
04439 hsa_status_t HSA_API hsa_executable_get_info(
04440     hsa_executable_t executable,
04441     hsa_executable_info_t attribute,
04442     void *value);
04443
04482 hsa_status_t HSA_API hsa_executable_global_variable_define(
04483     hsa_executable_t executable,
04484     const char *variable_name,
04485     void *address);
04486
04529 hsa_status_t HSA_API hsa_executable_agent_global_variable_define(
04530     hsa_executable_t executable,
04531     hsa_agent_t agent,
04532     const char *variable_name,
04533     void *address);
04534
04581 hsa_status_t HSA_API hsa_executable_readonly_variable_define(
04582     hsa_executable_t executable,
04583     hsa_agent_t agent,
04584     const char *variable_name,
04585     void *address);
04586
04609 hsa_status_t HSA_API hsa_executable_validate(
04610     hsa_executable_t executable,
04611     uint32_t *result);
04612
04640 hsa_status_t HSA_API hsa_executable_validate_alt(
04641     hsa_executable_t executable,
04642     const char *options,
04643     uint32_t *result);
04644
04652 typedef struct hsa_executable_symbol_s {
04657   uint64_t handle;
04658 } hsa_executable_symbol_t;
04659
04695 hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_get_symbol(
04696     hsa_executable_t executable,
04697     const char *module_name,
04698     const char *symbol_name,
04699     hsa_agent_t agent,
04700     int32_t call_convention,
04701     hsa_executable_symbol_t *symbol);
04702
04732 hsa_status_t HSA_API hsa_executable_get_symbol_by_name(
04733     hsa_executable_t executable,
04734     const char *symbol_name,
04735     const hsa_agent_t *agent,
04736     hsa_executable_symbol_t *symbol);
04737
04741 typedef enum {
04745   HSA_SYMBOL_KIND_VARIABLE = 0,
04749   HSA_SYMBOL_KIND_KERNEL = 1,
04753   HSA_SYMBOL_KIND_INDIRECT_FUNCTION = 2
04754 } hsa_symbol_kind_t;
04755
04759 typedef enum {
04763   HSA_SYMBOL_LINKAGE_MODULE = 0,
04767   HSA_SYMBOL_LINKAGE_PROGRAM = 1
04768 } hsa_symbol_linkage_t;
04769
04773 typedef enum {
04777   HSA_VARIABLE_ALLOCATION_AGENT = 0,
04781   HSA_VARIABLE_ALLOCATION_PROGRAM = 1
04782 } hsa_variable_allocation_t;
04783
04787 typedef enum {
04791   HSA_VARIABLE_SEGMENT_GLOBAL = 0,
04795   HSA_VARIABLE_SEGMENT_READONLY = 1
04796 } hsa_variable_segment_t;
04797
04801 typedef enum {
04805   HSA_EXECUTABLE_SYMBOL_INFO_TYPE = 0,
04810   HSA_EXECUTABLE_SYMBOL_INFO_NAME_LENGTH = 1,
04816   HSA_EXECUTABLE_SYMBOL_INFO_NAME = 2,
04824   HSA_EXECUTABLE_SYMBOL_INFO_MODULE_NAME_LENGTH = 3,
04833   HSA_EXECUTABLE_SYMBOL_INFO_MODULE_NAME = 4,
04842   HSA_EXECUTABLE_SYMBOL_INFO_AGENT = 20,
04850   HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_ADDRESS = 21,
```

```
04855   HSA_EXECUTABLE_SYMBOL_INFO_LINKAGE = 5,
04860   HSA_EXECUTABLE_SYMBOL_INFO_IS_DEFINITION = 17,
04868   HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_ALLOCATION = 6,
04876   HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_SEGMENT = 7,
04886   HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_ALIGNMENT = 8,
04896   HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_SIZE = 9,
04904   HSA_EXECUTABLE_SYMBOL_INFO_VARIABLE_IS_CONST = 10,
04913   HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_OBJECT = 22,
04920   HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_KERNARG_SEGMENT_SIZE = 11,
04927   HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_KERNARG_SEGMENT_ALIGNMENT = 12,
04937   HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_GROUP_SEGMENT_SIZE = 13,
04949   HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_PRIVATE_SEGMENT_SIZE = 14,
04958   HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_DYNAMIC_CALLSTACK = 15,
04965   HSA_EXECUTABLE_SYMBOL_INFO_KERNEL_CALL_CONVENTION = 18,
04976   HSA_EXECUTABLE_SYMBOL_INFO_INDIRECT_FUNCTION_OBJECT = 23,
04985   HSA_EXECUTABLE_SYMBOL_INFO_INDIRECT_FUNCTION_CALL_CONVENTION = 16
04986 } hsa_executable_symbol_info_t;
04987
05010 hsa_status_t HSA_API hsa_executable_symbol_get_info(
05011     hsa_executable_symbol_t executable_symbol,
05012     hsa_executable_symbol_info_t attribute,
05013     void *value);
05014
05041 hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_iterate_symbols(
05042     hsa_executable_t executable,
05043     hsa_status_t (*callback)(hsa_executable_t exec,
05044                              hsa_executable_symbol_t symbol,
05045                              void *data),
05046     void *data);
05047
05075 hsa_status_t HSA_API hsa_executable_iterate_agent_symbols(
05076     hsa_executable_t executable,
05077     hsa_agent_t agent,
05078     hsa_status_t (*callback)(hsa_executable_t exec,
05079                              hsa_agent_t agent,
05080                              hsa_executable_symbol_t symbol,
05081                              void *data),
05082     void *data);
05083
05108 hsa_status_t HSA_API hsa_executable_iterate_program_symbols(
05109     hsa_executable_t executable,
05110     hsa_status_t (*callback)(hsa_executable_t exec,
05111                              hsa_executable_symbol_t symbol,
05112                              void *data),
05113     void *data);
05114
05129 typedef struct hsa_code_object_s {
05134   uint64_t handle;
05135 } hsa_code_object_t;
05136
05143 typedef struct hsa_callback_data_s {
05147   uint64_t handle;
05148 } hsa_callback_data_t;
05149
05196 hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_serialize(
05197     hsa_code_object_t code_object,
05198     hsa_status_t (*alloc_callback)(size_t size,
05199                                    hsa_callback_data_t data,
05200                                    void **address),
05201     hsa_callback_data_t callback_data,
05202     const char *options,
05203     void **serialized_code_object,
05204     size_t *serialized_code_object_size);
05205
05236 hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_deserialize(
05237     void *serialized_code_object,
05238     size_t serialized_code_object_size,
05239     const char *options,
05240     hsa_code_object_t *code_object);
05241
05261 hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_destroy(
05262     hsa_code_object_t code_object);
05263
05269 typedef enum {
05274   HSA_CODE_OBJECT_TYPE_PROGRAM = 0
05275 } hsa_code_object_type_t;
05276
05282 typedef enum {
05289   HSA_CODE_OBJECT_INFO_VERSION = 0,
05294   HSA_CODE_OBJECT_INFO_TYPE = 1,
05299   HSA_CODE_OBJECT_INFO_ISA = 2,
05304   HSA_CODE_OBJECT_INFO_MACHINE_MODEL = 3,
05309   HSA_CODE_OBJECT_INFO_PROFILE = 4,
05315   HSA_CODE_OBJECT_INFO_DEFAULT_FLOAT_ROUNDING_MODE = 5
05316 } hsa_code_object_info_t;
05317
05341 hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_get_info(
```

