Multicore Task Management API (MTAPI)

The Multicore Task Management API (MTAPI) provides application-level management of tasks in multicore embedded systems.

Get the specification at www.multicore-association.org.

- [n.n] refers to the sections in the MTAPI API 1.0 specification.
- MTAPI_IN and MTAPI_OUT distinguish between input and output parameters. Parameters that are both read and written are declared as MTAPI_INOUT.
- In function prototypes, qualifiers are shown in blue, **function names are bold**, types are shown in green, and *parameters are italic*.

THE Multicore® ASSOCIATION

Data Types [2.9]

Data Types

mtapi_status_t	Status of MTAPI API call
mtapi_timeout_t	Specifies a timeout value
mtapi_size_t	Size of data
mtapi_domain_t	MTAPI domain
mtapi_node_t	MTAPI node
mtapi_info_t	Provides information about the MTAPI runtime
mtapi_*_hndl_t * = job, action, task, queue, or group	References job, action, task, queue, or group
mtapi_*_id_t * = job, task, queue, or group	Identifier used to get handle
mtapi_int_t, mtapi_uint_t	Define platform-specific integer
mtapi_uint*n*_t *n* = 64, 32, 16, or 8	64-, 32-, 16-, and 8-bit scalars
mtapi_uint*n*_t *n* = 64, 32, 16, or 8	Unsigned 64-, 32-, 16-, and 8-bit scalars
mtapi_action_function_t	Function pointer to an action function
mtapi_task_context_t	Get information about the task in an action
mtapi_affinity_t	Represents a task-to-core affinity mask
mtapi_task_state_t	Task state
mtapi_notification_t	Used for MTAPI extensions

Other Data Types

Additional types and enums are defined in header files mca.h and mtapi.h [6].

Concepts

Domains [2.2]

Domains are comprised of one or more MTAPI nodes in a multicore topology, and used for routing purposes. Comparable to a subnet in a network or a namespace for unique names and identifiers.

Nodes [2.3]

A node is an independent unit of execution, such as a process, thread, thread pool, processor, hardware accelerator, or instance of an operating system. A node ID is specified in the call to **mtapi_initialize()**.

Tasks and Actions [2.4]

A task represents the computation associated with the data to be processed. A task is associated with at least one action object representing the calculation. The association is indirect: one or more actions implement a job, one job is associated with a task. A *task* is a particular invocation of a job. A *job* refers to one or more actions. An *action* is a hardware or software implementation of a job.

Starting a task consists of three steps:

1. Create the action object with a job ID.
2. Obtain an job reference.
3. Start the task using the job reference.

Queues [2.5]

Queues are used for guaranteeing scheduling policies, such as the sequential order of execution of tasks. Set up and use a queue with the following steps:

1. Create the action object.
2. Obtain a job reference.
3. Create a queue object and attach the job to the queue.
4. Obtain a queue handle if the queue was created on a different node or if it is hardware-implemented.
5. Use the queue: enqueue the work using the queue.

Error and Status Codes [2.7] [6]

The following codes begin with MTAPI_ERR (except for MTAPI_GROUP_COMPLETED and MTAPI_TIMEOUT).

