1.1.9 DB\_REMOTE

NAME

db\_remote -- "Remote Request"

SYNOPSIS

uint db\_remote (cpuid, request, &rval, argl, ..., argN)

uint cpuid; /\* Identifies remote cpu \*/

uint request; /\* Identifies request to be performed \*/

uint rval; /\* Return value of remote call - returned by this call \*/

uint argl; /\* First argument of request \*/

uint argN; /\* Last argument of request \*/

DESCRIPTION

The *db\_remote* directive will cause a directive to be executed on a remote cpu.

The *cpuid* identifies the remote cpu, the *request* specifies which RTEID request (including debug extensions) is to be performed, and *arg1-argN* specify the arguments.

*Arg1-argN* are the arguments for the request and their meaning is specific to the directive identified by *request*. Any addresses specific to the calling task are treated as external physical addresses.

RETURN VALUE

If *db\_remote* successfully completes, then *rval* contains the return value of the remote directive, and 0 is returned.

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

ERROR CONDITIONS

Invalid *cpuid.*

Invalid request.

Other error returns are based on the specific directive identified by *request.*

NOTES

Thu request operates as if a task on the remote system issues the request on behalf of the caller.

The actual execution or the remote request may be performed by the ISR which processes remote requests, or may be performed by a system task on the target system.

Since not all RTEID directives may be executed on & non-local cpu, the *db\_remote* directive will provide this function. It is especially important for debuggers which need to create tasks and manage resources on the target cpu.

This directive is also needed to access resources that are local to a remote cpu. For example, this directive could be used to suspend a task which does not have the GLOBAL flag set (assuming the task is local to a remote cpu).

Several directives have the address of return buffers as input parameters. The caller or *db\_remote*

must specify addresses which are external to the target processor (designated by *cpuid).*

1.1.10 DB\_BLOCK

NAME

db\_block -- "Prevent a Task Under Debug Control from Running"

SYNOPSIS

uint db\_block ( tid )

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

DESCRIPTION

The *db\_block* directive prevents the task identified in the *tid* field from executing. The controlling relationship must have been previously established using the *db\_control* directive.

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 flag set (see *t\_create).*

RETURN VALUE

If *db\_block* is successful, then 0 is returned.

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

ERROR CONDITIONS

Invalid *tid.*

Task not in controlled state.

Task already blocked.

NOTES

Not callable from ISR.

1.1.11 DB\_UNBLOCK

NAME

db\_unblock -- "Release a Task"

SYNOPSIS

uint db\_unblock ( tid )

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

DESCRIPTION

*Db\_unblock* allows the task identified by the *tid* field to resume execution under control of the requesting task. The controlling relationship must have been previously established using the *db\_control* directive.

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 flag set (see *t\_create).*

RETURN VALUE

If *db\_unblock* is successful, then 0 is returned.

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

ERROR CONDITIONS

Invalid *tid.*

Task not in controlled state.

Task not blocked.

NOTES

Not callable from ISR.

May cause a preempt.

1.1.12 DB\_GETMEM

NAME

db\_getmem -- "Get a Task's Memory"

SYNOPSIS

uint db\_getmem (tid, laddr, bufaddr, length )

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

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

char \*bufaddr; /\* buffer address \*/

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

DESCRIPTION

The executive reads memory from the task identified in the *tid* field, starting at the task's logical address *laddr,* and copies it to the buffer identified in the *bufaddr* field for the length identified in *length.*

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 flag set (see *t\_create).* This directive may be used to transfer data between a logical address belonging to the task identified by the *tid* and the requesting task's buffer.

RETURN VALUE

If *db\_getmem* successfully read the memory into the buffer, then 0 is returned.

If the memory was not successfully read into the buffer, an error code is returned.

ERROR CONDITIONS

Invalid *tid.*

Invalid *laddr* for the task.

Bus Error occurred during the read.

NOTES

Not callable from ISR.

Will not cause a preempt.

