CRITICAL SOFTWARE, SA
EN1/IC2, KM 185,6
BANHOS SECOS, SANTA CLARA
3040-032 COIMBRA, PORTUGAL
TEL. +351.239.801300, FAX. +351.239.801319
WWW.CRITICALSOFTWARE.COM

CRITICAL SOFTWARE, LIMITED
111 NORTH MARKET STREET, SUITE 670
SAN JOSE, CALIFORNIA, USA, 95113
TEL. +1(408) 9711231,
FAX: +1(408) 3513330

# RTEMS 4.5.0 Evaluation Report - Annex

## RAMS Call-off Order 2

**Contract Ref.:** *CSW-RAMS-2003-CTR-1306*

*ESTEC/Contract Nº 16582/02/NL/PA*



**Date:** *25-11-2003*
**Pages:** *51*
**State:** *Approved*
**Access:** *See Access List*
**Reference:** *DL-RAMS02-01-05*
*CSW-RAMS-2003-RPT-1334-05*

***Partners / Clients:***

# RTEMS 4.5.0 Evaluation Report - Annex

## RAMS Call-off Order 2

| Approved Version: 1.5 | | | |
|---|---|---|---|
| **Name** | **Function** | **Signature** | **Date** |
| Ricardo Maia | Project Manager | | 25-11-2003 |
| Jose Silva | SQA Engineer | | 25-11-2003 |

| Authors and Contributors: | | | |
|---|---|---|---|
| **Name** | **Contact** | **Description** | **Date** |
| Luís Henriques | lhenriques@criticalsoftware.com | Senior Engineer | 05-11-2003 |
| Ricardo Maia | rmaia@criticalsoftware.com | Project Manager | 25-11-2003 |

| Access List: |
|---|
| **Internal Access** |
| Project Team Members |
| **External Access** |
| ESA-ESTEC |

| Revision History: | | | |
|---|---|---|---|
| **Version** | **Date** | **Description** | **Author(s)** |
| 0.1 | 05-11-2003 | First issue | Luís Henriques |
| 0.2 | 07-11-2003 | Updated after internal review | Ricardo Maia |
| 1.0 | 25-11-2003 | Updated disclaimer. | Ricardo Maia |

# Table of Contents

This annex is part of the deliverable DL-RAMS02-01-05 of the Call-off Order number 02 under project Software Dependability and Safety Evaluations, ESTEC/Contract Nº 16582/02/NL/PA.

# Annex A. RTEMS Data Structures Definition

The following sections presents the data structures definitions as they are defined in the RTEMS source code. These are the data structures that shall be identified by the Xception RTEMS plug-in.

## A.1. itimerspec

```
struct timespec {
  time_t  tv_sec;   /* Seconds */
  long    tv_nsec;  /* Nanoseconds */
};
```

## A.2. mq_attr

```
struct mq_attr {
  long  mq_flags;     /* Message queue flags */
  long  mq_maxmsg;    /* Maximum number of messages */
  long  mq_msgsize;   /* Maximum message size */
  long  mq_curmsgs;   /* Number of messages currently queued */
};
```

## A.3. pthread_attr_t

```
typedef struct {
  int is_initialized;
  void *stackaddr;
  int stacksize;
  int contentionscope;
  int inheritsched;
  int schedpolicy;
  struct sched_param schedparam;

#if defined(_POSIX_THREAD_CPUTIME)
  int  cputime_clock_allowed;  /* see time.h */
#endif
  int  detachstate;

} pthread_attr_t;
```

## A.4. pthread_condattr_t

```
typedef struct {
  int   is_initialized;
#if defined(_POSIX_THREAD_PROCESS_SHARED)
  int   process_shared;
#endif
} pthread_condattr_t;
```

## A.5. pthread_mutexattr_t

```
typedef struct {
  int   is_initialized;
#if defined(_POSIX_THREAD_PROCESS_SHARED)
  int   process_shared;
 #endif
#if defined(_POSIX_THREAD_PRIO_PROTECT)
  int   prio_ceiling;
  int   protocol;
#endif
  int   recursive;
} pthread_mutexattr_t;
```

## A.6. pthread_once_t

```
typedef struct {
  int   is_initialized;
  int   init_executed;
 } pthread_once_t;
```

## A.7. rtems_time_of_day

Typedef'ed from:

- TOD_Control

```
typedef struct {                    /* RTEID style time/date */
  unsigned32 year;                  /* year, A.D. */
  unsigned32 month;                 /* month, 1 -> 12 */
  unsigned32 day;                   /* day, 1 -> 31 */
  unsigned32 hour;                  /* hour, 0 -> 23 */
  unsigned32 minute;                /* minute, 0 -> 59 */
  unsigned32 second;                /* second, 0 -> 59 */
  unsigned32 ticks;                 /* elapsed ticks between secs */
}   TOD_Control;
```

## A.8. sched_param

```
struct sched_param {
  int sched_priority;
#if defined(_POSIX_SPORADIC_SERVER)
  int ss_low_priority;
  struct timespec ss_replenish_period;
  struct timespec ss_initial_budget;
 #endif
};
```

## A.9. sigaction

```
struct sigaction {
  int         sa_flags;
  sigset_t    sa_mask;
  union {
```

```
    void      (*_handler)();
    void      (*_sigaction)( int, siginfo_t *, void * );
  } _signal_handlers;
};
```

## A.10. sigevent

```
struct sigevent {
  int              sigev_notify;
  int              sigev_signo;
  union sigval     sigev_value;
#if defined(_POSIX_THREADS)
  void           (*sigev_notify_function)( union sigval );
  pthread_attr_t  *sigev_notify_attributes;
#endif
};
```

## A.11. siginfo_t

```
typedef struct {
  int          si_signo;
  int          si_code;
  union sigval si_value;
} siginfo_t;
```

## A.12. timespec

```
struct timespec {
  time_t  tv_sec;
  long    tv_nsec;
};
```

## A.13. rtems_extensions_table

Typedef'ed from:

- `User_extensions_Table`

```
typedef struct {
  User_extensions_thread_create_extension       thread_create;
  User_extensions_thread_start_extension        thread_start;
  User_extensions_thread_restart_extension      thread_restart;
  User_extensions_thread_delete_extension       thread_delete;
  User_extensions_thread_switch_extension       thread_switch;
  User_extensions_thread_begin_extension        thread_begin;
  User_extensions_thread_exitted_extension      thread_exitted;
  User_extensions_fatal_extension               fatal;
}   User_extensions_Table;
```

# Annex B. Metrics

This annex presents the metrics collected during the Message Manager robustness testing, for the RTEMS Classic API workload.
The source code here presented belongs to the files that contain code executed during the following RTEMS directives:

- *rtems_message_queue_create;*
- *rtems_message_queue_delete;*
- *rtems_message_queue_send;*
- *rtems_message_queue_receive;*
- *rtems_message_queue_get_number_pending;*
- *rtems_message_queue_ident;*
- *rtems_message_queue_urgent;*
- *rtems_message_queue_flush;*
- *rtems_message_queue_broadcast.*

The code listed in the next sections are only those that contain error handling code. This error handling code is highlighted.
The numbers on the left are the number of executions a certain line was executed.

## B.1. chain.inl

```
0       /*  inline/chain.inl
0        *
0        *  This include file contains the bodies of the routines which are
0        *  associated with doubly linked chains and inlined.
0        *
0        *  NOTE:  The routines in this file are ordered from simple
0        *         to complex.  No other Chain Handler routine is referenced
0        *         unless it has already been defined.
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: chain.inl.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #ifndef __INLINE_CHAIN_inl
0       #define __INLINE_CHAIN_inl
0
0       /*PAGE
0        *
0        *  _Chain_Are_nodes_equal
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if LEFT and RIGHT are equal,
0        *  and FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Chain_Are_nodes_equal(
0         Chain_Node *left,
0         Chain_Node *right
0       )
0       {
0         return left == right;
0       }
0
```

```
0       /*PAGE
0        *
0        *  _Chain_Is_null
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the_chain is NULL and FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Chain_Is_null(
0         Chain_Control *the_chain
0       )
0       {
0         return ( the_chain == NULL );
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Is_null_node
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the_node is NULL and FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Chain_Is_null_node(
0         Chain_Node *the_node
0       )
0       {
0         return ( the_node == NULL );
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Head
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns a pointer to the first node on the chain.
0        */
0
0       RTEMS_INLINE_ROUTINE Chain_Node *_Chain_Head(
0         Chain_Control *the_chain
0       )
42      {
99         return (Chain_Node *) the_chain;
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Tail
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns a pointer to the last node on the chain.
0        */
0
0       RTEMS_INLINE_ROUTINE Chain_Node *_Chain_Tail(
0         Chain_Control *the_chain
0       )
0       {
106        return (Chain_Node *) &the_chain->permanent_null;
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Is_empty
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if there a no nodes on the_chain and
0        *  FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Chain_Is_empty(
0         Chain_Control *the_chain
0       )
```

```
0       {
2397      return ( the_chain->first == _Chain_Tail( the_chain ) );
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Is_first
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the_node is the first node on a chain and
0        *  FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Chain_Is_first(
0         Chain_Node *the_node
0       )
0       {
0         return ( the_node->previous == NULL );
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Is_last
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the_node is the last node on a chain and
0        *  FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Chain_Is_last(
0         Chain_Node *the_node
0       )
0       {
0         return ( the_node->next == NULL );
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Has_only_one_node
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if there is only one node on the_chain and
0        *  FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Chain_Has_only_one_node(
0         Chain_Control *the_chain
0       )
0       {
150       return ( the_chain->first == the_chain->last );
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Is_head
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the_node is the head of the_chain and
0        *  FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Chain_Is_head(
0         Chain_Control *the_chain,
0         Chain_Node    *the_node
0       )
0       {
0          return ( the_node == _Chain_Head( the_chain ) );
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Is_tail
0        *
```

```
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the_node is the tail of the_chain and
0        *  FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Chain_Is_tail(
0         Chain_Control *the_chain,
0         Chain_Node    *the_node
0       )
0       {
0          return ( the_node == _Chain_Tail( the_chain ) );
0       }
0
0       /*PAGE
0        *
0        *  Chain_Initialize_empty
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine initializes the specified chain to contain zero nodes.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Chain_Initialize_empty(
0         Chain_Control *the_chain
0       )
102     {
114       the_chain->first          = _Chain_Tail( the_chain );
108       the_chain->permanent_null = NULL;
108       the_chain->last           = _Chain_Head( the_chain );
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Extract_unprotected
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine extracts the_node from the chain on which it resides.
0        *  It does NOT disable interrupts to insure the atomicity of the
0        *  extract operation.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Chain_Extract_unprotected(
0         Chain_Node *the_node
0       )
0       {
0         Chain_Node *next;
0         Chain_Node *previous;
0
0         next           = the_node->next;
0         previous       = the_node->previous;
0         next->previous = previous;
0         previous->next = next;
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Get_first_unprotected
0        *
0        *  DESCRIPTION:
0        *
0        *  This function removes the first node from the_chain and returns
0        *  a pointer to that node.  It does NOT disable interrupts to insure
0        *  the atomicity of the get operation.
0        */
0
0       RTEMS_INLINE_ROUTINE Chain_Node *_Chain_Get_first_unprotected(
0         Chain_Control *the_chain
0       )
0       {
0         Chain_Node  *return_node;
0         Chain_Node  *new_first;
0
0         return_node         = the_chain->first;
0         new_first           = return_node->next;
91        the_chain->first    = new_first;
```

```
91        new_first->previous = _Chain_Head( the_chain );
0
0         return return_node;
0       }
0
0       /*PAGE
0        *
0        *  Chain_Get_unprotected
0        *
0        *  DESCRIPTION:
0        *
0        *  This function removes the first node from the_chain and returns
0        *  a pointer to that node.  If the_chain is empty, then NULL is returned.
0        *  It does NOT disable interrupts to insure the atomicity of the
0        *  get operation.
0        */
0
0       RTEMS_INLINE_ROUTINE Chain_Node *_Chain_Get_unprotected(
0         Chain_Control *the_chain
0       )
0       {
0         if ( !_Chain_Is_empty( the_chain ) )
0           return _Chain_Get_first_unprotected( the_chain );
0         else
0           return NULL;
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Insert_unprotected
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine inserts the_node on a chain immediately following
0        *  after_node.  It does NOT disable interrupts to insure the atomicity
0        *  of the extract operation.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Chain_Insert_unprotected(
0         Chain_Node *after_node,
0         Chain_Node *the_node
0       )
49      {
0         Chain_Node *before_node;
0
0         the_node->previous    = after_node;
0         before_node           = after_node->next;
0         after_node->next      = the_node;
0         the_node->next        = before_node;
0         before_node->previous = the_node;
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Append_unprotected
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine appends the_node onto the end of the_chain.
0        *  It does NOT disable interrupts to insure the atomicity of the
0        *  append operation.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Chain_Append_unprotected(
0         Chain_Control *the_chain,
0         Chain_Node    *the_node
0       )
198     {
0         Chain_Node *old_last_node;
0
0         the_node->next      = _Chain_Tail( the_chain );
0         old_last_node       = the_chain->last;
0         the_chain->last     = the_node;
0         old_last_node->next = the_node;
0         the_node->previous  = old_last_node;
0       }
0
```

```
0       /*PAGE
0        *
0        *  _Chain_Prepend_unprotected
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine prepends the_node onto the front of the_chain.
0        *  It does NOT disable interrupts to insure the atomicity of the
0        *  prepend operation.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Chain_Prepend_unprotected(
0         Chain_Control *the_chain,
0         Chain_Node    *the_node
0       )
0       {
0         _Chain_Insert_unprotected( _Chain_Head( the_chain ), the_node );
0
0       }
0
0       /*PAGE
0        *
0        *  _Chain_Prepend
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine prepends the_node onto the front of the_chain.
0        *  It disables interrupts to insure the atomicity of the
0        *  prepend operation.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Chain_Prepend(
0         Chain_Control *the_chain,
0         Chain_Node    *the_node
0       )
0       {
0         _Chain_Insert( _Chain_Head( the_chain ), the_node );
0       }
0
0       #endif
0       /* end of include file */
```