_ACTION_CANCELLED	Execution was cancelled.
_ACTION_DELETED	Actions associated with the task have been deleted.
_ACTION_DISABLED	Actions associated with the task have been disabled before the execution of the task was started.
_ACTION_EXISTS	This action is already created.
_ACTION_FAILED	Error set by action.
_ACTION_INVALID	Argument not a valid action handle.
_ACTION_LIMIT	Exceeded max. number of actions allowed.
_ACTION_NOAFFINITY	Action was created with an invalid affinity attribute.
_AFFINITY_MASK	Invalid mask parameter.
_ARG_SIZE	Size of arguments expected differs from arguments size of caller.
_ATTR_NUM	Unknown attribute number.
_ATTR_READONLY	Attribute cannot be modified.
_ATTR_SIZE	Incorrect attribute size.
CONTEXT OUTOFCONTEXT	Not called in the context of a task execution.
_CORE_NUM	Unknown core number.
_DOMAIN_INVALID	The domain_id parameter is not valid.
_DOMAIN_NOTSHARED	This resource cannot be shared by this domain.
_GROUP_INVALID	Argument not a valid group or task handle.
_GROUP_LIMIT	Exceeded maximum number of groups allowed.
_JOB_INVALID	The associated job is not valid.
MTAPI_GROUP_COMPLETED	No more tasks to wait for in the group.
MTAPI_TIMEOUT	Timeout was reached.
_NODE_FINALFAILED	MTAPI environment could not be finalized.
_NODE_INITFAILED	MTAPI environment could not be initialized.
_NODE_INITIALIZED	MTAPI environment already initialized.
_NODE_INVALID	The node_id parameter is not valid.
_NODE_NOTINIT	The calling node is not initialized.
_PARAMETER	Invalid attributes parameter.
_QUEUE_DELETED	Queue no longer exists.
_QUEUE_DISABLED	Queue has been disabled.
_QUEUE_EXISTS	This queue is already created.
_QUEUE_INVALID	Argument is not a valid queue handle.
_QUEUE_LIMIT	Exceeded maximum number of queues allowed.
_RESULT_SIZE	Size of result buffer expected differs from result buffer size of the caller.
_TASK_CANCELLED	Task has been cancelled.
_TASK_INVALID	Argument is not a valid task handle.
_TASK_LIMIT	Exceeded maximum number of tasks allowed.
_WAIT_PENDING	Previously called wait function is still pending.

General Functions

Initialize Node Attributes Object [3.2.1]

void **mtapi_nodeattr_init(**
  MTAPI_OUT mtapi_node_attributes_t * *attributes*,
  MTAPI_OUT mtapi_status_t * *status***);**

Set Node Attribute [3.2.2]

void **mtapi_nodeattr_set(**
  MTAPI_INOUT mtapi_node_attributes_t * *attributes*,
  MTAPI_IN mtapi_uint_t *attribute_num*,
  MTAPI_IN void * *attribute*,
  MTAPI_IN mtapi_size_t *attribute_size*,
  MTAPI_OUT mtapi_status_t * *status***);**

Initialize MTAPI [3.2.3]

void **mtapi_initialize(**
  MTAPI_IN mtapi_domain_t *domain_id*,
  MTAPI_IN mtapi_node_t *node_id*,
  MTAPI_IN mtapi_node_attributes_t * *attributes*,
  MTAPI_OUT mtapi_info_t * *mtapi_info*,
  MTAPI_OUT mtapi_status_t * *status***);**

Finalize MTAPI Environment [3.2.5]

void **mtapi_finalize(**
  MTAPI_OUT mtapi_status_t * *status***);**

Get Node Attribute Values [3.2.4]

void **mtapi_node_get_attribute(**
  MTAPI_IN mtapi_node_t *node*,
  MTAPI_IN mtapi_uint_t *attribute_num*,
  MTAPI_OUT void * *attribute*,
  MTAPI_IN mtapi_size_t *attribute_size*,
  MTAPI_OUT mtapi_status_t * *status***);**

Get Domain ID [3.2.6]

mtapi_domain_t **mtapi_domain_id_get(**
  MTAPI_OUT mtapi_status_t * *status***);**

Get Node ID [3.2.7]

mtapi_node_t **mtapi_node_id_get(**
  MTAPI_OUT mtapi_status_t * *status***);**

Also see the Multicore Communications API (MCAPI) for communication and synchronization between processing cores in embedded systems, and the Multicore Resource Management API (MRAPI) for managing shared resources in a closely distributed embedded system.

Learn more at www.multicore-association.org.

