activated. After executing the exception_return operation in this XSR the routine corresponding to the bit with the second highest bit-number will be activated etc. An XSR running without the NOXSR bit in its mode will be interrupted by an exception of higher priority, i.e. with a higher bit-number. Exceptions of equal and lower priority will be latched.

The exception_return operation will return either to the interrupted task, reinstating its original mode, or to the interrupted XSR with its original mode. This is also true in case of explicit change of an XSR's mode via task_set_mode.

10.1. EXCEPTION_CATCH

Specify a task's Exception Service Routine for a given exception bit.

Synopsis

```
exception_catch( bit_number, new_xsr, new_mode, old_xsr, old_mode )
```

Input Parameters

bit_number	: integer	exception bit-number
new_xsr	: address	address of XSR
new_mode	: bit_field	execution mode to be ored in

Output Parameters

old_xsr	: address	address of old XSR
old_mode	: bit_field	mode of old XSR

Literal Values

new_xsr	= NULL_XSR	task henceforth will have no XSR for the given exception bit
new_mode	+ NOXSR	XSRs cannot be activated
	+ NOTERMINATION	task cannot be restarted or deleted
	+ NOPREEMPT	task cannot be preempted
	+ NOINTERRUPT	task cannot be interrupted
	= ZERO	no mode set
old_mode		same as new_mode
old_xsr	= NULL_XSR	task previously had no XSR for the given exception bit

Completion Status

OK	exception_catch successful
ILLEGAL_USE	exception_catch not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_MODE	invalid mode value
INVALID_BIT	invalid exception bit-number

Description

This operation designates a new Exception Service Routine (XSR) for the exception given by bit_number for the calling task. The task supplies the start address of the XSR, and the mode which will be ored to the active mode of the interrupted task or XSR to produce the active mode of this XSR. If this operation returns a successful completion status, the exception given by bit_number will henceforth cause the XSR at the given address to be activated, if the running task does not have the NOXSR mode set.

The kernel returns the address of the previous XSR and the mode of that

XSR for the specified exception.
Note that if a task has no XSR defined for the given exception a 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 for this exception without designating a new one.

Observation:

This operation can be used for defining the corresponding XSR for the first time and when a task wishes to use a different XSR temporarily. Once finished with the temporary XSR, the original one can be simply reinstated using the old_xsr and old_mode values.

10.2. EXCEPTION_RAISE

Raise exception(s) to a task.

Synopsis

```
exception_raise( tid, exception )
```

Input Parameters

tid	: task_id	kernel defined task id
exception	: bit_field	exception(s) to be raised

Output Parameters

<none>

Completion Status

OK	exception_raise successful
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	task does not exist
OBJECT_DELETED	originally existing task has been deleted before operation
XSR_NOT_SET	no handler routine for given exception(s)
NODE_NOT_REACHABLE	node on which task resides is not reachable

Description

This operation raises one or more exceptions to a task. If the task in question has XSR(s) defined for the given exception(s), then unless it has the NOXSR mode value set, the highest priority XSR will be activated immediately and will run when the task would be normally scheduled. If NOXSR is set, this XSR will be activated as soon as the task clears this parameter.

If the task has no XSR(s) for the given exception(s), then this operation returns the XSR_NOT_SET completion status.

10.3. EXCEPTION_RETURN

Return from Exception Service 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.