## B.2. coremsg.c

```
0       /*
0        *  CORE Message Queue Handler
0        *
0        *  DESCRIPTION:
0        *
0        *  This package is the implementation of the CORE Message Queue Handler.
0        *  This core object provides task synchronization and communication functions
0        *  via messages passed to queue objects.
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: coremsg.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
0       #endif
0
```

```
0       /*PAGE
0        *
0        *  _CORE_message_queue_Initialize
0        *
0        *  This routine initializes a newly created message queue based on the
0        *  specified data.
0        *
0        *  Input parameters:
0        *    the_message_queue            - the message queue to initialize
0        *    the_class                    - the API specific object class
0        *    the_message_queue_attributes - the message queue's attributes
0        *    maximum_pending_messages     - maximum message and reserved buffer count
0        *    maximum_message_size         - maximum size of each message
0        *    proxy_extract_callout        - remote extract support
0        *
0        *  Output parameters:
0        *    TRUE   - if the message queue is initialized
0        *    FALSE  - if the message queue is NOT initialized
0        */
0
0       boolean _CORE_message_queue_Initialize(
0         CORE_message_queue_Control    *the_message_queue,
0         Objects_Classes                the_class,
0         CORE_message_queue_Attributes *the_message_queue_attributes,
0         unsigned32                     maximum_pending_messages,
0         unsigned32                     maximum_message_size,
0         Thread_queue_Extract_callout   proxy_extract_callout
0       )
0       {
0         unsigned32 message_buffering_required;
0         unsigned32 allocated_message_size;
0
101       the_message_queue->maximum_pending_messages   = maximum_pending_messages;
101       the_message_queue->number_of_pending_messages = 0;
101       the_message_queue->maximum_message_size       = maximum_message_size;
0         _CORE_message_queue_Set_notify( the_message_queue, NULL, NULL );
0
0         /*
0          * round size up to multiple of a ptr for chain init
0          */
0
101       allocated_message_size = maximum_message_size;
101       if (allocated_message_size & (sizeof(unsigned32) - 1)) {
2             allocated_message_size += sizeof(unsigned32);
2             allocated_message_size &= ~(sizeof(unsigned32) - 1);
0         }
0
101       message_buffering_required = maximum_pending_messages *
0              (allocated_message_size + sizeof(CORE_message_queue_Buffer_control));
0
0         the_message_queue->message_buffers = (CORE_message_queue_Buffer *)
0            _Workspace_Allocate( message_buffering_required );
0
101       if (the_message_queue->message_buffers == 0)
2           return FALSE;
0
99        _Chain_Initialize (
0           &the_message_queue->Inactive_messages,
0           the_message_queue->message_buffers,
0           maximum_pending_messages,
0           allocated_message_size + sizeof( CORE_message_queue_Buffer_control )
0         );
0
0         _Chain_Initialize_empty( &the_message_queue->Pending_messages );
0
0         _Thread_queue_Initialize(
0           &the_message_queue->Wait_queue,
0           the_class,
0           _CORE_message_queue_Is_priority( the_message_queue_attributes ) ?
0              THREAD_QUEUE_DISCIPLINE_PRIORITY : THREAD_QUEUE_DISCIPLINE_FIFO,
0           STATES_WAITING_FOR_MESSAGE,
0           proxy_extract_callout,
0           CORE_MESSAGE_QUEUE_STATUS_TIMEOUT
99        );
0
98        return TRUE;
100     }
```

## B.3. coremsgseize.c

```
0       /*
0        *  CORE Message Queue Handler
0        *
0        *  DESCRIPTION:
0        *
0        *  This package is the implementation of the CORE Message Queue Handler.
0        *  This core object provides task synchronization and communication functions
0        *  via messages passed to queue objects.
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: coremsgseize.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
0       #endif
0
0       /*PAGE
0        *
0        *  _CORE_message_queue_Seize
0        *
0        *  This kernel routine dequeues a message, copies the message buffer to
0        *  a given destination buffer, and frees the message buffer to the
0        *  inactive message pool.  The thread will be blocked if wait is TRUE,
0        *  otherwise an error will be given to the thread if no messages are available.
0        *
0        *  Input parameters:
0        *    the_message_queue - pointer to message queue
0        *    id                - id of object we are waitig on
0        *    buffer            - pointer to message buffer to be filled
0        *    size              - pointer to the size of buffer to be filled
0        *    wait              - TRUE if wait is allowed, FALSE otherwise
0        *    timeout           - time to wait for a message
0        *
0        *  Output parameters:  NONE
0        *
0        *  NOTE: Dependent on BUFFER_LENGTH
0        *
0        *  INTERRUPT LATENCY:
0        *    available
0        *    wait
0        */
0
0       void _CORE_message_queue_Seize(
0         CORE_message_queue_Control      *the_message_queue,
0         Objects_Id                       id,
0         void                            *buffer,
0         unsigned32                      *size,
0         boolean                          wait,
0         Watchdog_Interval                timeout
0       )
0       {
0         ISR_Level                          level;
0         CORE_message_queue_Buffer_control *the_message;
0         Thread_Control                    *executing;
0         Thread_Control                    *the_thread;
0
124       executing = _Thread_Executing;
0         executing->Wait.return_code = CORE_MESSAGE_QUEUE_STATUS_SUCCESSFUL;
124       _ISR_Disable( level );
```

```
248       if ( the_message_queue->number_of_pending_messages != 0 ) {
19          the_message_queue->number_of_pending_messages -= 1;
0
0           the_message = _CORE_message_queue_Get_pending_message( the_message_queue );
19          _ISR_Enable( level );
0
19          *size = the_message->Contents.size;
19          _Thread_Executing->Wait.count = the_message->priority;
0           _CORE_message_queue_Copy_buffer(the_message->Contents.buffer,buffer,*size);
0
0           /*
0            *  There could be a thread waiting to send a message.  If there
0            *  is not, then we can go ahead and free the buffer.
0            *
0            *  NOTE: If we note that the queue was not full before this receive,
0            *  then we can avoid this dequeue.
0            */
0
19          the_thread = _Thread_queue_Dequeue( &the_message_queue->Wait_queue );
19          if ( !the_thread ) {
0             _CORE_message_queue_Free_message_buffer( the_message_queue, the_message );
19            return;
0           }
0
0           /*
0            *  There was a thread waiting to send a message.  This code
0            *  puts the messages in the message queue on behalf of the
0            *  waiting task.
0            */
0
0           the_message->priority  = the_thread->Wait.count;
0           the_message->Contents.size = (unsigned32)the_thread->Wait.return_argument_1;
0           _CORE_message_queue_Copy_buffer(
0             the_thread->Wait.return_argument,
0             the_message->Contents.buffer,
0             the_message->Contents.size
0           );
0
0           _CORE_message_queue_Insert_message(
0              the_message_queue,
0              the_message,
0              the_message->priority
0           );
0           return;
0         }
0
105       if ( !wait ) {
2           _ISR_Enable( level );
2           executing->Wait.return_code = CORE_MESSAGE_QUEUE_STATUS_UNSATISFIED_NOWAIT;
2           return;
0         }
0
0         _Thread_queue_Enter_critical_section( &the_message_queue->Wait_queue );
0         executing->Wait.queue             = &the_message_queue->Wait_queue;
103       executing->Wait.id                = id;
103       executing->Wait.return_argument   = (void *)buffer;
103       executing->Wait.return_argument_1 = (void *)size;
0         /* Wait.count will be filled in with the message priority */
206       _ISR_Enable( level );
0
103       _Thread_queue_Enqueue( &the_message_queue->Wait_queue, timeout );
124     }
0
```

