

Asynchronous Task and Memory Interface (ATMI)

Version 0.3

Generated by Doxygen 1.8.6

Sun Jul 10 2016 18:19:16

Contents

1	Module Index	1
1.1	Modules	1
2	Class Index	3
2.1	Class List	3
3	Module Documentation	5
3.1	Kernel Handles	5
3.1.1	Detailed Description	5
3.1.2	Typedef Documentation	5
3.1.2.1	atmi_kernel_t	6
3.1.3	Function Documentation	6
3.1.3.1	atmi_kernel_add_cpu_impl	6
3.1.3.2	atmi_kernel_add_gpu_impl	6
3.1.3.3	atmi_kernel_create_empty	6
3.1.3.4	atmi_kernel_release	7
3.2	ATMI Context Setup and Finalize	8
3.2.1	Detailed Description	8
3.2.2	Function Documentation	8
3.2.2.1	atmi_finalize	8
3.2.2.2	atmi_init	8
3.3	ATMI Module	9
3.3.1	Detailed Description	9
3.3.2	Function Documentation	9
3.3.2.1	atmi_module_register	9
3.3.2.2	atmi_module_register_from_memory	9
3.4	ATMI Machine	11
3.4.1	Detailed Description	11
3.4.2	Function Documentation	11
3.4.2.1	atmi_machine_get_info	11
3.5	ATMI Task	12
3.5.1	Detailed Description	12

3.5.2	Function Documentation	12
3.5.2.1	atmi_task_group_sync	12
3.5.2.2	atmi_task_launch	12
3.5.2.3	atmi_task_wait	12
3.6	ATMI Data Management	14
3.6.1	Detailed Description	14
3.6.2	Function Documentation	14
3.6.2.1	atmi_free	14
3.6.2.2	atmi_malloc	14
3.6.2.3	atmi_memcpy	15
3.6.2.4	atmi_memcpy_async	16
3.7	ATMI CPU Device Runtime	17
3.7.1	Detailed Description	17
3.7.2	Function Documentation	17
3.7.2.1	get_atmi_task_handle	17
3.8	Enumerated Types	18
3.8.1	Detailed Description	18
3.8.2	Enumeration Type Documentation	18
3.8.2.1	atmi_arg_type_s	18
3.8.2.2	atmi_devtype_s	19
3.8.2.3	atmi_memtype_s	19
3.8.2.4	atmi_platform_type_t	19
3.8.2.5	atmi_scheduler_s	19
3.8.2.6	atmi_state_s	19
3.8.2.7	atmi_status_t	19
3.9	Common ATMI Structures	20
3.9.1	Detailed Description	21
4	Class Documentation	23
4.1	atmi_context_s Struct Reference	23
4.2	atmi_cparm_s Struct Reference	23
4.2.1	Member Data Documentation	23
4.2.1.1	group	23
4.2.1.2	groupable	23
4.2.1.3	num_required	24
4.2.1.4	profilable	24
4.2.1.5	requires	24
4.2.1.6	synchronous	24
4.2.1.7	task_info	24
4.3	atmi_data_s Struct Reference	24

4.3.1	Member Data Documentation	24
4.3.1.1	place	24
4.3.1.2	ptr	24
4.3.1.3	size	24
4.4	atmi_device_s Struct Reference	25
4.4.1	Member Data Documentation	25
4.4.1.1	memory_pool_count	25
4.4.1.2	type	25
4.5	atmi_kernel_s Struct Reference	25
4.5.1	Detailed Description	25
4.5.2	Member Data Documentation	25
4.5.2.1	handle	25
4.6	atmi_lparm_s Struct Reference	26
4.6.1	Member Data Documentation	26
4.6.1.1	acquire_scope	26
4.6.1.2	atmi_id	26
4.6.1.3	continuation_task	26
4.6.1.4	gridDim	26
4.6.1.5	group	26
4.6.1.6	groupable	26
4.6.1.7	groupDim	27
4.6.1.8	kernel_id	27
4.6.1.9	needs_any	27
4.6.1.10	num_needs_any	27
4.6.1.11	num_required	27
4.6.1.12	place	27
4.6.1.13	profilable	27
4.6.1.14	release_scope	27
4.6.1.15	requires	27
4.6.1.16	synchronous	27
4.6.1.17	task_info	27
4.7	atmi_machine_s Struct Reference	28
4.7.1	Member Data Documentation	28
4.7.1.1	device_count_by_type	28
4.7.1.2	devices_by_type	28
4.8	atmi_mem_place_s Struct Reference	28
4.8.1	Member Data Documentation	28
4.8.1.1	dev_id	28
4.8.1.2	dev_type	28
4.8.1.3	mem_id	28

4.8.1.4	node_id	29
4.9	atmi_place_s Struct Reference	29
4.9.1	Member Data Documentation	29
4.9.1.1	cu_mask	29
4.9.1.2	device_id	29
4.9.1.3	node_id	29
4.9.1.4	type	29
4.10	atmi_task_group_s Struct Reference	29
4.10.1	Member Data Documentation	30
4.10.1.1	id	30
4.10.1.2	maxsize	30
4.10.1.3	place	30
4.11	atmi_task_s Struct Reference	30
4.11.1	Member Data Documentation	30
4.11.1.1	continuation	30
4.11.1.2	handle	30
4.11.1.3	profile	30
4.11.1.4	state	31
4.12	atmi_tprofile_s Struct Reference	31
4.12.1	Member Data Documentation	31
4.12.1.1	dispatch_time	31
4.12.1.2	end_time	31
4.12.1.3	ready_time	31
4.12.1.4	start_time	31
Index		32

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Kernel Handles	5
ATMI Context Setup and Finalize	8
ATMI Module	9
ATMI Machine	11
ATMI Task	12
ATMI Data Management	14
ATMI CPU Device Runtime	17
Enumerated Types	18
Common ATMI Structures	20

Chapter 2

Class Index

2.1 Class List

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

atmi_context_s	23
atmi_cparm_s	
The ATMI Data Copy Parameter Data Structure	23
atmi_data_s	
High-level data abstraction	24
atmi_device_s	
ATMI Device Structure	25
atmi_kernel_s	
Opaque handle representing an ATMI Kernel	25
atmi_lparm_s	
The ATMI Launch Parameter Data Structure	26
atmi_machine_s	
ATMI Machine Structure	28
atmi_mem_place_s	
ATMI Memory Place	28
atmi_place_s	
ATMI Compute Place	29
atmi_task_group_s	
ATMI Task Group Data Structure	29
atmi_task_s	30
atmi_tprofile_s	
ATMI Task Profile Data Structure	31

Chapter 3

Module Documentation

3.1 Kernel Handles

Classes

- struct [atmi_kernel_s](#)
Opaque handle representing an ATMI Kernel.

