Real Time Executive Interface Definition January 22, 1988

3.4.9 TM\_TICK

NAME

tm\_tick - ”Announce Tick”

SYNOPSIS

uint tm\_tick ( )

DESCRIPTION

This call is used to inform the executive that a system clock tick has occurred. This information is used by the time manager to maintain correct calendar time, execute timeslicing, and decrement ticks from tasks which are currently being delayed or timing out. When a timeslice or timeout expires, the task is made ready.

RETURN VALUE

Tm.\_tick always succeeds and returns 0.

ERROR CONDITIONS

None.

NOTES

Can be called from within an ISR.

 Page 63

January 22, 1988 Real Time Executive Interface Definition

3.5 Interrupt Handling .

Fast interrupt response and the ability to preempt from an Interrupt Service Routine (ISR) are important features of a real time executive.

In order to provide the fastest possible interrupt service mechanism, the executive will allow tasks and lSRs to directly claim interrupt vectors by writing directly to the vector table.

An ISR usually communicates with tasks within the system using RTEID directives. The directives which are callable from ISRs are identified in the NOTES section of each directive. Directives called from an ISR will always return immediately to the ISR, without going though the normal dispatch cycle. The postponed dispatch is required to complete the ISR. before any tasks are dispatched.

The i\_return directive provides the real-time exit mechanism for ISRs. Since an ISR can make a task other than the running task ready to run, i.e. by sending a message from the ISR, it becomes extremely important NOT to exit the ISR with the RTE instruction. This would return control to the running task at the time of the interrupt, which may not be the highest priority task ready to run. To ensure the highest priority task runs, all ISRs must exit using the i\_return directive, which may cause the running task to be preempted.

The directives provided by the interrupt manager are:

|  |  |
| --- | --- |
| Directive | Function |
| i\_return | Return from Interrupt |

Page 64

Real Time Executive Interface Definition January 22, 1988

3.5.1 I\_RETURN

NAME

i\_return - “Return from Interrupt”

SYNOPSIS

void i\_return ()

DESCRIPTION

The i\_return directive will allow the executive to return control to the highest priority task in the system following the interrupt processing. The interrupt routine may have caused a task of higher priority than the task running at the time of interrupt, to become ready.

RETURN VALUE

None.

ERROR CONDITIONS

None.

NOTES

Can only be called from an ISR.

 Page 65

January 22, 1988 Real Time Executive Interface Definition

3.6 Fatal Errors.

Occasionally, the executive, application or system software will detect an unrecoverable error condition. Such a condition is called a fatal error and normally halts execution on the local node. Such errors include checksum errors, not enough memory, etc.

The executive will provide a fatal error handler which is responsible for processing fatal errors. The exact manner in which fatal errors are processed is implementation dependent. For example, the executive may simply STOP, or it may pass control to a debugger or other user provided fatal error handling routine.

There are three sources for fatal errors:

1. the executive

2. system code

3. user application code

When the executive detects a fatal error, control is automatically passed to the fatal error handler. When system code or user application code detects a fatal error, the k\_fatal directive should be used to pass control to the fatal error handler. The error code passed to the fatal error handler describes the type of fatal error.

Fatal errors only halt execution on the local node. Remote nodes are not directly affected.

The directive provided to report fatal errors is:

|  |  |
| --- | --- |
| Directive | Function |
| k\_fatal | Fatal Error |

Page 66

Real Time Executive Interface Definition January 22, 1988

3.6.1 K\_FATAL

NAME

k\_fatal - "Fatal Error”

SYNOPSIS

void k\_fatal ( errcode )

 uint errcode; /\* type of error to be reported \*/

DESCRIPTION

The k\_fatal directive will allow the executive to halt execution of the system in a manner as described by the errcode. This directive does not return to the caller.

RETURN VALUE

None.

ERROR CONDITIONS

None.

NOTES

Can be called from within an ISR.

Page 67

January 22, 1988 Real Time Executive Interface Definition

3.7 Memory Management

The executive will support two different memory managers. A region manager provides allocation of variable sized memory segments. A partition manager provides allocation of fixed sized buffers.

3.7.1 Region Manager

A region is an area of physical contiguous memory from which the executive can dynamically allocate segments to an application. A segment is a variable length block of memory.

A region is created with the rn\_create directive. Like all objects managed by the executive, a region has a 4 character name, and, once created, a 32-bit region id ( rnid). Tasks other than the creator can use the rn\_ident directive to obtain a region’s rnid. The directives rn\_getseg and rn\_retseg allocate and return segments from the region.

Each region has an associated pagesize, specified when the region is created. The pagesize must be a power of 2. Segment lengths are always in multiples of this pagesize. For example, if a task requests a 700 byte segment from a region having a 512 byte pagesìze, a 1024 byte segment is allocated.

When requesting a segment, if the request cannot immediately be satisfied, the requesting task may optionally wait ( with or without timeout ) for a segment to become available. If it elects to wait, the task is placed in a memory wait queue associated with the region. Tasks can be queued either by priority or FIFO. When a segment is returned, if possible it is merged with its neighbor segments and then the wait queue is searched. The first task, if any, whose request can be satisfied receives the segment.