## B.4. coremsgsubmit.c

```
0       /*
0        *  CORE Message Queue Handler
0        *
0        *  DESCRIPTION:
0        *
0        *  This package is the implementation of the CORE Message Queue Handler.
0        *  This core object provides task synchronization and communication functions
0        *  via messages passed to queue objects.
0        *
```

```
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: coremsgsubmit.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
0       #endif
0
0       /*PAGE
0        *
0        *  _CORE_message_queue_Submit
0        *
0        *  This routine implements the send and urgent message functions. It
0        *  processes a message that is to be submitted to the designated
0        *  message queue.  The message will either be processed as a
0        *  send message which it will be inserted at the rear of the queue
0        *  or it will be processed as an urgent message which will be inserted
0        *  at the front of the queue.
0        *
0        *  Input parameters:
0        *    the_message_queue            - message is submitted to this message queue
0        *    buffer                       - pointer to message buffer
0        *    size                         - size in bytes of message to send
0        *    id                           - id of message queue
0        *    api_message_queue_mp_support - api specific mp support callout
0        *    submit_type                  - send or urgent message
0        *
0        *  Output parameters:
0        *    CORE_MESSAGE_QUEUE_SUCCESSFUL - if successful
0        *    error code                    - if unsuccessful
0        */
0
0       void _CORE_message_queue_Submit(
0         CORE_message_queue_Control                *the_message_queue,
0         void                                      *buffer,
0         unsigned32                                 size,
0         Objects_Id                                 id,
0         CORE_message_queue_API_mp_support_callout  api_message_queue_mp_support,
0         CORE_message_queue_Submit_types            submit_type,
0         boolean                                    wait,
0         Watchdog_Interval                          timeout
0       )
0       {
0         ISR_Level                          level;
0         CORE_message_queue_Buffer_control   *the_message;
0         Thread_Control                      *the_thread;
0         Thread_Control                      *executing;
0
179       _Thread_Executing->Wait.return_code = CORE_MESSAGE_QUEUE_STATUS_SUCCESSFUL;
0
180       if ( size > the_message_queue->maximum_message_size ) {
0           _Thread_Executing->Wait.return_code =
0              CORE_MESSAGE_QUEUE_STATUS_INVALID_SIZE;
3           return;
0         }
0
0         /*
0          *  Is there a thread currently waiting on this message queue?
0          */
0
86        if ( the_message_queue->number_of_pending_messages == 0 ) {
45          the_thread = _Thread_queue_Dequeue( &the_message_queue->Wait_queue );
45          if ( the_thread ) {
```

```
0             _CORE_message_queue_Copy_buffer(
0               buffer,
0               the_thread->Wait.return_argument,
0               size
0             );
0             *(unsigned32 *)the_thread->Wait.return_argument_1 = size;
0             the_thread->Wait.count = submit_type;
0
0       #if defined(RTEMS_MULTIPROCESSING)
0             if ( !_Objects_Is_local_id( the_thread->Object.id ) )
0               (*api_message_queue_mp_support) ( the_thread, id );
0       #endif
0             return;
0           }
0         }
0
0         /*
0          *  No one waiting on the message queue at this time, so attempt to
0          *  queue the message up for a future receive.
0          */
0
45        if ( the_message_queue->number_of_pending_messages <
0              the_message_queue->maximum_pending_messages ) {
0
0           the_message =
0               _CORE_message_queue_Allocate_message_buffer( the_message_queue );
0
0           /*
0            *  NOTE: If the system is consistent, this error should never occur.
0            */
85          if ( !the_message ) {
0             _Thread_Executing->Wait.return_code =
0                 CORE_MESSAGE_QUEUE_STATUS_UNSATISFIED;
0             return;
0           }
0
0           _CORE_message_queue_Copy_buffer(
0             buffer,
0             the_message->Contents.buffer,
0             size
0           );
85          the_message->Contents.size = size;
0           the_message->priority  = submit_type;
0
85          _CORE_message_queue_Insert_message(
0             the_message_queue,
0             the_message,
0             submit_type
0           );
85          return;
0         }
0
0         /*
0          *  No message buffers were available so we may need to return an
0          *  overflow error or block the sender until the message is placed
0          *  on the queue.
0          */
0
1         if ( !wait ) {
1           _Thread_Executing->Wait.return_code = CORE_MESSAGE_QUEUE_STATUS_TOO_MANY;
1           return;
0         }
0
0         executing = _Thread_Executing;
0
0         _ISR_Disable( level );
0         _Thread_queue_Enter_critical_section( &the_message_queue->Wait_queue );
0         executing->Wait.queue             = &the_message_queue->Wait_queue;
0         executing->Wait.id                = id;
0         executing->Wait.return_argument   = (void *)buffer;
0         executing->Wait.return_argument_1 = (void *)size;
0         executing->Wait.count             = submit_type;
0         _ISR_Enable( level );
0
0         _Thread_queue_Enqueue( &the_message_queue->Wait_queue, timeout );
89      }
```

## B.5. heapallocate.c

```
0       /*
0        *  Heap Handler
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: heapallocate.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0
0       #include <rtems/system.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/heap.h>
0
0       /*PAGE
0        *
0        *  _Heap_Allocate
0        *
0        *  This kernel routine allocates the requested size of memory
0        *  from the specified heap.
0        *
0        *  Input parameters:
0        *    the_heap  - pointer to heap header.
0        *    size      - size in bytes of the memory block to allocate.
0        *
0        *  Output parameters:
0        *    returns - starting address of memory block allocated
0        */
0
0       void *_Heap_Allocate(
0         Heap_Control        *the_heap,
0         unsigned32           size
0       )
0       {
0         unsigned32  excess;
0         unsigned32  the_size;
0         Heap_Block *the_block;
0         Heap_Block *next_block;
0         Heap_Block *temporary_block;
0         void       *ptr;
0         unsigned32  offset;
0
673       excess   = size % the_heap->page_size;
673       the_size = size + the_heap->page_size + HEAP_BLOCK_USED_OVERHEAD;
0
673       if ( excess )
208         the_size += the_heap->page_size - excess;
0
673       if ( the_size < sizeof( Heap_Block ) )
0           the_size = sizeof( Heap_Block );
0
673       for ( the_block = the_heap->first;
0               ;
2               the_block = the_block->next ) {
675         if ( the_block == _Heap_Tail( the_heap ) )
0             return( NULL );
673         if ( the_block->front_flag >= the_size )
0             break;
0         }
0
671       if ( (the_block->front_flag - the_size) >
0              (the_heap->page_size + HEAP_BLOCK_USED_OVERHEAD) ) {
671         the_block->front_flag -= the_size;
0           next_block             = _Heap_Next_block( the_block );
671         next_block->back_flag  = the_block->front_flag;
0
0           temporary_block            = _Heap_Block_at( next_block, the_size );
0           temporary_block->back_flag =
0           next_block->front_flag     = _Heap_Build_flag( the_size,
671                                         HEAP_BLOCK_USED );
```

```
0           ptr = _Heap_Start_of_user_area( next_block );
671       } else {
0           next_block                = _Heap_Next_block( the_block );
0           next_block->back_flag     = _Heap_Build_flag( the_block->front_flag,
0                                         HEAP_BLOCK_USED );
0           the_block->front_flag     = next_block->back_flag;
0           the_block->next->previous = the_block->previous;
0           the_block->previous->next = the_block->next;
0           ptr = _Heap_Start_of_user_area( the_block );
0         }
0
0         /*
0          * round ptr up to a multiple of page size
0          * Have to save the bump amount in the buffer so that free can figure it out
0          */
0
671       offset = the_heap->page_size - (((unsigned32) ptr) & (the_heap->page_size - 1));
0         ptr = _Addresses_Add_offset( ptr, offset );
671       *(((unsigned32 *) ptr) - 1) = offset;
0
0       #ifdef RTEMS_DEBUG
0         {
0             unsigned32 ptr_u32;
0             ptr_u32 = (unsigned32) ptr;
0             if (ptr_u32 & (the_heap->page_size - 1))
0                 abort();
0         }
0       #endif
0
0         return ptr;
673     }
0
```

## B.6. heapfree.c

```
0       /*
0        *  Heap Handler
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: heapfree.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0
0       #include <rtems/system.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/heap.h>
0
0       /*PAGE
0        *
0        *  _Heap_Free
0        *
0        *  This kernel routine returns the memory designated by the
0        *  given heap and given starting address to the memory pool.
0        *
0        *  Input parameters:
0        *    the_heap         - pointer to heap header
0        *    starting_address - starting address of the memory block to free.
0        *
0        *  Output parameters:
0        *    TRUE  - if starting_address is valid heap address
0        *    FALSE - if starting_address is invalid heap address
0        */
0
0       boolean _Heap_Free(
0         Heap_Control        *the_heap,
0         void                *starting_address
0       )
145     {
0         Heap_Block        *the_block;
```

```
0         Heap_Block        *next_block;
0         Heap_Block        *new_next_block;
0         Heap_Block        *previous_block;
0         Heap_Block        *temporary_block;
0         unsigned32         the_size;
0
0         the_block = _Heap_User_block_at( starting_address );
0
0         if ( !_Heap_Is_block_in( the_heap, the_block ) ||
145            _Heap_Is_block_free( the_block ) ) {
0             return( FALSE );
145       }
0
0         the_size   = _Heap_Block_size( the_block );
0         next_block = _Heap_Block_at( the_block, the_size );
0
0         if ( !_Heap_Is_block_in( the_heap, next_block ) ||
145            (the_block->front_flag != next_block->back_flag) ) {
0             return( FALSE );
145       }
0
145       if ( _Heap_Is_previous_block_free( the_block ) ) {
0           previous_block = _Heap_Previous_block( the_block );
0
41          if ( !_Heap_Is_block_in( the_heap, previous_block ) ) {
0               return( FALSE );
41          }
0
41          if ( _Heap_Is_block_free( next_block ) ) {      /* coalesce both */
0             previous_block->front_flag += next_block->front_flag + the_size;
0             temporary_block             = _Heap_Next_block( previous_block );
0             temporary_block->back_flag  = previous_block->front_flag;
0             next_block->next->previous  = next_block->previous;
0             next_block->previous->next  = next_block->next;
0           }
0           else {                       /* coalesce prev */
41            previous_block->front_flag =
0             next_block->back_flag      = previous_block->front_flag + the_size;
0           }
41        }
104       else if ( _Heap_Is_block_free( next_block ) ) { /* coalesce next */
24          the_block->front_flag     = the_size + next_block->front_flag;
0           new_next_block            = _Heap_Next_block( the_block );
24          new_next_block->back_flag = the_block->front_flag;
24          the_block->next           = next_block->next;
24          the_block->previous       = next_block->previous;
24          next_block->previous->next = the_block;
24          next_block->next->previous = the_block;
0
24          if (the_heap->first == next_block)
0               the_heap->first = the_block;
24        }
0         else {                           /* no coalesce */
80          next_block->back_flag     =
0           the_block->front_flag     = the_size;
80          the_block->previous       = _Heap_Head( the_heap );
80          the_block->next           = the_heap->first;
80          the_heap->first           = the_block;
80          the_block->next->previous = the_block;
0         }
0
121       return( TRUE );
145     }
0
```