Typedefs

- typedef struct [atmi_kernel_s](#) [atmi_kernel_t](#)
Opaque handle representing an ATMI Kernel.
- typedef void(* [atmi_generic_fp](#))(void)
A generic function pointer representing CPU tasks.

Functions

- [atmi_status_t](#) [atmi_kernel_create_empty](#) ([atmi_kernel_t](#) *kernel, const int num_args, const size_t *arg_sizes)
Create an empty kernel opaque structure.
- [atmi_status_t](#) [atmi_kernel_add_gpu_impl](#) ([atmi_kernel_t](#) atmi_kernel, const char *impl, const unsigned int ID)
Add a GPU kernel implementation.
- [atmi_status_t](#) [atmi_kernel_add_cpu_impl](#) ([atmi_kernel_t](#) atmi_kernel, [atmi_generic_fp](#) impl, const unsigned int ID)
Add a CPU kernel implementation.
- [atmi_status_t](#) [atmi_kernel_release](#) ([atmi_kernel_t](#) kernel)
Release the kernel and all of its implementations.

3.1.1 Detailed Description

This module includes all kernel-related classes, structs and functions.

3.1.2 Typedef Documentation

3.1.2.1 typedef struct atmi_kernel_s atmi_kernel_t

ATMI kernels are instantiated in two steps. First, an empty kernel is created. Next, architecture specific implementations are added. Each kernel can have several implementations, but should have at least one implementation

3.1.3 Function Documentation

3.1.3.1 atmi_status_t atmi_kernel_add_cpu_impl (atmi_kernel_t *atmi_kernel*, atmi_generic_fp *impl*, const unsigned int *ID*)

An ATMI CPU kernel implementation is identified by a function pointer. The implementation must have the same number of arguments as the kernel. A unique user-specified identifier is associated with each implementation. The advanced user may want to run the specific implementation of the kernel by using the unique identifier in the launch parameter of task launch functions.

Parameters

in	<i>kernel</i>	The opaque kernel handle.
in	<i>impl</i>	The kernel implementation function pointer.
in	<i>ID</i>	The user-specified unique kernel identifier.

Return values

ATMI_STATUS_SUCCESS	The function has executed successfully.
ATMI_STATUS_ERROR	The function encountered errors.
ATMI_STATUS_UNKNOWN	The function encountered errors.

3.1.3.2 atmi_status_t atmi_kernel_add_gpu_impl (atmi_kernel_t *atmi_kernel*, const char * *impl*, const unsigned int *ID*)

An ATMI GPU kernel implementation is identified by a char array. The implementation must have the same number of arguments as the kernel. A unique user-specified identifier is associated with each implementation. The advanced user may want to run the specific implementation of the kernel by using the unique identifier in the launch parameter of task launch functions.

Parameters

in	<i>kernel</i>	The opaque kernel handle.
in	<i>impl</i>	The kernel implementation name.
in	<i>ID</i>	The user-specified unique kernel identifier.

Return values

ATMI_STATUS_SUCCESS	The function has executed successfully.
ATMI_STATUS_ERROR	The function encountered errors.
ATMI_STATUS_UNKNOWN	The function encountered errors.

3.1.3.3 atmi_status_t atmi_kernel_create_empty (atmi_kernel_t * *kernel*, const int *num_args*, const size_t * *arg_sizes*)

ATMI kernels are instantiated in two steps. First, an empty kernel is created. Next, architecture specific implementations are added. Each kernel can have several implementations, but should have at least one implementation. The opaque kernel handle acts as a key to identify the set of kernel implementations.

Parameters

out	<i>kernel</i>	The opaque kernel handle.
in	<i>num_args</i>	Number of arguments of the kernel. All implementations must have the same number of input arguments. May be 0.
in	<i>arg_sizes</i>	Size of each argument. May be NULL only if <i>num_args</i> is 0.

Return values

<i>ATMI_STATUS_SUCCESS</i>	The function has executed successfully.
<i>ATMI_STATUS_ERROR</i>	The function encountered errors.
<i>ATMI_STATUS_UNKNOWN</i>	The function encountered errors.

3.1.3.4 `atmi_status_t atmi_kernel_release (atmi_kernel_t kernel)`

After the kernel is released, its implementations may not be used to launch any ATMI tasks.

Parameters

in	<i>kernel</i>	The opaque kernel handle.
----	---------------	---------------------------

Return values

<i>ATMI_STATUS_SUCCESS</i>	The function has executed successfully.
<i>ATMI_STATUS_ERROR</i>	The function encountered errors.
<i>ATMI_STATUS_UNKNOWN</i>	The function encountered errors.

3.2 ATMI Context Setup and Finalize

Functions

- [atmi_status_t atmi_init \(atmi_devtype_t type\)](#)
Initialize the ATMI runtime environment.
- [atmi_status_t atmi_finalize \(\)](#)
Finalize the ATMI runtime environment.

3.2.1 Detailed Description

3.2.2 Function Documentation

3.2.2.1 [atmi_status_t atmi_finalize \(\)](#)

ATMI runtime functions will fail if called after finalize.

Return values

ATMI_STATUS_SUCCESS	The function has executed successfully.
ATMI_STATUS_ERROR	The function encountered errors.
ATMI_STATUS_UNKNOWN	The function encountered errors.

3.2.2.2 [atmi_status_t atmi_init \(atmi_devtype_t type \)](#)

All ATMI runtime functions will fail if this function is not called at least once. The user may initialize different device types at different regions in the program in order for optimization purposes.

