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 tank 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 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 exits on a remote processor in a multiprocessor configuration.

3.1.8 T\_SETPRI

NAME

t\_setpri – “Set Task Priority”

SYNOPSIS

uint t\_settpri ( 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 \*/

Unit 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.

3.1.9 T\_MODE

NAME

t\_mode - "Change Task Mode”

SYNOPSIS

uint t\_mode ( mode, mask, &pmode )

uint mode; /\* new mode \*/

uint mask; /\* mask \*/

uint pmode; /\* previous mode - returned by this call \*/

The mode and mask values are 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

T\_mode enables and disables several modes of execution for the calling task. A task may enable/disable timeslicing, enable/disable preempting, enable/disable asynchronous signal processing, or execute in supervisor mode at an optional interrupt level.

Tasks have the ability to process signals asynchronously. Any task with a valid asynchronous signal routine (asr) which needs to temporarily disable asynchronous processing should use this directive.

To change a particular mode, the user must indicate which mode is being changed by setting the appropriate value in the mask parameter, and then set the appropriate value in the mode parameter to the new mode. For example, if the user only wants to change the preempt mode characteristic, he would set the mask value to NOPREEMPT and the mode value to NOPREEMPT to disable preempting, or the mode field to 0 to enable preempting.

If the preempt mode is not in effect, timeslicing will not take place.

RETURN VALUE

The t\_mode call always succeeds, pmode is filled in, and 0 is returned.

NOTES

Not callable from ISR.

May cause a preempt if the running task enables preempting.

Refer to as\_catch for discussion on receiving asynchronous signals.

3.1.10 T\_GETREG

NAME

t\_getreg — “Get a task’s register”

SYNOPSIS

uint t\_getreg ( tid, regnum, &regval )

uint tid; /\* task id as returned from t\_create or t\_ident \*/

uint regnum; /\* register number \*/

uint regval; /\* register value - returned by this call \*/

The regnum field value: are:

S\_REG0 System defined register 0

S\_REG1 System defined register 1

S\_REG2 System defined register 2

S\_REG3 System defined register 3

S\_REG4 System defined register 4

S\_REG5 System defined register 5

S\_REG6 System defined register 6

S\_REG7 System defined register 7

U\_REG0 User defined register 0

U\_REG1 User defined register 1

U\_REG2 User defined register 2

U\_REG3 User defined register 3

U\_REG4 User defined register 4

U\_REG5 User defined register 5

U\_REG6 User defined register 6

U\_REG7 User defined register 7

DESCRIPTION

The executive returns the register value in the regval field for the register identified in the regnum field and the task identified by the tid.

The task identified in the tid field may exist on the local processor, or any remote processor in the multiprocessing configuration if the task was created with the GLOBAL flags value set (see t\_create).

RETURN VALUE

If t\_getreg is successful, regval is filled in, and 0 is returned.

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

ERROR CONDITIONS

Invalid tid.

Invalid register number.

ISR cannot reference remote node.

NOTES

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

Will not cause a preempt.

3.1.11 T\_SETREG

NAME

t\_setreg — “Set a task’s register”

SYNOPSIS

uint t\_setreg ( tid, regnum, regval )

uint tid; /\* task id as returned from t\_create or t\_ident \*/

uint regnum; /\* register number \*/

uint regval; /\* register value \*/

The regnum field values are:

S\_REG0 System defined register 0

S\_REG1 System defined register 1

S\_REG2 System defined register 2

S\_REG3 System defined register 3

S\_REG4 System defined register 4

S\_REG5 System defined register 5

S\_REG6 System defined register 6

S\_REG7 System defined register 7

U\_REG0 User defined register 0

U\_REG1 User defined register 1

U\_REG2 User defined register 2

U\_REG3 User defined register 3

U\_REG4 User defined register 4

U\_REG5 User defined register 5

U\_REG6 User defined register 6

U\_REG7 User defined register 7

DESCRIPTION

The executive sets the register identified in the regnum field for the task identified by the tid with the value in the regval field.

The task identified in the tid field may exist on the local processor, or any remote processor in the multiprocessing configuration if the task was created with the GLOBAL flags value set (see t\_create).

RETURN VALUE

If t\_setreg successfully set the register value, 0 is returned.

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