## B.7. msgqbroadcast.c

```
0       /*
0        *  Message Queue Manager
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
```

```
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: msgqbroadcast.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
0       #endif
0       #include <rtems/rtems/status.h>
0       #include <rtems/rtems/attr.h>
0       #include <rtems/rtems/message.h>
0       #include <rtems/rtems/options.h>
0       #include <rtems/rtems/support.h>
0
0       /*PAGE
0        *
0        *  rtems_message_queue_broadcast
0        *
0        *  This directive sends a message for every thread waiting on the queue
0        *  designated by id.
0        *
0        *  Input parameters:
0        *    id     - pointer to message queue
0        *    buffer - pointer to message buffer
0        *    size   - size of message to broadcast
0        *    count  - pointer to area to store number of threads made ready
0        *
0        *  Output parameters:
0        *    count              - number of threads made ready
0        *    RTEMS_SUCCESSFUL   - if successful
0        *    error code         - if unsuccessful
0        */
0
0       rtems_status_code rtems_message_queue_broadcast(
0         Objects_Id            id,
0         void                 *buffer,
0         unsigned32            size,
0         unsigned32           *count
0       )
0       {
0         register Message_queue_Control *the_message_queue;
0         Objects_Locations               location;
0         CORE_message_queue_Status       core_status;
0
0         the_message_queue = _Message_queue_Get( id, &location );
47        switch ( location ) {
0           case OBJECTS_REMOTE:
0       #if defined(RTEMS_MULTIPROCESSING)
0             _Thread_Executing->Wait.return_argument = count;
0
0             return
0               _Message_queue_MP_Send_request_packet(
0                 MESSAGE_QUEUE_MP_BROADCAST_REQUEST,
0                 id,
0                 buffer,
0                 &size,
0                 0,                                /* option_set not used */
0                 MPCI_DEFAULT_TIMEOUT
0               );
0       #endif
0
0           case OBJECTS_ERROR:
14            return RTEMS_INVALID_ID;
0
0           case OBJECTS_LOCAL:
33            core_status = _CORE_message_queue_Broadcast(
0                           &the_message_queue->message_queue,
```

```
0                           buffer,
0                           size,
0                           id,
0       #if defined(RTEMS_MULTIPROCESSING)
0                           _Message_queue_Core_message_queue_mp_support,
0       #else
0                           NULL,
0       #endif
0                           count
0                         );
0
0             _Thread_Enable_dispatch();
32            return
0               _Message_queue_Translate_core_message_queue_return_code( core_status );
0
0           }
0           return RTEMS_INTERNAL_ERROR;   /* unreached - only to remove warnings */

46      }
```

## B.8. msgqcreate.c

```
0       /*
0        *  Message Queue Manager
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: msgqcreate.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
0       #endif
0       #include <rtems/rtems/status.h>
0       #include <rtems/rtems/attr.h>
0       #include <rtems/rtems/message.h>
0       #include <rtems/rtems/options.h>
0       #include <rtems/rtems/support.h>
0
0       /*PAGE
0        *
0        *  rtems_message_queue_create
0        *
0        *  This directive creates a message queue by allocating and initializing
0        *  a message queue data structure.
0        *
0        *  Input parameters:
0        *    name             - user defined queue name
0        *    count            - maximum message and reserved buffer count
0        *    max_message_size - maximum size of each message
0        *    attribute_set    - process method
0        *    id               - pointer to queue
0        *
0        *  Output parameters:
0        *    id                - queue id
0        *    RTEMS_SUCCESSFUL  - if successful
0        *    error code        - if unsuccessful
0        */
0
0       rtems_status_code rtems_message_queue_create(
```

```
0         rtems_name          name,
0         unsigned32          count,
0         unsigned32          max_message_size,
0         rtems_attribute     attribute_set,
0         Objects_Id         *id
0       )
0       {
0         register Message_queue_Control *the_message_queue;
0         CORE_message_queue_Attributes   the_msgq_attributes;
0         void                           *handler;
0       #if defined(RTEMS_MULTIPROCESSING)
0         boolean                         is_global;
0       #endif
0
106       if ( !rtems_is_name_valid( name ) )

1           return RTEMS_INVALID_NAME;

0
0       #if defined(RTEMS_MULTIPROCESSING)

0         if ( (is_global = _Attributes_Is_global( attribute_set ) ) &&
0              !_System_state_Is_multiprocessing )
0           return RTEMS_MP_NOT_CONFIGURED;
0       #endif

0
105       if ( count == 0 )

2             return RTEMS_INVALID_NUMBER;
0
103       if ( max_message_size == 0 )
2             return RTEMS_INVALID_SIZE;
0
0       #if defined(RTEMS_MULTIPROCESSING)
0       #if 1
0         /*
0          * I am not 100% sure this should be an error.
0          * It seems reasonable to create a que with a large max size,
0          * and then just send smaller msgs from remote (or all) nodes.
0          */
0
0         if ( is_global && (_MPCI_table->maximum_packet_size < max_message_size) )
0           return RTEMS_INVALID_SIZE;
0       #endif
0       #endif
0
0         _Thread_Disable_dispatch();              /* protects object pointer */
0
101       the_message_queue = _Message_queue_Allocate( count, max_message_size );
0
202       if ( !the_message_queue ) {
0           _Thread_Enable_dispatch();
0           return RTEMS_TOO_MANY;
0         }
0
0       #if defined(RTEMS_MULTIPROCESSING)
0         if ( is_global &&
0           !( _Objects_MP_Allocate_and_open( &_Message_queue_Information,
0                                        name, the_message_queue->Object.id, FALSE ) ) ) {
0           _Message_queue_Free( the_message_queue );
0           _Thread_Enable_dispatch();
0           return RTEMS_TOO_MANY;
0         }
0       #endif
0
0         the_message_queue->attribute_set = attribute_set;
0
101       if (_Attributes_Is_priority( attribute_set ) )
2           the_msgq_attributes.discipline = CORE_MESSAGE_QUEUE_DISCIPLINES_PRIORITY;
0         else
99          the_msgq_attributes.discipline = CORE_MESSAGE_QUEUE_DISCIPLINES_FIFO;
0


0         handler = NULL;
0       #if defined(RTEMS_MULTIPROCESSING)
0         handler = _Message_queue_MP_Send_extract_proxy;
```

```
0       #endif
0
101       if ( ! _CORE_message_queue_Initialize(
0                   &the_message_queue->message_queue,
0                   OBJECTS_RTEMS_MESSAGE_QUEUES,
0                   &the_msgq_attributes,
0                   count,
0                   max_message_size,
0                   handler ) ) {
0       #if defined(RTEMS_MULTIPROCESSING)
0           if ( is_global )
0               _Objects_MP_Close(
0                 &_Message_queue_Information, the_message_queue->Object.id);
0       #endif
0
0           _Message_queue_Free( the_message_queue );
0           _Thread_Enable_dispatch();
2           return RTEMS_TOO_MANY;
98        }
0
0         _Objects_Open(
0           &_Message_queue_Information,
0           &the_message_queue->Object,
0           &name
0         );
0
98        *id = the_message_queue->Object.id;
0
0       #if defined(RTEMS_MULTIPROCESSING)
0         if ( is_global )
0           _Message_queue_MP_Send_process_packet(
0             MESSAGE_QUEUE_MP_ANNOUNCE_CREATE,
0             the_message_queue->Object.id,
0             name,
0             0
0           );
0       #endif
0
0         _Thread_Enable_dispatch();
0         return RTEMS_SUCCESSFUL;
105     }
```

## B.9. msgqdelete.c

```
0       /*
0        *  Message Queue Manager
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: msgqdelete.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
0       #endif
0       #include <rtems/rtems/status.h>
0       #include <rtems/rtems/attr.h>
0       #include <rtems/rtems/message.h>
0       #include <rtems/rtems/options.h>
0       #include <rtems/rtems/support.h>
```

```
0
0       /*PAGE
0        *
0        *  rtems_message_queue_delete
0        *
0        *  This directive allows a thread to delete the message queue specified
0        *  by the given queue identifier.
0        *
0        *  Input parameters:
0        *    id - queue id
0        *
0        *  Output parameters:
0        *    RTEMS_SUCCESSFUL - if successful
0        *    error code       - if unsuccessful
0        */
0
0       rtems_status_code rtems_message_queue_delete(
0         Objects_Id id
0       )
0       {
84        register Message_queue_Control *the_message_queue;
0         Objects_Locations               location;
0
0         the_message_queue = _Message_queue_Get( id, &location );
84        switch ( location ) {
0
0           case OBJECTS_REMOTE:
0       #if defined(RTEMS_MULTIPROCESSING)
0             _Thread_Dispatch();
0             return RTEMS_ILLEGAL_ON_REMOTE_OBJECT;
0       #endif
0
0           case OBJECTS_ERROR:
21            return RTEMS_INVALID_ID;
0
0           case OBJECTS_LOCAL:
0             _Objects_Close( &_Message_queue_Information,
0                             &the_message_queue->Object );
0
63            _CORE_message_queue_Close(
0               &the_message_queue->message_queue,
0       #if defined(RTEMS_MULTIPROCESSING)
0               _Message_queue_MP_Send_object_was_deleted,
0       #else
0               NULL,
0       #endif
0               CORE_MESSAGE_QUEUE_STATUS_WAS_DELETED
0             );
0
0             _Message_queue_Free( the_message_queue );
0
0       #if defined(RTEMS_MULTIPROCESSING)
0             if ( _Attributes_Is_global( the_message_queue->attribute_set ) ) {
0               _Objects_MP_Close(
0                 &_Message_queue_Information,
0                 the_message_queue->Object.id
0               );
0
0               _Message_queue_MP_Send_process_packet(
0                 MESSAGE_QUEUE_MP_ANNOUNCE_DELETE,
0                 the_message_queue->Object.id,
0                 0,                                /* Not used */
0                 0
0               );
0             }
0       #endif
0
0             _Thread_Enable_dispatch();
63            return RTEMS_SUCCESSFUL;
0         }
0
0         return RTEMS_INTERNAL_ERROR;   /* unreached - only to remove warnings */
84      }
```

## B.10. msgqflush.c

```
0       /*
0        *  Message Queue Manager
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: msgqflush.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
0       #endif
0       #include <rtems/rtems/status.h>
0       #include <rtems/rtems/attr.h>
0       #include <rtems/rtems/message.h>
0       #include <rtems/rtems/options.h>
0       #include <rtems/rtems/support.h>
0
0       /*PAGE
0        *
0        *  rtems_message_queue_flush
0        *
0        *  This directive removes all pending messages from a queue and returns
0        *  the number of messages removed.  If no messages were present then
0        *  a count of zero is returned.
0        *
0        *  Input parameters:
0        *    id    - queue id
0        *    count - return area for count
0        *
0        *  Output parameters:
0        *    count             - number of messages removed ( 0 = empty queue )
0        *    RTEMS_SUCCESSFUL - if successful
0        *    error code        - if unsuccessful
0        */
0
0       rtems_status_code rtems_message_queue_flush(
0         Objects_Id  id,
0         unsigned32 *count
0       )
0       {
0         register Message_queue_Control *the_message_queue;
0         Objects_Locations               location;
0
0         the_message_queue = _Message_queue_Get( id, &location );
49        switch ( location ) {
0           case OBJECTS_REMOTE:
0       #if defined(RTEMS_MULTIPROCESSING)
0             _Thread_Executing->Wait.return_argument = count;
0
0             return
0               _Message_queue_MP_Send_request_packet(
0                 MESSAGE_QUEUE_MP_FLUSH_REQUEST,
0                 id,
0                 0,                                /* buffer not used */
0                 0,                                /* size */
0                 0,                                /* option_set not used */
0                 MPCI_DEFAULT_TIMEOUT
0               );
0       #endif
```

```
0
0           case OBJECTS_ERROR:
12            return RTEMS_INVALID_ID;
0
0           case OBJECTS_LOCAL:
35            *count = _CORE_message_queue_Flush( &the_message_queue->message_queue );
0             _Thread_Enable_dispatch();
35            return RTEMS_SUCCESSFUL;
0         }
0
0         return RTEMS_INTERNAL_ERROR;   /* unreached - only to remove warnings */
47      }
```

