3.2.5 Q_CREATE

NAME

q_create – "Create a Message Queue"

SYNOPSIS

```
#include <message.h>
uint q_create (name, count, flags, &qid )

        uint name;    /* user defined 4-byte name */
        uint count;   /* maximum message and reserved buffer count */
        uint flags;   /* process method */
        uint qid;     /* message queue id - returned by this call */
```

The flags values are:

PRIOR	set	to process by priority
	clear	to process by FIFO
GLOBAL	set	to indicate the queue is a multiprocessor global resource.
	clear	to indicate the queue is local
TYPE	set	to process typed messages
	clear	to process messages without regard to type
LIMIT	set	to limit queue entries to number in count field
	clear	NO limit on queue entries and no reserved buffers
RESVD	set	to reserve system buffers equal to count when LIMIT is set
	clear	NO reserved system buffers when LIMIT is set

DESCRIPTION

The *q_create* directive creates a message queue by allocating and initialising a message queue data structure. A message queue is created by name. A message *qid* is returned. Subsequent sending and receiving calls must reference the message queue with its message *qid*.

By setting the **PRIOR** value in the flags field, tasks waiting for messages in the queue will be processed by task priority order. Otherwise the tasks waiting for messages will be processed by first in, first out (FIFO) order.

By setting the **TYPE** value in the flags field, messages sent to this queue may be processed by type.

The user may put a limit on the number of messages at the message queue by setting the **LIMIT** value in the flags field, and placing the count in the *count* field. The user may additionally reserve a number of system message buffers equal to the count in the *count* field by setting the **RESVD** value in the flags field.

By setting the **GLOBAL** value in the flags field, the message *qid* 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 message queue is always created on the local node.

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

The maximum number of system message buffers is a configuration parameter.

RETURN VALUE

If the *q_create* directive succeeds, the *qid* is filled in, and 0 is returned.

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

ERROR CONDITIONS

Too many message queues.

No more system message buffers.

NOTES

Not callable from ISR.

Will not cause a preempt.

3.2.6 Q_IDENT

NAME

q_ident -- "Obtain id of a Message Queue"

SYNOPSIS

```
#include <message.h>
uint q_ident ( name, node, &qid )

        uint name;    /* user defined 4-byte name */
        uint node;    /* node identifier */
                      /* 0 indicates any node */
        uint qid;     /* message queue id - returned by this call */
```

DESCRIPTION

The *q_ident* directive allows a task to identify a previously created message queue by name and receive the message *qid* to use for send and receive directives for the queue.

If the message queue name is not unique, the message *qid* returned may not correspond to the message queue named in this call.

The message queue may exist on the local processor or any remote processor in a multiprocessor configuration, as long as the queue was created with the GLOBAL flags value set (see *q_create*). If the message queue name is not unique within the multiprocessor configuration, a non-zero node identifier must be specified in the *node* field.

RETURN VALUE

If the *q_ident* directive succeeds, the *qid* is filled in, and 0 is returned.

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

ERROR CONDITIONS

Named message queue does not exist.

Invalid node identifier.

NOTES

Can be called from within an ISR.

Will not cause a preempt.

3.2.7 Q_DELETE

NAME

q_delete – "Delete a Message Queue"

SYNOPSIS

```
#include <message.h>
uint q_delete ( qid )

        uint qid;    /* message queue id returned from q_create or q_ident */
```

DESCRIPTION

The *q_delete* directive deletes the message queue identified by the *qid*, freeing the data structure.

When a message queue is deleted, it could be in one of three states: empty, tasks waiting for messages, messages waiting for tasks. If empty, the data structure of the message queue is returned to the system. If tasks are waiting, each is made ready and given a return code indicating a deleted message queue. If messages are waiting, then each system message buffer is returned to the system message buffer pool, and the message it is carrying is therefore lost.

The message queue must exist on the local processor. If the message queue was created with the **GLOBAL** flags value set in a multiprocessor configuration, a notification will be sent to all processors in the system, so the *qid* can be deleted from the global resource table.

The requester does not have to be the creator of the message queue. Any task knowing the *qid* can delete it.

RETURN VALUE

If the *q_delete* directive successfully deleted the message queue, then 0 is returned.

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

ERROR CONDITIONS

Message *qid* is invalid.

Message queue not created from local node.

NOTES

Cannot be called from within an ISR.

May cause a preempt if a task waiting at the message queue has a higher priority than the running task, and the preempt mode is in effect. A preempt will not occur if all tasks waiting at the message queue exist on a remote processor in a multiprocessor configuration.

3.2.8 Q_SEND

NAME

q_send -- "Send a Message to a Message Queue"

SYNOPSIS

```
#include <message.h>
uint q_send ( qid, buffer )

        uint qid;           /* message queue id returned from q_create or q_ident */
        long (*buffer)[4];  /* pointer to message buffer */
```

DESCRIPTION

The *q_send* directive sends a message to the queue identified by the *qid*.

If a task is already waiting at the queue, the message is copied to that task's indicated receiving buffer. The waiting task is then made ready. If there is no task waiting, the message is copied to a system message buffer which is then placed at the end of the message queue.

Once sent, the task's message buffer may be reused immediately. A message is fixed length, 16-bytes.

The message queue may exist on the local processor or any remote processor in a multiprocessor configuration, as long as the queue was created with the GLOBAL flags value set (see *q_create*).

RETURN VALUE

If the *q_send* directive successfully sent a message, then 0 is returned.

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

ERROR CONDITIONS

Message *qid* is invalid.

Out of system message buffers.

Message queue at maximum count.

ISR cannot reference remote node.

NOTES

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

May cause a preempt if a task waiting at the message queue has a higher priority than the running task, and the preempt mode is in effect. A preempt will not occur if a task waiting exists on a remote processor in a multiprocessor configuration.