Parameters

<code>in</code>	<code>type</code>	The types of devices that will be used by the application.
-----------------	-------------------	--

Return values

ATMI_STATUS_SUCCESS	The function has executed successfully.
ATMI_STATUS_ERROR	The function encountered errors.
ATMI_STATUS_UNKNOWN	The function encountered errors.

3.3 ATMI Module

Functions

- [atmi_status_t atmi_module_register](#) (const char **filenames, [atmi_platform_type_t](#) *types, const int num_modules)
Register the ATMI code module from file.
- [atmi_status_t atmi_module_register_from_memory](#) (void **modules, size_t *module_sizes, [atmi_platform_type_t](#) *types, const int num_modules)
Register the ATMI code module from memory.

3.3.1 Detailed Description

3.3.2 Function Documentation

3.3.2.1 [atmi_status_t atmi_module_register](#) (const char ** filenames, [atmi_platform_type_t](#) * types, const int num_modules)

Currently, only GPU devices need explicit module registration because of their specific ISAs that require a separate compilation phase. On the other hand, CPU devices execute regular x86 functions that are compiled with the host program.

Parameters

in	<i>filenames</i>	A collection of files that contain the GPU modules targeting either the BRIG or AMDGCN platform types. Value cannot be NULL.
in	<i>types</i>	A collection of platform types corresponding to the files. Value cannot be NULL.
in	<i>num_modules</i>	Size of <i>filenames</i> and <i>types</i> . Value should be greater than 0.

Return values

ATMI_STATUS_SUCCESS	The function has executed successfully.
ATMI_STATUS_ERROR	The function encountered errors.
ATMI_STATUS_UNKNOWN	The function encountered errors.

3.3.2.2 [atmi_status_t atmi_module_register_from_memory](#) (void ** modules, size_t * module_sizes, [atmi_platform_type_t](#) * types, const int num_modules)

Currently, only GPU devices need explicit module registration because of their specific ISAs that require a separate compilation phase. On the other hand, CPU devices execute regular x86 functions that are compiled with the host program.

Parameters

in	<i>modules</i>	A collection of memory regions that contain the GPU modules targeting either the BRIG or AMDGCN platform types. Value cannot be NULL.
in	<i>module_sizes</i>	Sizes of each module region in <i>modules</i> . Value cannot be NULL.
in	<i>types</i>	A collection of platform types corresponding to the modules. Value cannot be NULL.

in	<i>num_modules</i>	Size of <code>modules</code> , <code>module_sizes</code> and <code>types</code> . Value should be greater than 0.
----	--------------------	---

Return values

<i>ATMI_STATUS_SUCCESS</i>	The function has executed successfully.
<i>ATMI_STATUS_ERROR</i>	The function encountered errors.
<i>ATMI_STATUS_UNKNOWN</i>	The function encountered errors.

3.4 ATMI Machine

Functions

- `atmi_machine_t * atmi_machine_get_info ()`

ATMI's device discovery function to get the current machine's topology.

3.4.1 Detailed Description

3.4.2 Function Documentation

3.4.2.1 `atmi_machine_t * atmi_machine_get_info ()`

The `atmi_machine_t` structure is a tree-based representation of the compute and memory elements in the current node. Once ATMI is initialized, this function can be called to retrieve the pointer to this global structure.

Returns

Returns a pointer to a global structure of type `atmi_machine_t`. Returns NULL if ATMI is not initialized.

3.5 ATMI Task

Functions

- [atmi_task_handle_t atmi_task_launch \(atmi_lparm_t *lparm, atmi_kernel_t kernel, void **args\)](#)
The ATMI task launcher function.
- [atmi_status_t atmi_task_wait \(atmi_task_handle_t task\)](#)
Wait for a launched task or a data movement operation.
- [atmi_status_t atmi_task_group_sync \(atmi_task_group_t *group\)](#)
Wait for the launched task group, which could be a group of compute tasks and data movement tasks.

3.5.1 Detailed Description

3.5.2 Function Documentation

3.5.2.1 [atmi_status_t atmi_task_group_sync \(atmi_task_group_t * group \)](#)

Parameters

in	group	The task group of already launched tasks or an in-flight data movement operations.
--------------------	-----------------------	--

Return values

ATMI_STATUS_SUCCESS	The function has executed successfully.
ATMI_STATUS_ERROR	The function encountered errors.
ATMI_STATUS_UNKNOWN	The function encountered errors.

3.5.2.2 [atmi_task_handle_t atmi_task_launch \(atmi_lparm_t * lparm, atmi_kernel_t kernel, void ** args \)](#)

This function is used to launch any ATMI task (CPU or GPU). The `kernel` parameter specifies what has to be launched. The `lparm` structure defines the task's launch parameters, which will guide the ATMI runtime how to launch and manage the task.

Parameters

in	lparm	The structure describing how the task has to be managed.
in	kernel	The opaque kernel handle, which denotes what has to be launched.
in	args	The bag of arguments all passed by reference. Their sizes should be consistent with the kernel's <code>arg_sizes</code> parameter.

Returns

A handle to the ATMI task. The task handle may be used to setup dependencies with other copy and compute tasks or for explicit synchronization by the host.

3.5.2.3 [atmi_status_t atmi_task_wait \(atmi_task_handle_t task \)](#)

Parameters

<code>in</code>	<code>task</code>	The handle to an already launched task or an in-flight data movement operation.
-----------------	-------------------	---

Return values

<code>ATMI_STATUS_SUCCESS</code>	The function has executed successfully.
<code>ATMI_STATUS_ERROR</code>	The function encountered errors.
<code>ATMI_STATUS_UNKNOWN</code>	The function encountered errors.

3.6 ATMI Data Management

Functions

- [atmi_status_t atmi_malloc](#) (void **ptr, size_t size, [atmi_mem_place_t](#) place)
Allocate memory from the specified memory place.
- [atmi_status_t atmi_free](#) (void *ptr)
Frees memory that was previously allocated.
- [atmi_status_t atmi_memcpy](#) (void *dest, const void *src, size_t size)
Synchronously copy memory from the source to destination memory locations.
- [atmi_task_handle_t atmi_memcpy_async](#) ([atmi_cparm_t](#) *cparm, void *dest, const void *src, size_t size)
Asynchronously copy memory from the source to destination memory locations.