## B.11. msgqgetnumberpending.c

```
0       /*
0        *  Message Queue Manager
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: msgqgetnumberpending.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
0       #endif
0       #include <rtems/rtems/status.h>
0       #include <rtems/rtems/attr.h>
0       #include <rtems/rtems/message.h>
0       #include <rtems/rtems/options.h>
0       #include <rtems/rtems/support.h>
0
0       /*PAGE
0        *
0        *  rtems_message_queue_get_number_pending
0        *
0        *  This directive returns the number of messages pending.
0        *
0        *  Input parameters:
0        *    id    - queue id
0        *    count - return area for count
0        *
0        *  Output parameters:
0        *    count             - number of messages removed ( 0 = empty queue )
0        *    RTEMS_SUCCESSFUL - if successful
0        *    error code        - if unsuccessful
0        */
0
0       rtems_status_code rtems_message_queue_get_number_pending(
0         Objects_Id  id,
0         unsigned32 *count
0       )
0       {
0         register Message_queue_Control *the_message_queue;
0         Objects_Locations               location;
0
0         the_message_queue = _Message_queue_Get( id, &location );
49        switch ( location ) {
0           case OBJECTS_REMOTE:
```

```
0       #if defined(RTEMS_MULTIPROCESSING)
0             _Thread_Executing->Wait.return_argument = count;
0
0             return _Message_queue_MP_Send_request_packet(
0                 MESSAGE_QUEUE_MP_GET_NUMBER_PENDING_REQUEST,
0                 id,
0                 0,                                /* buffer not used */
0                 0,                                /* size */
0                 0,                                /* option_set not used */
0                 MPCI_DEFAULT_TIMEOUT
0               );
0       #endif
0
0           case OBJECTS_ERROR:
10            return RTEMS_INVALID_ID;
0
0           case OBJECTS_LOCAL:
77            *count = the_message_queue->message_queue.number_of_pending_messages;
0             _Thread_Enable_dispatch();
38            return RTEMS_SUCCESSFUL;
0         }
0
0         return RTEMS_INTERNAL_ERROR;   /* unreached - only to remove warnings */
48      }
```

## B.12. msgqreceive.c

```
0       /*
0        *  Message Queue Manager
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: msgqreceive.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
0       #endif
0       #include <rtems/rtems/status.h>
0       #include <rtems/rtems/attr.h>
0       #include <rtems/rtems/message.h>
0       #include <rtems/rtems/options.h>
0       #include <rtems/rtems/support.h>
0
0       /*PAGE
0        *
0        *  rtems_message_queue_receive
0        *
0        *  This directive dequeues a message from the designated message queue
0        *  and copies it into the requesting thread's buffer.
0        *
0        *  Input parameters:
0        *    id         - queue id
0        *    buffer     - pointer to message buffer
0        *    size       - size of message receive
0        *    option_set - options on receive
0        *    timeout    - number of ticks to wait
0        *
0        *  Output parameters:
0        *    RTEMS_SUCCESSFUL - if successful
```

```
0        *    error code       - if unsuccessful
0        */
0
0       rtems_status_code rtems_message_queue_receive(
0         Objects_Id            id,
0         void                 *buffer,
0         unsigned32           *size,
0         unsigned32            option_set,
0         rtems_interval        timeout
0       )
0       {
0         register Message_queue_Control *the_message_queue;
0         Objects_Locations               location;
0         boolean                        wait;
0
0         the_message_queue = _Message_queue_Get( id, &location );
139       switch ( location ) {
0
0           case OBJECTS_REMOTE:
0       #if defined(RTEMS_MULTIPROCESSING)
0             return _Message_queue_MP_Send_request_packet(
0                 MESSAGE_QUEUE_MP_RECEIVE_REQUEST,
0                 id,
0                 buffer,
0                 size,
0                 option_set,
0                 timeout
0               );
0       #endif
0
0           case OBJECTS_ERROR:
15            return RTEMS_INVALID_ID;
0
0           case OBJECTS_LOCAL:
0             if ( _Options_Is_no_wait( option_set ) )
0               wait = FALSE;
0             else
0               wait = TRUE;
0
124           _CORE_message_queue_Seize(
0               &the_message_queue->message_queue,
0               the_message_queue->Object.id,
0               buffer,
0               size,
0               wait,
0               timeout
0             );
0             _Thread_Enable_dispatch();
118           return _Message_queue_Translate_core_message_queue_return_code(
0               _Thread_Executing->Wait.return_code
0             );
0
0         }
0
0         return RTEMS_INTERNAL_ERROR;   /* unreached - only to remove warnings */
133     }
```

## B.13. msgqsubmit.c

```
0       /*
0        *  Message Queue Manager
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: msgqsubmit.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/sysstate.h>
```

```
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
0       #endif
0       #include <rtems/rtems/status.h>
0       #include <rtems/rtems/attr.h>
0       #include <rtems/rtems/message.h>
0       #include <rtems/rtems/options.h>
0       #include <rtems/rtems/support.h>
0
0       /*PAGE
0        *
0        *  _Message_queue_Submit
0        *
0        *  This routine implements the directives rtems_message_queue_send
0        *  and rtems_message_queue_urgent.  It processes a message that is
0        *  to be submitted to the designated message queue.  The message will
0        *  either be processed as a send send message which it will be inserted
0        *  at the rear of the queue or it will be processed as an urgent message
0        *  which will be inserted at the front of the queue.
0        *
0        *  Input parameters:
0        *    id          - pointer to message queue
0        *    buffer      - pointer to message buffer
0        *    size        - size in bytes of message to send
0        *    submit_type - send or urgent message
0        *
0        *  Output parameters:
0        *    RTEMS_SUCCESSFUL - if successful
0        *    error code       - if unsuccessful
0        */
0
0       rtems_status_code _Message_queue_Submit(
0         Objects_Id                  id,
0         void                       *buffer,
0         unsigned32                  size,
0         Message_queue_Submit_types  submit_type
0       )
0       {
0         register Message_queue_Control  *the_message_queue;
0         Objects_Locations                location;
0
0         the_message_queue = _Message_queue_Get( id, &location );
104       switch ( location )
0         {
0           case OBJECTS_REMOTE:
0       #if defined(RTEMS_MULTIPROCESSING)
0             switch ( submit_type ) {
0               case MESSAGE_QUEUE_SEND_REQUEST:
0                 return _Message_queue_MP_Send_request_packet(
0                     MESSAGE_QUEUE_MP_SEND_REQUEST,
0                     id,
0                     buffer,
0                     &size,
0                     0,                                /* option_set */
0                     MPCI_DEFAULT_TIMEOUT
0                   );
0
0               case MESSAGE_QUEUE_URGENT_REQUEST:
0                 return _Message_queue_MP_Send_request_packet(
0                     MESSAGE_QUEUE_MP_URGENT_REQUEST,
0                     id,
0                     buffer,
0                     &size,
0                     0,                                /* option_set */
0                     MPCI_DEFAULT_TIMEOUT
0                   );
0             }
0             break;
0       #endif
0
```

```
0           case OBJECTS_ERROR:
14            return RTEMS_INVALID_ID;
0
0           case OBJECTS_LOCAL:
90            switch ( submit_type ) {
0               case MESSAGE_QUEUE_SEND_REQUEST:
0                 _CORE_message_queue_Send(
0                   &the_message_queue->message_queue,
0                   buffer,
0                   size,
0                   id,
0       #if defined(RTEMS_MULTIPROCESSING)
0                   _Message_queue_Core_message_queue_mp_support,
0       #else
0                   NULL,
0       #endif
0                   FALSE,   /* sender does not block */
0                   0        /* no timeout */
0                 );
46                break;
0               case MESSAGE_QUEUE_URGENT_REQUEST:
0                 _CORE_message_queue_Urgent(
0                   &the_message_queue->message_queue,
0                   buffer,
0                   size,
0                   id,
0       #if defined(RTEMS_MULTIPROCESSING)
0                   _Message_queue_Core_message_queue_mp_support,
0       #else
0                   NULL,
0       #endif
0                   FALSE,   /* sender does not block */
0                   0        /* no timeout */
0                 );
0                 break;
0               default:
0                 return RTEMS_INTERNAL_ERROR;   /* should never get here */
89            }
0
0             _Thread_Enable_dispatch();
89            return _Message_queue_Translate_core_message_queue_return_code(
0               _Thread_Executing->Wait.return_code
0             );
0
0         }
0         return RTEMS_INTERNAL_ERROR;   /* unreached - only to remove warnings */
103     }
```

## B.14. msgqtranslatereturncode.c

```
0       /*
0        *  Message Queue Manager
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: msgqtranslatereturncode.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/coremsg.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/mpci.h>
```

```
0       #endif
0       #include <rtems/rtems/status.h>
0       #include <rtems/rtems/attr.h>
0       #include <rtems/rtems/message.h>
0       #include <rtems/rtems/options.h>
0       #include <rtems/rtems/support.h>
0
0       /*PAGE
0        *
0        *  _Message_queue_Translate_core_message_queue_return_code
0        *
0        *  Input parameters:
0        *    the_message_queue_status - message_queue status code to translate
0        *
0        *  Output parameters:
0        *    rtems status code - translated RTEMS status code
0        *
0        */
0
0       rtems_status_code _Message_queue_Translate_core_message_queue_return_code (
0         unsigned32 the_message_queue_status
0       )
0       {
239       switch ( the_message_queue_status ) {
0           case  CORE_MESSAGE_QUEUE_STATUS_SUCCESSFUL:
0             return RTEMS_SUCCESSFUL;
0           case  CORE_MESSAGE_QUEUE_STATUS_INVALID_SIZE:
0             return RTEMS_INVALID_SIZE;
0           case  CORE_MESSAGE_QUEUE_STATUS_TOO_MANY:
1             return RTEMS_TOO_MANY;
0           case CORE_MESSAGE_QUEUE_STATUS_UNSATISFIED:
0             return RTEMS_UNSATISFIED;
0           case CORE_MESSAGE_QUEUE_STATUS_UNSATISFIED_NOWAIT:
2             return RTEMS_UNSATISFIED;
0           case CORE_MESSAGE_QUEUE_STATUS_WAS_DELETED:
0             return RTEMS_OBJECT_WAS_DELETED;
0           case CORE_MESSAGE_QUEUE_STATUS_TIMEOUT:
0             return RTEMS_TIMEOUT;
0           case THREAD_STATUS_PROXY_BLOCKING:
0             return RTEMS_PROXY_BLOCKING;
0         }
0         _Internal_error_Occurred(         /* XXX */
0           INTERNAL_ERROR_RTEMS_API,
0           TRUE,
0           the_message_queue_status
0         );
239       return RTEMS_INTERNAL_ERROR;   /* unreached - only to remove warnings */
0       }
```

