8.5. QUEUE_JUMP

Send a message to the head of a given queue.

Synopsis

```
queue_jump( qid, msg_buff, msg_length )
```

Input Parameters

qid	: queue_id	kernel defined queue identifier
msg_buff	: address	message starting address
msg_length	: integer	length of message in bytes

Output Parameters

<none>

Completion Status

OK	queue_jump successful
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	queue does not exist
OBJECT_DELETED	originally existing queue has been deleted before operation
INVALID_LENGTH	message length greater than queue's buffer length
QUEUE_FULL	no more buffers available
NODE_NOT_REACHABLE	node on which queue resides is not reachable

Description

This operations sends a message to the head of a queue.

If the queue's wait queue contains a number of tasks waiting on messages, then the message is delivered to the task at the head of the wait queue. This task is then removed from the wait queue, unblocked and will be returned a successful completion status along with the message. Otherwise the message is prepended at the head of the queue.

If the maximum queue length has been reached, then the QUEUE_FULL completion status is returned.

8.6. QUEUE_BROADCAST

Broadcast message to all tasks blocked on a queue.

Synopsis

```
queue_broadcast( qid, msg_buff, msg_length, count )
```

Input Parameters

qid	: queue_id	kernel defined queue identifier
msg_buff	: address	message starting address
msg_length	: integer	message length in bytes

Output Parameters

count	: integer	number of unblocked tasks

Completion Status

OK	queue_broadcast successful
ILLEGAL_USE	queue_broadcast not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	queue does not exist
OBJECT_DELETED	originally existing queue has been deleted before operation
INVALID_LENGTH	message length greater than queue's buffer length
NODE_NOT_REACHABLE	node on which queue resides is not reachable

Description

This operation sends a message to all tasks waiting on a queue.

If the wait queue is empty, then no messages are sent, no tasks are unblocked and the count passed back will be zero. If the wait queue contains a number of tasks waiting on messages, then the message is delivered to each task in the wait queue. All tasks are then removed from the wait queue, unblocked and returned a successful completion status. The number of tasks unblocked is passed back in the count parameter.

This operation is atomic with respect to other operations on the queue.

8.7. QUEUE_RECEIVE

Receive a message from a queue.

Synopsis

```
queue_receive( qid, msg_buff, buff_length, options, time_out,
               msg_length )
```

Input Parameters

qid	: queue_id	kernel defined queue identifier
msg_buff	: address	starting address of receive buffer
buff_length	: integer	length of receive buffer in bytes
options	: bit_field	queue receive options
time_out	: integer	ticks to wait before timing out

Output Parameters

msg_length	: integer	received message length in bytes

Literal Values

options	+ NOWAIT	do not wait - return immediately if no message in queue
time_out	= FOREVER	wait forever - do not time out

Completion Status

OK	queue_receive successful
ILLEGAL_USE	queue_receive not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	queue does not exist
OBJECT_DELETED	originally existing queue has been deleted before operation
INVALID_LENGTH	receive buffer smaller than queue's message buffer
INVALID_OPTIONS	invalid options value
TIME_OUT	queue-receive timed out
QUEUE_DELETED	queue deleted while blocked in queue_receive
QUEUE_EMPTY	queue empty with NOWAIT option
NODE_NOT_REACHABLE	node on which queue resides is not reachable

Description

This operation receives a message from a given queue. The operation first checks if the receive buffer is smaller than the queue's message buffer. If this is the case the INVALID_LENGTH completion status is returned.

Otherwise, if there are one or more messages on the queue, then the message at the head of the queue is removed and copied into the receive

buffer and a successful completion status returned.

If the message queue is empty, and NOWAIT was not specified in the options, then the task is blocked and put on the queue's wait queue. At that moment the time-out period is started. If the time-out expires then the TIME_OUT completion status is returned.

If NOWAIT was specified and the queue is empty, then the QUEUE_EMPTY completion status is returned.

If the queue is deleted while the task is waiting on a message from it, then the QUEUE_DELETED completion status is returned.

Otherwise, when the task reaches the head of the queue and a message is sent, or if a message is broadcast while the task is anywhere in the queue, then the task receives the message and is returned a successful completion status.

8.8. QUEUE_FLUSH

Flush all messages on a queue.

Synopsis

queue_flush(qid, count)

Input Parameters

qid	: queue_id	kernel defined queue identifier

Output Parameters

count	: integer	number of flushed messages

Completion Status

OK	queue_flush successful
ILLEGAL_USE	queue_flush not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	queue does not exist
OBJECT_DELETED	originally existing queue has been deleted before operation
NODE_NOT_REACHABLE	node on which queue resides is not reachable

Description

If there were one or more messages in the specified queue, then they are removed from the queue, their buffers deallocated and their number returned in count. If there were no messages in the queue, then a count of zero is returned.