Real Time Executive Interface Definition January 22, 1988

3.8 MMU Management

The executive can optionally support the PMMU (M68851 and M68030) to provide memory protection, dynamic task loading, and dynamic memory allocation.

To provide these services, the executive adopts an MMU model which defines the pagesize, the structure and depth of the memory map tree, and the degree of control each task has over its own memory map. Different implementations of the RTEID are free to choose different models. However, the model chosen should allow the standard memory management services (regions and partitions) to operate in a consistent and intuitive manner in both an MMU and non-MMU environment.

Logically, the RTEID adopts a sectioned view of the logical address space associated with each task. Memory objects are mapped into a task’s logical address space in variable size MÍMU sections. A single section is contiguous in the logical and possibly the underlying physical address spaces. Thus, the MMU is used to define a set of mappings for each task in the form:

 (Logical address, length) → Physical address range

Based on this model, the RTEID defines how the memory management services should operate, and defines additional services to manage the MMU directly.

3.8.1 Segments vs. Sections

MMU sections should not be confused with region segments. A segment is a block of memory allocated from a region. It can exist on any M68000 family. A section is only meaningful on the M68030or M88020/M88851 combination, and refers to a contiguous block of memory which is mapped into a task’s address space.

3.8.2 Regions

When a task calls rn\_getsegto obtain a segment from a region, the segment is automatically mapped into the task’s logical address space at an executive assigned address. Because rn\_getsegperforms the mapping, the corresponding region is not mapped into the address space of tasks using it. This means that allocated sections are accessible only by the allocating task, and those tasks which explicitly are given access to the segment using the MMU directives. Thus, a segment is fully protected from inadvertent access by other tasks.

3.8.3 Partition.

When a task executes a pt\_create or pt\_ident directive, the entire partition is mapped into the task’s address space. Thus, tasks which share a partition can share and access any buffers allocated from the partition. However, protection is on the partition level, and individual buffers are not protected.

 Page 83

January 22, 1988 Real Time Executive Interface Definition

The directives provided by the manager are:

|  |  |
| --- | --- |
| Directive  | Function |
| mm\_l2p | Logical to physical |
| mm\_p2l  | Physical to logical |
| mm\_pmap  | Map physical  |
| mm\_unmap | Unmap logical |
| mm\_pread | Physical read |
| mm\_pwrite  | Physical write |
| mm\_ptcreate  | Create logical partition |

Page 84

Real Time Executive Interface Definition January 22, 1988

3.8.4 MM\_L2P

NAME

mm\_l2p- "Logical to Physical"

SYNOPSIS

#include <memory.h>

uint mn\_l2p *(tid*, laddr, &paddr, &length**)**

uint tid; */*\* task Id as returned by t\_create or t\_ident */* char \*laddr; */\** logical start address */* char \*paddr; */\** physical startaddress **-** returned by this call *\*/\*/* uint length**;** */\** remaining lengthinbytes - returned by this call \**/*

DESCRIPTION

This directive calculates the physical address within the section associated with the logical address belonging to the task identified by the tid.

The physical start address is returned in the paddr field. The number of bytes remaining in the section is in the length field.

RETURN VALUE

If mm\_l2p was successful, then the physical start address is returned in paddr, the number of bytes remaining is returned in length, and 0 is returned.

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

ERROR CONDITIONS

Invalid tid.

Unmapped logical address.

Task not created on local node.

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.

Page 85

 January 22, 1988 Real Time Executive Interface Definition

3.8.5 MM\_P2L

NAME

Mm\_p2l— **\***Physicalto Logical\*

SYNOPSIS

 **#**include **<**memory.h>
uint mm\_p2l (tid, paddr, &laddr, length)

 uint tid**;** / \*taskidasreturned byt\_create ort\_ident **\*/** char\*paddr; / \*physical startaddress **\***/
 char \*laddr /\* logical start address - returned by this call \*/\* /
 uint length; /\* remaining length in bytes - returned by this call \*/

DESCRIPTION

This directive returns the logical address within the section associated with the physical address belonging to the task identified by the tid. The executive will only return the first valid mapping of the physical address it finds, and the logical address returned may be ambiguous if the task has a many­-to-one mapping of the physical address range.

The logical start address is returned in the laddr field, and the number of bytes remaining in the section is returned in the length field.

RETURN VALUE

If mm\_p2lwas successful, then the logical address is returned in laddr, the number of bytes remaining is returned in length, and 0 is returned.

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

ERROR CONDITIONS

Invalid tid.

Unmapped logical address.

Task not created on local node.

NOTES

Not callable from ISR.

Will not cause a preempt.