Action Functions

Initialize & Set Action Attributes [3.3.1, 3.3.2]

void **mtapi_actionattr_init(**
  MTAPI_OUT mtapi_action_attributes_t * *attributes*,
  MTAPI_OUT mtapi_status_t * *status***);**

void **mtapi_actionattr_set(**
  MTAPI_INOUT mtapi_action_attributes_t * *attributes*,
  MTAPI_IN mtapi_uint_t *attribute_num*,
  MTAPI_IN void * *attribute*,
  MTAPI_IN mtapi_size_t *attribute_size*,
  MTAPI_OUT mtapi_status_t * *status***);**

Create Action [3.3.3]

mtapi_action_hndl_t **mtapi_action_create(**
  MTAPI_IN mtapi_job_id_t *job_id*,
  MTAPI_IN mtapi_action_function_t *function*,
  MTAPI_IN void * *node_local_data*,
  MTAPI_IN mtapi_size_t *node_local_data_size*,
  MTAPI_IN mtapi_action_attributes_t * *attributes*,
  MTAPI_OUT mtapi_status_t * *status***);**

Set & Get Attribute Value [3.3.4, 3.3.5]

void **mtapi_action_set_attribute(**
  MTAPI_IN mtapi_action_hndl_t *action*,
  MTAPI_IN mtapi_uint_t *attribute_num*,
  MTAPI_IN void * *attribute*,
  MTAPI_IN mtapi_size_t *attribute_size*,
  MTAPI_OUT mtapi_status_t * *status***);**

void **mtapi_action_get_attribute(**
  MTAPI_IN mtapi_action_hndl_t *action*,
  MTAPI_IN mtapi_uint_t *attribute_num*,
  MTAPI_OUT void * *attribute*,
  MTAPI_IN mtapi_size_t *attribute_size*,
  MTAPI_OUT mtapi_status_t * *status***);**

Continued on next page >

Action Functions (Continued)

Delete Action [3.3.6]

```
void mtapi_action_delete(
    MTAPI_IN mtapi_action_hndl_t action,
    MTAPI_IN mtapi_timeout_t timeout,
    MTAPI_OUT mtapi_status_t * status);
```

Disable & Enable Action [3.3.7, 3.3.8]

```
void mtapi_action_disable(
    MTAPI_IN mtapi_action_hndl_t action,
    MTAPI_IN mtapi_timeout_t timeout,
    MTAPI_OUT mtapi_status_t * status);
void mtapi_action_enable(
    MTAPI_IN mtapi_action_hndl_t action,
    MTAPI_OUT mtapi_status_t * status);
```

Task Functions

Initialize Task Attributes Object [3.8.1]

```
void mtapi_taskattr_init(
    MTAPI_OUT mtapi_task_attributes_t * attributes,
    MTAPI_OUT mtapi_status_t * status);
```

Set & Get Task Attribute Value [3.8.2, 3.8.5]

```
void mtapi_taskattr_set(
    MTAPI_INOUT mtapi_task_attributes_t * attributes,
    MTAPI_IN mtapi_uint_t attribute_num,
    MTAPI_IN void * attribute,
    MTAPI_IN mtapi_size_t attribute_size,
    MTAPI_OUT mtapi_status_t * status);
void mtapi_task_get_attribute(
    MTAPI_IN mtapi_task_hndl_t task,
    MTAPI_IN mtapi_uint_t attribute_num,
    MTAPI_OUT void * attribute,
    MTAPI_IN mtapi_size_t attribute_size,
    MTAPI_OUT mtapi_status_t * status);
```

Schedule a Task [3.8.3, 3.8.4]

```
mtapi_task_hndl_t mtapi_task_start(
    MTAPI_IN mtapi_task_id_t task_id,
    MTAPI_IN mtapi_job_hndl_t job,
    MTAPI_IN void * arguments,
    MTAPI_IN mtapi_size_t arguments_size,
    MTAPI_OUT void * result_buffer,
    MTAPI_IN mtapi_size_t result_size,
    MTAPI_IN mtapi_task_attributes_t * attributes,
    MTAPI_IN mtapi_group_hndl_t group,
    MTAPI_OUT mtapi_status_t * status);
mtapi_task_hndl_t mtapi_task_enqueue(
    MTAPI_IN mtapi_task_id_t task_id,
    MTAPI_IN mtapi_queue_hndl_t queue,
    MTAPI_IN void * arguments,
    MTAPI_IN mtapi_size_t arguments_size,
    MTAPI_OUT void * result_buffer,
    MTAPI_IN mtapi_size_t result_size,
    MTAPI_IN mtapi_task_attributes_t * attributes,
    MTAPI_IN mtapi_group_hndl_t group,
    MTAPI_OUT mtapi_status_t * status);
```