3.6.1 Detailed Description

3.6.2 Function Documentation

3.6.2.1 [atmi_status_t atmi_free](#) (void * ptr)

This function frees memory that was previously allocated by calling [atmi_malloc](#). It throws an error otherwise. It is illegal to access a pointer after a call to this function.

Parameters

in	<i>ptr</i>	The pointer to the memory that has to be freed.
----	------------	---

Return values

ATMI_STATUS_SUCCESS	The function has executed successfully.
ATMI_STATUS_ERROR	The function encountered errors.
ATMI_STATUS_UNKNOWN	The function encountered errors.

3.6.2.2 [atmi_status_t atmi_malloc](#) (void ** ptr, size_t size, [atmi_mem_place_t](#) place)

This function allocates memory from the specified memory place. If the memory place belongs primarily to the CPU, then the memory will be accessible by other GPUs and CPUs in the system. If the memory place belongs primarily to a GPU, then it cannot be accessed by other devices in the system.

Parameters

in	<i>ptr</i>	The pointer to the memory that will be allocated.
in	<i>size</i>	The size of the allocation in bytes.
in	<i>place</i>	The memory place in the system to perform the allocation.

Return values

ATMI_STATUS_SUCCESS	The function has executed successfully.
ATMI_STATUS_ERROR	The function encountered errors.
ATMI_STATUS_UNKNOWN	The function encountered errors.

3.6.2.3 `atmi_status_t atmi_memcpy (void * dest, const void * src, size_t size)`

This function assumes that the source and destination regions are non-overlapping. The runtime determines the memory place of the source and the destination and executes the appropriate optimized data movement methodology.

Parameters

in	<i>dest</i>	The destination pointer previously allocated by a system allocator or <code>atmi_malloc</code> .
in	<i>src</i>	The source pointer previously allocated by a system allocator or <code>atmi_malloc</code> .
in	<i>size</i>	The size of the data to be copied in bytes.

Return values

<code>ATMI_STATUS_SUCCESS</code>	The function has executed successfully.
<code>ATMI_STATUS_ERROR</code>	The function encountered errors.
<code>ATMI_STATUS_UNKNOWN</code>	The function encountered errors.

3.6.2.4 `atmi_task_handle_t atmi_memcpy_async (atmi_cparm_t * cparm, void * dest, const void * src, size_t size)`

This function assumes that the source and destination regions are non-overlapping. The runtime determines the memory place of the source and the destination and executes the appropriate optimized data movement methodology. This function is equivalent to an asynchronous task, which means that it can be used to setup dependencies with other memory copy routines or compute tasks. The `cparm` structure can be used to provide additional information about the copy operation.

Parameters

in	<i>cparm</i>	The structure describing how the copy task has to be managed.
in	<i>dest</i>	The destination pointer previously allocated by a system allocator or <code>atmi_malloc</code> .
in	<i>src</i>	The source pointer previously allocated by a system allocator or <code>atmi_malloc</code> .
in	<i>size</i>	The size of the data to be copied in bytes.

Returns

A handle to the ATMI task. The task handle may be used to setup dependencies with other copy and compute tasks or for explicit synchronization by the host.

3.7 ATMI CPU Device Runtime

Functions

- [atmi_task_handle_t get_atmi_task_handle \(\)](#)

Retrieve the task handle of the currently running task. This function is valid only within the body of a CPU task.

3.7.1 Detailed Description

3.7.2 Function Documentation

3.7.2.1 `atmi_task_handle_t get_atmi_task_handle ()`

Returns

A handle to the ATMI task.

3.8 Enumerated Types

Typedefs

- typedef enum [atmi_status_t](#) [atmi_status_t](#)
Status codes.
- typedef enum [atmi_devtype_s](#) [atmi_devtype_t](#)
Device Types.
- typedef enum [atmi_memtype_s](#) [atmi_memtype_t](#)
Memory Access Type.
- typedef enum [atmi_state_s](#) [atmi_state_t](#)
Task States.
- typedef enum [atmi_scheduler_s](#) [atmi_scheduler_t](#)
Scheduler Types.
- typedef enum [atmi_arg_type_s](#) [atmi_arg_type_t](#)
ATMI data arg types.

Enumerations

- enum [atmi_status_t](#) { [ATMI_STATUS_SUCCESS](#) = 0, [ATMI_STATUS_UNKNOWN](#) = 1, [ATMI_STATUS_ERROR](#) = 2 }
- Status codes.*
- enum [atmi_platform_type_t](#) { [BRIG](#) = 0, [AMDGCN](#) }
- Platform Types.*
- enum [atmi_devtype_s](#) {
[ATMI_DEVTYPE_CPU](#) = 0x0001, [ATMI_DEVTYPE_iGPU](#) = 0x0010, [ATMI_DEVTYPE_dGPU](#) = 0x0100, [ATMI_DEVTYPE_GPU](#) = [ATMI_DEVTYPE_iGPU](#) | [ATMI_DEVTYPE_dGPU](#),
[ATMI_DEVTYPE_DSP](#) = 0x1000, [ATMI_DEVTYPE_ALL](#) = 0x1111 }
- Device Types.*
- enum [atmi_memtype_s](#) { [ATMI_MEMTYPE_FINE_GRAINED](#) = 0, [ATMI_MEMTYPE_COARSE_GRAINED](#) = 1, [ATMI_MEMTYPE_ANY](#) }
- Memory Access Type.*
- enum [atmi_state_s](#) {
[ATMI_INITIALIZED](#) = 0, [ATMI_READY](#) = 1, [ATMI_DISPATCHED](#) = 2, [ATMI_COMPLETED](#) = 3,
[ATMI_FAILED](#) = -1 }
- Task States.*
- enum [atmi_scheduler_s](#) { [ATMI_SCHED_NONE](#) = 0, [ATMI_SCHED_RR](#) }
- Scheduler Types.*
- enum [atmi_arg_type_s](#) { [ATMI_IN](#), [ATMI_OUT](#), [ATMI_IN_OUT](#) }
- ATMI data arg types.*