```
05342     hsa_code_object_t code_object,
05343     hsa_code_object_info_t attribute,
05344     void *value);
05345
05397 hsa_status_t HSA_API HSA_DEPRECATED hsa_executable_load_code_object(
05398     hsa_executable_t executable,
05399     hsa_agent_t agent,
05400     hsa_code_object_t code_object,
05401     const char *options);
05402
05412 typedef struct hsa_code_symbol_s {
05417   uint64_t handle;
05418 } hsa_code_symbol_t;
05419
05445 hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_get_symbol(
05446     hsa_code_object_t code_object,
05447     const char *symbol_name,
05448     hsa_code_symbol_t *symbol);
05449
05478 hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_get_symbol_from_name(
05479     hsa_code_object_t code_object,
05480     const char *module_name,
05481     const char *symbol_name,
05482     hsa_code_symbol_t *symbol);
05483
05489 typedef enum {
05493   HSA_CODE_SYMBOL_INFO_TYPE = 0,
05498   HSA_CODE_SYMBOL_INFO_NAME_LENGTH = 1,
05504   HSA_CODE_SYMBOL_INFO_NAME = 2,
05510   HSA_CODE_SYMBOL_INFO_MODULE_NAME_LENGTH = 3,
05517   HSA_CODE_SYMBOL_INFO_MODULE_NAME = 4,
05522   HSA_CODE_SYMBOL_INFO_LINKAGE = 5,
05527   HSA_CODE_SYMBOL_INFO_IS_DEFINITION = 17,
05533   HSA_CODE_SYMBOL_INFO_VARIABLE_ALLOCATION = 6,
05539   HSA_CODE_SYMBOL_INFO_VARIABLE_SEGMENT = 7,
05547   HSA_CODE_SYMBOL_INFO_VARIABLE_ALIGNMENT = 8,
05555   HSA_CODE_SYMBOL_INFO_VARIABLE_SIZE = 9,
05561   HSA_CODE_SYMBOL_INFO_VARIABLE_IS_CONST = 10,
05568   HSA_CODE_SYMBOL_INFO_KERNEL_KERNARG_SEGMENT_SIZE = 11,
05575   HSA_CODE_SYMBOL_INFO_KERNEL_KERNARG_SEGMENT_ALIGNMENT = 12,
05585   HSA_CODE_SYMBOL_INFO_KERNEL_GROUP_SEGMENT_SIZE = 13,
05597   HSA_CODE_SYMBOL_INFO_KERNEL_PRIVATE_SEGMENT_SIZE = 14,
05606   HSA_CODE_SYMBOL_INFO_KERNEL_DYNAMIC_CALLSTACK = 15,
05611   HSA_CODE_SYMBOL_INFO_KERNEL_CALL_CONVENTION = 18,
05617   HSA_CODE_SYMBOL_INFO_INDIRECT_FUNCTION_CALL_CONVENTION = 16
05618 } hsa_code_symbol_info_t;
05619
05643 hsa_status_t HSA_API HSA_DEPRECATED hsa_code_symbol_get_info(
05644     hsa_code_symbol_t code_symbol,
05645     hsa_code_symbol_info_t attribute,
05646     void *value);
05647
05674 hsa_status_t HSA_API HSA_DEPRECATED hsa_code_object_iterate_symbols(
05675     hsa_code_object_t code_object,
05676     hsa_status_t (*callback)(hsa_code_object_t code_object,
05677                              hsa_code_symbol_t symbol,
05678                              void *data),
05679     void *data);
05680
05683 #ifdef __cplusplus
05684 }  // end extern "C" block
05685 #endif
05686
05687 #endif  // header guard
```

7.8 hsa_api_trace.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
```

```
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 #ifndef HSA_RUNTIME_INC_HSA_API_TRACE_H
00044 #define HSA_RUNTIME_INC_HSA_API_TRACE_H
00045
00046 #include "hsa.h"
00047 #ifdef AMD_INTERNAL_BUILD
00048 #include "hsa_ext_image.h"
00049 #include "hsa_ext_amd.h"
00050 #include "hsa_ext_finalize.h"
00051 #else
00052 #include "inc/hsa_ext_image.h"
00053 #include "inc/hsa_ext_amd.h"
00054 #include "inc/hsa_ext_finalize.h"
00055 #endif
00056
00057 #include <string.h>
00058 #include <assert.h>
00059 #include <stddef.h>
00060
00061 // Major Ids of the Api tables exported by Hsa Core Runtime
00062 #define HSA_API_TABLE_MAJOR_VERSION               0x01
00063 #define HSA_CORE_API_TABLE_MAJOR_VERSION          0x01
00064 #define HSA_AMD_EXT_API_TABLE_MAJOR_VERSION       0x01
00065 #define HSA_FINALIZER_API_TABLE_MAJOR_VERSION     0x01
00066 #define HSA_IMAGE_API_TABLE_MAJOR_VERSION         0x01
00067 #define HSA_AQLPROFILE_API_TABLE_MAJOR_VERSION    0x01
00068
00069 // Step Ids of the Api tables exported by Hsa Core Runtime
00070 #define HSA_API_TABLE_STEP_VERSION                0x00
00071 #define HSA_CORE_API_TABLE_STEP_VERSION           0x00
00072 #define HSA_AMD_EXT_API_TABLE_STEP_VERSION        0x00
00073 #define HSA_FINALIZER_API_TABLE_STEP_VERSION      0x00
00074 #define HSA_IMAGE_API_TABLE_STEP_VERSION          0x00
00075 #define HSA_AQLPROFILE_API_TABLE_STEP_VERSION     0x00
00076
00077 // Min function used to copy Api Tables
00078 static inline uint32_t Min(const uint32_t a, const uint32_t b) {
00079   return (a > b) ? b : a;
00080 }
00081
00082 // Declarations of APIs intended for use only by tools.
00083 typedef void (*hsa_amd_queue_intercept_packet_writer)(const void* pkts, uint64_t pkt_count);
00084 typedef void (*hsa_amd_queue_intercept_handler)(const void* pkts, uint64_t pkt_count,
00085                                                 uint64_t user_pkt_index, void* data,
00086                                                 hsa_amd_queue_intercept_packet_writer writer);
00087 hsa_status_t hsa_amd_queue_intercept_register(hsa_queue_t* queue,
00088                                               hsa_amd_queue_intercept_handler callback,
00089                                               void* user_data);
00090 hsa_status_t hsa_amd_queue_intercept_create(
00091     hsa_agent_t agent_handle, uint32_t size, hsa_queue_type32_t type,
00092     void (*callback)(hsa_status_t status, hsa_queue_t* source, void* data), void* data,
00093     uint32_t private_segment_size, uint32_t group_segment_size, hsa_queue_t** queue);
00094
00095 typedef void (*hsa_amd_runtime_queue_notifier)(const hsa_queue_t* queue, hsa_agent_t agent,
00096                                                void* data);
00097 hsa_status_t hsa_amd_runtime_queue_create_register(hsa_amd_runtime_queue_notifier callback,
00098                                                    void* user_data);
00099
00100 // Structure of Version used to identify an instance of Api table
00101 // Must be the first member (offsetof == 0) of all API tables.
00102 // This is the root of the table passing ABI.
00103 struct ApiTableVersion {
00104   uint32_t major_id;
```

```
00105   uint32_t minor_id;
00106   uint32_t step_id;
00107   uint32_t reserved;
00108 };
00109
00110 // Table to export HSA Finalizer Extension Apis
00111 struct FinalizerExtTable {
00112   ApiTableVersion version;
00113     decltype(hsa_ext_program_create)* hsa_ext_program_create_fn;
00114     decltype(hsa_ext_program_destroy)* hsa_ext_program_destroy_fn;
00115     decltype(hsa_ext_program_add_module)* hsa_ext_program_add_module_fn;
00116     decltype(hsa_ext_program_iterate_modules)* hsa_ext_program_iterate_modules_fn;
00117     decltype(hsa_ext_program_get_info)* hsa_ext_program_get_info_fn;
00118     decltype(hsa_ext_program_finalize)* hsa_ext_program_finalize_fn;
00119 };
00120
00121 // Table to export HSA Image Extension Apis
00122 struct ImageExtTable {
00123   ApiTableVersion version;
00124     decltype(hsa_ext_image_get_capability)* hsa_ext_image_get_capability_fn;
00125     decltype(hsa_ext_image_data_get_info)* hsa_ext_image_data_get_info_fn;
00126     decltype(hsa_ext_image_create)* hsa_ext_image_create_fn;
00127     decltype(hsa_ext_image_import)* hsa_ext_image_import_fn;
00128     decltype(hsa_ext_image_export)* hsa_ext_image_export_fn;
00129     decltype(hsa_ext_image_copy)* hsa_ext_image_copy_fn;
00130     decltype(hsa_ext_image_clear)* hsa_ext_image_clear_fn;
00131     decltype(hsa_ext_image_destroy)* hsa_ext_image_destroy_fn;
00132     decltype(hsa_ext_sampler_create)* hsa_ext_sampler_create_fn;
00133     decltype(hsa_ext_sampler_destroy)* hsa_ext_sampler_destroy_fn;
00134   decltype(hsa_ext_image_get_capability_with_layout)* hsa_ext_image_get_capability_with_layout_fn;
00135   decltype(hsa_ext_image_data_get_info_with_layout)* hsa_ext_image_data_get_info_with_layout_fn;
00136   decltype(hsa_ext_image_create_with_layout)* hsa_ext_image_create_with_layout_fn;
00137 };
00138
00139 // Table to export AMD Extension Apis
00140 struct AmdExtTable {
00141   ApiTableVersion version;
00142     decltype(hsa_amd_coherency_get_type)* hsa_amd_coherency_get_type_fn;
00143     decltype(hsa_amd_coherency_set_type)* hsa_amd_coherency_set_type_fn;
00144   decltype(hsa_amd_profiling_set_profiler_enabled)* hsa_amd_profiling_set_profiler_enabled_fn;
00145   decltype(hsa_amd_profiling_async_copy_enable) *hsa_amd_profiling_async_copy_enable_fn;
00146   decltype(hsa_amd_profiling_get_dispatch_time)* hsa_amd_profiling_get_dispatch_time_fn;
00147   decltype(hsa_amd_profiling_get_async_copy_time) *hsa_amd_profiling_get_async_copy_time_fn;
00148   decltype(hsa_amd_profiling_convert_tick_to_system_domain)*
      hsa_amd_profiling_convert_tick_to_system_domain_fn;
00149   decltype(hsa_amd_signal_async_handler)* hsa_amd_signal_async_handler_fn;
00150   decltype(hsa_amd_async_function)* hsa_amd_async_function_fn;
00151   decltype(hsa_amd_signal_wait_any)* hsa_amd_signal_wait_any_fn;
00152   decltype(hsa_amd_queue_cu_set_mask)* hsa_amd_queue_cu_set_mask_fn;
00153   decltype(hsa_amd_memory_pool_get_info)* hsa_amd_memory_pool_get_info_fn;
00154   decltype(hsa_amd_agent_iterate_memory_pools)* hsa_amd_agent_iterate_memory_pools_fn;
00155   decltype(hsa_amd_memory_pool_allocate)* hsa_amd_memory_pool_allocate_fn;
00156   decltype(hsa_amd_memory_pool_free)* hsa_amd_memory_pool_free_fn;
00157   decltype(hsa_amd_memory_async_copy)* hsa_amd_memory_async_copy_fn;
00158   decltype(hsa_amd_agent_memory_pool_get_info)* hsa_amd_agent_memory_pool_get_info_fn;
00159   decltype(hsa_amd_agents_allow_access)* hsa_amd_agents_allow_access_fn;
00160   decltype(hsa_amd_memory_pool_can_migrate)* hsa_amd_memory_pool_can_migrate_fn;
00161   decltype(hsa_amd_memory_migrate)* hsa_amd_memory_migrate_fn;
00162   decltype(hsa_amd_memory_lock)* hsa_amd_memory_lock_fn;
00163   decltype(hsa_amd_memory_unlock)* hsa_amd_memory_unlock_fn;
00164   decltype(hsa_amd_memory_fill)* hsa_amd_memory_fill_fn;
00165   decltype(hsa_amd_interop_map_buffer)* hsa_amd_interop_map_buffer_fn;
00166   decltype(hsa_amd_interop_unmap_buffer)* hsa_amd_interop_unmap_buffer_fn;
00167   decltype(hsa_amd_image_create)* hsa_amd_image_create_fn;
00168   decltype(hsa_amd_pointer_info)* hsa_amd_pointer_info_fn;
00169   decltype(hsa_amd_pointer_info_set_userdata)* hsa_amd_pointer_info_set_userdata_fn;
00170   decltype(hsa_amd_ipc_memory_create)* hsa_amd_ipc_memory_create_fn;
00171   decltype(hsa_amd_ipc_memory_attach)* hsa_amd_ipc_memory_attach_fn;
00172   decltype(hsa_amd_ipc_memory_detach)* hsa_amd_ipc_memory_detach_fn;
00173   decltype(hsa_amd_signal_create)* hsa_amd_signal_create_fn;
00174   decltype(hsa_amd_ipc_signal_create)* hsa_amd_ipc_signal_create_fn;
00175   decltype(hsa_amd_ipc_signal_attach)* hsa_amd_ipc_signal_attach_fn;
00176   decltype(hsa_amd_register_system_event_handler)* hsa_amd_register_system_event_handler_fn;
00177   decltype(hsa_amd_queue_intercept_create)* hsa_amd_queue_intercept_create_fn;
00178   decltype(hsa_amd_queue_intercept_register)* hsa_amd_queue_intercept_register_fn;
00179   decltype(hsa_amd_queue_set_priority)* hsa_amd_queue_set_priority_fn;
00180   decltype(hsa_amd_memory_async_copy_rect)* hsa_amd_memory_async_copy_rect_fn;
00181   decltype(hsa_amd_runtime_queue_create_register)* hsa_amd_runtime_queue_create_register_fn;
00182   decltype(hsa_amd_memory_lock_to_pool)* hsa_amd_memory_lock_to_pool_fn;
00183   decltype(hsa_amd_register_deallocation_callback)* hsa_amd_register_deallocation_callback_fn;
00184   decltype(hsa_amd_deregister_deallocation_callback)* hsa_amd_deregister_deallocation_callback_fn;
00185   decltype(hsa_amd_signal_value_pointer)* hsa_amd_signal_value_pointer_fn;
00186   decltype(hsa_amd_svm_attributes_set)* hsa_amd_svm_attributes_set_fn;
00187   decltype(hsa_amd_svm_attributes_get)* hsa_amd_svm_attributes_get_fn;
00188   decltype(hsa_amd_svm_prefetch_async)* hsa_amd_svm_prefetch_async_fn;
00189   decltype(hsa_amd_queue_cu_get_mask)* hsa_amd_queue_cu_get_mask_fn;
00190 };
```

```
00191
00192 // Table to export HSA Core Runtime Apis
00193 struct CoreApiTable {
00194   ApiTableVersion version;
00195   decltype(hsa_init)* hsa_init_fn;
00196   decltype(hsa_shut_down)* hsa_shut_down_fn;
00197   decltype(hsa_system_get_info)* hsa_system_get_info_fn;
00198   decltype(hsa_system_extension_supported)* hsa_system_extension_supported_fn;
00199   decltype(hsa_system_get_extension_table)* hsa_system_get_extension_table_fn;
00200   decltype(hsa_iterate_agents)* hsa_iterate_agents_fn;
00201   decltype(hsa_agent_get_info)* hsa_agent_get_info_fn;
00202   decltype(hsa_queue_create)* hsa_queue_create_fn;
00203   decltype(hsa_soft_queue_create)* hsa_soft_queue_create_fn;
00204   decltype(hsa_queue_destroy)* hsa_queue_destroy_fn;
00205   decltype(hsa_queue_inactivate)* hsa_queue_inactivate_fn;
00206   decltype(hsa_queue_load_read_index_scacquire)* hsa_queue_load_read_index_scacquire_fn;
00207   decltype(hsa_queue_load_read_index_relaxed)* hsa_queue_load_read_index_relaxed_fn;
00208   decltype(hsa_queue_load_write_index_scacquire)* hsa_queue_load_write_index_scacquire_fn;
00209   decltype(hsa_queue_load_write_index_relaxed)* hsa_queue_load_write_index_relaxed_fn;
00210   decltype(hsa_queue_store_write_index_relaxed)* hsa_queue_store_write_index_relaxed_fn;
00211   decltype(hsa_queue_store_write_index_screlease)* hsa_queue_store_write_index_screlease_fn;
00212   decltype(hsa_queue_cas_write_index_scacq_screl)* hsa_queue_cas_write_index_scacq_screl_fn;
00213   decltype(hsa_queue_cas_write_index_scacquire)* hsa_queue_cas_write_index_scacquire_fn;
00214   decltype(hsa_queue_cas_write_index_relaxed)* hsa_queue_cas_write_index_relaxed_fn;
00215   decltype(hsa_queue_cas_write_index_screlease)* hsa_queue_cas_write_index_screlease_fn;
00216   decltype(hsa_queue_add_write_index_scacq_screl)* hsa_queue_add_write_index_scacq_screl_fn;
00217   decltype(hsa_queue_add_write_index_scacquire)* hsa_queue_add_write_index_scacquire_fn;
00218   decltype(hsa_queue_add_write_index_relaxed)* hsa_queue_add_write_index_relaxed_fn;
00219   decltype(hsa_queue_add_write_index_screlease)* hsa_queue_add_write_index_screlease_fn;
00220   decltype(hsa_queue_store_read_index_relaxed)* hsa_queue_store_read_index_relaxed_fn;
00221   decltype(hsa_queue_store_read_index_screlease)* hsa_queue_store_read_index_screlease_fn;
00222   decltype(hsa_agent_iterate_regions)* hsa_agent_iterate_regions_fn;
00223   decltype(hsa_region_get_info)* hsa_region_get_info_fn;
00224   decltype(hsa_agent_get_exception_policies)* hsa_agent_get_exception_policies_fn;
00225   decltype(hsa_agent_extension_supported)* hsa_agent_extension_supported_fn;
00226   decltype(hsa_memory_register)* hsa_memory_register_fn;
00227   decltype(hsa_memory_deregister)* hsa_memory_deregister_fn;
00228   decltype(hsa_memory_allocate)* hsa_memory_allocate_fn;
00229   decltype(hsa_memory_free)* hsa_memory_free_fn;
00230   decltype(hsa_memory_copy)* hsa_memory_copy_fn;
00231   decltype(hsa_memory_assign_agent)* hsa_memory_assign_agent_fn;
00232   decltype(hsa_signal_create)* hsa_signal_create_fn;
00233   decltype(hsa_signal_destroy)* hsa_signal_destroy_fn;
00234   decltype(hsa_signal_load_relaxed)* hsa_signal_load_relaxed_fn;
00235   decltype(hsa_signal_load_scacquire)* hsa_signal_load_scacquire_fn;
00236   decltype(hsa_signal_store_relaxed)* hsa_signal_store_relaxed_fn;
00237   decltype(hsa_signal_store_screlease)* hsa_signal_store_screlease_fn;
00238   decltype(hsa_signal_wait_relaxed)* hsa_signal_wait_relaxed_fn;
00239   decltype(hsa_signal_wait_scacquire)* hsa_signal_wait_scacquire_fn;
00240   decltype(hsa_signal_and_relaxed)* hsa_signal_and_relaxed_fn;
00241   decltype(hsa_signal_and_scacquire)* hsa_signal_and_scacquire_fn;
00242   decltype(hsa_signal_and_screlease)* hsa_signal_and_screlease_fn;
00243   decltype(hsa_signal_and_scacq_screl)* hsa_signal_and_scacq_screl_fn;
00244   decltype(hsa_signal_or_relaxed)* hsa_signal_or_relaxed_fn;
00245   decltype(hsa_signal_or_scacquire)* hsa_signal_or_scacquire_fn;
00246   decltype(hsa_signal_or_screlease)* hsa_signal_or_screlease_fn;
00247   decltype(hsa_signal_or_scacq_screl)* hsa_signal_or_scacq_screl_fn;
00248   decltype(hsa_signal_xor_relaxed)* hsa_signal_xor_relaxed_fn;
00249   decltype(hsa_signal_xor_scacquire)* hsa_signal_xor_scacquire_fn;
00250   decltype(hsa_signal_xor_screlease)* hsa_signal_xor_screlease_fn;
00251   decltype(hsa_signal_xor_scacq_screl)* hsa_signal_xor_scacq_screl_fn;
00252   decltype(hsa_signal_exchange_relaxed)* hsa_signal_exchange_relaxed_fn;
00253   decltype(hsa_signal_exchange_scacquire)* hsa_signal_exchange_scacquire_fn;
00254   decltype(hsa_signal_exchange_screlease)* hsa_signal_exchange_screlease_fn;
00255   decltype(hsa_signal_exchange_scacq_screl)* hsa_signal_exchange_scacq_screl_fn;
00256   decltype(hsa_signal_add_relaxed)* hsa_signal_add_relaxed_fn;
00257   decltype(hsa_signal_add_scacquire)* hsa_signal_add_scacquire_fn;
00258   decltype(hsa_signal_add_screlease)* hsa_signal_add_screlease_fn;
00259   decltype(hsa_signal_add_scacq_screl)* hsa_signal_add_scacq_screl_fn;
00260   decltype(hsa_signal_subtract_relaxed)* hsa_signal_subtract_relaxed_fn;
00261   decltype(hsa_signal_subtract_scacquire)* hsa_signal_subtract_scacquire_fn;
00262   decltype(hsa_signal_subtract_screlease)* hsa_signal_subtract_screlease_fn;
00263   decltype(hsa_signal_subtract_scacq_screl)* hsa_signal_subtract_scacq_screl_fn;
00264   decltype(hsa_signal_cas_relaxed)* hsa_signal_cas_relaxed_fn;
00265   decltype(hsa_signal_cas_scacquire)* hsa_signal_cas_scacquire_fn;
00266   decltype(hsa_signal_cas_screlease)* hsa_signal_cas_screlease_fn;
00267   decltype(hsa_signal_cas_scacq_screl)* hsa_signal_cas_scacq_screl_fn;
00268
00269   //====-- Instruction Set Architecture ----------------------------------===//
00270
00271   decltype(hsa_isa_from_name)* hsa_isa_from_name_fn;
00272   // Deprecated since v1.1.
00273   decltype(hsa_isa_get_info)* hsa_isa_get_info_fn;
00274   // Deprecated since v1.1.
00275   decltype(hsa_isa_compatible)* hsa_isa_compatible_fn;
00276
00277   //====-- Code Objects (deprecated) -------------------------------------===//
```

```
00278
00279   // Deprecated since v1.1.
00280   decltype(hsa_code_object_serialize)* hsa_code_object_serialize_fn;
00281   // Deprecated since v1.1.
00282   decltype(hsa_code_object_deserialize)* hsa_code_object_deserialize_fn;
00283   // Deprecated since v1.1.
00284   decltype(hsa_code_object_destroy)* hsa_code_object_destroy_fn;
00285   // Deprecated since v1.1.
00286   decltype(hsa_code_object_get_info)* hsa_code_object_get_info_fn;
00287   // Deprecated since v1.1.
00288   decltype(hsa_code_object_get_symbol)* hsa_code_object_get_symbol_fn;
00289   // Deprecated since v1.1.
00290   decltype(hsa_code_symbol_get_info)* hsa_code_symbol_get_info_fn;
00291   // Deprecated since v1.1.
00292   decltype(hsa_code_object_iterate_symbols)* hsa_code_object_iterate_symbols_fn;
00293
00294   //====== Executable ========================================================//
00295
00296   // Deprecated since v1.1.
00297   decltype(hsa_executable_create)* hsa_executable_create_fn;
00298   decltype(hsa_executable_destroy)* hsa_executable_destroy_fn;
00299   // Deprecated since v1.1.
00300   decltype(hsa_executable_load_code_object)* hsa_executable_load_code_object_fn;
00301   decltype(hsa_executable_freeze)* hsa_executable_freeze_fn;
00302   decltype(hsa_executable_get_info)* hsa_executable_get_info_fn;
00303   decltype(hsa_executable_global_variable_define)*
00304       hsa_executable_global_variable_define_fn;
00305   decltype(hsa_executable_agent_global_variable_define)*
00306       hsa_executable_agent_global_variable_define_fn;
00307   decltype(hsa_executable_readonly_variable_define)*
00308       hsa_executable_readonly_variable_define_fn;
00309   decltype(hsa_executable_validate)* hsa_executable_validate_fn;
00310   // Deprecated since v1.1.
00311   decltype(hsa_executable_get_symbol)* hsa_executable_get_symbol_fn;
00312   decltype(hsa_executable_symbol_get_info)* hsa_executable_symbol_get_info_fn;
00313   // Deprecated since v1.1.
00314   decltype(hsa_executable_iterate_symbols)* hsa_executable_iterate_symbols_fn;
00315
00316   //====== Runtime Notifications =============================================//
00317
00318   decltype(hsa_status_string)* hsa_status_string_fn;
00319
00320   // Start HSA v1.1 additions
00321   decltype(hsa_extension_get_name)* hsa_extension_get_name_fn;
00322   decltype(hsa_system_major_extension_supported)* hsa_system_major_extension_supported_fn;
00323   decltype(hsa_system_get_major_extension_table)* hsa_system_get_major_extension_table_fn;
00324   decltype(hsa_agent_major_extension_supported)* hsa_agent_major_extension_supported_fn;
00325   decltype(hsa_cache_get_info)* hsa_cache_get_info_fn;
00326   decltype(hsa_agent_iterate_caches)* hsa_agent_iterate_caches_fn;
00327   decltype(hsa_signal_silent_store_relaxed)* hsa_signal_silent_store_relaxed_fn;
00328   decltype(hsa_signal_silent_store_screlease)* hsa_signal_silent_store_screlease_fn;
00329   decltype(hsa_signal_group_create)* hsa_signal_group_create_fn;
00330   decltype(hsa_signal_group_destroy)* hsa_signal_group_destroy_fn;
00331   decltype(hsa_signal_group_wait_any_scacquire)* hsa_signal_group_wait_any_scacquire_fn;
00332   decltype(hsa_signal_group_wait_any_relaxed)* hsa_signal_group_wait_any_relaxed_fn;
00333
00334   //====== Instruction Set Architecture - HSA v1.1 additions ================//
00335
00336   decltype(hsa_agent_iterate_isas)* hsa_agent_iterate_isas_fn;
00337   decltype(hsa_isa_get_info_alt)* hsa_isa_get_info_alt_fn;
00338   decltype(hsa_isa_get_exception_policies)* hsa_isa_get_exception_policies_fn;
00339   decltype(hsa_isa_get_round_method)* hsa_isa_get_round_method_fn;
00340   decltype(hsa_wavefront_get_info)* hsa_wavefront_get_info_fn;
00341   decltype(hsa_isa_iterate_wavefronts)* hsa_isa_iterate_wavefronts_fn;
00342
00343   //====== Code Objects (deprecated) - HSA v1.1 additions ====================//
00344
00345   // Deprecated since v1.1.
00346   decltype(hsa_code_object_get_symbol_from_name)*
00347       hsa_code_object_get_symbol_from_name_fn;
00348
00349   //====== Executable - HSA v1.1 additions ===================================//
00350
00351   decltype(hsa_code_object_reader_create_from_file)*
00352       hsa_code_object_reader_create_from_file_fn;
00353   decltype(hsa_code_object_reader_create_from_memory)*
00354       hsa_code_object_reader_create_from_memory_fn;
00355   decltype(hsa_code_object_reader_destroy)* hsa_code_object_reader_destroy_fn;
00356   decltype(hsa_executable_create_alt)* hsa_executable_create_alt_fn;
00357   decltype(hsa_executable_load_program_code_object)*
00358       hsa_executable_load_program_code_object_fn;
00359   decltype(hsa_executable_load_agent_code_object)*
00360       hsa_executable_load_agent_code_object_fn;
00361   decltype(hsa_executable_validate_alt)* hsa_executable_validate_alt_fn;
00362   decltype(hsa_executable_get_symbol_by_name)*
00363       hsa_executable_get_symbol_by_name_fn;
00364   decltype(hsa_executable_iterate_agent_symbols)*
```

```
00365       hsa_executable_iterate_agent_symbols_fn;
00366   decltype(hsa_executable_iterate_program_symbols)*
00367       hsa_executable_iterate_program_symbols_fn;
00368 };
00369
00370 // Table to export HSA Apis from Core Runtime, Amd Extensions
00371 // Finalizer and Images
00372 struct HsaApiTable {
00373
00374   // Version of Hsa Api Table
00375   ApiTableVersion version;
00376
00377   // Table of function pointers to HSA Core Runtime
00378     CoreApiTable* core_;
00379
00380   // Table of function pointers to AMD extensions
00381     AmdExtTable* amd_ext_;
00382
00383   // Table of function pointers to HSA Finalizer Extension
00384     FinalizerExtTable* finalizer_ext_;
00385
00386   // Table of function pointers to HSA Image Extension
00387     ImageExtTable* image_ext_;
00388 };
00389
00390 // Structure containing instances of different api tables
00391 struct HsaApiTableContainer {
00392   HsaApiTable root;
00393     CoreApiTable core;
00394     AmdExtTable amd_ext;
00395     FinalizerExtTable finalizer_ext;
00396     ImageExtTable image_ext;
00397
00398   // Default initialization of a container instance
00399   HsaApiTableContainer() {
00400     root.version.major_id = HSA_API_TABLE_MAJOR_VERSION;
00401     root.version.minor_id = sizeof(HsaApiTable);
00402     root.version.step_id = HSA_API_TABLE_STEP_VERSION;
00403
00404     core.version.major_id = HSA_CORE_API_TABLE_MAJOR_VERSION;
00405     core.version.minor_id = sizeof(CoreApiTable);
00406     core.version.step_id = HSA_CORE_API_TABLE_STEP_VERSION;
00407     root.core_ = &core;
00408
00409     amd_ext.version.major_id = HSA_AMD_EXT_API_TABLE_MAJOR_VERSION;
00410     amd_ext.version.minor_id = sizeof(AmdExtTable);
00411     amd_ext.version.step_id = HSA_AMD_EXT_API_TABLE_STEP_VERSION;
00412     root.amd_ext_ = &amd_ext;
00413
00414     finalizer_ext.version.major_id = HSA_FINALIZER_API_TABLE_MAJOR_VERSION;
00415     finalizer_ext.version.minor_id = sizeof(FinalizerExtTable);
00416     finalizer_ext.version.step_id = HSA_FINALIZER_API_TABLE_STEP_VERSION;
00417     root.finalizer_ext_ = & finalizer_ext;
00418
00419     image_ext.version.major_id = HSA_IMAGE_API_TABLE_MAJOR_VERSION;
00420     image_ext.version.minor_id = sizeof(ImageExtTable);
00421     image_ext.version.step_id = HSA_IMAGE_API_TABLE_STEP_VERSION;
00422     root.image_ext_ = &image_ext;
00423   }
00424 };
00425
00426 // Api to copy function pointers of a table
00427 static
00428 void inline copyApi(void* src, void* dest, size_t size) {
00429   assert(size >= sizeof(ApiTableVersion));
00430   memcpy((char*)src + sizeof(ApiTableVersion),
00431          (char*)dest + sizeof(ApiTableVersion),
00432          (size - sizeof(ApiTableVersion)));
00433 }
00434
00435 // Copy Api child tables if valid.
00436 static void inline copyElement(ApiTableVersion* dest, ApiTableVersion* src) {
00437   if (src->major_id && (dest->major_id == src->major_id)) {
00438     dest->step_id = src->step_id;
00439     dest->minor_id = Min(dest->minor_id, src->minor_id);
00440     copyApi(dest, src, dest->minor_id);
00441   } else {
00442     dest->major_id = 0;
00443     dest->minor_id = 0;
00444     dest->step_id = 0;
00445   }
00446 }
00447
00448 // Copy constructor for all Api tables. The function assumes the
00449 // user has initialized an instance of tables container correctly
00450 // for the Major, Minor and Stepping Ids of Root and Child Api tables.
00451 // The function will overwrite the value of Minor Id by taking the
```

```
00452 // minimum of source and destination parameters. It will also overwrite
00453 // the stepping Id with value from source parameter.
00454 static void inline copyTables(const HsaApiTable* src, HsaApiTable* dest) {
00455   // Verify Major Id of source and destination tables match
00456   if (dest->version.major_id != src->version.major_id) {
00457     dest->version.major_id = 0;
00458     dest->version.minor_id = 0;
00459     dest->version.step_id = 0;
00460     return;
00461   }
00462
00463   // Initialize the stepping id and minor id of root table. For the
00464   // minor id which encodes struct size, take the minimum of source
00465   // and destination parameters
00466   dest->version.step_id = src->version.step_id;
00467   dest->version.minor_id = Min(dest->version.minor_id, src->version.minor_id);
00468
00469   // Copy child tables if present
00470   if ((offsetof(HsaApiTable, core_) < dest->version.minor_id))
00471     copyElement(&dest->core_->version, &src->core_->version);
00472   if ((offsetof(HsaApiTable, amd_ext_) < dest->version.minor_id))
00473     copyElement(&dest->amd_ext_->version, &src->amd_ext_->version);
00474   if ((offsetof(HsaApiTable, finalizer_ext_) < dest->version.minor_id))
00475     copyElement(&dest->finalizer_ext_->version, &src->finalizer_ext_->version);
00476   if ((offsetof(HsaApiTable, image_ext_) < dest->version.minor_id))
00477     copyElement(&dest->image_ext_->version, &src->image_ext_->version);
00478 }
00479 #endif
```