## B.15. object.inl

```
0       /*  object.inl
0        *
0        *  This include file contains the static inline implementation of all
0        *  of the inlined routines in the Object Handler.
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: object.inl.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #ifndef __OBJECTS_inl
0       #define __OBJECTS_inl
0
0       /*PAGE
0        *
0        *  _Objects_Build_id
0        *
0        *  DESCRIPTION:
0        *
```

```
0        *  This function builds an object's id from the processor node and index
0        *  values specified.
0        */
0
0       RTEMS_INLINE_ROUTINE Objects_Id _Objects_Build_id(
0         Objects_Classes  the_class,
0         unsigned32       node,
0         unsigned32       index
0       )
0       {
0         return ( (the_class << OBJECTS_CLASS_START_BIT) |
0                  (node << OBJECTS_NODE_START_BIT)       |
0                  (index << OBJECTS_INDEX_START_BIT) );
0       }
0
0       /*PAGE
0        *
0        *  _Objects_Get_class
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns the class portion of the ID.
0        */
0
0       RTEMS_INLINE_ROUTINE Objects_Classes _Objects_Get_class(
0         Objects_Id id
0       )
104     {
104       return (Objects_Classes)
0           ((id >> OBJECTS_CLASS_START_BIT) & OBJECTS_CLASS_VALID_BITS);
0       }
0
0
0       /*PAGE
0        *
0        *  _Objects_Get_node
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns the node portion of the ID.
0        */
0
0       RTEMS_INLINE_ROUTINE unsigned32 _Objects_Get_node(
0         Objects_Id id
0       )
0       {
0         return (id >> OBJECTS_NODE_START_BIT) & OBJECTS_NODE_VALID_BITS;
0       }
0
0       /*PAGE
0        *
0        *  _Objects_Get_index
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns the index portion of the ID.
0        */
0
0       RTEMS_INLINE_ROUTINE unsigned32 _Objects_Get_index(
0         Objects_Id id
0       )
227     {
677       return (id >> OBJECTS_INDEX_START_BIT) & OBJECTS_INDEX_VALID_BITS;
0       }
0
0       /*PAGE
0        *
0        *  _Objects_Is_class_valid
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the class is valid.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Objects_Is_class_valid(
0         Objects_Classes the_class
0       )
```

```
0       {
0         return the_class && the_class <= OBJECTS_CLASSES_LAST;
0       }
0
0       /*PAGE
0        *
0        *  _Objects_Is_local_node
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the node is of the local object, and
0        *  FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Objects_Is_local_node(
0         unsigned32 node
0       )
0       {
0         return ( node == _Objects_Local_node );
0       }
0
0       /*PAGE
0        *
0        *  _Objects_Is_local_id
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the id is of a local object, and
0        *  FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Objects_Is_local_id(
0         Objects_Id id
0       )
0       {
0         return _Objects_Is_local_node( _Objects_Get_node(id) );
0       }
0
0       /*PAGE
0        *
0        *  _Objects_Are_ids_equal
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if left and right are equal,
0        *  and FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Objects_Are_ids_equal(
0         Objects_Id left,
0         Objects_Id right
0       )
0       {
0         return ( left == right );
0       }
0
0       /*PAGE
0        *
0        *  _Objects_Get_local_object
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns a pointer to the local_table object
0        *  referenced by the index.
0        */
0
0       RTEMS_INLINE_ROUTINE Objects_Control *_Objects_Get_local_object(
0         Objects_Information *information,
0         unsigned32           index
0       )
0       {
1610      if ( index > information->maximum )
0           return NULL;
1610      return ( information->local_table[ index ] );
0       }
0
0       /*PAGE
```

```
0        *
0        *  _Objects_Set_local_object
0        *
0        *  DESCRIPTION:
0        *
0        *  This function sets the pointer to the local_table object
0        *  referenced by the index.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Objects_Set_local_object(
0         Objects_Information *information,
0         unsigned32           index,
0         Objects_Control     *the_object
0       )
0       {
208       if ( index <= information->maximum )
288         information->local_table[ index ] = the_object;
0       }
0
0
0       /*PAGE
0        *
0        *  _Objects_Get_information
0        *
0        *  DESCRIPTION:
0        *
0        *  This function return the information structure given
0        *  an id of an object.
0        */
0
0       RTEMS_INLINE_ROUTINE Objects_Information *_Objects_Get_information(
0         Objects_Id  id
0       )
0       {
0         Objects_Classes  the_class;
0
0         the_class = _Objects_Get_class( id );
0
0         if ( !_Objects_Is_class_valid( the_class ) )
0           return NULL;
0
0         return _Objects_Information_table[ the_class ];
0       }
0
0       /*PAGE
0        *
0        *  _Objects_Open
0        *
0        *  DESCRIPTION:
0        *
0        *  This function places the_object control pointer and object name
0        *  in the Local Pointer and Local Name Tables, respectively.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Objects_Open(
0         Objects_Information *information,
0         Objects_Control     *the_object,
0         Objects_Name         name
0       )
0       {
0         unsigned32  index;
0
0         index = _Objects_Get_index( the_object->id );
0         _Objects_Set_local_object( information, index, the_object );
0
104       if ( information->is_string )
0           _Objects_Copy_name_string( name, the_object->name );
0         else
104         _Objects_Copy_name_raw( name, the_object->name, information->name_length );
0       }
0
0       /*PAGE
0        *
0        *  _Objects_Close
0        *
0        *  DESCRIPTION:
0        *
```

```
0        *  This function removes the_object control pointer and object name
0        *  in the Local Pointer and Local Name Tables.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Objects_Close(
0         Objects_Information  *information,
0         Objects_Control      *the_object
0       )
0       {
63        unsigned32 index;
0
0         index = _Objects_Get_index( the_object->id );
0         _Objects_Set_local_object( information, index, NULL );
268       _Objects_Clear_name( the_object->name, information->name_length );
0       }
0
0       /*PAGE
0        *
0        *  _Objects_Namespace_remove
0        *
0        *  DESCRIPTION:
0        *
0        *  This function removes the_object from the namespace.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Objects_Namespace_remove(
0         Objects_Information  *information,
0         Objects_Control      *the_object
0       )
0       {
0         _Objects_Clear_name( the_object->name, information->name_length );
0       }
0
0       #endif
0       /* end of include file */
```

## B.16. objectget.c

```
0       /*
0        *  Object Handler
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: objectget.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/address.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/object.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/objectmp.h>
0       #endif
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/isr.h>
0
0       /*PAGE
0        *
0        * _Objects_Get
0        *
0        * This routine sets the object pointer for the given
0        * object id based on the given object information structure.
0        *
0        * Input parameters:
0        *   information - pointer to entry in table for this class
0        *   id          - object id to search for
0        *   location    - address of where to store the location
```

```
0        *
0        * Output parameters:
0        *   returns  - address of object if local
0        *   location - one of the following:
0        *                  OBJECTS_ERROR  - invalid object ID
0        *                  OBJECTS_REMOTE - remote object
0        *                  OBJECTS_LOCAL  - local object
0        */
0
0       Objects_Control *_Objects_Get(
0         Objects_Information *information,
0         Objects_Id           id,
0         Objects_Locations   *location
0       )
0       {
0         Objects_Control *the_object;
0         unsigned32       index;
0
0         index = _Objects_Get_index( id );
0
1634      if ( information->maximum >= index ) {
0           _Thread_Disable_dispatch();
1610        if ( (the_object = _Objects_Get_local_object( information, index )) != NULL ) {
0             *location = OBJECTS_LOCAL;
1548          return( the_object );
62          }
0           _Thread_Enable_dispatch();
0           *location = OBJECTS_ERROR;
0           return( NULL );
0         }
86        *location = OBJECTS_ERROR;
0       #if defined(RTEMS_MULTIPROCESSING)
0         _Objects_MP_Is_remote(
0           information,
0           _Objects_Build_id( information->the_class, _Objects_Local_node, index ),
0           location,
0           &the_object
0         );
0         return the_object;
0       #else
86        return NULL;
0       #endif
1634    }
```

## B.17. objectnametoid.c

```
0       /*
0        *  Object Handler
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: objectnametoid.c.cov.tmp,v 1.1 2003/10/28 14:41:51 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/address.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/object.h>
0       #if defined(RTEMS_MULTIPROCESSING)
0       #include <rtems/score/objectmp.h>
0       #endif
0       #include <rtems/score/thread.h>
0       #include <rtems/score/wkspace.h>
0       #include <rtems/score/sysstate.h>
0       #include <rtems/score/isr.h>
0
0       /*PAGE
0        *
0        *  _Objects_Name_to_id
```

```
0        *
0        *  These kernel routines search the object table(s) for the given
0        *  object name and returns the associated object id.
0        *
0        *  Input parameters:
0        *    information - object information
0        *    name        - user defined object name
0        *    node        - node indentifier (0 indicates any node)
0        *    id          - address of return ID
0        *
0        *  Output parameters:
0        *    id                  - object id
0        *    OBJECTS_SUCCESSFUL - if successful
0        *    error code          - if unsuccessful
0        */
0
0       Objects_Name_to_id_errors _Objects_Name_to_id(
0         Objects_Information *information,
0         Objects_Name         name,
0         unsigned32           node,
0         Objects_Id          *id
0       )
0       {
0         boolean                    search_local_node;
0         Objects_Control           *the_object;
0         unsigned32                 index;
0         unsigned32                 name_length;
0         Objects_Name_comparators   compare_them;
0
94        if ( name == 0 )
0           return OBJECTS_INVALID_NAME;
0
0         search_local_node = FALSE;
0
94        if ( information->maximum != 0 &&
0             (node == OBJECTS_SEARCH_ALL_NODES || node == OBJECTS_SEARCH_LOCAL_NODE ||
0             _Objects_Is_local_node( node ) ) )
94         search_local_node = TRUE;
0
94        if ( search_local_node ) {
0           name_length = information->name_length;
0
94          if ( information->is_string ) compare_them = _Objects_Compare_name_string;
94          else                          compare_them = _Objects_Compare_name_raw;
0
150         for ( index = 1; index <= information->maximum; index++ ) {
0
94            the_object = information->local_table[ index ];
0
187           if ( !the_object || !the_object->name )
0               continue;
0
111           if ( (*compare_them)( name, the_object->name, name_length ) ) {
0               *id = the_object->id;
55              return OBJECTS_SUCCESSFUL;
0             }
0           }
39        }
0
39        if ( _Objects_Is_local_node( node ) || node == OBJECTS_SEARCH_LOCAL_NODE )
39          return OBJECTS_INVALID_NAME;
0
0       #if defined(RTEMS_MULTIPROCESSING)
0         return ( _Objects_MP_Global_name_search( information, name, node, id ) );
0       #else
39        return OBJECTS_INVALID_NAME;
0       #endif
94      }
```