3.8.1 Detailed Description

3.8.2 Enumeration Type Documentation

3.8.2.1 enum [atmi_arg_type_s](#)

Enumerator

- [ATMI_IN](#)** Input argument
- [ATMI_OUT](#)** Output argument
- [ATMI_IN_OUT](#)** In/out argument

3.8.2.2 enum atmi_devtype_s

Enumerator

ATMI_DEVTYPE_CPU CPU
ATMI_DEVTYPE_iGPU Integrated GPU
ATMI_DEVTYPE_dGPU Discrete GPU
ATMI_DEVTYPE_GPU Any GPU
ATMI_DEVTYPE_DSP Digital Signal Processor
ATMI_DEVTYPE_ALL Union of all device types

3.8.2.3 enum atmi_memtype_s

Enumerator

ATMI_MEMTYPE_FINE_GRAINED Fine grained memory type
ATMI_MEMTYPE_COARSE_GRAINED Coarse grained memory type
ATMI_MEMTYPE_ANY Any memory type

3.8.2.4 enum atmi_platform_type_t

Enumerator

BRIG Target Platform is BRIG (deprecated)
AMDGCN Target Platform is AMD GCN (default)

3.8.2.5 enum atmi_scheduler_s

Enumerator

ATMI_SCHED_NONE No scheduler, all tasks go to the same queue
ATMI_SCHED_RR Round-robin scheduler

3.8.2.6 enum atmi_state_s

Enumerator

ATMI_INITIALIZED Initialized state
ATMI_READY Ready state
ATMI_DISPATCHED Dispatched state
ATMI_COMPLETED Completed state
ATMI_FAILED Failed state

3.8.2.7 enum atmi_status_t

Enumerator

ATMI_STATUS_SUCCESS The function has been executed successfully.
ATMI_STATUS_UNKNOWN A undocumented error has occurred.
ATMI_STATUS_ERROR A generic error has occurred.

3.9 Common ATMI Structures

Classes

- struct [atmi_tprofile_s](#)
ATMI Task Profile Data Structure.
- struct [atmi_place_s](#)
ATMI Compute Place.
- struct [atmi_mem_place_s](#)
ATMI Memory Place.
- struct [atmi_device_s](#)
ATMI Device Structure.
- struct [atmi_machine_s](#)
ATMI Machine Structure.
- struct [atmi_task_group_s](#)
ATMI Task Group Data Structure.
- struct [atmi_task_s](#)
- struct [atmi_lparm_s](#)
The ATMI Launch Parameter Data Structure.
- struct [atmi_cparm_s](#)
The ATMI Data Copy Parameter Data Structure.
- struct [atmi_data_s](#)
High-level data abstraction.

Typedefs

- typedef struct [atmi_tprofile_s](#) [atmi_tprofile_t](#)
ATMI Task Profile Data Structure.
- typedef struct [atmi_place_s](#) [atmi_place_t](#)
ATMI Compute Place.
- typedef struct [atmi_mem_place_s](#) [atmi_mem_place_t](#)
ATMI Memory Place.
- typedef struct [atmi_device_s](#) [atmi_device_t](#)
ATMI Device Structure.
- typedef struct [atmi_machine_s](#) [atmi_machine_t](#)
ATMI Machine Structure.
- typedef struct [atmi_task_group_s](#) [atmi_task_group_t](#)
ATMI Task Group Data Structure.
- typedef void * [atmi_handle_t](#)
ATMI Task info structure.
- typedef struct [atmi_task_s](#) [atmi_task_t](#)
- typedef unsigned long int [atmi_task_handle_t](#)
The ATMI task handle.
- typedef struct [atmi_lparm_s](#) [atmi_lparm_t](#)
The ATMI Launch Parameter Data Structure.
- typedef struct [atmi_cparm_s](#) [atmi_cparm_t](#)
The ATMI Data Copy Parameter Data Structure.
- typedef struct [atmi_data_s](#) [atmi_data_t](#)
High-level data abstraction.

Variables

- [atmi_task_handle_t NULL_TASK](#)
The special NULL task handle.

3.9.1 Detailed Description

Chapter 4

Class Documentation

4.1 atmi_context_s Struct Reference

Public Attributes

- int **atmi_id**

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

- atmi.h

4.2 atmi_cparm_s Struct Reference

The ATMI Data Copy Parameter Data Structure.

```
#include <atmi.h>
```

Public Attributes

- [atmi_task_group_t](#) * group
- boolean groupable
- boolean profilable
- boolean synchronous
- int num_required
- [atmi_task_handle_t](#) * requires
- [atmi_task_t](#) * task_info

4.2.1 Member Data Documentation

4.2.1.1 atmi_task_group_t* atmi_cparm_s::group

Group for this task, Default= NULL

4.2.1.2 boolean atmi_cparm_s::groupable

Create signal for task, default = F

4.2.1.3 `int atmi_cparm_s::num_required`

of required parent tasks, default 0

4.2.1.4 `boolean atmi_cparm_s::profilable`

Points to tprofile if metrics desired

4.2.1.5 `atmi_task_handle_t* atmi_cparm_s::requires`

Array of required parent tasks

4.2.1.6 `boolean atmi_cparm_s::synchronous`

Async or Sync, default = F (async)

4.2.1.7 `atmi_task_t* atmi_cparm_s::task_info`

Optional user-created structure to store executed task's information

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

- `atmi.h`

4.3 `atmi_data_s` Struct Reference

High-level data abstraction.

```
#include <atmi.h>
```

Public Attributes

- `void * ptr`
- `unsigned int size`
- `atmi_mem_place_t place`

4.3.1 Member Data Documentation

4.3.1.1 `atmi_mem_place_t atmi_data_s::place`

The memory placement of data

4.3.1.2 `void* atmi_data_s::ptr`

The data pointer

4.3.1.3 `unsigned int atmi_data_s::size`

Data size

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

- `atmi.h`

4.4 atmi_device_s Struct Reference

ATMI Device Structure.

```
#include <atmi.h>
```

Public Attributes

- [atmi_devtype_t](#) type
- unsigned int [memory_pool_count](#)

4.4.1 Member Data Documentation

4.4.1.1 unsigned int atmi_device_s::memory_pool_count

Number of memory regions that are accessible from this device.