 Page 88

Real Time Executive Interface Definition January 22, 1988

3.8.6. MM\_PMAP

NAME

mm\_pmap — ‘Map Physical’

SYNOPSIS

#include <memory.h>

uint mm\_pmap (tid, laddr, paddr, length, flags)

uint tid; / \*taskidasreturned byt\_create ort\_ident **\*/**

char \*laddr; / logical start address ‘/

char \*paddr: / physical start address ‘/

uint length; / length in bytes /

uint flags; / section attributes /

The flags field values are defined as follows:

 RDONLY set read-only

 clear read-write

DESCRIPTION

This directive maps physical memory starting at paddr for the number of bytes specified in length, to a section at the logical start address laddr in the address space of the task identified by the tid.

The physical start address specified in paddr must be on the pagesize boundary. The logical start address specified in laddr must be on a section boundary.

If length is not a multiple of the pagesize, then more bytes than requested are mapped.

RETURN VALUE

II mm\_pmap was successful, and then 0 is returned.

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

ERROR. CONDITIONS

Invalid tid.

Paddr is not on a pagesize boundary.

Laddr is not on a section boundary.

Length specified is too large.

Page 87

January 22, 1988 Real Time Executive Interface

Duplicate logical address.

Task not created on local node.

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.

Page 88

 Real Time Executive Interface Definition January 22, 1988

3.8.7 MM\_UNMAP

NAME

mm\_unmap — ‘Unmap Logical’

SYNOPSIS

#include <memory.h>

uint mm\_unmap (tid, laddr)

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

char \* laddr\*; /\*logical start address \*/

DESCRIPTION

This directive removes the section starting at logical address laddr from the address space of the task identified by the tid.

RETURN VALUE

If mrn\_unmap was successful, then 0 is returned.

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

ERROR CONDITIONS

Invalid tid.

Unmapped logical address.

Task not created on local node.

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.

To return the segment to the region, the directive rn\_retseg must be used.

Page 89

 Real Time Executive Interface Definition January 22, 1988

3.8.8 MM\_PREAD

NAME

mm\_pread - 'Physical read"

SYNOPSIS

#include< memory.h >

uint mm\_pread(paddr, laddr, length)

 uint paddr; /\* physical start address\*/

 char \*laddr; /\*logical start address\*/

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

DESCRIPTION

The mm\_pread directive reads from a physical address, and writes to the logical address in the calling task’s address space. The length cannot span a section boundary.

RETURN VALUE

If mm\_pread was successful then 0 is returned.

If the call was not successful, no data is transferred and an error code is returned.

ERROR CONDITIONS

Unmapped logical address.

Length spans section boundary.

NOTES

Not callable from ISR.

Will not cause a preempt.

Page 90

Real Time Executive Interface Definition

3.8.9 MM\_PWRITE

NAME

mm\_pwrite - “Physical Write”

SYPNOSIS

#include <memory.h>

uint mm\_pwrite ( paddr, laddr, length )

 uint paddr; physical start address

 char laddr; logical start address

 uint length; length in bytes

DESCRIPTION

The mm\_pwrite directive reads from the logical address in the calling task’s address space, and writes to a physical address. The length may not span a section boundary.

RETURN VALUE

If mm\_pwrite was successful, then 0 is returned.

If the call was not successful, no data is transferred and an error code is returned.

ERROR CONDITIONS

Unmapped logical address.

Length spans section boundary.

NOTES

Not callable from ISR.

Will not cause a preempt.

Page 91

January 22, 1988 Real Time Executive Interface Definition

3.8.10 MM\_PTCREATE

NAME

mm\_ptcreate "Create a Logical Partition'

SYNOPSIS

#include <memory.h >

uint mm\_ptcreate (name, paddr, length, baise, laddr, flags, &ptid, &bnum)

 uint name; /\*user defined 4-byte partition name\*/
 char \*paddr; /\*physical start address of partition\*/
 uint length; /\*physical length in bytes\*/
 uint bsize; /\*size of buffers in bytes\*/
 char \*laddr; /\*physical start address of partition\*/
 uint flags; /\*partition attributes \*/
 uint ptid; /\*partition id- returned by this call \*/
 uint bnum; /\*number of buffers in partition - returned by this call\*/

Flags field values:

 GLOBAL set to indicate the partition is

 a multiprocessor global resource.

 clear to indicate the partition is local

DESCRIPTION

This directive allows the user to create a logical partition of fixed size buffers from a contiguous memory area. The partition is mapped into the caller’s address space at the logical address specified in laddr. By creating logical partitions at the same logical addresses, partitions can be easily shared between processors.

The partition id will be returned in ptid by the executive to use for pt\_getbuf and pt\_retbuf directives for the partition.

The partition physical start address must be on the pagesize boundary.

The number of buffers created by the executive will be returned in bnum. The executive may use memory within the partition for partition and buffer data structures. Therefore, the product of the buffer count and size will be slightly less than the length of the partition.

By setting the GLOBAL value in the flags field, the ptid will be sent to all processors in the system, to be entered into a global resource table. The system is defined as the collection of interconnected processors.

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

RETURN VALUE

Page 92