## B.18. thread.inl

```
0       /*  thread.inl
0        *
0        *  This file contains the macro implementation of the inlined
0        *  routines from the Thread handler.
```

```
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: thread.inl.cov.tmp,v 1.1 2003/10/28 14:41:52 rmaia Exp $
0        */
0
0       #ifndef __THREAD_inl
0       #define __THREAD_inl
0
0       /*PAGE
0        *
0        *  _Thread_Stop_multitasking
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine halts multitasking and returns control to
0        *  the "thread" (i.e. the BSP) which initially invoked the
0        *  routine which initialized the system.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Thread_Stop_multitasking( void )
0       {
0         _Context_Switch( &_Thread_Executing->Registers, &_Thread_BSP_context );
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Is_executing
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the_thread is the currently executing
0        *  thread, and FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Thread_Is_executing (
0         Thread_Control *the_thread
0       )
0       {
287       return ( the_thread == _Thread_Executing );
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Is_heir
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the_thread is the heir
0        *  thread, and FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Thread_Is_heir (
0         Thread_Control *the_thread
0       )
0       {
150       return ( the_thread == _Thread_Heir );
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Is_executing_also_the_heir
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the currently executing thread
0        *  is also the heir thread, and FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Thread_Is_executing_also_the_heir( void )
0       {
0         return ( _Thread_Executing == _Thread_Heir );
```

```
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Unblock
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine clears any blocking state for the_thread.  It performs
0        *  any necessary scheduling operations including the selection of
0        *  a new heir thread.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Thread_Unblock (
0         Thread_Control *the_thread
0       )
0       {
94        _Thread_Clear_state( the_thread, STATES_BLOCKED );
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Restart_self
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine resets the current context of the calling thread
0        *  to that of its initial state.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Thread_Restart_self( void )
0       {
0         if ( _Thread_Executing->fp_context != NULL )
0           _Context_Restore_fp( &_Thread_Executing->fp_context );
0
0         _CPU_Context_Restart_self( &_Thread_Executing->Registers );
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Calculate_heir
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns a pointer to the highest priority
0        *  ready thread.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Thread_Calculate_heir( void )
0       {
104       _Thread_Heir = (Thread_Control *)
0           _Thread_Ready_chain[ _Priority_Get_highest() ].first;
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Is_allocated_fp
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the floating point context of
0        *  the_thread is currently loaded in the floating point unit, and
0        *  FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Thread_Is_allocated_fp (
0         Thread_Control *the_thread
0       )
0       {
0         return ( the_thread == _Thread_Allocated_fp );
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Deallocate_fp
0        *
0        *  DESCRIPTION:
```

```
0        *
0        *  This routine is invoked when the currently loaded floating
0        *  point context is now longer associated with an active thread.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Thread_Deallocate_fp( void )
0       {
0         _Thread_Allocated_fp = NULL;
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Disable_dispatch
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine prevents dispatching.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Thread_Disable_dispatch( void )
0       {
1612      _Thread_Dispatch_disable_level += 1;
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Enable_dispatch
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine allows dispatching to occur again.  If this is
0        *  the outer most dispatching critical section, then a dispatching
0        *  operation will be performed and, if necessary, control of the
0        *  processor will be transferred to the heir thread.
0        */
0
0       #if ( CPU_INLINE_ENABLE_DISPATCH == TRUE )
0       RTEMS_INLINE_ROUTINE void _Thread_Enable_dispatch()
0       {
1116      if ( (--_Thread_Dispatch_disable_level) == 0 )
727         _Thread_Dispatch();
0       }
0       #endif
0
0       #if ( CPU_INLINE_ENABLE_DISPATCH == FALSE )
0       void _Thread_Enable_dispatch( void );
0       #endif
0
0       /*PAGE
0        *
0        *  _Thread_Unnest_dispatch
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine allows dispatching to occur again.  However,
0        *  no dispatching operation is performed even if this is the outer
0        *  most dispatching critical section.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Thread_Unnest_dispatch( void )
0       {
0         _Thread_Dispatch_disable_level -= 1;
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Is_dispatching_enabled
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if dispatching is disabled, and FALSE
0        *  otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Thread_Is_dispatching_enabled( void )
0       {
0         return ( _Thread_Dispatch_disable_level == 0 );
```

```
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Is_context_switch_necessary
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if dispatching is disabled, and FALSE
0        *  otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Thread_Is_context_switch_necessary( void )
0       {
0         return ( _Context_Switch_necessary );
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Dispatch_initialization
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine initializes the thread dispatching subsystem.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Thread_Dispatch_initialization( void )
0       {
0         _Thread_Dispatch_disable_level = 1;
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Is_null
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the_thread is NULL and FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Thread_Is_null (
0         Thread_Control *the_thread
0       )
0       {
0         return ( the_thread == NULL );
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Get
0        *
0        *  DESCRIPTION:
0        *
0        *  This function maps thread IDs to thread control
0        *  blocks.  If ID corresponds to a local thread, then it
0        *  returns the_thread control pointer which maps to ID
0        *  and location is set to OBJECTS_LOCAL.  If the thread ID is
0        *  global and resides on a remote node, then location is set
0        *  to OBJECTS_REMOTE, and the_thread is undefined.
0        *  Otherwise, location is set to OBJECTS_ERROR and
0        *  the_thread is undefined.
0        *
0        *  NOTE:  XXX... This routine may be able to be optimized.
0        */
0
0       RTEMS_INLINE_ROUTINE Thread_Control *_Thread_Get (
0         Objects_Id         id,
0         Objects_Locations *location
0       )
0       {
0         Objects_Classes      the_class;
0         Objects_Information *information;
0         Thread_Control      *tp = (Thread_Control *) 0;
0
104       if ( _Objects_Are_ids_equal( id, OBJECTS_ID_OF_SELF ) ) {
0           _Thread_Disable_dispatch();
0           *location = OBJECTS_LOCAL;
```

```
0           tp = _Thread_Executing;
0           goto done;
0         }
0
0         the_class = _Objects_Get_class( id );
0
104       if ( the_class > OBJECTS_CLASSES_LAST ) {
0           *location = OBJECTS_ERROR;
0           goto done;
0         }
0
104       information = _Objects_Information_table[ the_class ];
0
104       if ( !information || !information->is_thread ) {
0           *location = OBJECTS_ERROR;
0           goto done;
0         }
0
104       tp = (Thread_Control *) _Objects_Get( information, id, location );
0
0       done:
0         return tp;
0       }
0
0
0       /*
0        *  _Thread_Is_proxy_blocking
0        *
0        *  DESCRIPTION:
0        *
0        *  This function returns TRUE if the status code is equal to the
0        *  status which indicates that a proxy is blocking, and FALSE otherwise.
0        */
0
0       RTEMS_INLINE_ROUTINE boolean _Thread_Is_proxy_blocking (
0         unsigned32 code
0       )
0       {
0         return (code == THREAD_STATUS_PROXY_BLOCKING);
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Internal_allocate
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine allocates an internal thread.
0        */
0
0       RTEMS_INLINE_ROUTINE Thread_Control *_Thread_Internal_allocate( void )
0       {
0         return (Thread_Control *) _Objects_Allocate( &_Thread_Internal_information );
0       }
0
0       /*PAGE
0        *
0        *  _Thread_Internal_free
0        *
0        *  DESCRIPTION:
0        *
0        *  This routine frees an internal thread.
0        */
0
0       RTEMS_INLINE_ROUTINE void _Thread_Internal_free (
0         Thread_Control *the_task
0       )
0       {
0         _Objects_Free( &_Thread_Internal_information, &the_task->Object );
0       }
0
0       #endif
0       /* end of include file */
```

## B.19. threadqdequeue.c

```
0       /*
0        *  Thread Queue Handler
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: threadqdequeue.c.cov.tmp,v 1.1 2003/10/28 14:41:52 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/threadq.h>
0       #include <rtems/score/tqdata.h>
0
0       /*PAGE
0        *
0        *  _Thread_queue_Dequeue
0        *
0        *  This routine removes a thread from the specified threadq.  If the
0        *  threadq discipline is FIFO, it unblocks a thread, and cancels its
0        *  timeout timer.  Priority discipline is processed elsewhere.
0        *
0        *  Input parameters:
0        *    the_thread_queue - pointer to threadq
0        *
0        *  Output parameters:
0        *    returns - thread dequeued or NULL
0        *
0        *  INTERRUPT LATENCY:
0        *    check sync
0        */
0
0       Thread_Control *_Thread_queue_Dequeue(
0         Thread_queue_Control *the_thread_queue
0       )
0       {
0         Thread_Control *the_thread;
0
786       switch ( the_thread_queue->discipline ) {
0           case THREAD_QUEUE_DISCIPLINE_FIFO:
0             the_thread = _Thread_queue_Dequeue_fifo( the_thread_queue );
246           break;
0           case THREAD_QUEUE_DISCIPLINE_PRIORITY:
540           the_thread = _Thread_queue_Dequeue_priority( the_thread_queue );
0             break;
0           default:              /* this is only to prevent warnings */
0             the_thread = NULL;
0             break;
0         }
0
0         return( the_thread );
785     }
0
```

## B.20. threadqdequeuefifo.c

```
0       /*
0        *  Thread Queue Handler
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
```

```
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: threadqdequeuefifo.c.cov.tmp,v 1.1 2003/10/28 14:41:52 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/threadq.h>
0       #include <rtems/score/tqdata.h>
0
0       /*PAGE
0        *
0        *  _Thread_queue_Dequeue_fifo
0        *
0        *  This routine removes a thread from the specified threadq.
0        *
0        *  Input parameters:
0        *    the_thread_queue - pointer to threadq
0        *
0        *  Output parameters:
0        *    returns - thread dequeued or NULL
0        *
0        *  INTERRUPT LATENCY:
0        *    check sync
0        *    FIFO
0        */
0
0       Thread_Control *_Thread_queue_Dequeue_fifo(
0         Thread_queue_Control *the_thread_queue
0       )
0       {
0         ISR_Level              level;
0         Thread_Control *the_thread;
0
246       _ISR_Disable( level );
246       if ( !_Chain_Is_empty( &the_thread_queue->Queues.Fifo ) ) {
0
0           the_thread = (Thread_Control *)
0              _Chain_Get_first_unprotected( &the_thread_queue->Queues.Fifo );
0
91          if ( !_Watchdog_Is_active( &the_thread->Timer ) ) {
91            _ISR_Enable( level );
0             _Thread_Unblock( the_thread );
90          } else {
0             _Watchdog_Deactivate( &the_thread->Timer );
0             _ISR_Enable( level );
0             (void) _Watchdog_Remove( &the_thread->Timer );
0             _Thread_Unblock( the_thread );
0           }
0
0       #if defined(RTEMS_MULTIPROCESSING)
0           if ( !_Objects_Is_local_id( the_thread->Object.id ) )
0             _Thread_MP_Free_proxy( the_thread );
0       #endif
0
0           return the_thread;
0         }
0
155       switch ( the_thread_queue->sync_state ) {
0           case THREAD_QUEUE_SYNCHRONIZED:
0           case THREAD_QUEUE_SATISFIED:
155           _ISR_Enable( level );
155           return NULL;
0
0           case THREAD_QUEUE_NOTHING_HAPPENED:
0           case THREAD_QUEUE_TIMEOUT:
0             the_thread_queue->sync_state = THREAD_QUEUE_SATISFIED;
0             _ISR_Enable( level );
0             return _Thread_Executing;
0         }
0         return NULL;                /* this is only to prevent warnings */
```

```
245     }
0
```