4.4.1.2 atmi_devtype_t atmi_device_s::type

Device type

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

- atmi.h

4.5 atmi_kernel_s Struct Reference

Opaque handle representing an ATMI Kernel.

```
#include <atmi_runtime.h>
```

Public Attributes

- uint64_t [handle](#)

4.5.1 Detailed Description

ATMI kernels are instantiated in two steps. First, an empty kernel is created. Next, architecture specific implementations are added. Each kernel can have several implementations, but should have at least one implementation

4.5.2 Member Data Documentation

4.5.2.1 uint64_t atmi_kernel_s::handle

Opaque handle.

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

- atmi_runtime.h

4.6 atmi_lparm_s Struct Reference

The ATMI Launch Parameter Data Structure.

```
#include <atmi.h>
```

Public Attributes

- unsigned long [gridDim](#) [3]
- unsigned long [groupDim](#) [3]
- [atmi_task_group_t](#) * [group](#)
- boolean [groupable](#)
- boolean [synchronous](#)
- int [acquire_scope](#)
- int [release_scope](#)
- int [num_required](#)
- [atmi_task_handle_t](#) * [requires](#)
- int [num_needs_any](#)
- [atmi_task_handle_t](#) * [needs_any](#)
- boolean [profilable](#)
- int [atmi_id](#)
- int [kernel_id](#)
- [atmi_place_t](#) [place](#)
- [atmi_task_t](#) * [task_info](#)
- [atmi_task_handle_t](#) [continuation_task](#)

4.6.1 Member Data Documentation

4.6.1.1 int atmi_lparm_s::acquire_scope

Memory model, default = 2

4.6.1.2 int atmi_lparm_s::atmi_id

Constant that PIFs can check for

4.6.1.3 atmi_task_handle_t atmi_lparm_s::continuation_task

The continuation task of this current task

4.6.1.4 unsigned long atmi_lparm_s::gridDim[3]

of global threads for each dimension

4.6.1.5 atmi_task_group_t* atmi_lparm_s::group

Group for this task, Default= NULL

4.6.1.6 boolean atmi_lparm_s::groupable

Create signal for task, default = F

4.6.1.7 unsigned long atmi_lparm_s::groupDim[3]

Thread group size for each dimension

4.6.1.8 int atmi_lparm_s::kernel_id

Kernel ID if more than one kernel per task

4.6.1.9 atmi_task_handle_t* atmi_lparm_s::needs_any

Array of needed parent tasks

4.6.1.10 int atmi_lparm_s::num_needs_any

needed parents, only 1 must complete

4.6.1.11 int atmi_lparm_s::num_required

of required parent tasks, default 0

4.6.1.12 atmi_place_t atmi_lparm_s::place

Compute location to launch this task.

4.6.1.13 boolean atmi_lparm_s::profilable

Points to tprofile if metrics desired

4.6.1.14 int atmi_lparm_s::release_scope

Memory model, default = 2

4.6.1.15 atmi_task_handle_t* atmi_lparm_s::requires

Array of required parent tasks

4.6.1.16 boolean atmi_lparm_s::synchronous

Async or Sync, default = F (async)

4.6.1.17 atmi_task_t* atmi_lparm_s::task_info

Optional user-created structure to store executed task's information

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

- atmi.h

4.7 atmi_machine_s Struct Reference

ATMI Machine Structure.

```
#include <atmi.h>
```

Public Attributes

- unsigned int [device_count_by_type](#) [[ATMI_DEVTYPE_ALL](#)]
- [atmi_device_t](#) * [devices_by_type](#) [[ATMI_DEVTYPE_ALL](#)]

4.7.1 Member Data Documentation

4.7.1.1 unsigned int atmi_machine_s::device_count_by_type[ATMI_DEVTYPE_ALL]

The number of devices categorized by the device type

4.7.1.2 atmi_device_t* atmi_machine_s::devices_by_type[ATMI_DEVTYPE_ALL]

The device structures categorized by the device type

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

- [atmi.h](#)

4.8 atmi_mem_place_s Struct Reference

ATMI Memory Place.

```
#include <atmi.h>
```

Public Attributes

- unsigned int [node_id](#)
- [atmi_devtype_t](#) [dev_type](#)
- int [dev_id](#)
- int [mem_id](#)

4.8.1 Member Data Documentation

4.8.1.1 int atmi_mem_place_s::dev_id

Devices ordered by runtime; -1 for any

4.8.1.2 atmi_devtype_t atmi_mem_place_s::dev_type

CPU, GPU or DSP

4.8.1.3 int atmi_mem_place_s::mem_id

Memory spaces; -1 for any

4.8.1.4 unsigned int atmi_mem_place_s::node_id

node_id = 0 for local computations

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

- atmi.h

4.9 atmi_place_s Struct Reference

ATMI Compute Place.

```
#include <atmi.h>
```

Public Attributes

- unsigned int [node_id](#)
- [atmi_devtype_t](#) type
- int [device_id](#)
- unsigned long [cu_mask](#)

4.9.1 Member Data Documentation

4.9.1.1 unsigned long atmi_place_s::cu_mask

Compute Unit Mask (advanced feature)

4.9.1.2 int atmi_place_s::device_id

Devices ordered by runtime; -1 for any

4.9.1.3 unsigned int atmi_place_s::node_id

node_id = 0 for local computations

4.9.1.4 atmi_devtype_t atmi_place_s::type

CPU, GPU or DSP

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

- atmi.h

4.10 atmi_task_group_s Struct Reference

ATMI Task Group Data Structure.

```
#include <atmi.h>
```

Public Attributes

- int [id](#)
- boolean **ordered**
- [atmi_place_t](#) [place](#)
- int [maxsize](#)

4.10.1 Member Data Documentation

4.10.1.1 int [atmi_task_group_s::id](#)

Unique task group identifier

4.10.1.2 int [atmi_task_group_s::maxsize](#)