Cancel a Task [3.8.6]

```
void mtapi_task_cancel(
    MTAPI_IN mtapi_task_hndl_t task,
    MTAPI_OUT mtapi_status_t * status);
```

Wait for Task Completion [3.8.7]

```
void mtapi_task_wait(
    MTAPI_IN mtapi_task_hndl_t task,
    MTAPI_IN mtapi_timeout_t timeout,
    MTAPI_OUT mtapi_status_t * status);
```

Job Function

Get Job Handle [3.7.1]

```
mtapi_job_hndl_t mtapi_job_get(
    MTAPI_IN mtapi_job_id_t job_id,
    MTAPI_IN mtapi_domain_t domain_id,
    MTAPI_OUT mtapi_status_t * status);
```

Action & Action Context Functions

Set Context Status [3.4.1]

```
void mtapi_context_status_set(
    MTAPI_INOUT mtapi_task_context_t * task_context,
    MTAPI_IN mtapi_status_t error_code,
    MTAPI_OUT mtapi_status_t * status);
```

Notify Runtime System [3.4.2]

```
void mtapi_context_runtime_notify(
    MTAPI_IN mtapi_task_context_t * task_context,
    MTAPI_IN mtapi_notification_t notification,
    MTAPI_IN void * data,
    MTAPI_IN mtapi_size_t data_size,
    MTAPI_OUT mtapi_status_t * status);
```

Get Task State [3.4.3]

```
mtapi_task_state_t mtapi_context_taskstate_get(
    MTAPI_IN mtapi_task_context_t * task_context,
    MTAPI_OUT mtapi_status_t * status);
```

Query Task Instance Number [3.4.4]

```
mtapi_uint_t mtapi_context_instnum_get(
    MTAPI_IN mtapi_task_context_t * task_context,
    MTAPI_OUT mtapi_status_t * status);
```

Query Number of Task Instances [3.4.5]

```
mtapi_uint_t mtapi_context_numinst_get(
    MTAPI_IN mtapi_task_context_t * task_context,
    MTAPI_OUT mtapi_status_t * status);
```

Query Current Core Number [3.4.6]

```
mtapi_uint_t mtapi_context_corenum_get(
    MTAPI_IN mtapi_task_context_t * task_context,
    MTAPI_OUT mtapi_status_t * status);
```

Core Affinities

Initialize Affinity Mask Object [3.5.1]

```
void mtapi_affinity_init(
    MTAPI_OUT mtapi_affinity_t * mask,
    MTAPI_IN mtapi_boolean_t affinity,
    MTAPI_OUT mtapi_status_t * status);
```

Change Affinity Mask Object Default Values [3.5.2]

```
void mtapi_affinity_set(
    MTAPI_INOUT mtapi_affinity_t * mask,
    MTAPI_IN mtapi_uint_t core_num,
    MTAPI_IN mtapi_boolean_t affinity,
    MTAPI_OUT mtapi_status_t * status);
```

Get Affinity [3.5.3]

```
mtapi_boolean_t mtapi_affinity_get(
    MTAPI_IN mtapi_affinity_t * mask,
    MTAPI_IN mtapi_uint_t core_num,
    MTAPI_OUT mtapi_status_t * status);
```

Queue Functions

Initialize & Set Queue Attributes [3.6.1, 3.6.2]

```
void mtapi_queueattr_init(
    MTAPI_OUT mtapi_queue_attributes_t * attributes,
    MTAPI_OUT mtapi_status_t * status);
void mtapi_queueattr_set(
    MTAPI_INOUT mtapi_queue_attributes_t * attributes,
    MTAPI_IN mtapi_uint_t attribute_num,
    MTAPI_IN void * attribute,
    MTAPI_IN mtapi_size_t attribute_size,
    MTAPI_OUT mtapi_status_t * status);
```

Create Queue [3.6.3]

```
mtapi_queue_hndl_t mtapi_queue_create(
    MTAPI_IN mtapi_queue_id_t queue_id,
    MTAPI_IN mtapi_job_hndl_t job,
    MTAPI_IN mtapi_queue_attributes_t * attributes,
    MTAPI_OUT mtapi_status_t * status);
```