## B.21. threadqdequeuepriority.c

```
0       /*
0        *  Thread Queue Handler
0        *
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: threadqdequeuepriority.c.cov.tmp,v 1.1 2003/10/28 14:41:52 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
0       #include <rtems/score/chain.h>
0       #include <rtems/score/isr.h>
0       #include <rtems/score/object.h>
0       #include <rtems/score/states.h>
0       #include <rtems/score/thread.h>
0       #include <rtems/score/threadq.h>
0       #include <rtems/score/tqdata.h>
0
0       /*PAGE
0        *
0        *  _Thread_queue_Dequeue_priority
0        *
0        *  This routine removes a thread from the specified PRIORITY based
0        *  threadq, unblocks it, and cancels its timeout timer.
0        *
0        *  Input parameters:
0        *    the_thread_queue - pointer to thread queue
0        *
0        *  Output parameters:
0        *    returns - thread dequeued or NULL
0        *
0        *  INTERRUPT LATENCY:
0        *    only case
0        */
0
0       Thread_Control *_Thread_queue_Dequeue_priority(
0         Thread_queue_Control *the_thread_queue
0       )
0       {
0         unsigned32      index;
0         ISR_Level       level;
540       Thread_Control *the_thread = NULL;  /* just to remove warnings */
0         Thread_Control *new_first_thread;
0         Chain_Node     *new_first_node;
0         Chain_Node     *new_second_node;
0         Chain_Node     *last_node;
0         Chain_Node     *next_node;
0         Chain_Node     *previous_node;
0
540       _ISR_Disable( level );
540       for( index=0 ;
2688           index < TASK_QUEUE_DATA_NUMBER_OF_PRIORITY_HEADERS ;
2148           index++ ) {
2151        if ( !_Chain_Is_empty( &the_thread_queue->Queues.Priority[ index ] ) ) {
3             the_thread = (Thread_Control *)
0                            the_thread_queue->Queues.Priority[ index ].first;
0             goto dequeue;
0           }
0         }
0
537       switch ( the_thread_queue->sync_state ) {
0           case THREAD_QUEUE_SYNCHRONIZED:
0           case THREAD_QUEUE_SATISFIED:
537           _ISR_Enable( level );
537           return NULL;
```

```
0
0           case THREAD_QUEUE_NOTHING_HAPPENED:
0           case THREAD_QUEUE_TIMEOUT:
0             the_thread_queue->sync_state = THREAD_QUEUE_SATISFIED;
0             _ISR_Enable( level );
0             return _Thread_Executing;
0         }
0
0       dequeue:
3         new_first_node   = the_thread->Wait.Block2n.first;
0         new_first_thread = (Thread_Control *) new_first_node;
3         next_node        = the_thread->Object.Node.next;
0         previous_node    = the_thread->Object.Node.previous;
0
6         if ( !_Chain_Is_empty( &the_thread->Wait.Block2n ) ) {
0           last_node       = the_thread->Wait.Block2n.last;
0           new_second_node = new_first_node->next;
0
0           previous_node->next      = new_first_node;
0           next_node->previous      = new_first_node;
0           new_first_node->next     = next_node;
0           new_first_node->previous = previous_node;
0
0           if ( !_Chain_Has_only_one_node( &the_thread->Wait.Block2n ) ) {
0                                                       /* > two threads on 2-n */
0             new_second_node->previous =
0                       _Chain_Head( &new_first_thread->Wait.Block2n );
0
0             new_first_thread->Wait.Block2n.first = new_second_node;
0             new_first_thread->Wait.Block2n.last  = last_node;
0
0             last_node->next = _Chain_Tail( &new_first_thread->Wait.Block2n );
0           }
0         } else {
3           previous_node->next = next_node;
3           next_node->previous = previous_node;
0         }
0
3         if ( !_Watchdog_Is_active( &the_thread->Timer ) ) {
3           _ISR_Enable( level );
0           _Thread_Unblock( the_thread );
3         } else {
0           _Watchdog_Deactivate( &the_thread->Timer );
0           _ISR_Enable( level );
0           (void) _Watchdog_Remove( &the_thread->Timer );
0           _Thread_Unblock( the_thread );
0         }
0
0       #if defined(RTEMS_MULTIPROCESSING)
0         if ( !_Objects_Is_local_id( the_thread->Object.id ) )
0           _Thread_MP_Free_proxy( the_thread );
0       #endif
0         return( the_thread );
540     }
0
```

## B.22. userext.c

```
0       /*

0        *  User Extension Handler
0        *
0        *  NOTE: XXX
0        *
0        *  COPYRIGHT (c) 1989-1999.
0        *  On-Line Applications Research Corporation (OAR).
0        *
0        *  The license and distribution terms for this file may be
0        *  found in the file LICENSE in this distribution or at
0        *  http://www.OARcorp.com/rtems/license.html.
0        *
0        *  $Id: userext.c.cov.tmp,v 1.1 2003/10/28 14:41:52 rmaia Exp $
0        */
0
0       #include <rtems/system.h>
```

```
0       #include <rtems/score/userext.h>
0
0       /*PAGE
0        *
0        *  _User_extensions_Thread_create
0        */
0
0       boolean _User_extensions_Thread_create (
0         Thread_Control *the_thread
0       )
0       {
0         Chain_Node              *the_node;
0         User_extensions_Control *the_extension;
0         boolean                  status;
0
104       for ( the_node = _User_extensions_List.first ;
0               !_Chain_Is_tail( &_User_extensions_List, the_node ) ;
104             the_node = the_node->next ) {
0
0           the_extension = (User_extensions_Control *) the_node;
0
104         if ( the_extension->Callouts.thread_create != NULL ) {
312           status = (*the_extension->Callouts.thread_create)(
0               _Thread_Executing,
0               the_thread
0             );
312           if ( !status )
0               return FALSE;
0           }
0         }
0
104       return TRUE;
104     }
0
0       /*PAGE
0        *
0        *  _User_extensions_Thread_delete
0        */
0
0       void _User_extensions_Thread_delete (
0         Thread_Control *the_thread
0       )
0       {
0         Chain_Node              *the_node;
0         User_extensions_Control *the_extension;
0
0         for ( the_node = _User_extensions_List.last ;
0               !_Chain_Is_head( &_User_extensions_List, the_node ) ;
0               the_node = the_node->previous ) {
0
0           the_extension = (User_extensions_Control *) the_node;
0
0           if ( the_extension->Callouts.thread_delete != NULL )
0             (*the_extension->Callouts.thread_delete)(
0               _Thread_Executing,
0               the_thread
0             );
0         }
0       }
0
0       /*PAGE
0        *
0        *  _User_extensions_Thread_start
0        *
0        */
0
0       void _User_extensions_Thread_start (
0         Thread_Control *the_thread
0       )
0       {
0         Chain_Node              *the_node;
0         User_extensions_Control *the_extension;
0
104       for ( the_node = _User_extensions_List.first ;
0               !_Chain_Is_tail( &_User_extensions_List, the_node ) ;
```

```
416             the_node = the_node->next ) {
0
0           the_extension = (User_extensions_Control *) the_node;
0
104         if ( the_extension->Callouts.thread_start != NULL )
208           (*the_extension->Callouts.thread_start)(
0               _Thread_Executing,
0               the_thread
0             );
0         }
104     }
0
0       /*PAGE
0        *
0        *  _User_extensions_Thread_restart
0        *
0        */
0
0       void _User_extensions_Thread_restart (
0         Thread_Control *the_thread
0       )
0       {
0         Chain_Node              *the_node;
0         User_extensions_Control *the_extension;
0
0         for ( the_node = _User_extensions_List.first ;
0               !_Chain_Is_tail( &_User_extensions_List, the_node ) ;
0               the_node = the_node->next ) {
0
0           the_extension = (User_extensions_Control *) the_node;
0
0           if ( the_extension->Callouts.thread_restart != NULL )
0             (*the_extension->Callouts.thread_restart)(
0               _Thread_Executing,
0               the_thread
0             );
0         }
0       }
0
0       /*PAGE
0        *
0        *  _User_extensions_Thread_begin
0        *
0        */
0
0       void _User_extensions_Thread_begin (
0         Thread_Control *executing
0       )
0       {
0         Chain_Node              *the_node;
0         User_extensions_Control *the_extension;
0
0         for ( the_node = _User_extensions_List.first ;
0               !_Chain_Is_tail( &_User_extensions_List, the_node ) ;
0               the_node = the_node->next ) {
0
0           the_extension = (User_extensions_Control *) the_node;
0
0           if ( the_extension->Callouts.thread_begin != NULL )
0             (*the_extension->Callouts.thread_begin)( executing );
0         }
0       }
0
0       /*PAGE
0        *
0        *  _User_extensions_Thread_exitted
0        */
0
0       void _User_extensions_Thread_exitted (
0         Thread_Control *executing
0       )
0       {
0         Chain_Node              *the_node;
0         User_extensions_Control *the_extension;
0
0         for ( the_node = _User_extensions_List.last ;
0               !_Chain_Is_head( &_User_extensions_List, the_node ) ;
```

```
0               the_node = the_node->previous ) {
0
0           the_extension = (User_extensions_Control *) the_node;
0
0           if ( the_extension->Callouts.thread_exitted != NULL )
0             (*the_extension->Callouts.thread_exitted)( executing );
0         }
0       }
0
0       /*PAGE
0        *
0        *  _User_extensions_Fatal
0        */
0
0       void _User_extensions_Fatal (
0         Internal_errors_Source  the_source,
0         boolean                 is_internal,
0         unsigned32              the_error
0       )
0       {
0         Chain_Node              *the_node;
0         User_extensions_Control *the_extension;
0
0         for ( the_node = _User_extensions_List.last ;
0               !_Chain_Is_head( &_User_extensions_List, the_node ) ;
0               the_node = the_node->previous ) {
0
0           the_extension = (User_extensions_Control *) the_node;
0
0           if ( the_extension->Callouts.fatal != NULL )
0             (*the_extension->Callouts.fatal)( the_source, is_internal, the_error );
0         }
0       }
0
0
```