1.1.9 DB_REMOTE

NAME

db_remote -- "Remote Request"

SYNOPSIS

uint db_remote (cpuid, request, &rval, arg1, ..., 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 arg1;      /* 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

This request operates as if a task on the remote system issues the request on behalf of the caller. The actual execution of 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 a 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 of *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.