7.9 hsa_ext_amd.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 // HSA AMD extension.
00044
00045 #ifndef HSA_RUNTIME_EXT_AMD_H_
00046 #define HSA_RUNTIME_EXT_AMD_H_
00047
00048 #include "hsa.h"
00049 #include "hsa_ext_image.h"
00050
00051 #define HSA_AMD_INTERFACE_VERSION_MAJOR 1
00052 #define HSA_AMD_INTERFACE_VERSION_MINOR 0
00053
```

```
00054 #ifdef __cplusplus
00055 extern "C" {
00056 #endif
00057
00065 typedef uint32_t hsa_signal_condition32_t;
00066
00070 typedef enum {
00076   HSA_AMD_PACKET_TYPE_BARRIER_VALUE = 2,
00077 } hsa_amd_packet_type_t;
00078
00082 typedef uint8_t hsa_amd_packet_type8_t;
00083
00087 typedef struct hsa_amd_packet_header_s {
00092   uint16_t header;
00093
00097   hsa_amd_packet_type8_t AmdFormat;
00098
00102   uint8_t reserved;
00103 } hsa_amd_vendor_packet_header_t;
00104
00110 typedef struct hsa_amd_barrier_value_packet_s {
00114   hsa_amd_vendor_packet_header_t header;
00115
00119   uint32_t reserved0;
00120
00126   hsa_signal_t signal;
00127
00131   hsa_signal_value_t value;
00132
00136   hsa_signal_value_t mask;
00137
00141   hsa_signal_condition32_t cond;
00142
00146   uint32_t reserved1;
00147
00151   uint64_t reserved2;
00152
00156   uint64_t reserved3;
00157
00162   hsa_signal_t completion_signal;
00163 } hsa_amd_barrier_value_packet_t;
00164
00172 enum {
00176   HSA_STATUS_ERROR_INVALID_MEMORY_POOL = 40,
00177
00181   HSA_STATUS_ERROR_MEMORY_APERTURE_VIOLATION = 41,
00182
00186   HSA_STATUS_ERROR_ILLEGAL_INSTRUCTION = 42,
00187
00193   HSA_STATUS_ERROR_MEMORY_FAULT = 43,
00194
00200   HSA_STATUS_CU_MASK_REDUCED = 44,
00201 };
00202
00206 typedef enum hsa_amd_agent_info_s {
00210   HSA_AMD_AGENT_INFO_CHIP_ID = 0xA000,
00214   HSA_AMD_AGENT_INFO_CACHELINE_SIZE = 0xA001,
00219   HSA_AMD_AGENT_INFO_COMPUTE_UNIT_COUNT = 0xA002,
00224   HSA_AMD_AGENT_INFO_MAX_CLOCK_FREQUENCY = 0xA003,
00228   HSA_AMD_AGENT_INFO_DRIVER_NODE_ID = 0xA004,
00234   HSA_AMD_AGENT_INFO_MAX_ADDRESS_WATCH_POINTS = 0xA005,
00239   HSA_AMD_AGENT_INFO_BDFID = 0xA006,
00244   HSA_AMD_AGENT_INFO_MEMORY_WIDTH = 0xA007,
00248   HSA_AMD_AGENT_INFO_MEMORY_MAX_FREQUENCY = 0xA008,
00253   HSA_AMD_AGENT_INFO_PRODUCT_NAME = 0xA009,
00258   HSA_AMD_AGENT_INFO_MAX_WAVES_PER_CU = 0xA00A,
00263   HSA_AMD_AGENT_INFO_NUM_SIMDS_PER_CU = 0xA00B,
00268   HSA_AMD_AGENT_INFO_NUM_SHADER_ENGINES = 0xA00C,
00273   HSA_AMD_AGENT_INFO_NUM_SHADER_ARRAYS_PER_SE = 0xA00D,
00279   HSA_AMD_AGENT_INFO_HDP_FLUSH = 0xA00E,
00285   HSA_AMD_AGENT_INFO_DOMAIN = 0xA00F,
00290   HSA_AMD_AGENT_INFO_COOPERATIVE_QUEUES = 0xA010,
00300   HSA_AMD_AGENT_INFO_UUID = 0xA011,
00308   HSA_AMD_AGENT_INFO_ASIC_REVISION = 0xA012,
00314   HSA_AMD_AGENT_INFO_SVM_DIRECT_HOST_ACCESS = 0xA013,
00321   HSA_AMD_AGENT_INFO_COOPERATIVE_COMPUTE_UNIT_COUNT = 0xA014,
00327   HSA_AMD_AGENT_INFO_MEMORY_AVAIL = 0xA015,
00333   HSA_AMD_AGENT_INFO_TIMESTAMP_FREQUENCY = 0xA016,
00338   HSA_AMD_AGENT_INFO_ASIC_FAMILY_ID = 0xA107
00339
00340 } hsa_amd_agent_info_t;
00341
00342 typedef struct hsa_amd_hdp_flush_s {
00343   uint32_t* HDP_MEM_FLUSH_CNTL;
00344   uint32_t* HDP_REG_FLUSH_CNTL;
00345 } hsa_amd_hdp_flush_t;
00346
```

```
00350 typedef enum hsa_amd_region_info_s {
00355   HSA_AMD_REGION_INFO_HOST_ACCESSIBLE = 0xA000,
00359   HSA_AMD_REGION_INFO_BASE = 0xA001,
00364   HSA_AMD_REGION_INFO_BUS_WIDTH = 0xA002,
00369   HSA_AMD_REGION_INFO_MAX_CLOCK_FREQUENCY = 0xA003
00370 } hsa_amd_region_info_t;
00371
00375 typedef enum hsa_amd_coherency_type_s {
00379   HSA_AMD_COHERENCY_TYPE_COHERENT = 0,
00383   HSA_AMD_COHERENCY_TYPE_NONCOHERENT = 1
00384 } hsa_amd_coherency_type_t;
00385
00403 hsa_status_t HSA_API hsa_amd_coherency_get_type(hsa_agent_t agent,
00404                                                 hsa_amd_coherency_type_t* type);
00405
00424 hsa_status_t HSA_API hsa_amd_coherency_set_type(hsa_agent_t agent,
00425                                                 hsa_amd_coherency_type_t type);
00426
00433 typedef struct hsa_amd_profiling_dispatch_time_s {
00437   uint64_t start;
00441   uint64_t end;
00442 } hsa_amd_profiling_dispatch_time_t;
00443
00450 typedef struct hsa_amd_profiling_async_copy_time_s {
00454   uint64_t start;
00458   uint64_t end;
00459 } hsa_amd_profiling_async_copy_time_t;
00460
00477 hsa_status_t HSA_API
00478     hsa_amd_profiling_set_profiler_enabled(hsa_queue_t* queue, int enable);
00479
00500 hsa_status_t HSA_API
00501     hsa_amd_profiling_async_copy_enable(bool enable);
00502
00530 hsa_status_t HSA_API hsa_amd_profiling_get_dispatch_time(
00531     hsa_agent_t agent, hsa_signal_t signal,
00532     hsa_amd_profiling_dispatch_time_t* time);
00533
00555 hsa_status_t HSA_API hsa_amd_profiling_get_async_copy_time(
00556     hsa_signal_t signal, hsa_amd_profiling_async_copy_time_t* time);
00557
00579 hsa_status_t HSA_API
00580     hsa_amd_profiling_convert_tick_to_system_domain(hsa_agent_t agent,
00581                                                     uint64_t agent_tick,
00582                                                     uint64_t* system_tick);
00583
00587 typedef enum {
00592   HSA_AMD_SIGNAL_AMD_GPU_ONLY = 1,
00600   HSA_AMD_SIGNAL_IPC = 2,
00601 } hsa_amd_signal_attribute_t;
00602
00638 hsa_status_t HSA_API hsa_amd_signal_create(hsa_signal_value_t initial_value, uint32_t num_consumers,
00639                                           const hsa_agent_t* consumers, uint64_t attributes,
00640                                           hsa_signal_t* signal);
00641
00670 hsa_status_t hsa_amd_signal_value_pointer(hsa_signal_t signal,
00671                                          volatile hsa_signal_value_t** value_ptr);
00672
00697 typedef bool (*hsa_amd_signal_handler)(hsa_signal_value_t value, void* arg);
00698
00740 hsa_status_t HSA_API
00741     hsa_amd_signal_async_handler(hsa_signal_t signal,
00742                                  hsa_signal_condition_t cond,
00743                                  hsa_signal_value_t value,
00744                                  hsa_amd_signal_handler handler, void* arg);
00745
00770 hsa_status_t HSA_API
00771     hsa_amd_async_function(void (*callback)(void* arg), void* arg);
00772
00782 uint32_t HSA_API
00783     hsa_amd_signal_wait_any(uint32_t signal_count, hsa_signal_t* signals,
00784                             hsa_signal_condition_t* conds,
00785                             hsa_signal_value_t* values, uint64_t timeout_hint,
00786                             hsa_wait_state_t wait_hint,
00787                             hsa_signal_value_t* satisfying_value);
00788
00810 hsa_status_t HSA_API hsa_amd_image_get_info_max_dim(hsa_agent_t agent,
00811                                                     hsa_agent_info_t attribute,
00812                                                     void* value);
00813
00847 hsa_status_t HSA_API hsa_amd_queue_cu_set_mask(const hsa_queue_t* queue,
00848                                                uint32_t num_cu_mask_count,
00849                                                const uint32_t* cu_mask);
00850
00875 hsa_status_t HSA_API hsa_amd_queue_cu_get_mask(const hsa_queue_t* queue, uint32_t num_cu_mask_count,
00876                                                uint32_t* cu_mask);
00877
```

