9.1. EXCEPTION_CATCH

Specify a task's asynchronous exception handling routine.

Synopsis

```
exception_catch( new_XSR, mode, old_XSR, old_mode )
```

Input Parameters

new_XSR	: address	address of exception handling routine
mode	: bit_field	startup execution mode of XSR

Output Parameters

old_XSR	: address	address of previous XSR
old_mode	: bit-filed	mode associated with old XSR

Literal Values

new_XSR	= NULL_XSR	task henceforth will have no XSR
mode	+ NOXHR	XSR cannot be activated
	+ NOTERMINATION	task cannot be restarted or deleted
	+ NOPREEMPT	task cannot be preempted
	+ NOINTERRUPT	interrupt handling routine cannot be activated
old_XSR	= NULL_XSR	task previously had no XSR

Completion Status

OK	exceptions_catch operation successful
ILLEGAL_USE	operation not callable from ISR
INVALID_PARAMETER	a parameter refers to an illegal address
INVALID_ADDRESS	new_XSR refers to an illegal address
INVALID_MODE	invalid mode value

Description

This operation designates a new exception handling routine (XSR) for the current task. The task supplies the start address of the XSR, and the mode in which it will be started. If this operation returns a successful completion status, an exception sent to the task will henceforth cause the XSR at the given address to be activated.

The kernel returns the address of the previous XSR and the mode associated with that XSR.

Observation:

This can be used when a task wishes to use a different XSR temporarily. Once finished with the temporary XSR, the original one can be simply reinstated.

Note that if tasks are created without an XSR in a particular

implementation, the first call to exception_catch will return the symbolic value NULL_XSR in old_XSR. This same value can be passed as the new_XSR input parameter, which removes the current XSR from the task without designating a new one.

9.2. EXCEPTION_RAISE

Raise exceptions to a task.

Synopsis

```
exception_raise( tid, exceptions )
```

Input Parameters

tid	: task_id	kernel defined task id
exceptions	: bit_field	exceptions to be raised

Output Parameters

<none>

Completion Status

OK	exceptions_send operation successful
INVALID_PARAMETER	a parameter refers to an illegal address
INVALID_ID	task does not exist
OBJECT_DELETED	task specified has been deleted
XSR_NOT_SET	task has no exception handler routine
NODE_NOT_REACHABLE	node on which semaphore resides is not reachable

Description

This operation raises one or more exceptions to a task. If the task in question has an XSR, then unless it has the NOXHR modal parameter set, the XSR will be activated immediately and run not later than the task would normally be scheduled. If NOXHR is set, the XSR will be activated as soon as the task clears this parameter.

If the task has no current XSR, then this operation returns the XSR_NOT_SET completion status.

9.3. EXCEPTION_RETURN

Return from Asynchronous Exception Handling Routine.

Synopsis

```
exception_return( )
```

Input Parameters

<none>

Output Parameters

<none>

Completion Status

<not applicable>

Description

This operation transfers control from an XSR back to the code which it interrupted. It has no parameters and does not produce a completion status. This operation must be used to deactivate an XSR.

The behavior of exception_return when not called from an XSR is undefined.

10. CLOCK

Each ORKID kernel maintains a node clock. This is a single data object in the kernel data structure which contains the current date and time. The clock is updated at every tick, the frequency of which is node dependent. The range of dates the clock is allowed to take is implementation dependent.

In a multi-node system, the different node clocks will very likely be synchronized, although this is not necessarily done automatically by the kernel. Since nodes could be in different time zones in widely distributed systems, the node clock specifies the local time zone, so that all nodes can synchronize their clocks to the same absolute time.

The data structure containing the clock value passed in clock operations is language binding dependent. It identifies the date and time down to the nearest tick, along with the local time zone. The time zone value is defined as the number of hours ahead (positive value) or behind (negative value) Greenwich Mean Time (GMT).

When the system starts up, the clock may be uninitialised. If this is the case, attempts at reading it before it has been set result in an error completion status, rather than returning a random value.