There is no requirement that the task identified by the tid be a controlled task.

*Db\_getmem* will attempt to only read the requested data and will not access memory beyond the *laddr+length*. If *length* is 1, a byte wide read is performed. If *length* is 2, a word wide read is performed.

1.1.13 DB\_SETMEM

NAME

db\_setmem -- "Set & Task's Memory"

SYNOPSIS

uint db\_setmem (tid, laddr, bufaddr, length )

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

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

char \*bufaddr; /\* buffer address \*/

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

DESCRIPTION

The executive writes memory to the task identified in the *tid* field from the buffer identified in the *bufaddr* starting at the task's logical address *laddr* field for the length identified in *length*.

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 flag set (see *t\_create*).

This directive may be used to transfer data between any requesting task's buffer and a logical

address belonging to the task identified by the *tid*.

RETURN VALUE

If *db\_setmem* successfully writes the memory from the buffer, then 0 is returned.

If the memory was not successfully written from the buffer, an error code is returned.

ERROR CONDITIONS

Invalid *tid*.

Invalid *laddr*.

Bus Error occurred during the write.

NOTES

Not callable from ISR.

Will not cause a preempt.

There is no requirement that the task identified by *tid* be a controlled task.

*Db\_setmem* will only read the requested data and will not access memory beyond the

*laddr+length*. If *length* is 1, a byte wide read is performed. If *length* is 2, a word wide read is

performed.

l.l.14 DB\_GETREG

NAME

Db\_getreg -- "Get a task's register"

SYNOPSIS

uint db\_getreg ( tid, regnum, &regptr )

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

uint regnum; /\* register number \*/

union regval \*regptr; /\* pointer to register value - returned by this call \*/

union regval {

uint i;

float f;

}

The *regnum* field values are:

S\_STAT Task's status byte values:

T\_WTMEM waiting for memory

T\_WTMSG waiting on message queue

T\_WTEVT waiting for event

T\_WTSEM waiting for semaphore

T\_WTTIM waiting for timeout

T\_WTCTL waiting on control

D\_REG0 Task's Processor Register D0

D\_REGl Task's Processor Register Dl

D\_REG2 Task's Processor Register D2

D\_REG3 Task's Processor Register D3

D\_REG4 Task's Processor Register D4

D\_REG5 Task's Processor Register D5

D\_REG6 Task's Processor Register D6

D\_REG7 Task's Processor Register D7

A\_REG0 Task's Processor Register A0

A\_REGl Task's Processor Register Al

A\_REG2 Task's Processor Register A2

A\_REG3 Task's Processor Register A3

A\_REG4 Task's Processor Register A4

A\_REG5 Task's Processor Register A5

A-REG6 Task's Processor Register A6

A-REG7 Task's Processor Register A7

H\_SR Status Register

H\_PC Program Counter

H\_VOR Vector Offset Register

H\_USP User Stack Pointer

H\_ISP Interrupt Stack Pointer

H\_MSP Master Stack Pointer

H\_VBR Vector Base Register

H\_CACR Cache Control Register

H\_CAAR Cache Address Register

H\_VBR Vector Base Register

H\_CACR Cache Control Register

H\_CAAR Cache Address Register

FP\_REG0 Task's Processor Register FP0

FP\_REGl Task's Processor Register FPl

FP\_REG2 Task's Processor Register FP2

FP\_REG3 Task's Processor Register FP3

FP\_REG4 Task's Processor Register FP4

FP\_REG5 Task's Processor Register FP5

FP\_REG6 Task's Processor Register FP6

FP\_REG7 Task's Processor Register FP7

FPCR Task's Coprocessor Control Register

FPSR Task's Coprocessor Status Register

FPIAR Task's Coprocessor Instruction Address Register

DESCRIPTION

The executive returns the register value in the *regptr* 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 *db\_getreg* is successful, *regptr* is filled in and 0 is returned.

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

ERROR CONDITIONS

Invalid *tid*.