14.1. INT_TO_EXT

Translate processor address to external port address.

Synopsis

```
int_to_ext( int_addr, port, ext_addr )
```

Input Parameters

int_addr	: address	processor address to be translated
port	: integer	port designation

Output Parameters

ext_addr	: address	correspondig address for designated port

Completion Status

OK	int_to_ext successful
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_PORT	port does not exist
NO_TRANSLATION	int_addr can not be accessed through port

Description

This operation translates a processor address of a multi-port memory location to the address accessing the same location via the given port. The port parameter encodes the bus and address space to be used, e.g. VMEbus with a certain address modifier. If the given port does not exist the INVALID_PORT completion status is returned. If the given location cannot be accessed via the port the NO_TRANSLATION completion status is returned.

Observation:

It is assumed that the various bus standard authorities will define literals for the encoding of ports for their respective bus architectures.

14.2. EXT_TO_INT

Translate external port address to processor address.

Synopsis

```
ext_to_int( ext_addr, port, int_addr )
```

Input Parameters

ext_addr	: address	port address to be translated
port	: integer	port designation

Output Parameters

int_addr	: address	correspondig processor address

Completion Status

OK	ext_to_int successful
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_PORT	port does not exist
NO_TRANSLATION	ext_addr can not be accessed by processor

Description

This operation translates an external port address of a multi-port memory to the processor address accessing the same location. The port parameter encodes the bus and address space to be used, e.g. VMEbus with a certain address modifier. If the given port does not exist the INVALID_PORT completion status is returned. If the given location can not be accessed by the processor the NO_TRANSLATION completion status is returned (see also 14.1. Observation).

A. COMPLETION STATUSES

CLOCK_NOT_SET	clock has not been initialized
ILLEGAL_USE	operation not callable from ISR
INVALID_ARGUMENTS	invalid number or type or size of arguments
INVALID_BIT	invalid exception bit-number
INVALID_BUFF	no buffer allocated from partition at buff_addr
INVALID_BUFF_SIZE	buff_size not supported
INVALID_CLOCK	invalid clock value
INVALID_COUNT	initial 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_OPTIONS	invalid options value
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_PRIORITY	invalid priority value
INVALID_SEGMENT	no segment allocated from this region at seg_addr
NAME_NOT_FOUND	object name does not exist on node
NODE_NOT_REACHABLE	node on which object resides is not reachable
NO_EVENT	event(s) not set and NOWAIT option given
NO_MORE_MEMORY	not enough memory to satisfy request
OBJECT_DELETED	originally existing task has been deleted before operation
OBJECT_NOT_LOCAL	operation not allowed on non-local object
OBJECT_PROTECTED	task in NOTERMINATION mode
OK	operation successful
POOL_IN_USE	buffers from this pool are still allocated
POOL_NOT_SHARED	pool not in shared memory subsystem
POOL_OVERLAP	area given overlaps an existing pool
QUEUE_DELETED	queue deleted while blocked in queue_receive
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_claim
SEMAPHORE_NOT_AVAILABLE	semaphore unavailable with NOWAIT option
SEMAPHORE_OVERFLOW	semaphore counter overflowed
SEMAPHORE_UNDERFLOW	semaphore counter underflowed
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_OBJECTS	too many objects of given type on the node or in the system
XSR_NOT_SET	no handler routine for given exception(s)

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 OBJECT_NOT_LOCAL completion status must be returned.

Observation:

The SELF literal is meaningless for ORKID operations called from an ISR and will lead to the INVALID_ID completion status.

NODE OPERATIONS

```
node_fail ( nid, code, options )
```

Task Operations

```
task_suspend         ( tid )
task_resume          ( tid )
task_read_note-pad   ( tid, loc_number, loc_value )
task_write_note-pad  ( tid, loc_number, loc_value )
```

Semaphore Operations

```
sem_release          ( sid )
```

Queue Operations

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

Event Operations

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

Exception Operations

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

Clock Operations

```
clock-get            ( clock )
clock-tick           ( )
```

Interrupt Operations

```
int_enter            ( )
int_return           ( )
```

C. SUMMARY OF ORKID OPERATIONS

In the following, output parameters are printed in bold characters.

Node Operations

```
node_ident          ( name, nid )
node_fail           ( nid, code, options )
node_info           ( nid, ticks_per_sec )
```

Task Operations

```
task_create         ( name, priority, stack_size, mode, options, tid )
task_delete         ( tid )
task_ident          ( name, nid, 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       ( new_mode, mask, old_mode )
task_read_note_pad  ( tid, loc_number, loc_value )
task_write_note_pad ( tid, loc_number, loc_value )
task_info           ( tid, priority, mode, options, event, exception )
```

Region Operations

```
region_create       ( name, addr, length, granularity, options, rid )
region_delete       ( rid )
region_ident        ( name, rid )
region_get_seg      ( rid, seg_size, seg_addr )
region_ret_seg      ( rid, seg_addr )
region_info         ( rid, size, max_segment, granularity, options )
```

Pool Operations

```
pool_create         ( name, addr, length, buff_size, options, pid )
pool_delete         ( pid )
pool_ident          ( name, nid, pid)
pool_get_buff       ( pid, buff_addr )
pool_ret_buff       ( pid, buff_addr )
pool_info           ( pid, buffers, free_buffers, buff_size, options )
```

Semaphore Operations

```
sem_create          ( name, init_count, options, sid )
sem_delete          ( sid )
sem_ident           ( name, nid, sid )
sem_claim           ( sid, options, time_out )
sem_release         ( sid )
sem_info            ( sid, options, count, tasks_waiting )
```

Queue Operations

```
queue_create        ( name, max_buff, length, options, qid )
```