12.1. INT_ENTER

Announce interrupt handler entry.

Synopsis

```
int_enter( )
```

Input Parameters

<none>

Output Parameters

<none>

Completion Status

OK	int_enter operation successful

Description

This operation call announces the start of an interrupt handling routine to the kernel. Its functionality is implementation dependent. The operation takes no parameters and always returns a successful completion status. It is up to a user task to set up vectors to the handler which makes this call.

12.2. INT_EXIT

Exit from an interrupt handler.

Synopsis

```
int_exit( )
```

Input Parameters

<none>

Output Parameters

<none>

Completion Status

<not applicable>

Description

This operation announces the end of an interrupt handling routine to the kernel. Its exact functionality is implementation dependent, but will involve returning to interrupted code or scheduling another task. The operation takes no parameters and does not return to the calling code.

The behavior of int_return when not called from an Interrupt Service Routine is undefined.

A. RETURN CODES

CLOCK_NOT_SET	clock has not been initialized
ILLEGAL_USE	operation not callable from XSR or ISR
INVALID OPTIONS	invalid options value
INVALID_ADDRESS	a specific parameter refers to an illegal address
INVALID_ARGUMENTS	invalid number or type or size of arguments
INVALID_BLOCK	no block allocated from partition at blk_addr
INVALID_BLOCK_SIZE	block_size not supported
INVALID_CLOCK	invalid clock value
INVALID_COUNT	init count is negative
INVALID_GRANULARITY	granularity not supported
INVALID_ID	object does not exist
INVALID_LENGTH	buffer length not supported
INVALID_LOCATION	note-pad number does not exist
INVALID_MODE	invalid mode or mask value
INVALID_NODE	node does not exist
INVALID_OPTIONS	invalid options value
INVALID_PARAMETER	a parameter refers to an illegal address
INVALID_PRIORITY	invalid priority value
INVALID_SEGMENT	no segment allocated from this region at seg_addr
NAME_NOT_FOUND	name does not exist on node
NODE_NOT_REACHABLE	node on which object resides is not reachable
NO_EVENTS	event(s) not set and NOWAIT option given
NO_MORE_MEMORY	not enough memory to satisfy request
OBJECT_DELETED	specified object has been deleted
OBJECT_PROTECTED	task has NOPREEMPT or NOTERMINATION parameter set
OK	operation successful
PARTITION_IN_USE	blocks from this partition are still allocated
PARTITION_OVERLAP	Area given overlaps an existing partition
QUEUE_DELETED	queue deleted while blocked in queue_receive operation
QUEUE_EMPTY	queue empty with NOWAIT option
QUEUE_FULL	no more buffers available
REGION_IN_USE	segments from this region are still allocated
REGION_OVERLAP	area given overlaps an existing region
SEMAPHORE_DELETED	semaphore deleted while blocked in sem_p operation
SEMAPHORE_NOT_AVAILABLE	semaphore unavailable with NOWAIT option
SEM_OVERFLOW	the counter of semaphore overflows
TASK_ALREADY_STARTED	task has been started already
TASK_ALREADY_SUSPENDED	task already suspended
TASK_NOT_STARTED	task has not yet been started
TASK_NOT_SUSPENDED	task not suspended
TIME_OUT	operation timed out
TOO_MANY_PARTITIONS	too many partitions on the node
TOO_MANY_QUEUES	too many queues on node
TOO_MANY_REGIONS	too many regions on the node
TOO_MANY_SEMAPHORES	too many semaphores on node
TOO_MANY_TASKS	too many tasks on the node
TOO_MANY_TIMERS	too many timers on node
XSR_NOT_SET	task has no exception handler routine

B. MINIMUM REQUIREMENTS FOR OPERATIONS FROM AN ISR.

ORKID requires that at least the following operations are supported from an Interrupt Service Routine. Only operations on local objects need to be supported. If the object resides on a remote node and remote operations are not supported, then the INVALID_ID completion status must be returned.

Task Operations

```
task_suspend        ( tid )
task_resume         ( tid )
task_read_notepad   ( tid, loc_number, loc_value )
task_write_notepad  ( tid, loc_number, loc_value )
```

Semaphore Operations

```
sem_v               ( sid )
```

Queue Operations

```
queue_send          ( qid, message, length )
queue_urgent        ( qid, message, length )
```

Event Operations

```
event_send          ( tid, event )
```

Exception Operations

```
exceptions_raise    ( tid, exceptions )
```

Clock Operations

```
clock_tick          ( ) clock_get          ( clock )
```

Interrupt Operations

```
int_enter           ( )
int_exit            ( )
```

C. MINIMUM REQUIREMENTS FOR OPERATIONS FROM AN XSR.

ORKID requires that at least the following operations are supported from an Exception Service Routine.

Task Operations

```
task_delete          ( tid )
task_start           ( tid, start_addr, arguments )
task_restart         ( tid, arguments )
task_suspend         ( tid )
task_resume          ( tid )
task_set_priority    ( tid, new_prio, old_prio )
task_set_mode        ( mode, mask, old_mode )
task_read_notepad    ( tid, loc_number, loc_value )
task_write_notepad   ( tid, loc_number, loc_value )
```

Region Operations

```
region_delete        ( rid, options )
region_get_seg       ( rid, seg_size, seg_addr )
region_ret_seg       ( rid, seg_addr )
region_info          ( rid, size, max_segment, granularity )
```

Partition Operations

```
partition_delete     ( pid, options )
partition_get_blk    ( pid, blk_addr )
partition_ret_blk    ( pid, blk_addr )
partition_info       ( pid, blocks, free_blocks, block_size )
```

Semaphore Operations

```
sem_delete           ( sid )
sem_p                ( sid, time_out )
sem_v                ( sid )
sem_info             ( sit, options, count, tasks_waiting )
```

Queue Operations

```
queue_delete         ( qid )
queue_send           ( qid, message, length )
queue_urgent         ( qid, message, length )
queue_broadcast      ( qid, message, length, count )
queue_receive        ( qid, message, time_out )
queue_flush          ( qid, count )
queue_info           ( qid, max_buf, length, options, messages_waiting,
                       tasks_waiting )
```

Event Operations

```
event_send           ( tid, event )
event_receive        ( events, options, time_out, events_caught )
```