Number of tasks allowed in group

4.10.1.3 [atmi_place_t](#) [atmi_task_group_s::place](#)

CUs to execute tasks; default: any

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

- [atmi.h](#)

4.11 [atmi_task_s](#) Struct Reference

Public Attributes

- [atmi_handle_t](#) [handle](#)
- [atmi_state_t](#) [state](#)
- [atmi_tprofile_t](#) [profile](#)
- struct [atmi_task_s](#) * [continuation](#)

4.11.1 Member Data Documentation

4.11.1.1 struct [atmi_task_s](#)* [atmi_task_s::continuation](#)

The continuation task of this current task

4.11.1.2 [atmi_handle_t](#) [atmi_task_s::handle](#)

Temp storage location for current task handle for DP

4.11.1.3 [atmi_tprofile_t](#) [atmi_task_s::profile](#)

Previously consistent profile information

4.11.1.4 atmi_state_t atmi_task_s::state

Previously consistent state of task

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

- atmi.h

4.12 atmi_tprofile_s Struct Reference

ATMI Task Profile Data Structure.

```
#include <atmi.h>
```

Public Attributes

- unsigned long int [dispatch_time](#)
- unsigned long int [ready_time](#)
- unsigned long int [start_time](#)
- unsigned long int [end_time](#)

4.12.1 Member Data Documentation

4.12.1.1 unsigned long int atmi_tprofile_s::dispatch_time

Timestamp of task dispatch.

4.12.1.2 unsigned long int atmi_tprofile_s::end_time

Timestamp when the task completed execution.

4.12.1.3 unsigned long int atmi_tprofile_s::ready_time

Timestamp when the task's dependencies were all met and ready to be dispatched.

4.12.1.4 unsigned long int atmi_tprofile_s::start_time

Timestamp when the task started execution.

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

- atmi.h

Index

AMDGCN
 Enumerated Types, [19](#)
ATMI_COMPLETED
 Enumerated Types, [19](#)
ATMI_DEVTYPE_ALL
 Enumerated Types, [19](#)
ATMI_DEVTYPE_CPU
 Enumerated Types, [19](#)
ATMI_DEVTYPE_DSP
 Enumerated Types, [19](#)
ATMI_DEVTYPE_GPU
 Enumerated Types, [19](#)
ATMI_DEVTYPE_dGPU
 Enumerated Types, [19](#)
ATMI_DEVTYPE_iGPU
 Enumerated Types, [19](#)
ATMI_DISPATCHED
 Enumerated Types, [19](#)
ATMI_FAILED
 Enumerated Types, [19](#)
ATMI_IN
 Enumerated Types, [18](#)
ATMI_IN_OUT
 Enumerated Types, [18](#)
ATMI_INITIALIZED
 Enumerated Types, [19](#)
ATMI_MEMTYPE_ANY
 Enumerated Types, [19](#)
ATMI_MEMTYPE_COARSE_GRAINED
 Enumerated Types, [19](#)
ATMI_MEMTYPE_FINE_GRAINED
 Enumerated Types, [19](#)
ATMI_OUT
 Enumerated Types, [18](#)
ATMI_READY
 Enumerated Types, [19](#)
ATMI_SCHED_NONE
 Enumerated Types, [19](#)
ATMI_SCHED_RR
 Enumerated Types, [19](#)
ATMI_STATUS_ERROR
 Enumerated Types, [19](#)
ATMI_STATUS_SUCCESS
 Enumerated Types, [19](#)
ATMI_STATUS_UNKNOWN
 Enumerated Types, [19](#)
ATMI CPU Device Runtime, [17](#)
 get_atmi_task_handle, [17](#)
ATMI Context Setup and Finalize, [8](#)

 atmi_finalize, [8](#)
 atmi_init, [8](#)
ATMI Data Management, [14](#)
 atmi_free, [14](#)
 atmi_malloc, [14](#)
 atmi_memcpy, [14](#)
 atmi_memcpy_async, [16](#)
ATMI Machine, [11](#)
 atmi_machine_get_info, [11](#)
ATMI Module, [9](#)
 atmi_module_register, [9](#)
 atmi_module_register_from_memory, [9](#)
ATMI Task, [12](#)
 atmi_task_group_sync, [12](#)
 atmi_task_launch, [12](#)
 atmi_task_wait, [12](#)
acquire_scope
 atmi_lparm_s, [26](#)
atmi_arg_type_s
 Enumerated Types, [18](#)
atmi_context_s, [23](#)
atmi_cparm_s, [23](#)
 group, [23](#)
 groupable, [23](#)
 num_required, [23](#)
 profilable, [24](#)
 requires, [24](#)
 synchronous, [24](#)
 task_info, [24](#)
atmi_data_s, [24](#)
 place, [24](#)
 ptr, [24](#)
 size, [24](#)
atmi_device_s, [25](#)
 memory_pool_count, [25](#)
 type, [25](#)
atmi_devtype_s
 Enumerated Types, [18](#)
atmi_finalize
 ATMI Context Setup and Finalize, [8](#)
atmi_free
 ATMI Data Management, [14](#)
atmi_id
 atmi_lparm_s, [26](#)
atmi_init
 ATMI Context Setup and Finalize, [8](#)
atmi_kernel_add_cpu_impl
 Kernel Handles, [6](#)
atmi_kernel_add_gpu_impl

