12.1. TIMER_WAKE_AFTER

Wake after a specified time interval.

Synopsis

```
timer_wake_after( ticks )
```

Input Parameters

ticks	: integer	number of ticks to wait

Output Parameters

<none>

Completion Status

OK	timer_wake_after successful
ILLEGAL_USE	timer_wake_after not callable from ISR

Description

This operation causes the calling task to be blocked for the given number of ticks. The task is woken after this interval has expired, and is returned a successful completion status. If the node clock is set using the clock_set operation during this interval, the number of ticks left does not change.

12.2. TIMER_WAKE_WHEN

Wake at a specified wall time and date.

Synopsis

```
timer_wake_when( clock )
```

Input Parameters

clock	: clock_buff	time and date to wake

Output Parameters

<none>

Completion Status

OK	timer_wake_when successful
ILLEGAL_USE	timer_wake_when not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_CLOCK	invalid clock value
CLOCK_NOT_SET	clock has not been initialized

Description

This operation causes the calling task to be blocked up until a given date and time. The task is woken at this time, and is returned a successful completion status. The kernel checks the supplied clock_buf data for validity. The exact structure of that data is language binding dependent.

If the node clock is set while the timer is running, the wall time at which the task is woken remains valid. If the node time is set to after the timer wake time, then the timer is deemed expired and the task is woken immediately and returned a successful completion status.

12.3. TIMER_EVENT_AFTER

Send event after a specified time interval.

Synopsis

```
timer_event_after( ticks, event, tmid )
```

Input Parameters

ticks	: integer	number of ticks to wait
event	: bit_field	event to send

Output Parameters

tmid	: timer_id	kernel defined timer identifier

Completion Status

OK	timer_event_after successful
ILLEGAL_USE	timer_event_after not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
TOO_MANY_OBJECTS	too many timers on the node

Description

This operation starts an event timer which will send the given events to the calling task after the specified number of ticks. The kernel returns an identifier which can be used to cancel the timer. If the node clock is set using the clock_set operation during this interval, the number of ticks left does not change.

12.4. TIMER_EVENT_WHEN

Send event at the specified wall time and date.

Synopsis

```
   timer_event_when( clock, event, tmid )
```

Input Parameters

clock	: clock_buff	time and date to send event
event	: bit_field	event(s) to send

Output Parameters

tmid	: timer_id	kernel defined timer identifier

Completion Status

OK	timer_event_when successful
ILLEGAL_USE	timer_event_when not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_CLOCK	invalid clock value
TOO_MANY_OBJECTS	too many timers on the node
CLOCK_NOT_SET	clock has not been initialized

Description

This operation starts an event timer which will send the given events to the calling task at the given date and time. The kernel returns an identifier which can be used to cancel the timer.

If the node clock is set while the timer is running, the wall time at which the envent(s) are sent remains valid. If the node time is set to after the value specified in the clock parameter, then the timer is deemed expired and the events are sent to the calling task immediately.

12.5. TIMER_EVENT_EVERY

Send periodic event.

Synopsis

```
   timer_event_every( ticks, event, tmid )
```

Input Parameters

ticks	: integer	number of ticks to wait between events
event	: bit_field	event to send

Output Parameters

tmid	: timer_id	kernel defined timer identifier

Completion Status

OK	timer_event_every successful
ILLEGAL_USE	timer_event_every not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
TOO_MANY_OBJECTS	too many timers on the node

Description

This operation starts an event timer which will periodically send the given events to the calling task with the periodicity specified by the number of ticks. The kernel returns an identifier which can be used to cancel the timer. If the node clock is set using the clock_set operation during the life time of the timer, the number of ticks left until the next event does not change.

Observation:

This provides a drift-free mechanism for sending an event at periodic intervals.