```
00881 typedef enum {
00885   HSA_AMD_SEGMENT_GLOBAL = 0,
00890   HSA_AMD_SEGMENT_READONLY = 1,
00894   HSA_AMD_SEGMENT_PRIVATE = 2,
00899   HSA_AMD_SEGMENT_GROUP = 3,
00900 } hsa_amd_segment_t;
00901
00923 typedef struct hsa_amd_memory_pool_s {
00927   uint64_t handle;
00928 } hsa_amd_memory_pool_t;
00929
00930 typedef enum hsa_amd_memory_pool_global_flag_s {
00936   HSA_AMD_MEMORY_POOL_GLOBAL_FLAG_KERNARG_INIT = 1,
00942   HSA_AMD_MEMORY_POOL_GLOBAL_FLAG_FINE_GRAINED = 2,
00946   HSA_AMD_MEMORY_POOL_GLOBAL_FLAG_COARSE_GRAINED = 4
00947 } hsa_amd_memory_pool_global_flag_t;
00948
00952 typedef enum {
00957   HSA_AMD_MEMORY_POOL_INFO_SEGMENT = 0,
00966   HSA_AMD_MEMORY_POOL_INFO_GLOBAL_FLAGS = 1,
00970   HSA_AMD_MEMORY_POOL_INFO_SIZE = 2,
00978   HSA_AMD_MEMORY_POOL_INFO_RUNTIME_ALLOC_ALLOWED = 5,
00987   HSA_AMD_MEMORY_POOL_INFO_RUNTIME_ALLOC_GRANULE = 6,
00994   HSA_AMD_MEMORY_POOL_INFO_RUNTIME_ALLOC_ALIGNMENT = 7,
01001   HSA_AMD_MEMORY_POOL_INFO_ACCESSIBLE_BY_ALL = 15,
01006   HSA_AMD_MEMORY_POOL_INFO_ALLOC_MAX_SIZE = 16,
01007 } hsa_amd_memory_pool_info_t;
01008
01013 typedef enum hsa_amd_memory_pool_flag_s {
01017   HSA_AMD_MEMORY_POOL_STANDARD_FLAG = 0,
01023   HSA_AMD_MEMORY_POOL_PCIE_FLAG = 1,
01024
01025 } hsa_amd_memory_pool_flag_t;
01026
01041 hsa_status_t HSA_API
01042     hsa_amd_memory_pool_get_info(hsa_amd_memory_pool_t memory_pool,
01043                                  hsa_amd_memory_pool_info_t attribute,
01044                                  void* value);
01045
01079 hsa_status_t HSA_API hsa_amd_agent_iterate_memory_pools(
01080     hsa_agent_t agent,
01081     hsa_status_t (*callback)(hsa_amd_memory_pool_t memory_pool, void* data),
01082     void* data);
01083
01120 hsa_status_t HSA_API
01121     hsa_amd_memory_pool_allocate(hsa_amd_memory_pool_t memory_pool, size_t size,
01122                                  uint32_t flags, void** ptr);
01123
01137 hsa_status_t HSA_API hsa_amd_memory_pool_free(void* ptr);
01138
01197 hsa_status_t HSA_API
01198     hsa_amd_memory_async_copy(void* dst, hsa_agent_t dst_agent, const void* src,
01199                               hsa_agent_t src_agent, size_t size,
01200                               uint32_t num_dep_signals,
01201                               const hsa_signal_t* dep_signals,
01202                               hsa_signal_t completion_signal);
01203
01204 /*
01205 [Provisional API]
01206 Pitched memory descriptor.
01207 All elements must be 4 byte aligned.  Pitch and slice are in bytes.
01208 */
01209 typedef struct hsa_pitched_ptr_s {
01210   void* base;
01211   size_t pitch;
01212   size_t slice;
01213 } hsa_pitched_ptr_t;
01214
01215 /*
01216 [Provisional API]
01217 Copy direction flag.
01218 */
01219 typedef enum {
01220   hsaHostToHost = 0,
01221   hsaHostToDevice = 1,
01222   hsaDeviceToHost = 2,
01223   hsaDeviceToDevice = 3
01224 } hsa_amd_copy_direction_t;
01225
01226 /*
01227 [Provisional API]
01228 SDMA 3D memory copy API.  The same requirements must be met by src and dst as in
01229 hsa_amd_memory_async_copy.
01230 Both src and dst must be directly accessible to the copy_agent during the copy, src and dst rects
01231 must not overlap.
01232 CPU agents are not supported.  API requires SDMA and will return an error if SDMA is not available.
01233 Offsets and range carry x in bytes, y and z in rows and layers.
```

```
01234 */
01235 hsa_status_t HSA_API hsa_amd_memory_async_copy_rect(
01236     const hsa_pitched_ptr_t* dst, const hsa_dim3_t* dst_offset, const hsa_pitched_ptr_t* src,
01237     const hsa_dim3_t* src_offset, const hsa_dim3_t* range, hsa_agent_t copy_agent,
01238     hsa_amd_copy_direction_t dir, uint32_t num_dep_signals, const hsa_signal_t* dep_signals,
01239     hsa_signal_t completion_signal);
01240
01244 typedef enum {
01248   HSA_AMD_MEMORY_POOL_ACCESS_NEVER_ALLOWED = 0,
01253   HSA_AMD_MEMORY_POOL_ACCESS_ALLOWED_BY_DEFAULT = 1,
01259   HSA_AMD_MEMORY_POOL_ACCESS_DISALLOWED_BY_DEFAULT = 2
01260 } hsa_amd_memory_pool_access_t;
01261
01265 typedef enum {
01269   HSA_AMD_LINK_INFO_TYPE_HYPERTRANSPORT = 0,
01270
01274   HSA_AMD_LINK_INFO_TYPE_QPI = 1,
01275
01279   HSA_AMD_LINK_INFO_TYPE_PCIE = 2,
01280
01284   HSA_AMD_LINK_INFO_TYPE_INFINBAND = 3,
01285
01289   HSA_AMD_LINK_INFO_TYPE_XGMI = 4
01290
01291 } hsa_amd_link_info_type_t;
01292
01297 typedef struct hsa_amd_memory_pool_link_info_s {
01301   uint32_t min_latency;
01302
01306   uint32_t max_latency;
01307
01311   uint32_t min_bandwidth;
01312
01316   uint32_t max_bandwidth;
01317
01321   bool atomic_support_32bit;
01322
01326   bool atomic_support_64bit;
01327
01331   bool coherent_support;
01332
01336   hsa_amd_link_info_type_t link_type;
01337
01341   uint32_t numa_distance;
01342 } hsa_amd_memory_pool_link_info_t;
01343
01347 typedef enum {
01368   HSA_AMD_AGENT_MEMORY_POOL_INFO_ACCESS = 0,
01369
01377   HSA_AMD_AGENT_MEMORY_POOL_INFO_NUM_LINK_HOPS = 1,
01378
01385   HSA_AMD_AGENT_MEMORY_POOL_INFO_LINK_INFO = 2
01386
01387 } hsa_amd_agent_memory_pool_info_t;
01388
01406 hsa_status_t HSA_API hsa_amd_agent_memory_pool_get_info(
01407     hsa_agent_t agent, hsa_amd_memory_pool_t memory_pool,
01408     hsa_amd_agent_memory_pool_info_t attribute, void* value);
01409
01447 hsa_status_t HSA_API
01448     hsa_amd_agents_allow_access(uint32_t num_agents, const hsa_agent_t* agents,
01449                                 const uint32_t* flags, const void* ptr);
01450
01478 hsa_status_t HSA_API
01479     hsa_amd_memory_pool_can_migrate(hsa_amd_memory_pool_t src_memory_pool,
01480                                     hsa_amd_memory_pool_t dst_memory_pool,
01481                                     bool* result);
01482
01519 hsa_status_t HSA_API hsa_amd_memory_migrate(const void* ptr,
01520                                             hsa_amd_memory_pool_t memory_pool,
01521                                             uint32_t flags);
01522
01557 hsa_status_t HSA_API hsa_amd_memory_lock(void* host_ptr, size_t size,
01558                                         hsa_agent_t* agents, int num_agent,
01559                                         void** agent_ptr);
01560
01606 hsa_status_t HSA_API hsa_amd_memory_lock_to_pool(void* host_ptr, size_t size, hsa_agent_t* agents,
01607                                                 int num_agent, hsa_amd_memory_pool_t pool,
01608                                                 uint32_t flags, void** agent_ptr);
01609
01626 hsa_status_t HSA_API hsa_amd_memory_unlock(void* host_ptr);
01627
01650 hsa_status_t HSA_API
01651     hsa_amd_memory_fill(void* ptr, uint32_t value, size_t count);
01652
01689 hsa_status_t HSA_API hsa_amd_interop_map_buffer(uint32_t num_agents,
01690                                               hsa_agent_t* agents,
```

```
01691                                         int interop_handle,
01692                                         uint32_t flags,
01693                                         size_t* size,
01694                                         void** ptr,
01695                                         size_t* metadata_size,
01696                                         const void** metadata);
01697
01702 hsa_status_t HSA_API hsa_amd_interop_unmap_buffer(void* ptr);
01703
01709 typedef struct hsa_amd_image_descriptor_s {
01710   /*
01711   Version number of the descriptor
01712   */
01713   uint32_t version;
01714
01715   /*
01716   Vendor and device PCI IDs for the format as VENDOR_ID«16|DEVICE_ID.
01717   */
01718   uint32_t deviceID;
01719
01720   /*
01721   Start of vendor specific data.
01722   */
01723   uint32_t data[1];
01724 } hsa_amd_image_descriptor_t;
01725
01751 hsa_status_t HSA_API hsa_amd_image_create(
01752     hsa_agent_t agent,
01753     const hsa_ext_image_descriptor_t *image_descriptor,
01754     const hsa_amd_image_descriptor_t *image_layout,
01755     const void *image_data,
01756     hsa_access_permission_t access_permission,
01757     hsa_ext_image_t *image
01758 );
01759
01763 typedef enum {
01764   /*
01765   Memory is not known to the HSA driver.  Unallocated or unlocked system memory.
01766   */
01767   HSA_EXT_POINTER_TYPE_UNKNOWN = 0,
01768   /*
01769   Memory was allocated with an HSA memory allocator.
01770   */
01771   HSA_EXT_POINTER_TYPE_HSA = 1,
01772   /*
01773   System memory which has been locked for use with an HSA agent.
01774
01775   Memory of this type is normal malloc'd memory and is always accessible to
01776   the CPU.  Pointer info queries may not include CPU agents in the accessible
01777   agents list as the CPU has implicit access.
01778   */
01779   HSA_EXT_POINTER_TYPE_LOCKED = 2,
01780   /*
01781   Memory originated in a graphics component and is shared with ROCr.
01782   */
01783   HSA_EXT_POINTER_TYPE_GRAPHICS = 3,
01784   /*
01785   Memory has been shared with the local process via ROCr IPC APIs.
01786   */
01787   HSA_EXT_POINTER_TYPE_IPC = 4
01788 } hsa_amd_pointer_type_t;
01789
01794 typedef struct hsa_amd_pointer_info_s {
01795   /*
01796   Size in bytes of this structure.  Used for version control within a major ROCr
01797   revision.  Set to sizeof(hsa_amd_pointer_t) prior to calling
01798   hsa_amd_pointer_info.  If the runtime supports an older version of pointer
01799   info then size will be smaller on return.  Members starting after the return
01800   value of size will not be updated by hsa_amd_pointer_info.
01801   */
01802   uint32_t size;
01803   /*
01804   The type of allocation referenced.
01805   */
01806   hsa_amd_pointer_type_t type;
01807   /*
01808   Base address at which non-host agents may access the allocation.
01809   */
01810   void* agentBaseAddress;
01811   /*
01812   Base address at which the host agent may access the allocation.
01813   */
01814   void* hostBaseAddress;
01815   /*
01816   Size of the allocation
01817   */
01818   size_t sizeInBytes;
```

```
01819   /*
01820   Application provided value.
01821   */
01822   void* userData;
01823   /*
01824   Reports an agent which "owns" (ie has preferred access to) the pool in which the allocation was
01825   made.  When multiple agents share equal access to a pool (ex: multiple CPU agents, or multi-die
01826   GPU boards) any such agent may be returned.
01827   */
01828   hsa_agent_t agentOwner;
01829   /*
01830   Contains a bitfield of hsa_amd_memory_pool_global_flag_t values.
01831   Reports the effective global flags bitmask for the allocation.  This field is not meaningful if
01832   the type of the allocation is HSA_EXT_POINTER_TYPE_UNKNOWN.
01833   */
01834   uint32_t global_flags;
01835 } hsa_amd_pointer_info_t;
01836
01870 hsa_status_t HSA_API hsa_amd_pointer_info(const void* ptr,
01871                                           hsa_amd_pointer_info_t* info,
01872                                           void* (*alloc)(size_t),
01873                                           uint32_t* num_agents_accessible,
01874                                           hsa_agent_t** accessible);
01875
01894 hsa_status_t HSA_API hsa_amd_pointer_info_set_userdata(const void* ptr,
01895                                                        void* userdata);
01896
01901 typedef struct hsa_amd_ipc_memory_s {
01902   uint32_t handle[8];
01903 } hsa_amd_ipc_memory_t;
01904
01934 hsa_status_t HSA_API hsa_amd_ipc_memory_create(void* ptr, size_t len,
01935                                                hsa_amd_ipc_memory_t* handle);
01936
01968 hsa_status_t HSA_API hsa_amd_ipc_memory_attach(
01969     const hsa_amd_ipc_memory_t* handle, size_t len,
01970     uint32_t num_agents,
01971     const hsa_agent_t* mapping_agents,
01972     void** mapped_ptr);
01973
01989 hsa_status_t HSA_API hsa_amd_ipc_memory_detach(void* mapped_ptr);
01990
01994 typedef hsa_amd_ipc_memory_t hsa_amd_ipc_signal_t;
01995
02020 hsa_status_t HSA_API hsa_amd_ipc_signal_create(hsa_signal_t signal, hsa_amd_ipc_signal_t* handle);
02021
02042 hsa_status_t HSA_API hsa_amd_ipc_signal_attach(const hsa_amd_ipc_signal_t* handle,
02043                                                hsa_signal_t* signal);
02044
02048 typedef enum hsa_amd_event_type_s {
02049   /*
02050    AMD GPU memory fault.
02051    */
02052   HSA_AMD_GPU_MEMORY_FAULT_EVENT = 0,
02053 } hsa_amd_event_type_t;
02054
02058 typedef enum {
02059   // Page not present or supervisor privilege.
02060   HSA_AMD_MEMORY_FAULT_PAGE_NOT_PRESENT = 1 << 0,
02061   // Write access to a read-only page.
02062   HSA_AMD_MEMORY_FAULT_READ_ONLY = 1 << 1,
02063   // Execute access to a page marked NX.
02064   HSA_AMD_MEMORY_FAULT_NX = 1 << 2,
02065   // GPU attempted access to a host only page.
02066   HSA_AMD_MEMORY_FAULT_HOST_ONLY = 1 << 3,
02067   // DRAM ECC failure.
02068   HSA_AMD_MEMORY_FAULT_DRAMECC = 1 << 4,
02069   // Can't determine the exact fault address.
02070   HSA_AMD_MEMORY_FAULT_IMPRECISE = 1 << 5,
02071   // SRAM ECC failure (ie registers, no fault address).
02072   HSA_AMD_MEMORY_FAULT_SRAMECC = 1 << 6,
02073   // GPU reset following unspecified hang.
02074   HSA_AMD_MEMORY_FAULT_HANG = 1 << 31
02075 } hsa_amd_memory_fault_reason_t;
02076
02080 typedef struct hsa_amd_gpu_memory_fault_info_s {
02081   /*
02082   The agent where the memory fault occurred.
02083   */
02084   hsa_agent_t agent;
02085   /*
02086   Virtual address accessed.
02087   */
02088   uint64_t virtual_address;
02089   /*
02090   Bit field encoding the memory access failure reasons. There could be multiple bits set
02091   for one fault.  Bits are defined in hsa_amd_memory_fault_reason_t.
```