Set & Get Attribute Value [3.6.4, 3.6.5]

```
void mtapi_queue_set_attribute(
    MTAPI_IN mtapi_queue_hndl_t queue,
    MTAPI_IN mtapi_uint_t attribute_num,
    MTAPI_IN void * attribute,
    MTAPI_IN mtapi_size_t attribute_size,
    MTAPI_OUT mtapi_status_t * status);
void mtapi_queue_get_attribute(
    MTAPI_IN mtapi_queue_hndl_t queue,
    MTAPI_IN mtapi_uint_t attribute_num,
    MTAPI_OUT void * attribute,
    MTAPI_IN mtapi_size_t attribute_size,
    MTAPI_OUT mtapi_status_t * status);
```

Convert Queue From Domain to Local [3.6.6]

```
mtapi_queue_hndl_t mtapi_queue_get(
    MTAPI_IN mtapi_queue_id_t queue_id,
    MTAPI_IN mtapi_domain_t domain_id,
    MTAPI_OUT mtapi_status_t * status);
```

Delete Queue [3.6.7]

```
void mtapi_queue_delete(
    MTAPI_IN mtapi_queue_hndl_t queue,
    MTAPI_IN mtapi_timeout_t timeout,
    MTAPI_OUT mtapi_status_t * status);
```

Disable & Enable Queue [3.6.8, 3.6.9]

```
void mtapi_queue_disable(
    MTAPI_IN mtapi_queue_hndl_t queue,
    MTAPI_IN mtapi_timeout_t timeout,
    MTAPI_OUT mtapi_status_t * status);
void mtapi_queue_enable(
    MTAPI_IN mtapi_queue_hndl_t queue,
    MTAPI_OUT mtapi_status_t * status);
```

Task Group Functions

Initialize & Set Task Group Attributes [3.9.1, 3.9.2]

```
void mtapi_groupattr_init(
    MTAPI_OUT mtapi_group_attributes_t * attributes,
    MTAPI_OUT mtapi_status_t * status);
void mtapi_groupattr_set(
    MTAPI_INOUT mtapi_group_attributes_t * attributes,
    MTAPI_IN mtapi_uint_t attribute_num,
    MTAPI_IN void * attribute,
    MTAPI_IN mtapi_size_t attribute_size,
    MTAPI_OUT mtapi_status_t * status);
```

Create Task Group [3.9.3]

```
mtapi_group_hndl_t mtapi_group_create(
    MTAPI_IN mtapi_group_id_t group_id,
    MTAPI_IN mtapi_group_attributes_t * attributes,
    MTAPI_OUT mtapi_status_t * status);
```

Set & Get Attribute Value [3.9.4, 3.9.5]

```
void mtapi_group_set_attribute(
    MTAPI_IN mtapi_ group_hndl_t group,
    MTAPI_IN mtapi_uint_t attribute_num,
    MTAPI_OUT void * attribute,
    MTAPI_IN mtapi_size_t attribute_size,
    MTAPI_OUT mtapi_status_t * status);
void mtapi_group_get_attribute(
    MTAPI_IN mtapi_group_hndl_t group,
    MTAPI_IN mtapi_uint_t attribute_num,
    MTAPI_OUT void * attribute,
    MTAPI_IN mtapi_size_t attribute_size,
    MTAPI_OUT mtapi_status_t * status);
```

Wait for Completion of Tasks in Group [3.9.6, 3.9.7]

```
void mtapi_group_wait_all(
    MTAPI_IN mtapi_group_hndl_t group,
    MTAPI_IN mtapi_timeout_t timeout,
    MTAPI_OUT mtapi_status_t * status);
void mtapi_group_wait_any(
    MTAPI_IN mtapi_group_hndl_t group,
    MTAPI_OUT void ** result,
    MTAPI_IN mtapi_timeout_t timeout,
    MTAPI_OUT mtapi_status_t * status);
```

Delete Group [3.9.8]

```
void mtapi_group_delete(
    MTAPI_IN mtapi_group_hndl_t group,
    MTAPI_OUT mtapi_status_t * status);
```

The Multicore Association (MCA) is an open membership organization that includes leading companies implementing products that embrace multicore technology.