ERROR CONDITIONS

Invalid *tid*.

Task has never been started.

Task not created from local node.

NOTES

Not callable from ISR.

May cause a preempt if the task being restarted has a higher priority than the running task, and the preempt mode is in effect.

3.1.5 T_DELETE

NAME

t_delete – "Delete a Task"

SYNOPSIS

uint t_delete (tid)

```
uint tid;   /* task id as returned from t_create or t_ident  */
            /* 0 indicates requesting task */
```

DESCRIPTION

This directive allows a task to delete itself, or the task identified in the *tid* field. The executive halts execution of the task and frees the task data structure.

The task identified by the *tid* must exist on the local processor, even if the task was created with the GLOBAL flags value set (see *t_create*).

RETURN VALUE

If the task identified in the *tid* field is the requesting task, then *t_delete* always succeeds, and there is no return.

If the task identified in the *tid* field is not the requesting task, and *t_delete* successfully deleted the task, then 0 is returned to the requesting task.

If the task identified in the *tid* field is not the requesting task, and the call was not successful, an error code is returned to the requesting task.

ERROR CONDITIONS

Invalid *tid*.

Task not created on local node.

NOTES

Not callable from ISR.

A new task is scheduled when the requesting task deletes itself, and there is no return.

Tasks are responsible for returning resources to the executive before deleting itself. It is suggested that a task needing to delete another task use *as_send* or *t_restart* to inform the task to return its resources and then delete itself.

3.1.6 T_SUSPEND

NAME

t_suspend -- "Suspend Task"

SYNOPSIS

uint t_suspend (tid)

```
uint tid;   /* task id as returned from t_create or t_ident */
            /* 0 indicates requesting task */
```

DESCRIPTION

The executive will prevent future execution of the task identified in the *tid* field. The task identified by the *tid* is placed in a suspended state. The suspended state is in addition to the other wait states; waiting for memory, for a message, for an event, for a semaphore, or for a timeout.

The *t_resume* directive issued by another task removes the suspended state. The task is made ready unless blocked by any other wait state.

The task identified by the *tid* may exist on the local processor or any remote processor in a multiprocessor configuration, as long as the task was created with the GLOBAL flags value set (see *t_create).*

RETURN VALUE

If the task identified in the *tid* field is the requesting task, then *t_suspend* always succeeds and returns 0 when the task runs.

If the task identified in the *tid* field is not the requesting task, and *t_suspend* successfully put the task in the suspend state, then 0 is returned to the requesting task.

If the task identified in the *tid* field is not the requesting task, and the call was not successful, an error code is returned to the requesting task.

ERROR CONDITIONS

Invalid *tid.*

Task already suspended.

NOTES

Not callable from ISR.

The running task will be blocked if suspending itself.

3.1.7 T_RESUME

NAME

t_resume – "Resume a Task"

SYNOPSIS

```
uint t_resume ( tid )

        uint tid;    /* task id as returned from t_create or t_ident */
```

DESCRIPTION

The *t_resume* directive removes the task identified in the *tid* field from the suspended state.

If the task was waiting for memory, for a message, for an event, for a semaphore, or for a timeout, then the task will not be scheduled. Otherwise, the task is scheduled to await execution. If the task is the highest priority ready to run task, it will cause a preempt.

The task identified by the *tid* may exist on the local processor or any remote processor in a multiprocessor configuration, as long as the task was created with the GLOBAL flags value set (see *t_create*).

RETURN VALUE

If *t_resume* successfully resumed the task, then 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Invalid *tid*.

Task not suspended.

ISR cannot reference remote node.

NOTES

Can be called from within an ISR, except when the task was not created on the local node.

May cause a preempt if the the resumed task is ready to run and has a higher priority than the running task, and the preempt mode is in effect. A preempt will not occur if the resumed task exists on a remote processor in a multiprocessor configuration.

3.1.8 T_SETPRI

NAME

t_setpri – "Set Task Priority"

SYNOPSIS

uint t_setpri (tid, priority, &ppriority)

```
uint tid;       /* task id as returned from t_create or t_ident */
                /* 0 indicates requesting task */
uint priority;  /* task priority */
                /* 0 indicates current priority */
uint ppriority; /* previous priority - returned by this call */
```

DESCRIPTION

This directive changes the current priority of the task identified in the *tid* field to the new value specified by *taskattr*. A task may change its own priority or the priority of another task. The task will be scheduled according to the new priority.

Priority level zero is reserved by the system, and may not be used as a priority. If zero is specified in the *priority* field, the task's current priority will be returned. The executive will support a minimum of 32 priorities.

The task identified by the *tid* may exist on the local processor or any remote processor in a multiprocessor configuration, as long as the task was created with the GLOBAL flags value set (see *t_create*).

RETURN VALUE

If *t_setpri* successfully changed the task priority, the *ppriority* is filled in, and 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Invalid *tid*.

Invalid priority.

NOTES

Not callable from ISR.

May cause a preempt if the running task lowers its own priority, or raises the priority of another task, and the preempt mode is in effect. A preempt will not occur if the task having its priority raised exists on a remote processor in a multiprocessor configuration.