```
02092   */
02093   uint32_t fault_reason_mask;
02094 } hsa_amd_gpu_memory_fault_info_t;
02095
02099 typedef struct hsa_amd_event_s {
02100   /*
02101   The event type.
02102   */
02103   hsa_amd_event_type_t event_type;
02104   union {
02105     /*
02106     The memory fault info, only valid when @p event_type is HSA_AMD_GPU_MEMORY_FAULT_EVENT.
02107     */
02108     hsa_amd_gpu_memory_fault_info_t memory_fault;
02109   };
02110 } hsa_amd_event_t;
02111
02112 typedef hsa_status_t (*hsa_amd_system_event_callback_t)(const hsa_amd_event_t* event, void* data);
02113
02129 hsa_status_t HSA_API hsa_amd_register_system_event_handler(hsa_amd_system_event_callback_t callback,
02130                                                            void* data);
02131
02135 typedef enum hsa_amd_queue_priority_s {
02136   /*
02137   Below normal/high priority compute and all graphics
02138   */
02139   HSA_AMD_QUEUE_PRIORITY_LOW = 0,
02140   /*
02141   Above low priority compute, below high priority compute and all graphics
02142   */
02143   HSA_AMD_QUEUE_PRIORITY_NORMAL = 1,
02144   /*
02145   Above low/normal priority compute and all graphics
02146   */
02147   HSA_AMD_QUEUE_PRIORITY_HIGH = 2,
02148 } hsa_amd_queue_priority_t;
02149
02166 hsa_status_t HSA_API hsa_amd_queue_set_priority(hsa_queue_t* queue,
02167                                                 hsa_amd_queue_priority_t priority);
02168
02172 typedef void (*hsa_amd_deallocation_callback_t)(void* ptr, void* user_data);
02173
02206 hsa_status_t HSA_API hsa_amd_register_deallocation_callback(void* ptr,
02207                                                             hsa_amd_deallocation_callback_t callback,
02208                                                             void* user_data);
02209
02226 hsa_status_t HSA_API hsa_amd_deregister_deallocation_callback(void* ptr,
02227                                                               hsa_amd_deallocation_callback_t callback);
02228
02229 typedef enum hsa_amd_svm_model_s {
02234   HSA_AMD_SVM_GLOBAL_FLAG_FINE_GRAINED = 0,
02239   HSA_AMD_SVM_GLOBAL_FLAG_COARSE_GRAINED = 1,
02249   HSA_AMD_SVM_GLOBAL_FLAG_INDETERMINATE = 2
02250 } hsa_amd_svm_model_t;
02251
02252 typedef enum hsa_amd_svm_attribute_s {
02253   // Memory model attribute.
02254   // Type of this attribute is hsa_amd_svm_model_t.
02255   HSA_AMD_SVM_ATTRIB_GLOBAL_FLAG = 0,
02256   // Marks the range read only.  This allows multiple physical copies to be
02257   // placed local to each accessing device.
02258   // Type of this attribute is bool.
02259   HSA_AMD_SVM_ATTRIB_READ_ONLY = 1,
02260   // Automatic migrations should attempt to keep the memory within the xgmi hive
02261   // containing accessible agents.
02262   // Type of this attribute is bool.
02263   HSA_AMD_SVM_ATTRIB_HIVE_LOCAL = 2,
02264   // Page granularity to migrate at once.  Page granularity is specified as
02265   // log2(page_count).
02266   // Type of this attribute is uint64_t.
02267   HSA_AMD_SVM_ATTRIB_MIGRATION_GRANULARITY = 3,
02268   // Physical location to prefer when automatic migration occurs.
02269   // Set to the null agent handle (handle == 0) to indicate there
02270   // is no preferred location.
02271   // Type of this attribute is hsa_agent_t.
02272   HSA_AMD_SVM_ATTRIB_PREFERRED_LOCATION = 4,
02273   // This attribute can not be used in ::hsa_amd_svm_attributes_set (see
02274   // ::hsa_amd_svm_prefetch_async).
02275   // Queries the physical location of most recent prefetch command.
02276   // If the prefetch location has not been set or is not uniform across the
02277   // address range then returned hsa_agent_t::handle will be 0.
02278   // Querying this attribute will return the destination agent of the most
02279   // recent ::hsa_amd_svm_prefetch_async targeting the address range.  If
02280   // multiple async prefetches have been issued targeting the region and the
02281   // most recently issued prefetch has completed then the query will return
02282   // the location of the most recently completed prefetch.
02283   // Type of this attribute is hsa_agent_t.
```

```
02284   HSA_AMD_SVM_ATTRIB_PREFETCH_LOCATION = 5,
02285   // Optimizes with the anticipation that the majority of operations to the
02286   // range will be read operations.
02287   // Type of this attribute is bool.
02288   HSA_AMD_SVM_ATTRIB_READ_MOSTLY = 6,
02289   // Allows the execution on GPU.
02290   // Type of this attribute is bool.
02291   HSA_AMD_SVM_ATTRIB_GPU_EXEC = 7,
02292   // This attribute can not be used in ::hsa_amd_svm_attributes_get.
02293   // Enables an agent for access to the range.  Access may incur a page fault
02294   // and associated memory migration.  Either this or
02295   // HSA_AMD_SVM_ATTRIB_AGENT_ACCESSIBLE_IN_PLACE is required prior to SVM
02296   // access if HSA_AMD_SYSTEM_INFO_SVM_ACCESSIBLE_BY_DEFAULT is false.
02297   // Type of this attribute is hsa_agent_t.
02298   HSA_AMD_SVM_ATTRIB_AGENT_ACCESSIBLE = 0x200,
02299   // This attribute can not be used in ::hsa_amd_svm_attributes_get.
02300   // Enables an agent for access to the range without page faults.  Access
02301   // will not incur a page fault and will not cause access based migration.
02302   // and associated memory migration.  Either this or
02303   // HSA_AMD_SVM_ATTRIB_AGENT_ACCESSIBLE is required prior to SVM access if
02304   // HSA_AMD_SYSTEM_INFO_SVM_ACCESSIBLE_BY_DEFAULT is false.
02305   // Type of this attribute is hsa_agent_t.
02306   HSA_AMD_SVM_ATTRIB_AGENT_ACCESSIBLE_IN_PLACE = 0x201,
02307   // This attribute can not be used in ::hsa_amd_svm_attributes_get.
02308   // Denies an agent access to the memory range.  Access will cause a terminal
02309   // segfault.
02310   // Type of this attribute is hsa_agent_t.
02311   HSA_AMD_SVM_ATTRIB_AGENT_NO_ACCESS = 0x202,
02312   // This attribute can not be used in ::hsa_amd_svm_attributes_set.
02313   // Returns the access attribute associated with the agent.
02314   // The agent to query must be set in the attribute value field.
02315   // The attribute enum will be replaced with the agent's current access
02316   // attribute for the address range.
02317   // TODO: Clarify KFD return value for non-uniform access attribute.
02318   // Type of this attribute is hsa_agent_t.
02319   HSA_AMD_SVM_ATTRIB_ACCESS_QUERY = 0x203,
02320 } hsa_amd_svm_attribute_t;
02321
02322 // List type for hsa_amd_svm_attributes_set/get.
02323 typedef struct hsa_amd_svm_attribute_pair_s {
02324   // hsa_amd_svm_attribute_t value.
02325   uint64_t attribute;
02326   // Attribute value.  Bit values should be interpreted according to the type
02327   // given in the associated attribute description.
02328   uint64_t value;
02329 } hsa_amd_svm_attribute_pair_t;
02330
02350 hsa_status_t hsa_amd_svm_attributes_set(void* ptr, size_t size,
02351                                         hsa_amd_svm_attribute_pair_t* attribute_list,
02352                                         size_t attribute_count);
02353
02372 hsa_status_t hsa_amd_svm_attributes_get(void* ptr, size_t size,
02373                                         hsa_amd_svm_attribute_pair_t* attribute_list,
02374                                         size_t attribute_count);
02375
02400 hsa_status_t hsa_amd_svm_prefetch_async(void* ptr, size_t size, hsa_agent_t agent,
02401                                         uint32_t num_dep_signals, const hsa_signal_t* dep_signals,
02402                                         hsa_signal_t completion_signal);
02403
02404 #ifdef __cplusplus
02405 }  // end extern "C" block
02406 #endif
02407
02408 #endif  // header guard
```

7.10 hsa_ext_finalize.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
```

```
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 #ifndef HSA_RUNTIME_INC_HSA_EXT_FINALIZE_H_
00044 #define HSA_RUNTIME_INC_HSA_EXT_FINALIZE_H_
00045
00046 #include "hsa.h"
00047
00048 #undef HSA_API
00049 #ifdef HSA_EXPORT_FINALIZER
00050 #define HSA_API HSA_API_EXPORT
00051 #else
00052 #define HSA_API HSA_API_IMPORT
00053 #endif
00054
00055 #ifdef __cplusplus
00056 extern "C" {
00057 #endif // __cplusplus
00058
00059 struct BrigModuleHeader;
00060 typedef struct BrigModuleHeader* BrigModule_t;
00061
00069 enum {
00073   HSA_EXT_STATUS_ERROR_INVALID_PROGRAM = 0x2000,
00077   HSA_EXT_STATUS_ERROR_INVALID_MODULE = 0x2001,
00082   HSA_EXT_STATUS_ERROR_INCOMPATIBLE_MODULE = 0x2002,
00086   HSA_EXT_STATUS_ERROR_MODULE_ALREADY_INCLUDED = 0x2003,
00090   HSA_EXT_STATUS_ERROR_SYMBOL_MISMATCH = 0x2004,
00095   HSA_EXT_STATUS_ERROR_FINALIZATION_FAILED = 0x2005,
00100   HSA_EXT_STATUS_ERROR_DIRECTIVE_MISMATCH = 0x2006
00101 };
00102
00113 typedef BrigModule_t hsa_ext_module_t;
00114
00120 typedef struct hsa_ext_program_s {
00124   uint64_t handle;
00125 } hsa_ext_program_t;
00126
00154 hsa_status_t HSA_API hsa_ext_program_create(
00155     hsa_machine_model_t machine_model,
00156     hsa_profile_t profile,
00157     hsa_default_float_rounding_mode_t default_float_rounding_mode,
00158     const char *options,
00159     hsa_ext_program_t *program);
00160
00181 hsa_status_t HSA_API hsa_ext_program_destroy(
00182     hsa_ext_program_t program);
00183
00227 hsa_status_t HSA_API hsa_ext_program_add_module(
00228     hsa_ext_program_t program,
00229     hsa_ext_module_t module);
00230
00255 hsa_status_t HSA_API hsa_ext_program_iterate_modules(
00256     hsa_ext_program_t program,
00257     hsa_status_t (*callback)(hsa_ext_program_t program, hsa_ext_module_t module,
00258                              void* data),
00259     void* data);
00260
00264 typedef enum {
00269   HSA_EXT_PROGRAM_INFO_MACHINE_MODEL = 0,
00274   HSA_EXT_PROGRAM_INFO_PROFILE = 1,
00279   HSA_EXT_PROGRAM_INFO_DEFAULT_FLOAT_ROUNDING_MODE = 2
00280 } hsa_ext_program_info_t;
00281
00303 hsa_status_t HSA_API hsa_ext_program_get_info(
```

```
00304     hsa_ext_program_t program,
00305     hsa_ext_program_info_t attribute,
00306     void *value);
00307
00311 typedef enum {
00315   HSA_EXT_FINALIZER_CALL_CONVENTION_AUTO = -1
00316 } hsa_ext_finalizer_call_convention_t;
00317
00322 typedef struct hsa_ext_control_directives_s {
00333   uint64_t control_directives_mask;
00340   uint16_t break_exceptions_mask;
00347   uint16_t detect_exceptions_mask;
00353   uint32_t max_dynamic_group_size;
00366   uint64_t max_flat_grid_size;
00380   uint32_t max_flat_workgroup_size;
00384   uint32_t reserved1;
00398   uint64_t required_grid_size[3];
00412   hsa_dim3_t required_workgroup_size;
00425   uint8_t required_dim;
00429   uint8_t reserved2[75];
00430 } hsa_ext_control_directives_t;
00431
00487 hsa_status_t HSA_API hsa_ext_program_finalize(
00488     hsa_ext_program_t program,
00489     hsa_isa_t isa,
00490     int32_t call_convention,
00491     hsa_ext_control_directives_t control_directives,
00492     const char *options,
00493     hsa_code_object_type_t code_object_type,
00494     hsa_code_object_t *code_object);
00495
00498 #define hsa_ext_finalizer_1_00
00499
00500 typedef struct hsa_ext_finalizer_1_00_pfn_s {
00501   hsa_status_t (*hsa_ext_program_create)(
00502       hsa_machine_model_t machine_model, hsa_profile_t profile,
00503       hsa_default_float_rounding_mode_t default_float_rounding_mode,
00504       const char *options, hsa_ext_program_t *program);
00505
00506   hsa_status_t (*hsa_ext_program_destroy)(hsa_ext_program_t program);
00507
00508   hsa_status_t (*hsa_ext_program_add_module)(hsa_ext_program_t program,
00509                                              hsa_ext_module_t module);
00510
00511   hsa_status_t (*hsa_ext_program_iterate_modules)(
00512       hsa_ext_program_t program,
00513       hsa_status_t (*callback)(hsa_ext_program_t program,
00514                                hsa_ext_module_t module, void *data),
00515       void *data);
00516
00517   hsa_status_t (*hsa_ext_program_get_info)(
00518       hsa_ext_program_t program, hsa_ext_program_info_t attribute,
00519       void *value);
00520
00521   hsa_status_t (*hsa_ext_program_finalize)(
00522       hsa_ext_program_t program, hsa_isa_t isa, int32_t call_convention,
00523       hsa_ext_control_directives_t control_directives, const char *options,
00524       hsa_code_object_type_t code_object_type, hsa_code_object_t *code_object);
00525 } hsa_ext_finalizer_1_00_pfn_t;
00526
00527 #ifdef __cplusplus
00528 } // extern "C" block
00529 #endif // __cplusplus
00530
00531 #endif // HSA_RUNTIME_INC_HSA_EXT_FINALIZE_H_
```

7.11 hsa_ext_image.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
00015 //
```