- Kernel Handles, [6](#)
- atmi_kernel_create_empty
 - Kernel Handles, [6](#)
- atmi_kernel_release
 - Kernel Handles, [7](#)
- atmi_kernel_s, [25](#)
 - handle, [25](#)
- atmi_kernel_t
 - Kernel Handles, [5](#)
- atmi_lparm_s, [26](#)
 - acquire_scope, [26](#)
 - atmi_id, [26](#)
 - continuation_task, [26](#)
 - gridDim, [26](#)
 - group, [26](#)
 - groupDim, [26](#)
 - groupable, [26](#)
 - kernel_id, [27](#)
 - needs_any, [27](#)
 - num_needs_any, [27](#)
 - num_required, [27](#)
 - place, [27](#)
 - profilable, [27](#)
 - release_scope, [27](#)
 - requires, [27](#)
 - synchronous, [27](#)
 - task_info, [27](#)
- atmi_machine_get_info
 - ATMI Machine, [11](#)
- atmi_machine_s, [28](#)
 - device_count_by_type, [28](#)
 - devices_by_type, [28](#)
- atmi_malloc
 - ATMI Data Management, [14](#)
- atmi_mem_place_s, [28](#)
 - dev_id, [28](#)
 - dev_type, [28](#)
 - mem_id, [28](#)
 - node_id, [28](#)
- atmi_memcpy
 - ATMI Data Management, [14](#)
- atmi_memcpy_async
 - ATMI Data Management, [16](#)
- atmi_memtype_s
 - Enumerated Types, [19](#)
- atmi_module_register
 - ATMI Module, [9](#)
- atmi_module_register_from_memory
 - ATMI Module, [9](#)
- atmi_place_s, [29](#)
 - cu_mask, [29](#)
 - device_id, [29](#)
 - node_id, [29](#)
 - type, [29](#)
- atmi_platform_type_t
 - Enumerated Types, [19](#)
- atmi_scheduler_s
 - Enumerated Types, [19](#)
- atmi_state_s
 - Enumerated Types, [19](#)
- atmi_status_t
 - Enumerated Types, [19](#)
- atmi_task_group_s, [29](#)
 - id, [30](#)
 - maxsize, [30](#)
 - place, [30](#)
- atmi_task_group_sync
 - ATMI Task, [12](#)
- atmi_task_launch
 - ATMI Task, [12](#)
- atmi_task_s, [30](#)
 - continuation, [30](#)
 - handle, [30](#)
 - profile, [30](#)
 - state, [30](#)
- atmi_task_wait
 - ATMI Task, [12](#)
- atmi_tprofile_s, [31](#)
 - dispatch_time, [31](#)
 - end_time, [31](#)
 - ready_time, [31](#)
 - start_time, [31](#)
- BRIG
 - Enumerated Types, [19](#)
- Common ATMI Structures, [20](#)
- continuation
 - atmi_task_s, [30](#)
- continuation_task
 - atmi_lparm_s, [26](#)
- cu_mask
 - atmi_place_s, [29](#)
- dev_id
 - atmi_mem_place_s, [28](#)
- dev_type
 - atmi_mem_place_s, [28](#)
- device_count_by_type
 - atmi_machine_s, [28](#)
- device_id
 - atmi_place_s, [29](#)
- devices_by_type
 - atmi_machine_s, [28](#)
- dispatch_time
 - atmi_tprofile_s, [31](#)
- end_time
 - atmi_tprofile_s, [31](#)
- Enumerated Types, [18](#)
 - AMDGCN, [19](#)
 - ATMI_COMPLETED, [19](#)
 - ATMI_DEVTYPE_ALL, [19](#)
 - ATMI_DEVTYPE_CPU, [19](#)
 - ATMI_DEVTYPE_DSP, [19](#)
 - ATMI_DEVTYPE_GPU, [19](#)
 - ATMI_DEVTYPE_dGPU, [19](#)

- ATMI_DEVTYPE_iGPU, 19
- ATMI_DISPATCHED, 19
- ATMI_FAILED, 19
- ATMI_IN, 18
- ATMI_IN_OUT, 18
- ATMI_INITIALIZED, 19
- ATMI_MEMTYPE_ANY, 19
- ATMI_MEMTYPE_COARSE_GRAINED, 19
- ATMI_MEMTYPE_FINE_GRAINED, 19
- ATMI_OUT, 18
- ATMI_READY, 19
- ATMI_SCHED_NONE, 19
- ATMI_SCHED_RR, 19
- ATMI_STATUS_ERROR, 19
- ATMI_STATUS_SUCCESS, 19
- ATMI_STATUS_UNKNOWN, 19
- atmi_arg_type_s, 18
- atmi_devtype_s, 18
- atmi_memtype_s, 19
- atmi_platform_type_t, 19
- atmi_scheduler_s, 19
- atmi_state_s, 19
- atmi_status_t, 19
- BRIG, 19
- get_atmi_task_handle
 - ATMI CPU Device Runtime, 17
- gridDim
 - atmi_lparm_s, 26
- group
 - atmi_cparm_s, 23
 - atmi_lparm_s, 26
- groupDim
 - atmi_lparm_s, 26
- groupable
 - atmi_cparm_s, 23
 - atmi_lparm_s, 26
- handle
 - atmi_kernel_s, 25
 - atmi_task_s, 30
- id
 - atmi_task_group_s, 30
- Kernel Handles, 5
 - atmi_kernel_add_cpu_impl, 6
 - atmi_kernel_add_gpu_impl, 6
 - atmi_kernel_create_empty, 6
 - atmi_kernel_release, 7
 - atmi_kernel_t, 5
- kernel_id
 - atmi_lparm_s, 27
- maxsize
 - atmi_task_group_s, 30
- mem_id
 - atmi_mem_place_s, 28
- memory_pool_count
 - atmi_device_s, 25
- needs_any
 - atmi_lparm_s, 27
- node_id
 - atmi_mem_place_s, 28
 - atmi_place_s, 29
- num_needs_any
 - atmi_lparm_s, 27
- num_required
 - atmi_cparm_s, 23
 - atmi_lparm_s, 27
- place
 - atmi_data_s, 24
 - atmi_lparm_s, 27
 - atmi_task_group_s, 30
- profilable
 - atmi_cparm_s, 24
 - atmi_lparm_s, 27
- profile
 - atmi_task_s, 30
- ptr
 - atmi_data_s, 24
- ready_time
 - atmi_tprofile_s, 31
- release_scope
 - atmi_lparm_s, 27
- requires
 - atmi_cparm_s, 24
 - atmi_lparm_s, 27
- size
 - atmi_data_s, 24
- start_time
 - atmi_tprofile_s, 31
- state
 - atmi_task_s, 30
- synchronous
 - atmi_cparm_s, 24
 - atmi_lparm_s, 27
- task_info
 - atmi_cparm_s, 24
 - atmi_lparm_s, 27
- type
 - atmi_device_s, 25
 - atmi_place_s, 29