7.4. SEM_CLAIM

Claim a semaphore (P operation).

Synopsis

```
sem_claim( sid, options, time_out )
```

Input Parameters

sid	: sem_id	kernel defined semaphore identifier
options	: bit_field	semaphore wait options
time_out	: integer	ticks to wait before timing out

Output Parameters

<none>

Literal Values

options	+ NOWAIT	do not wait - return immediately if semaphore not available
time_out	= FOREVER	wait forever - do not time out

Completion Status

OK	sem_claim successful
ILLEGAL_USE	sem_claim not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	semaphore does not exist
OBJECT_DELETED	originally existing semaphore has been deleted before operation
TIME_OUT	sem_claim timed out
SEMAPHORE_DELETED	semaphore deleted while blocked in sem_claim
SEMAPHORE_NOT_AVAILABLE	semaphore unavailable with NOWAIT option
SEMAPHORE_UNDERFLOW	semaphore counter underflowed
NODE_NOT_REACHABLE	node on which semaphore resides is not reachable

Description

This operation performs a claim from the given semaphore. It first checks if the NOWAIT option has been specified and the counter is zero or less, in which case the SEMAPHORE_NOT_AVAILABLE completion status is returned. Otherwise, the counter is decreased. If the counter is now zero or more, then the claim is successful, otherwise the calling task is put on the semaphore queue. If the counter underflowed the SEMAPHORE_UNDERFLOW completion status is returned. If the semaphore is deleted while a task is waiting on its queue, then the task is unblocked and the sem_claim operation returns the SEMAPHORE_DELETED completion status to the task. Otherwise the task is blocked either until the timeout expires, in which case the TIME_OUT completion status is returned, or until the task reaches the head of the queue and another task performs a sem_release operation on this semaphore, leading to the return of the successful completion status.

7.5. SEM_RELEASE

Release a semaphore (V operation).

Synopsis

```
sem_release( sid )
```

Input Parameters

sid	: sem_id	kernel defined semaphore identifier

Output Parameters

<none>

Completion Status

OK	sem_release successful
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	semaphore does not exist
OBJECT_DELETED	originally existing semaphore has been deleted before operation
SEMAPHORE_OVERFLOW	semaphore counter overflowed
NODE_NOT_REACHABLE	node on which semaphore resides is not reachable

Description

This operation increments the semaphore counter by one. If the resulting semaphore count is less than or equal to zero then the first task in the semaphore queue is unblocked, and returned the successful completion status. If the counter overflowed the SEMAPHORE_OVERFLOW completion status is returned.

7.6. SEM_INFO

Obtain information on a semaphore.

Synopsis

```
sem_info( sid, options, count, tasks_waiting )
```

Input Parameters

sid	: sem-id	kernel defined semaphore identifier

Output Parameters

options	: bit_field	semaphore create options
count	: integer	semaphore count at time of call
tasks_waiting	: integer	number of tasks waiting in the semaphore queue

Completion Status

OK	sem_info successful
ILLEGAL_USE	sem_info not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	semaphore does not exist
OBJECT_DELETED	originally existing semaphore has been deleted before operation
NODE_NOT_REACHABLE	node on which semaphore resides is not reachable

Description

This operation provides information on the specified semaphore. It returns its create options, the value of it's counter, and the number of tasks waiting on the semaphore queue. The latter two values should be used with care as they are just a snap-shot of the semaphore's state at the time of executing the operation.

8.3. QUEUE_IDENT

Obtain the identifier of a queue on a given node with a given name.

Synopsis

```
queue_ident( name, nid, qid )
```

Input Parameters

name	: string	user defined queue name
nid	: node_id	node identifier

Output Parameters

qid	: queue_id	kernel defined queue identifier

Literal Values

nid	= LOCAL_NODE	the node containing the calling task
	= OTHER_NODES	all nodes in the system except the local node
	= ALL_NODES	all nodes in the system

Completion Status

OK	queue_ident successful
ILLEGAL_USE	queue_ident not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	node does not exist
NAME_NOT_FOUND	queue name does not exist on node
NODE_NOT_REACHABLE	node is not reachable

Description

This operation searches the kernel data structure in the node(s) specified for a queue with the given name, and returns its identifier if found. If OTHER_NODES or ALL_NODES is specified, the node search order is implementation dependent. If there is more than one queue with the same name in the node(s) specified, then the qid of the first one found is returned.

8.4. QUEUE_SEND

Send a message to a given queue.

Synopsis

```
queue_send( qid, msg_buff, msg_length )
```

Input Parameters

qid	: queue_id	kernel defined queue identifier
msg_buff	: address	message starting address
msg_length	: integer	length of message in bytes

Output Parameters

<none>

Completion Status

OK	queue_send successful
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	queue does not exist
OBJECT_DELETED	originally existing queue has been deleted before operation
INVALID_LENGTH	message length greater than queue's buffer length
QUEUE_FULL	no more buffers available
NODE_NOT_REACHABLE	node on which queue resides is not reachable

Description

This operations sends a message to a queue.

If the queue's wait queue contains a number of tasks waiting on messages, then the message is delivered to the task at the head of the wait queue. This task is then removed from the wait queue, unblocked and will be returned a successful completion status along with the message. Otherwise the message is appended at the end of the queue.

If the maximum queue length has been reached, then the QUEUE_FULL completion status is returned.