## 3.5. TASK_RESTART

Restart a task.

Synopsis

```
   task_restart( tid, arguments )
```

Input Parameters

| | | |
|---|---|---|
| tid | : task_id | kernel defined identifier |
| arguments | : * | arguments passed to task |

Output Parameters

&lt;none&gt;

Literal Values

| | | |
|---|---|---|
| tid | = SELF | The calling task restarts itself |

Completion Status

| | |
|---|---|
| OK | task_restart operation successful |
| ILLEGAL_USE | operation not callable from ISR |
| INVALID_PARAMETER | a parameter refers to an illegal address |
| INVALID_ID | task does not exist |
| OBJECT_DELETED | task specified has been deleted |
| INVALID_ARGUMENTS | invalid number or type or size of arguments |
| TASK_NOT_STARTED | task has not yet been started |
| OBJECT_PROTECTED | task has NOTERMINATION parameter set |
| NODE_NOT_REACHABLE | node on which task resides is not reachable |

Description

The task_restart operation interrupts the current thread of execution of the specified task and forces the task to restart at the address given in the task_start call which originally started the task. The stack pointer is reset to its original value. No assumption can be made about the original content of the stack at this time.

Any resources allocated to the task are not affected during the task_restart operation. The tasks themselves are responsible for the proper management of such resources through task_restart.

If the task's active mode has the parameter NOTERMINATION set, then the task will not be restarted and the completion status OBJECT_PROTECTED will be returned.

* The specification of the number and type of the arguments is language binding dependent. For a high level language, it is likely that these arguments will be passed as parameters to the procedure whose name was given as start address in the original task_start call.

## 3.6. TASK_SUSPEND

Suspend a task.

Synopsis

```
task_suspend( tid )
```

Input Parameters

| | | |
|---|---|---|
| tid | : task_id | kernel defined task identifier |

Output Parameters

<none>

Literal Values

| | | |
|---|---|---|
| tid | = SELF | The calling task suspends itself |

Completion Status

| | |
|---|---|
| OK | task_suspend operation successful |
| INVALID_PARAMETER | a parameter refers to an illegal address |
| INVALID_ID | task does not exist |
| OBJECT_DELETED | task specified has been deleted |
| OBJECT_PROTECTED | task has NOPREEMPT parameter set |
| TASK_ALREADY_SUSPENDED | task already suspended |
| NODE_NOT_REACHABLE | node on which task resides is not reachable |

Description

This operation temporarily suspends the specified task until the suspension is lifted by a call to task_resume. While it is suspended, a task cannot be scheduled to run.

If the task's active mode has the parameter NOPREEMPT set the operation will fail and return the completions status OBJECT_PROTECTED, unless the task suspends itself. In which case the operation will always be successful.

## 3.7. TASK_RESUME

Resume a suspended task.

Synopsis

```
task_resume( tid )
```

Input Parameters

| | | |
|---|---|---|
| tid | : task_id | kernel defined task identifier |

Output Parameters

<none>

Completion Status

| | |
|---|---|
| OK | task_resume operation successful |
| INVALID_PARAMETER | a parameter refers to an illegal address |
| INVALID_ID | task does not exist |
| OBJECT_DELETED | task specified has been deleted |
| TASK_NOT_SUSPENDED | task not suspended |
| NODE_NOT_REACHABLE | node on which task resides is not reachable |

Description

The task_resume operation lifts the task's suspension immediately after the point at which it was suspended. The task must have been suspended with a call to the task_suspend operation.

## 3.8. TASK_SET_PRIORITY

Set priority of a task.

Synopsis

```
   task_set_priority( tid, new_prio, old_prio )
```

Input Parameters

| | | |
|---|---|---|
| tid | : task_id | kernel defined task id |
| new_prio | : prio | task's new priority |

Output Parameters

| | | |
|---|---|---|
| old_prio | : prio | task's previous priority |

Literal Values

| | | |
|---|---|---|
| tid | = SELF | The calling task sets its own priority |
| new_prio | = CURRENT | There will be no change in priority |

Completion Status

| | |
|---|---|
| OK | task_set_priority operation successful |
| ILLEGAL_USE | operation not callable from ISR |
| INVALID_PARAMETER | a parameter refers to an illegal address |
| INVALID_ID | task does not exist |
| OBJECT_DELETED | task specified has been deleted |
| INVALID_PRIORITY | invalid priority value |
| NODE_NOT_REACHABLE | node on which task resides is not reachable |

Description

This operation sets the priority of the specified task to new_prio. The new_prio parameter is specified as CURRENT if the calling task merely wishes to find out the current value of the specified task's priority. ( see also 3. Task Priority )

## 3.9. TASK_SET_MODE

Set mode of own task.

Synopsis

```
   task_set_mode( new_mode, mask, old_mode )
```

Input Parameters

| | | |
|---|---|---|
| new_mode | : bit_field | new task mode settings |
| mask | : bit_field | significant bits in mode |

Output Parameters

| | | |
|---|---|---|
| old_mode | : bit_field | task's previous mode |

Literal Values

| | | |
|---|---|---|
| new_mode | + NOXSR | XSRs cannot be activated |
| | + NOTERMINATION | task cannot be restarted or deleted |
| | + NOPREEMPT | task cannot be preempted |
| | + NOINTERRUPT | interrupt handling routine cannot be activated |
| old_mode | + NOXSR | XSRs cannot be activated |
| | + NOTERMINATION | task cannot be restarted or deleted |
| | + NOPREEMPT | task cannot be preempted |
| | + NOINTERRUPT | interrupt handling routine cannot be activated |
| mask | | (same as mode) |

Completion Status

| | |
|---|---|
| OK | task_set_mode operation successful |
| ILLEGAL_USE | operation not callable from ISR |
| INVALID_PARAMETER | a parameter refers to an illegal address |
| INVALID_MODE | invalid mode or mask value |

Description

This operation sets a new active mode for the task or its XSR. If called from a task's XSR then the XSR mode is changed, otherwise the main task's mode is changed.

The mode parameters which are to be changed are given in mask. If a parameter is to be set then it is also given in mode, otherwise it is left out. For both mask and mode, the logical OR (!) of the symbolic values for the mode parameters are passed to the operation.

For example, to clear NOINTERRUPT and set NOPREEMPT, mask = NOINTERRUPT ! NOPREEMPT, and mode = NOPREEMPT. To return the current mode without altering it, the mask should simply be set to zero. ( see also 3. Task Modes )