```
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 #ifndef HSA_EXT_IMAGE_H
00044 #define HSA_EXT_IMAGE_H
00045
00046 #include "hsa.h"
00047
00048 #undef HSA_API
00049 #ifdef HSA_EXPORT_IMAGES
00050 #define HSA_API HSA_API_EXPORT
00051 #else
00052 #define HSA_API HSA_API_IMPORT
00053 #endif
00054
00055 #ifdef __cplusplus
00056 extern "C" {
00057 #endif /*__cplusplus*/
00058
00068 enum {
00072     HSA_EXT_STATUS_ERROR_IMAGE_FORMAT_UNSUPPORTED = 0x3000,
00076     HSA_EXT_STATUS_ERROR_IMAGE_SIZE_UNSUPPORTED = 0x3001,
00080     HSA_EXT_STATUS_ERROR_IMAGE_PITCH_UNSUPPORTED = 0x3002,
00084     HSA_EXT_STATUS_ERROR_SAMPLER_DESCRIPTOR_UNSUPPORTED = 0x3003
00085 };
00086
00093 enum {
00098   HSA_EXT_AGENT_INFO_IMAGE_1D_MAX_ELEMENTS = 0x3000,
00103   HSA_EXT_AGENT_INFO_IMAGE_1DA_MAX_ELEMENTS = 0x3001,
00108   HSA_EXT_AGENT_INFO_IMAGE_1DB_MAX_ELEMENTS = 0x3002,
00114   HSA_EXT_AGENT_INFO_IMAGE_2D_MAX_ELEMENTS = 0x3003,
00120   HSA_EXT_AGENT_INFO_IMAGE_2DA_MAX_ELEMENTS = 0x3004,
00126   HSA_EXT_AGENT_INFO_IMAGE_2DDEPTH_MAX_ELEMENTS = 0x3005,
00132   HSA_EXT_AGENT_INFO_IMAGE_2DADEPTH_MAX_ELEMENTS = 0x3006,
00138   HSA_EXT_AGENT_INFO_IMAGE_3D_MAX_ELEMENTS = 0x3007,
00143   HSA_EXT_AGENT_INFO_IMAGE_ARRAY_MAX_LAYERS = 0x3008,
00148   HSA_EXT_AGENT_INFO_MAX_IMAGE_RD_HANDLES = 0x3009,
00154   HSA_EXT_AGENT_INFO_MAX_IMAGE_RORW_HANDLES = 0x300A,
00159   HSA_EXT_AGENT_INFO_MAX_SAMPLER_HANDLERS = 0x300B,
00165   HSA_EXT_AGENT_INFO_IMAGE_LINEAR_ROW_PITCH_ALIGNMENT = 0x300C
00166 };
00167
00174 typedef struct hsa_ext_image_s {
00179     uint64_t handle;
00180
00181 } hsa_ext_image_t;
00182
00191 typedef enum {
00195   HSA_EXT_IMAGE_GEOMETRY_1D = 0,
00196
00200   HSA_EXT_IMAGE_GEOMETRY_2D = 1,
00201
00205   HSA_EXT_IMAGE_GEOMETRY_3D = 2,
00206
00211   HSA_EXT_IMAGE_GEOMETRY_1DA = 3,
00212
00217   HSA_EXT_IMAGE_GEOMETRY_2DA = 4,
00218
00226   HSA_EXT_IMAGE_GEOMETRY_1DB = 5,
00227
00231   HSA_EXT_IMAGE_GEOMETRY_2DDEPTH = 6,
00232
00237   HSA_EXT_IMAGE_GEOMETRY_2DADEPTH = 7
00238 } hsa_ext_image_geometry_t;
```

```
00239
00247 typedef enum {
00248     HSA_EXT_IMAGE_CHANNEL_TYPE_SNORM_INT8 = 0,
00249     HSA_EXT_IMAGE_CHANNEL_TYPE_SNORM_INT16 = 1,
00250     HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_INT8 = 2,
00251     HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_INT16 = 3,
00252     HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_INT24 = 4,
00253     HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_SHORT_555 = 5,
00254     HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_SHORT_565 = 6,
00255     HSA_EXT_IMAGE_CHANNEL_TYPE_UNORM_SHORT_101010 = 7,
00256     HSA_EXT_IMAGE_CHANNEL_TYPE_SIGNED_INT8 = 8,
00257     HSA_EXT_IMAGE_CHANNEL_TYPE_SIGNED_INT16 = 9,
00258     HSA_EXT_IMAGE_CHANNEL_TYPE_SIGNED_INT32 = 10,
00259     HSA_EXT_IMAGE_CHANNEL_TYPE_UNSIGNED_INT8 = 11,
00260     HSA_EXT_IMAGE_CHANNEL_TYPE_UNSIGNED_INT16 = 12,
00261     HSA_EXT_IMAGE_CHANNEL_TYPE_UNSIGNED_INT32 = 13,
00262     HSA_EXT_IMAGE_CHANNEL_TYPE_HALF_FLOAT = 14,
00263     HSA_EXT_IMAGE_CHANNEL_TYPE_FLOAT = 15
00264 } hsa_ext_image_channel_type_t;
00265
00269 typedef uint32_t hsa_ext_image_channel_type32_t;
00270
00279 typedef enum {
00280     HSA_EXT_IMAGE_CHANNEL_ORDER_A = 0,
00281     HSA_EXT_IMAGE_CHANNEL_ORDER_R = 1,
00282     HSA_EXT_IMAGE_CHANNEL_ORDER_RX = 2,
00283     HSA_EXT_IMAGE_CHANNEL_ORDER_RG = 3,
00284     HSA_EXT_IMAGE_CHANNEL_ORDER_RGX = 4,
00285     HSA_EXT_IMAGE_CHANNEL_ORDER_RA = 5,
00286     HSA_EXT_IMAGE_CHANNEL_ORDER_RGB = 6,
00287     HSA_EXT_IMAGE_CHANNEL_ORDER_RGBX = 7,
00288     HSA_EXT_IMAGE_CHANNEL_ORDER_RGBA = 8,
00289     HSA_EXT_IMAGE_CHANNEL_ORDER_BGRA = 9,
00290     HSA_EXT_IMAGE_CHANNEL_ORDER_ARGB = 10,
00291     HSA_EXT_IMAGE_CHANNEL_ORDER_ABGR = 11,
00292     HSA_EXT_IMAGE_CHANNEL_ORDER_SRGB = 12,
00293     HSA_EXT_IMAGE_CHANNEL_ORDER_SRGBX = 13,
00294     HSA_EXT_IMAGE_CHANNEL_ORDER_SRGBA = 14,
00295     HSA_EXT_IMAGE_CHANNEL_ORDER_SBGRA = 15,
00296     HSA_EXT_IMAGE_CHANNEL_ORDER_INTENSITY = 16,
00297     HSA_EXT_IMAGE_CHANNEL_ORDER_LUMINANCE = 17,
00298     HSA_EXT_IMAGE_CHANNEL_ORDER_DEPTH = 18,
00299     HSA_EXT_IMAGE_CHANNEL_ORDER_DEPTH_STENCIL = 19
00300 } hsa_ext_image_channel_order_t;
00301
00305 typedef uint32_t hsa_ext_image_channel_order32_t;
00306
00307
00311 typedef struct hsa_ext_image_format_s {
00315     hsa_ext_image_channel_type32_t channel_type;
00316
00320     hsa_ext_image_channel_order32_t channel_order;
00321 } hsa_ext_image_format_t;
00322
00326 typedef struct hsa_ext_image_descriptor_s {
00330     hsa_ext_image_geometry_t geometry;
00334     size_t width;
00341     size_t height;
00346     size_t depth;
00352     size_t array_size;
00356     hsa_ext_image_format_t format;
00357 } hsa_ext_image_descriptor_t;
00358
00362 typedef enum  {
00367     HSA_EXT_IMAGE_CAPABILITY_NOT_SUPPORTED = 0x0,
00372     HSA_EXT_IMAGE_CAPABILITY_READ_ONLY = 0x1,
00377     HSA_EXT_IMAGE_CAPABILITY_WRITE_ONLY = 0x2,
00382     HSA_EXT_IMAGE_CAPABILITY_READ_WRITE = 0x4,
00387     HSA_EXT_IMAGE_CAPABILITY_READ_MODIFY_WRITE = 0x8,
00393     HSA_EXT_IMAGE_CAPABILITY_ACCESS_INVARIANT_DATA_LAYOUT = 0x10
00394 } hsa_ext_image_capability_t;
00395
00411 typedef enum  {
00417     HSA_EXT_IMAGE_DATA_LAYOUT_OPAQUE = 0x0,
00434     HSA_EXT_IMAGE_DATA_LAYOUT_LINEAR = 0x1
00435 } hsa_ext_image_data_layout_t;
00436
00462 hsa_status_t HSA_API hsa_ext_image_get_capability(
00463     hsa_agent_t agent,
00464     hsa_ext_image_geometry_t geometry,
00465     const hsa_ext_image_format_t *image_format,
00466     uint32_t *capability_mask);
00467
00498 hsa_status_t HSA_API hsa_ext_image_get_capability_with_layout(
00499     hsa_agent_t agent,
00500     hsa_ext_image_geometry_t geometry,
00501     const hsa_ext_image_format_t *image_format,
```

```
00502     hsa_ext_image_data_layout_t image_data_layout,
00503     uint32_t *capability_mask);
00504
00509 typedef struct hsa_ext_image_data_info_s {
00513   size_t size;
00514
00518   size_t alignment;
00519
00520 } hsa_ext_image_data_info_t;
00521
00574 hsa_status_t HSA_API hsa_ext_image_data_get_info(
00575     hsa_agent_t agent,
00576     const hsa_ext_image_descriptor_t *image_descriptor,
00577     hsa_access_permission_t access_permission,
00578     hsa_ext_image_data_info_t *image_data_info);
00579
00657 hsa_status_t HSA_API hsa_ext_image_data_get_info_with_layout(
00658     hsa_agent_t agent,
00659     const hsa_ext_image_descriptor_t *image_descriptor,
00660     hsa_access_permission_t access_permission,
00661     hsa_ext_image_data_layout_t image_data_layout,
00662     size_t image_data_row_pitch,
00663     size_t image_data_slice_pitch,
00664     hsa_ext_image_data_info_t *image_data_info);
00665
00730 hsa_status_t HSA_API hsa_ext_image_create(
00731     hsa_agent_t agent,
00732     const hsa_ext_image_descriptor_t *image_descriptor,
00733     const void *image_data,
00734     hsa_access_permission_t access_permission,
00735     hsa_ext_image_t *image);
00736
00833 hsa_status_t HSA_API hsa_ext_image_create_with_layout(
00834     hsa_agent_t agent,
00835     const hsa_ext_image_descriptor_t *image_descriptor,
00836     const void *image_data,
00837     hsa_access_permission_t access_permission,
00838     hsa_ext_image_data_layout_t image_data_layout,
00839     size_t image_data_row_pitch,
00840     size_t image_data_slice_pitch,
00841     hsa_ext_image_t *image);
00842
00863 hsa_status_t HSA_API hsa_ext_image_destroy(
00864     hsa_agent_t agent,
00865     hsa_ext_image_t image);
00866
00916 hsa_status_t HSA_API hsa_ext_image_copy(
00917     hsa_agent_t agent,
00918     hsa_ext_image_t src_image,
00919     const hsa_dim3_t* src_offset,
00920     hsa_ext_image_t dst_image,
00921     const hsa_dim3_t* dst_offset,
00922     const hsa_dim3_t* range);
00923
00927 typedef struct hsa_ext_image_region_s {
00931     hsa_dim3_t offset;
00932
00937     hsa_dim3_t range;
00938 } hsa_ext_image_region_t;
00939
00989 hsa_status_t HSA_API hsa_ext_image_import(
00990     hsa_agent_t agent,
00991     const void *src_memory,
00992     size_t src_row_pitch,
00993     size_t src_slice_pitch,
00994     hsa_ext_image_t dst_image,
00995     const hsa_ext_image_region_t *image_region);
00996
01044 hsa_status_t HSA_API hsa_ext_image_export(
01045     hsa_agent_t agent,
01046     hsa_ext_image_t src_image,
01047     void *dst_memory,
01048     size_t dst_row_pitch,
01049     size_t dst_slice_pitch,
01050     const hsa_ext_image_region_t *image_region);
01051
01085 hsa_status_t HSA_API hsa_ext_image_clear(
01086     hsa_agent_t agent,
01087     hsa_ext_image_t image,
01088     const void* data,
01089     const hsa_ext_image_region_t *image_region);
01090
01096 typedef struct hsa_ext_sampler_s {
01101     uint64_t handle;
01102 } hsa_ext_sampler_t;
01103
01111 typedef enum {
```

```
01115   HSA_EXT_SAMPLER_ADDRESSING_MODE_UNDEFINED = 0,
01116
01120   HSA_EXT_SAMPLER_ADDRESSING_MODE_CLAMP_TO_EDGE = 1,
01121
01125   HSA_EXT_SAMPLER_ADDRESSING_MODE_CLAMP_TO_BORDER = 2,
01126
01131   HSA_EXT_SAMPLER_ADDRESSING_MODE_REPEAT = 3,
01132
01138   HSA_EXT_SAMPLER_ADDRESSING_MODE_MIRRORED_REPEAT = 4
01139
01140 } hsa_ext_sampler_addressing_mode_t;
01141
01145 typedef uint32_t hsa_ext_sampler_addressing_mode32_t;
01146
01154 typedef enum {
01155
01159   HSA_EXT_SAMPLER_COORDINATE_MODE_UNNORMALIZED = 0,
01160
01165   HSA_EXT_SAMPLER_COORDINATE_MODE_NORMALIZED = 1
01166
01167 } hsa_ext_sampler_coordinate_mode_t;
01168
01172 typedef uint32_t hsa_ext_sampler_coordinate_mode32_t;
01173
01174
01181 typedef enum {
01186   HSA_EXT_SAMPLER_FILTER_MODE_NEAREST = 0,
01187
01193   HSA_EXT_SAMPLER_FILTER_MODE_LINEAR = 1
01194
01195 } hsa_ext_sampler_filter_mode_t;
01196
01200 typedef uint32_t hsa_ext_sampler_filter_mode32_t;
01201
01205 typedef struct hsa_ext_sampler_descriptor_s {
01209   hsa_ext_sampler_coordinate_mode32_t coordinate_mode;
01210
01214   hsa_ext_sampler_filter_mode32_t filter_mode;
01215
01220   hsa_ext_sampler_addressing_mode32_t address_mode;
01221
01222 } hsa_ext_sampler_descriptor_t;
01223
01253 hsa_status_t HSA_API hsa_ext_sampler_create(
01254     hsa_agent_t agent,
01255     const hsa_ext_sampler_descriptor_t *sampler_descriptor,
01256     hsa_ext_sampler_t *sampler);
01257
01276 hsa_status_t HSA_API hsa_ext_sampler_destroy(
01277     hsa_agent_t agent,
01278     hsa_ext_sampler_t sampler);
01279
01280
01281 #define hsa_ext_images_1_00
01282
01286 typedef struct hsa_ext_images_1_00_pfn_s {
01287
01288   hsa_status_t (*hsa_ext_image_get_capability)(
01289     hsa_agent_t agent,
01290     hsa_ext_image_geometry_t geometry,
01291     const hsa_ext_image_format_t *image_format,
01292     uint32_t *capability_mask);
01293
01294   hsa_status_t (*hsa_ext_image_data_get_info)(
01295     hsa_agent_t agent,
01296     const hsa_ext_image_descriptor_t *image_descriptor,
01297     hsa_access_permission_t access_permission,
01298     hsa_ext_image_data_info_t *image_data_info);
01299
01300   hsa_status_t (*hsa_ext_image_create)(
01301     hsa_agent_t agent,
01302     const hsa_ext_image_descriptor_t *image_descriptor,
01303     const void *image_data,
01304     hsa_access_permission_t access_permission,
01305     hsa_ext_image_t *image);
01306
01307   hsa_status_t (*hsa_ext_image_destroy)(
01308     hsa_agent_t agent,
01309     hsa_ext_image_t image);
01310
01311   hsa_status_t (*hsa_ext_image_copy)(
01312     hsa_agent_t agent,
01313     hsa_ext_image_t src_image,
01314     const hsa_dim3_t* src_offset,
01315     hsa_ext_image_t dst_image,
01316     const hsa_dim3_t* dst_offset,
01317     const hsa_dim3_t* range);
```