In a multiprocessor system, regions may not be shared between processors. Segments may only be allocated or returned by tasks running on the processor from which the region was created. Hence, the GLOBAL flag used with the other create services is not supported by rn\_create.

When a region is created, the executive must build data structures to manage the region. The memory containing these structures may itself be allocated from the region, in which case, the amount of allocatable memory within the region may be slightly less than the original size of the region.

The maximum number of regions that may exist at any one time is a configuration parameter.

The directives provided by the region manager are:

Directive Function

|  |  |
| --- | --- |
| Directive | Function |
| rn\_create | Create a region |
| rn\_ident | Obtain id of a region |
| rn\_delete | Delete a region |
| rn\_getseg | Get a segment |
| rn\_retseg | Return a segment |

Page 68

Real Time Executive lnterface Definition January 22, 1988

3.7.2 Partition Manager

A partition is a pool of equal sized buffers. Pt\_create creates a partition in a physical contiguous memory area provided by the caller. Like all objects managed by the executive, partitions have a 4 character name, and, once created, a 32-bit partition id ( ptid). Tasks other than the creator can use the pt\_ident directive to obtain a partition’s ptid. Pt\_getbuf and pt\_retbuf allocate and return buffers from the partition.

Each partition contains a specified number of size buffers. The number and size of the buffers is specified when the partition is created.

In a shared memory multiprocessor configuration, partitions may be shared between processors. To do so, the caller must declare the partition GLOBAL when it is created. If a partition is GLOBAL, then the executive will arbitrate access to the partition.

Tasks may not wait for buffers. If no buffers are available an error number is returned.

When a partition is created, the executive must build data structures to manage the partition. The memory containing these structures may be allocated within the partition area provided by the caller, in which case, the partition may occupy slightly more memory than the simple product of the buffer count and buffer size.

The maximum number of partitions that may exist at any one time is a configuration parameter.

The directives provided by the partition manager are:

|  |  |
| --- | --- |
| Directive | Function |
| pt\_create | Create a partition |
| pt\_ident | Obtain id of a partition |
| pt\_delete | Delete a partition |
| pt\_getbuf | Get a buffer |
| pt\_retbuf | Return a buffer |

 Page 69

January 22, 1988 Real Time Executive Interface Definition

3.7.3 RN-CREATE

NAME

rn\_create- “Create a Region”

SYNOPSIS

#include <memory.h>

unit rn\_create (name, paddr, length, pagesize, flags, &rnid, &bytes)

 uint name; /\* user defined 4-byte region name \*/

 char \*paddr; /\* physical start address of region \*/

 uint length; /\* physical length in bytes \*/

 uint pagesize; /\* region pagesize \*/

 uint flags; /\* region attributes \*/

 uint rnid; /\* region id - returned by this call \*/

 uint bytes; /\* available number of bytes - returned by this call \*/

The flags field values are:

 PRIOR set to process wait list by priority

 clear to process wait list by FIFO

DESCRIPTION

This directive allows the user to create a region from a physical contiguous memory area. The region id will be returned in rnid by the executive to use in rn\_getseg and rn\_retseg directives for the region.

The region physical start address specified in paddr will be long­word aligned by the executive. In systems with an MMU, the region physical start address must be on the pagesize boundary.

The available number of bytes within the region will be returned by the executive in the bytes field. Since the executive may use memory within the region for a region data structure, the number of bytes in bytes may be less than the number of bytes in length.

By setting the PRIOR value in the flags field, tasks which wait for segments from the region will be processed in task priority order. Otherwise, the tasks will wait in first in, first out (FIFO) order.

Regions may not be shared between processors in a shared memory multiprocessor configuration.

The maximum number of regions that can be in existence at one time is a configuration parameter.

RETURN VALUE

If rn\_create successfully created the region, then rnid and bytes are filled in and 0 is returned.

Page 70

Real Time Executive Interface Definition January 22, 1988

If the region was not successfully created, an error code is returned.

ERROR CONDITIONS

Too many regions.

Paddr is not on a pagesize boundary (MMU only).

NOTES

Not callable from ISR.

 Page 71

January 22, 1988 Real Time Executive Interface Definition

3.7.4 RN\_IDENT

NAME

rn\_ident - “Obtain id of a Region”

SYNOPSIS

#include <memory.h>

uint rn\_ident (name, &rnid)

 uint name; /\* user defined 4-byte region name \*/

 uint rnid; /\* region id - returned by this call \*/

DESCRIPTION

This directive allows a task to identify a previously created region by name, and obtain the rnid to use for rn\_getseg and rn\_retseg directives for the region.

The region must have been created by a task on the local processor. It may not be shared between processors in a shared memory multiprocessor configuration.

If the region name is not unique, the region id returned in rnid may not correspond to the region named by this call.

RETURN VALUE

If rn\_ident directive succeeds, then the rnid is filled in and O is returned.

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

ERROR CONDITIONS

Named region does not exist.

NOTES

Can be called from within an ISR.

Will not cause a preempt.

Page 72