5.6. PARTITION_INFO

Obtain information on a partition.

Synopsis

```
partition_info( pid, blocks, free_blocks, block_size  )
```

Input Parameters

pid	: partition-id	kernel defined region id

Output Parameters

blocks	: integer	number of blocks in the partition
free_blocks	: integer	number of free blocks in the partition
block_size	: integer	partition block size in bytes

Completion Status

OK	partition_info operation successful
ILLEGAL_USE	operation not callable from ISR
INVALID_PARAMETER	a parameter refers to an illegal address
INVALID_ID	partition does not exist
OBJECT_DELETED	partition specified has been deleted
NODE_NOT_REACHABLE	node on which the partition resides is not reachable

Description

This operation provides information on the specified partition. It returns its overall number of blocks, the number of free blocks in the partition, and the block size. The number of free blocks in the partition should be used with care as it is just a snap-shot of the partitions's usage at the time of executing the operation.

6. SEMAPHORES

The semaphores defined in ORKID are standard Dijkstra counting semaphores. Semaphores provide for the fundamental need of synchronization in multi-tasking systems, i.e. mutual exclusion, resource management and sequencing.

Semaphore Behavior

The following should not be understood as a recipe for implementations.

The behavior of counting semaphores can be described as follows:

During a sem_p operation, the semaphore count is decremented by one. If the resulting semaphore count is greater than or equal to zero, than the calling task continues to execute. If the count is less than zero, the task blocks from CPU usage and is put on a waiting list for the semaphore.

During a sem_v operation, the semaphore count is incremented by one. If the resulting semaphore count is less than or equal to zero then the first task in the waiting list for this semaphore is unblocked and is made eligible for CPU usage.

Semaphore Usage

Mutual exclusion is achieved by creating a counting semaphore with an initial count of one. A resource is guarded with this semaphore by requiring all operations on the resource to be proceeded by a sem_p operation. Thus, if one task has claimed a resource, all other tasks requiring the resource will be blocked until the task releases the resource with a sem_v operation.

In situations where multiple instantiations of a resource exist, the semaphore may be created with an initial count equal to a number of instantiations. A resource is claimed from the pool with the sem_p operation. When all available copies of the resource have been claimed, a task requiring the resource will be blocked until one of the claimed resources is returned to the pool by a sem_v operation.

Sequencing is achieved by creating a semaphore with an initial count of zero. A task may pend the arrival of another task by performing a sem_p operation when it reaches a synchronization point. The other tasks performs a sem_v operation when it reaches its synchronization point, unblocking the pended task.

Semaphore Options

ORKID defines the following option symbols, which may be combined.

* GLOBAL — Semaphores created with the GLOBAL option set are visible and accessible from any node in the system.

* FIFO — Semaphores created with the FIFO option set enqueue blocked tasks in order of arrival of the sem_p

operations. Without this option, the tasks are enqueued in order of task priority.

6.1. SEM_CREATE

Create a semaphore.

Synopsis

```
sem_create( name, init_count, options, sid )
```

Input Parameters

name	: string	user defined semaphore name
init_count	: integer	initial semaphore count
options	: bit_field	semaphore create options

Output Parameters

sid	: sema_id	kernel defined semaphore identifier

Literal Values

options	+ GLOBAL	the new semaphore will be visible throughout the system
	+ FIFO	tasks will be queued in first in first out order

Completion Status

OK	sem_create operation successful
ILLEGAL_USE	operation not callable from XSR or ISR
INVALID_PARAMETER	a parameter refers to an illegal address
INVALID_COUNT	init count is negative
INVALID_OPTIONS	invalid options value
TOO_MANY_SEMAPHORES	too many semaphores on node

Description

This operation creates a new semaphore in the kernel data structure, and returns its identifier. The semaphore is created with its counter at the value given by the count parameter. The task queue, initially empty, will be ordered by task priority, unless the FIFO option is set, in which case it will be first in first out.

6.2. SEM_DELETE

Delete a semaphore.

Synopsis

```
sem_delete( sid )
```

Input Parameters

sid	: sema_id	kernel defined semaphore identifier

Output Parameters

<none>

Completion Status

OK	sem_delete operation successful
ILLEGAL_USE	operation not callable from ISR
INVALID_PARAMETER	a parameter refers to an illegal address
INVALID_ID	semaphore does not exist
OBJECT_DELETED	semaphore specified has been deleted
NODE_NOT_REACHABLE	node on which semaphore resides is not reachable

Description

The sem_delete operation deletes a semaphore from the kernel data structure. The semaphore is deleted immediately, even though there are tasks waiting in its queue. These latter are all unblocked and are returned the SEMAPHORE_DELETED completion status.