```
01318
01319   hsa_status_t (*hsa_ext_image_import)(
01320     hsa_agent_t agent,
01321     const void *src_memory,
01322     size_t src_row_pitch,
01323     size_t src_slice_pitch,
01324     hsa_ext_image_t dst_image,
01325     const hsa_ext_image_region_t *image_region);
01326
01327   hsa_status_t (*hsa_ext_image_export)(
01328     hsa_agent_t agent,
01329     hsa_ext_image_t src_image,
01330     void *dst_memory,
01331     size_t dst_row_pitch,
01332     size_t dst_slice_pitch,
01333     const hsa_ext_image_region_t *image_region);
01334
01335   hsa_status_t (*hsa_ext_image_clear)(
01336     hsa_agent_t agent,
01337     hsa_ext_image_t image,
01338     const void* data,
01339     const hsa_ext_image_region_t *image_region);
01340
01341   hsa_status_t (*hsa_ext_sampler_create)(
01342     hsa_agent_t agent,
01343     const hsa_ext_sampler_descriptor_t *sampler_descriptor,
01344     hsa_ext_sampler_t *sampler);
01345
01346   hsa_status_t (*hsa_ext_sampler_destroy)(
01347     hsa_agent_t agent,
01348     hsa_ext_sampler_t sampler);
01349
01350 } hsa_ext_images_1_00_pfn_t;
01351
01352 #define hsa_ext_images_1
01353
01357 typedef struct hsa_ext_images_1_pfn_s {
01358
01359   hsa_status_t (*hsa_ext_image_get_capability)(
01360     hsa_agent_t agent,
01361     hsa_ext_image_geometry_t geometry,
01362     const hsa_ext_image_format_t *image_format,
01363     uint32_t *capability_mask);
01364
01365   hsa_status_t (*hsa_ext_image_data_get_info)(
01366     hsa_agent_t agent,
01367     const hsa_ext_image_descriptor_t *image_descriptor,
01368     hsa_access_permission_t access_permission,
01369     hsa_ext_image_data_info_t *image_data_info);
01370
01371   hsa_status_t (*hsa_ext_image_create)(
01372     hsa_agent_t agent,
01373     const hsa_ext_image_descriptor_t *image_descriptor,
01374     const void *image_data,
01375     hsa_access_permission_t access_permission,
01376     hsa_ext_image_t *image);
01377
01378   hsa_status_t (*hsa_ext_image_destroy)(
01379     hsa_agent_t agent,
01380     hsa_ext_image_t image);
01381
01382   hsa_status_t (*hsa_ext_image_copy)(
01383     hsa_agent_t agent,
01384     hsa_ext_image_t src_image,
01385     const hsa_dim3_t* src_offset,
01386     hsa_ext_image_t dst_image,
01387     const hsa_dim3_t* dst_offset,
01388     const hsa_dim3_t* range);
01389
01390   hsa_status_t (*hsa_ext_image_import)(
01391     hsa_agent_t agent,
01392     const void *src_memory,
01393     size_t src_row_pitch,
01394     size_t src_slice_pitch,
01395     hsa_ext_image_t dst_image,
01396     const hsa_ext_image_region_t *image_region);
01397
01398   hsa_status_t (*hsa_ext_image_export)(
01399     hsa_agent_t agent,
01400     hsa_ext_image_t src_image,
01401     void *dst_memory,
01402     size_t dst_row_pitch,
01403     size_t dst_slice_pitch,
01404     const hsa_ext_image_region_t *image_region);
01405
01406   hsa_status_t (*hsa_ext_image_clear)(
01407     hsa_agent_t agent,
```

```
01408     hsa_ext_image_t image,
01409     const void* data,
01410     const hsa_ext_image_region_t *image_region);
01411
01412   hsa_status_t (*hsa_ext_sampler_create)(
01413     hsa_agent_t agent,
01414     const hsa_ext_sampler_descriptor_t *sampler_descriptor,
01415     hsa_ext_sampler_t *sampler);
01416
01417   hsa_status_t (*hsa_ext_sampler_destroy)(
01418     hsa_agent_t agent,
01419     hsa_ext_sampler_t sampler);
01420
01421   hsa_status_t (*hsa_ext_image_get_capability_with_layout)(
01422     hsa_agent_t agent,
01423     hsa_ext_image_geometry_t geometry,
01424     const hsa_ext_image_format_t *image_format,
01425     hsa_ext_image_data_layout_t image_data_layout,
01426     uint32_t *capability_mask);
01427
01428   hsa_status_t (*hsa_ext_image_data_get_info_with_layout)(
01429     hsa_agent_t agent,
01430     const hsa_ext_image_descriptor_t *image_descriptor,
01431     hsa_access_permission_t access_permission,
01432     hsa_ext_image_data_layout_t image_data_layout,
01433     size_t image_data_row_pitch,
01434     size_t image_data_slice_pitch,
01435     hsa_ext_image_data_info_t *image_data_info);
01436
01437   hsa_status_t (*hsa_ext_image_create_with_layout)(
01438     hsa_agent_t agent,
01439     const hsa_ext_image_descriptor_t *image_descriptor,
01440     const void *image_data,
01441     hsa_access_permission_t access_permission,
01442     hsa_ext_image_data_layout_t image_data_layout,
01443     size_t image_data_row_pitch,
01444     size_t image_data_slice_pitch,
01445     hsa_ext_image_t *image);
01446
01447 } hsa_ext_images_1_pfn_t;
01450 #ifdef __cplusplus
01451 }  // end extern "C" block
01452 #endif /*__cplusplus*/
01453
01454 #endif
```

7.12 hsa_ven_amd_aqlprofile.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2017-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
```

```
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 #ifndef OPENSRC_HSA_RUNTIME_INC_HSA_VEN_AMD_AQLPROFILE_H_
00044 #define OPENSRC_HSA_RUNTIME_INC_HSA_VEN_AMD_AQLPROFILE_H_
00045
00046 #include <stdint.h>
00047 #include "hsa.h"
00048
00049 #define HSA_AQLPROFILE_VERSION_MAJOR 2
00050 #define HSA_AQLPROFILE_VERSION_MINOR 0
00051
00052 #ifdef __cplusplus
00053 extern "C" {
00054 #endif  // __cplusplus
00055
00057 // Library version
00058 uint32_t hsa_ven_amd_aqlprofile_version_major();
00059 uint32_t hsa_ven_amd_aqlprofile_version_minor();
00060
00062 // Library API:
00063 // The library provides helper methods for instantiation of
00064 // the profile context object and for populating of the start
00065 // and stop AQL packets. The profile object contains a profiling
00066 // events list and needed for profiling buffers descriptors,
00067 // a command buffer and an output data buffer. To check if there
00068 // was an error the library methods return a status code. Also
00069 // the library provides methods for querying required buffers
00070 // attributes, to validate the event attributes and to get profiling
00071 // output data.
00072 //
00073 // Returned status:
00074 //     hsa_status_t - HSA status codes are used from hsa.h header
00075 //
00076 // Supported profiling features:
00077 //
00078 // Supported profiling events
00079 typedef enum {
00080   HSA_VEN_AMD_AQLPROFILE_EVENT_TYPE_PMC = 0,
00081   HSA_VEN_AMD_AQLPROFILE_EVENT_TYPE_TRACE = 1,
00082 } hsa_ven_amd_aqlprofile_event_type_t;
00083
00084 // Supported performance counters (PMC) blocks
00085 // The block ID is the same for a block instances set, for example
00086 // each block instance from the TCC block set, TCC0, TCC1, ..., TCCN
00087 // will have the same block ID HSA_VEN_AMD_AQLPROFILE_BLOCKS_TCC.
00088 typedef enum {
00089   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_CPC = 0,
00090   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_CPF = 1,
00091   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_GDS = 2,
00092   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_GRBM = 3,
00093   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_GRBMSE = 4,
00094   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_SPI = 5,
00095   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_SQ = 6,
00096   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_SQCS = 7,
00097   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_SRBM = 8,
00098   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_SX = 9,
00099   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_TA = 10,
00100   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_TCA = 11,
00101   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_TCC = 12,
00102   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_TCP = 13,
00103   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_TD = 14,
00104   // Memory related blocks
00105   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_MCARB = 15,
00106   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_MCHUB = 16,
00107   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_MCMCBVM = 17,
00108   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_MCSEQ = 18,
00109   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_MCVML2 = 19,
00110   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_MCXBAR = 20,
00111   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_ATC = 21,
00112   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_ATCL2 = 22,
00113   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_GCEA = 23,
00114   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_RPB = 24,
00115   // System blocks
00116   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_SDMA = 25,
00117   // GFX10 added blocks
00118   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_GL1A = 26,
00119   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_GL1C = 27,
00120   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_GL2A = 28,
00121   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_GL2C = 29,
00122   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_GCR = 30,
00123   HSA_VEN_AMD_AQLPROFILE_BLOCK_NAME_GUS = 31,
00124
00125   HSA_VEN_AMD_AQLPROFILE_BLOCKS_NUMBER
```

```
00126 } hsa_ven_amd_aqlprofile_block_name_t;
00127
00128 // PMC event object structure
00129 // 'counter_id' value is specified in GFXIPs perfcounter user guides
00130 // which is the counters select value, "Performance Counters Selection"
00131 // chapter.
00132 typedef struct {
00133   hsa_ven_amd_aqlprofile_block_name_t block_name;
00134   uint32_t block_index;
00135   uint32_t counter_id;
00136 } hsa_ven_amd_aqlprofile_event_t;
00137
00138 // Check if event is valid for the specific GPU
00139 hsa_status_t hsa_ven_amd_aqlprofile_validate_event(
00140     hsa_agent_t agent,                          // HSA handle for the profiling GPU
00141     const hsa_ven_amd_aqlprofile_event_t* event,  // [in] Pointer on validated event
00142     bool* result);                              // [out] True if the event valid, False otherwise
00143
00144 // Profiling parameters
00145 // All parameters are generic and if not applicable for a specific
00146 // profile configuration then error status will be returned.
00147 typedef enum {
00148   // Trace applicable parameters
00149   HSA_VEN_AMD_AQLPROFILE_PARAMETER_NAME_COMPUTE_UNIT_TARGET = 0,
00150   HSA_VEN_AMD_AQLPROFILE_PARAMETER_NAME_VM_ID_MASK = 1,
00151   HSA_VEN_AMD_AQLPROFILE_PARAMETER_NAME_MASK = 2,
00152   HSA_VEN_AMD_AQLPROFILE_PARAMETER_NAME_TOKEN_MASK = 3,
00153   HSA_VEN_AMD_AQLPROFILE_PARAMETER_NAME_TOKEN_MASK2 = 4,
00154   HSA_VEN_AMD_AQLPROFILE_PARAMETER_NAME_SE_MASK = 5,
00155   HSA_VEN_AMD_AQLPROFILE_PARAMETER_NAME_SAMPLE_RATE = 6,
00156   HSA_VEN_AMD_AQLPROFILE_PARAMETER_NAME_K_CONCURRENT = 7,
00157 } hsa_ven_amd_aqlprofile_parameter_name_t;
00158
00159 // Profile parameter object
00160 typedef struct {
00161   hsa_ven_amd_aqlprofile_parameter_name_t parameter_name;
00162   uint32_t value;
00163 } hsa_ven_amd_aqlprofile_parameter_t;
00164
00165 //
00166 // Profile context object:
00167 // The library provides a profile object structure which contains
00168 // the events array, a buffer for the profiling start/stop commands
00169 // and a buffer for the output data.
00170 // The buffers are specified by the buffer descriptors and allocated
00171 // by the application. The buffers allocation attributes, the command
00172 // buffer size, the PMC output buffer size as well as profiling output
00173 // data can be get using the generic get profile info helper _get_info.
00174 //
00175 // Buffer descriptor
00176 typedef struct {
00177   void* ptr;
00178   uint32_t size;
00179 } hsa_ven_amd_aqlprofile_descriptor_t;
00180
00181 // Profile context object structure, contains profiling events list and
00182 // needed for profiling buffers descriptors, a command buffer and
00183 // an output data buffer
00184 typedef struct {
00185   hsa_agent_t agent;                                    // GFXIP handle
00186   hsa_ven_amd_aqlprofile_event_type_t type;             // Events type
00187   const hsa_ven_amd_aqlprofile_event_t* events;         // Events array
00188   uint32_t event_count;                                 // Events count
00189   const hsa_ven_amd_aqlprofile_parameter_t* parameters; // Parameters array
00190   uint32_t parameter_count;                             // Parameters count
00191   hsa_ven_amd_aqlprofile_descriptor_t output_buffer;    // Output buffer
00192   hsa_ven_amd_aqlprofile_descriptor_t command_buffer;   // PM4 commands
00193 } hsa_ven_amd_aqlprofile_profile_t;
00194
00195 //
00196 // AQL packets populating methods:
00197 // The helper methods to populate provided by the application START and
00198 // STOP AQL packets which the application is required to submit before and
00199 // after profiled GPU task packets respectively.
00200 //
00201 // AQL Vendor Specific packet which carries a PM4 command
00202 typedef struct {
00203   uint16_t header;
00204   uint16_t pm4_command[27];
00205   hsa_signal_t completion_signal;
00206 } hsa_ext_amd_aql_pm4_packet_t;
00207
00208 // Method to populate the provided AQL packet with profiling start commands
00209 // Only 'pm4_command' fields of the packet are set and the application
00210 // is responsible to set Vendor Specific header type a completion signal
00211 hsa_status_t hsa_ven_amd_aqlprofile_start(
00212     hsa_ven_amd_aqlprofile_profile_t* profile,        // [in/out] profile contex object
```

```
00213     hsa_ext_amd_aql_pm4_packet_t* aql_start_packet);  // [out] profile start AQL packet
00214
00215 // Method to populate the provided AQL packet with profiling stop commands
00216 // Only 'pm4_command' fields of the packet are set and the application
00217 // is responsible to set Vendor Specific header type and a completion signal
00218 hsa_status_t hsa_ven_amd_aqlprofile_stop(
00219     const hsa_ven_amd_aqlprofile_profile_t* profile,  // [in] profile contex object
00220     hsa_ext_amd_aql_pm4_packet_t* aql_stop_packet);   // [out] profile stop AQL packet
00221
00222 // Method to populate the provided AQL packet with profiling read commands
00223 // Only 'pm4_command' fields of the packet are set and the application
00224 // is responsible to set Vendor Specific header type and a completion signal
00225 hsa_status_t hsa_ven_amd_aqlprofile_read(
00226     const hsa_ven_amd_aqlprofile_profile_t* profile,  // [in] profile contex object
00227     hsa_ext_amd_aql_pm4_packet_t* aql_read_packet);   // [out] profile stop AQL packet
00228
00229 // Legacy devices, PM4 profiling packet size
00230 const unsigned HSA_VEN_AMD_AQLPROFILE_LEGACY_PM4_PACKET_SIZE = 192;
00231 // Legacy devices, converting the profiling AQL packet to PM4 packet blob
00232 hsa_status_t hsa_ven_amd_aqlprofile_legacy_get_pm4(
00233     const hsa_ext_amd_aql_pm4_packet_t* aql_packet,  // [in] AQL packet
00234     void* data);                                     // [out] PM4 packet blob
00235
00236 //
00237 // Get profile info:
00238 // Generic method for getting various profile info including profile buffers
00239 // attributes like the command buffer size and the profiling PMC results.
00240 // It's implied that all counters are 64bit values.
00241 //
00242 // Profile generic output data:
00243 typedef struct {
00244   uint32_t sample_id;  // PMC sample or trace buffer index
00245   union {
00246     struct {
00247       hsa_ven_amd_aqlprofile_event_t event;  // PMC event
00248       uint64_t result;                       // PMC result
00249     } pmc_data;
00250     hsa_ven_amd_aqlprofile_descriptor_t trace_data;  // Trace output data descriptor
00251   };
00252 } hsa_ven_amd_aqlprofile_info_data_t;
00253
00254 // ID query type
00255 typedef struct {
00256   const char* name;
00257   uint32_t id;
00258   uint32_t instance_count;
00259 } hsa_ven_amd_aqlprofile_id_query_t;
00260
00261 // Profile attributes
00262 typedef enum {
00263   HSA_VEN_AMD_AQLPROFILE_INFO_COMMAND_BUFFER_SIZE = 0,  // get_info returns uint32_t value
00264   HSA_VEN_AMD_AQLPROFILE_INFO_PMC_DATA_SIZE = 1,        // get_info returns uint32_t value
00265   HSA_VEN_AMD_AQLPROFILE_INFO_PMC_DATA = 2,             // get_info returns PMC uint64_t value
00266                                                         // in info_data object
00267   HSA_VEN_AMD_AQLPROFILE_INFO_TRACE_DATA = 3,           // get_info returns trace buffer ptr/size
00268                                                         // in info_data object
00269                                                         //
00270   HSA_VEN_AMD_AQLPROFILE_INFO_BLOCK_COUNTERS = 4,       // get_info returns number of block counter
00271   HSA_VEN_AMD_AQLPROFILE_INFO_BLOCK_ID = 5,             // get_info returns block id, instances
00272                                                         // by name string using _id_query_t
00273                                                         //
00274   HSA_VEN_AMD_AQLPROFILE_INFO_ENABLE_CMD = 6,           // get_info returns size/pointer for
00275                                                         // counters enable command buffer
00276   HSA_VEN_AMD_AQLPROFILE_INFO_DISABLE_CMD = 7,          // get_info returns size/pointer for
00277                                                         // counters disable command buffer
00278 } hsa_ven_amd_aqlprofile_info_type_t;
00279
00280 // Definition of output data iterator callback
00281 typedef hsa_status_t (*hsa_ven_amd_aqlprofile_data_callback_t)(
00282     hsa_ven_amd_aqlprofile_info_type_t info_type,   // [in] data type, PMC or trace data
00283     hsa_ven_amd_aqlprofile_info_data_t* info_data,  // [in] info_data object
00284     void* callback_data);                           // [in/out] data passed to the callback
00285
00286 // Method for getting the profile info
00287 hsa_status_t hsa_ven_amd_aqlprofile_get_info(
00288     const hsa_ven_amd_aqlprofile_profile_t* profile,  // [in] profile context object
00289     hsa_ven_amd_aqlprofile_info_type_t attribute,     // [in] requested profile attribute
00290     void* value);                                     // [in/out] returned value
00291
00292 // Method for iterating the events output data
00293 hsa_status_t hsa_ven_amd_aqlprofile_iterate_data(
00294     const hsa_ven_amd_aqlprofile_profile_t* profile,  // [in] profile context object
00295     hsa_ven_amd_aqlprofile_data_callback_t callback,  // [in] callback to iterate the output data
00296     void* data);                                      // [in/out] data passed to the callback
00297
00298 // Return error string
00299 hsa_status_t hsa_ven_amd_aqlprofile_error_string(
```

```
00300     const char** str);  // [out] pointer on the error string
00301
00305 #define hsa_ven_amd_aqlprofile_VERSION_MAJOR 1
00306 #define hsa_ven_amd_aqlprofile_LIB(suff) "libhsa-amd-aqlprofile" suff ".so"
00307
00308 #ifdef HSA_LARGE_MODEL
00309 static const char kAqlProfileLib[] = hsa_ven_amd_aqlprofile_LIB("64");
00310 #else
00311 static const char kAqlProfileLib[] = hsa_ven_amd_aqlprofile_LIB("");
00312 #endif
00313
00317 typedef struct hsa_ven_amd_aqlprofile_1_00_pfn_s {
00318   uint32_t (*hsa_ven_amd_aqlprofile_version_major)();
00319   uint32_t (*hsa_ven_amd_aqlprofile_version_minor)();
00320
00321   hsa_status_t (*hsa_ven_amd_aqlprofile_error_string)(
00322       const char** str);
00323
00324   hsa_status_t (*hsa_ven_amd_aqlprofile_validate_event)(
00325       hsa_agent_t agent,
00326       const hsa_ven_amd_aqlprofile_event_t* event,
00327       bool* result);
00328
00329   hsa_status_t (*hsa_ven_amd_aqlprofile_start)(
00330       hsa_ven_amd_aqlprofile_profile_t* profile,
00331       hsa_ext_amd_aql_pm4_packet_t* aql_start_packet);
00332
00333   hsa_status_t (*hsa_ven_amd_aqlprofile_stop)(
00334       const hsa_ven_amd_aqlprofile_profile_t* profile,
00335       hsa_ext_amd_aql_pm4_packet_t* aql_stop_packet);
00336
00337   hsa_status_t (*hsa_ven_amd_aqlprofile_read)(
00338       const hsa_ven_amd_aqlprofile_profile_t* profile,
00339       hsa_ext_amd_aql_pm4_packet_t* aql_read_packet);
00340
00341   hsa_status_t (*hsa_ven_amd_aqlprofile_legacy_get_pm4)(
00342       const hsa_ext_amd_aql_pm4_packet_t* aql_packet,
00343       void* data);
00344
00345   hsa_status_t (*hsa_ven_amd_aqlprofile_get_info)(
00346       const hsa_ven_amd_aqlprofile_profile_t* profile,
00347       hsa_ven_amd_aqlprofile_info_type_t attribute,
00348       void* value);
00349
00350   hsa_status_t (*hsa_ven_amd_aqlprofile_iterate_data)(
00351       const hsa_ven_amd_aqlprofile_profile_t* profile,
00352       hsa_ven_amd_aqlprofile_data_callback_t callback,
00353       void* data);
00354 } hsa_ven_amd_aqlprofile_1_00_pfn_t;
00355
00356 typedef hsa_ven_amd_aqlprofile_1_00_pfn_t hsa_ven_amd_aqlprofile_pfn_t;
00357
00358 #ifdef __cplusplus
00359 }
00360 #endif  // __cplusplus
00361
00362 #endif  // OPENSRC_HSA_RUNTIME_INC_HSA_VEN_AMD_AQLPROFILE_H_
```

