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

ERROR CONDITIONS

Too many tasks.

No more memory for stack(s) segment.

Superstk too small.

Invalid priority.

NOTES

Not callable from ISR.

Will not cause a preempt.

3.1.2 T_IDENT

NAME

t_ident -- "Obtain id of a task"

SYNOPSIS

uint t_ident (name, node, &tid)

```
uint name;    /* user defined 4-byte task name */
              /* 0 indicates requesting task */
uint node;    /* node identifier */
              /* 0 indicates any node */
uint tid;     /* task id - returned by this call */
```

DESCRIPTION

This directive allows a task to obtain the *tid* of itself or another task in the system. The *tid* must then be used in all calls to the executive requiring a *tid*.

If the task name is not unique, the *tid* returned may not correspond to the task named in this call.

The task identified by its name 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*). If the task name is not unique within the multiprocessor configuration, a non-zero node identifier must be specified in the *node* field.

RETURN VALUE

If *t_ident* succeeded, the *tid* is filled in, and 0 is returned.

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

ERROR CONDITIONS

Task with this name does not exist.

Invalid node identifier.

NOTES

Can be called from within an ISR.

Will not cause a preempt.

3.1.3 T_START

NAME

t_start – "Start a Task"

SYNOPSIS

uint t_start (tid, saddr, mode, argp)

```
uint tid;          /* task id as returned from t_create or t_ident */
ptf saddr;         /* start execution address of task */
uint mode;         /* initial mode value of task */
long (*argp)[4];   /* pointer to argument list */
```

The *mode* value is defined as follows:

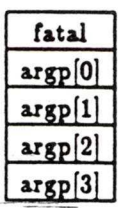
NOPREEMPT	set	to disable preempting
	clear	to enable preempting
TSLICE	set	to enable timeslicing
	clear	to disable timeslicing
NOASR	set	to disable asynchronous signal processing
	clear	to enable asynchronous signal processing
SUPV	set	to execute in supervisor mode
	clear	to execute in user mode
LEVEL		interrupt level when SUPV is set

DESCRIPTION

The task identified by the *tid* is made ready, based on its current priority, to await execution. A task can be started only from the dormant state.

Saddr is the logical address where the task wants to start execution. *Mode* contains the flag values to enable/disable preempting, timeslicing, asynchronous processing, supervisor mode and an optional interrupt level when the task starts execution.

Argp is a pointer to a list of four arguments. These arguments are pushed onto the stack of the task being started. A fifth argument, the executive's fatal error handler, is also pushed onto the task's stack. Should the task attempt to exit the procedure (which normally causes unpredictable behavior), the executive's fatal error handler will be executed. The user must take this frame into consideration when calculating the size of a task's stack(s).



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 *t_start* successfully started the task, then 0 is returned.

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

ERROR CONDITIONS

Invalid *tid*.

Task not in dormant state.

Task not created from local node.

NOTES

Not callable from ISR.

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

3.1.4 T_RESTART

NAME

t_restart – "Restart a Task"

SYNOPSIS

uint t_restart (tid, argp)

```
uint tid;       /* task id as returned from t_create or t_ident */
long argp[4];   /* pointer to argument list */
```

DESCRIPTION

The task identified by the *tid* is made ready. If the task was blocked, the executive unblocks it. The task's *superstk, userstk,* and *priority* are set to their original values established when the task was created using *t_create*. The task's start address *saddr* and *mode* are set to their original values established when the task was started using *t_start*. A task can be restarted from any state.

Argp is a pointer to a list of four arguments. These arguments are pushed onto the stack of the task being restarted. This argument list may be different from the original argument list. A fifth argument, the executive's fatal error handler, is also pushed onto the task's stack. Should the task attempt to exit the procedure (which normally causes unpredictable behavior), the executive's fatal error handler will be executed.

Tasks which anticipate being restarted can use the arguments to distinguish between initial startup and a restart.

Due to the capability of this call to unblock a task, this call is useful to delete a task in the system. Tasks which anticipate being deleted can use the arguments to distinguish between initial startup and deletion.

fatal
argp[0]
argp[1]
argp[2]
argp[3]

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 *t_restart* successfully restarted the task, then 0 is returned.

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