7.13 hsa_ven_amd_loader.h

```
00001
00002 //
00003 // The University of Illinois/NCSA
00004 // Open Source License (NCSA)
00005 //
00006 // Copyright (c) 2014-2020, Advanced Micro Devices, Inc. All rights reserved.
00007 //
00008 // Developed by:
00009 //
00010 //                 AMD Research and AMD HSA Software Development
00011 //
00012 //                 Advanced Micro Devices, Inc.
00013 //
00014 //                 www.amd.com
00015 //
00016 // Permission is hereby granted, free of charge, to any person obtaining a copy
00017 // of this software and associated documentation files (the "Software"), to
00018 // deal with the Software without restriction, including without limitation
00019 // the rights to use, copy, modify, merge, publish, distribute, sublicense,
00020 // and/or sell copies of the Software, and to permit persons to whom the
00021 // Software is furnished to do so, subject to the following conditions:
00022 //
00023 //  - Redistributions of source code must retain the above copyright notice,
```

```
00024 //    this list of conditions and the following disclaimers.
00025 //  - Redistributions in binary form must reproduce the above copyright
00026 //    notice, this list of conditions and the following disclaimers in
00027 //    the documentation and/or other materials provided with the distribution.
00028 //  - Neither the names of Advanced Micro Devices, Inc,
00029 //    nor the names of its contributors may be used to endorse or promote
00030 //    products derived from this Software without specific prior written
00031 //    permission.
00032 //
00033 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00034 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00035 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
00036 // THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
00037 // OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
00038 // ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
00039 // DEALINGS WITH THE SOFTWARE.
00040 //
00042
00043 // HSA AMD extension for additional loader functionality.
00044
00045 #ifndef HSA_VEN_AMD_LOADER_H
00046 #define HSA_VEN_AMD_LOADER_H
00047
00048 #include "hsa.h"
00049
00050 #ifdef __cplusplus
00051 extern "C" {
00052 #endif /* __cplusplus */
00053
00086 hsa_status_t hsa_ven_amd_loader_query_host_address(
00087   const void *device_address,
00088   const void **host_address);
00089
00094 typedef enum {
00099   HSA_VEN_AMD_LOADER_CODE_OBJECT_STORAGE_TYPE_NONE = 0,
00104   HSA_VEN_AMD_LOADER_CODE_OBJECT_STORAGE_TYPE_FILE = 1,
00109   HSA_VEN_AMD_LOADER_CODE_OBJECT_STORAGE_TYPE_MEMORY = 2
00110 } hsa_ven_amd_loader_code_object_storage_type_t;
00111
00123 typedef struct hsa_ven_amd_loader_segment_descriptor_s {
00128   hsa_agent_t agent;
00132   hsa_executable_t executable;
00136   hsa_ven_amd_loader_code_object_storage_type_t code_object_storage_type;
00146   const void *code_object_storage_base;
00156   size_t code_object_storage_size;
00164   size_t code_object_storage_offset;
00168   const void *segment_base;
00172   size_t segment_size;
00173 } hsa_ven_amd_loader_segment_descriptor_t;
00174
00223 hsa_status_t hsa_ven_amd_loader_query_segment_descriptors(
00224   hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors,
00225   size_t *num_segment_descriptors);
00226
00241 hsa_status_t hsa_ven_amd_loader_query_executable(
00242   const void *device_address,
00243   hsa_executable_t *executable);
00244
00245 //===----------------------------------------------------------------------===//
00246
00273 hsa_status_t hsa_ven_amd_loader_executable_iterate_loaded_code_objects(
00274   hsa_executable_t executable,
00275   hsa_status_t (*callback)(
00276     hsa_executable_t executable,
00277     hsa_loaded_code_object_t loaded_code_object,
00278     void *data),
00279   void *data);
00280
00284 typedef enum {
00288   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_KIND_PROGRAM = 1,
00292   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_KIND_AGENT = 2
00293 } hsa_ven_amd_loader_loaded_code_object_kind_t;
00294
00298 typedef enum hsa_ven_amd_loader_loaded_code_object_info_e {
00303   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_EXECUTABLE = 1,
00308   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_KIND = 2,
00316   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_AGENT = 3,
00322   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_CODE_OBJECT_STORAGE_TYPE = 4,
00330   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_CODE_OBJECT_STORAGE_MEMORY_BASE = 5,
00338   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_CODE_OBJECT_STORAGE_MEMORY_SIZE = 6,
00346   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_CODE_OBJECT_STORAGE_FILE = 7,
00354   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_LOAD_DELTA = 8,
00363   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_LOAD_BASE = 9,
00369   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_LOAD_SIZE = 10,
00374   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_URI_LENGTH = 11,
00415   HSA_VEN_AMD_LOADER_LOADED_CODE_OBJECT_INFO_URI = 12,
00416 } hsa_ven_amd_loader_loaded_code_object_info_t;
```

```
00417
00441 hsa_status_t hsa_ven_amd_loader_loaded_code_object_get_info(
00442   hsa_loaded_code_object_t loaded_code_object,
00443   hsa_ven_amd_loader_loaded_code_object_info_t attribute,
00444   void *value);
00445
00446 //===----------------------------------------------------------------------===//
00447
00487 hsa_status_t
00488 hsa_ven_amd_loader_code_object_reader_create_from_file_with_offset_size(
00489     hsa_file_t file,
00490     size_t offset,
00491     size_t size,
00492     hsa_code_object_reader_t *code_object_reader);
00493
00494 //===----------------------------------------------------------------------===//
00495
00521 hsa_status_t
00522 hsa_ven_amd_loader_iterate_executables(
00523     hsa_status_t (*callback)(
00524       hsa_executable_t executable,
00525       void *data),
00526     void *data);
00527
00528 //===----------------------------------------------------------------------===//
00529
00533 #define hsa_ven_amd_loader 001003
00534
00538 typedef struct hsa_ven_amd_loader_1_00_pfn_s {
00539   hsa_status_t (*hsa_ven_amd_loader_query_host_address)(
00540     const void *device_address,
00541     const void **host_address);
00542
00543   hsa_status_t (*hsa_ven_amd_loader_query_segment_descriptors)(
00544     hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors,
00545     size_t *num_segment_descriptors);
00546
00547   hsa_status_t (*hsa_ven_amd_loader_query_executable)(
00548     const void *device_address,
00549     hsa_executable_t *executable);
00550 } hsa_ven_amd_loader_1_00_pfn_t;
00551
00555 typedef struct hsa_ven_amd_loader_1_01_pfn_s {
00556   hsa_status_t (*hsa_ven_amd_loader_query_host_address)(
00557     const void *device_address,
00558     const void **host_address);
00559
00560   hsa_status_t (*hsa_ven_amd_loader_query_segment_descriptors)(
00561     hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors,
00562     size_t *num_segment_descriptors);
00563
00564   hsa_status_t (*hsa_ven_amd_loader_query_executable)(
00565     const void *device_address,
00566     hsa_executable_t *executable);
00567
00568   hsa_status_t (*hsa_ven_amd_loader_executable_iterate_loaded_code_objects)(
00569     hsa_executable_t executable,
00570     hsa_status_t (*callback)(
00571       hsa_executable_t executable,
00572       hsa_loaded_code_object_t loaded_code_object,
00573       void *data),
00574     void *data);
00575
00576   hsa_status_t (*hsa_ven_amd_loader_loaded_code_object_get_info)(
00577     hsa_loaded_code_object_t loaded_code_object,
00578     hsa_ven_amd_loader_loaded_code_object_info_t attribute,
00579     void *value);
00580 } hsa_ven_amd_loader_1_01_pfn_t;
00581
00585 typedef struct hsa_ven_amd_loader_1_02_pfn_s {
00586   hsa_status_t (*hsa_ven_amd_loader_query_host_address)(
00587     const void *device_address,
00588     const void **host_address);
00589
00590   hsa_status_t (*hsa_ven_amd_loader_query_segment_descriptors)(
00591     hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors,
00592     size_t *num_segment_descriptors);
00593
00594   hsa_status_t (*hsa_ven_amd_loader_query_executable)(
00595     const void *device_address,
00596     hsa_executable_t *executable);
00597
00598   hsa_status_t (*hsa_ven_amd_loader_executable_iterate_loaded_code_objects)(
00599     hsa_executable_t executable,
00600     hsa_status_t (*callback)(
00601       hsa_executable_t executable,
00602       hsa_loaded_code_object_t loaded_code_object,
```

```
00603       void *data),
00604     void *data);
00605
00606   hsa_status_t (*hsa_ven_amd_loader_loaded_code_object_get_info)(
00607     hsa_loaded_code_object_t loaded_code_object,
00608     hsa_ven_amd_loader_loaded_code_object_info_t attribute,
00609     void *value);
00610
00611   hsa_status_t
00612     (*hsa_ven_amd_loader_code_object_reader_create_from_file_with_offset_size)(
00613       hsa_file_t file,
00614       size_t offset,
00615       size_t size,
00616       hsa_code_object_reader_t *code_object_reader);
00617 } hsa_ven_amd_loader_1_02_pfn_t;
00618
00622 typedef struct hsa_ven_amd_loader_1_03_pfn_s {
00623   hsa_status_t (*hsa_ven_amd_loader_query_host_address)(
00624     const void *device_address,
00625     const void **host_address);
00626
00627   hsa_status_t (*hsa_ven_amd_loader_query_segment_descriptors)(
00628     hsa_ven_amd_loader_segment_descriptor_t *segment_descriptors,
00629     size_t *num_segment_descriptors);
00630
00631   hsa_status_t (*hsa_ven_amd_loader_query_executable)(
00632     const void *device_address,
00633     hsa_executable_t *executable);
00634
00635   hsa_status_t (*hsa_ven_amd_loader_executable_iterate_loaded_code_objects)(
00636     hsa_executable_t executable,
00637     hsa_status_t (*callback)(
00638       hsa_executable_t executable,
00639       hsa_loaded_code_object_t loaded_code_object,
00640       void *data),
00641     void *data);
00642
00643   hsa_status_t (*hsa_ven_amd_loader_loaded_code_object_get_info)(
00644     hsa_loaded_code_object_t loaded_code_object,
00645     hsa_ven_amd_loader_loaded_code_object_info_t attribute,
00646     void *value);
00647
00648   hsa_status_t
00649     (*hsa_ven_amd_loader_code_object_reader_create_from_file_with_offset_size)(
00650       hsa_file_t file,
00651       size_t offset,
00652       size_t size,
00653       hsa_code_object_reader_t *code_object_reader);
00654
00655   hsa_status_t
00656     (*hsa_ven_amd_loader_iterate_executables)(
00657       hsa_status_t (*callback)(
00658         hsa_executable_t executable,
00659         void *data),
00660       void *data);
00661 } hsa_ven_amd_loader_1_03_pfn_t;
00662
00663 #ifdef __cplusplus
00664 }
00665 #endif /* __cplusplus */
00666
00667 #endif /* HSA_VEN_AMD_LOADER_H */
```

Index