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 Robustness Testing Report - Annex

RAMS Call-off Order 2

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

ESTEC/Contract Nº 16582/02/NL/PA

DISCLAIMER
European Space Agency Contract Report
The work described in this report was performed under ESA contract. Responsibility for the contents resides in the author or organization that prepared it.
No conclusions on the quality of case studies used in this work shall be taken from this report. The only results that can be considered are the ones related with the techniques and methodologies applied.

Date: *25-11-2003*
Pages: *176*
State: *Approved*
Access: *See Access List*
Reference: *DL-RAMS02-02-02*
CSW-RAMS-2003-RPT-1335-02

Partners / Clients:

RTEMS 4.5.0 Robustness Testing Report - Annex

RAMS Call-off Order 2

Approved Version: 1.12			
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**
Lubomir Velkov	lvelkov@criticalsoftware.com	Senior Engineer	17-09-2003
Luís Henriques	lhenriques@criticalsoftware.com	Senior Engineer	19-09-2003
Ricardo Barbosa	rbarbosa@criticalsoftware.com	Project Engineer	17-09-2003
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	12-09-2003	First issue	Ricardo Barbosa
0.2	17-09-2003	Moved test suites to annexes.	Luís Henriques
1.0	19-09-2003	Updated after internal review meeting	Luís Henriques
1.1	25-11-2003	Updated disclaimer. Corrected some minor bugs.	Ricardo Maia

Table of Contents

This annex is part of the deliverable DL-RAMS02-02-02 of the Call-off Order number 02 under project Software Dependability and Safety Evaluations, ESTEC/Contract Nº 16582/02/NL/PA. It presents the test campaigns and test suites definition as well as the results of the execution and corresponding analysis. It also contains the source code of all of the workloads used during the robustness evaluation of the RTEMS 4.5.0.

Annex A. RTEMS Robustness Tests Definition and Results

This section contains the robustness tests definitions performed in the Real Time Executive for Multiprocessor Systems (RTEMS) Classic and POSIX APIs. It also contains the results of the execution and corresponding analysis.

One campaign was defined for each of the RTEMS resource managers. Each campaign includes one test suite for each directive of the corresponding manager. Each of the test suites originated several test cases.

This section has the following layout:

- Manager Test Campaigns
    - Manager Classic API Test Campaign
        - Test Suite
        - Test Case Results
        - Test Suite
        - Test Case Results
        - [. . .]
    - Manager POSIX API Test Campaign
        - [. . .]
- [. . .]

The first level of this hierarchical organisation will refer a specific RTEMS manager. Thus, there will be 15 entries at this level.

The next level will contain the test campaigns defined for each tested API (Classic and POSIX) in the specified RTEMS manager. It will contain, at most, 2 entries.

The final level will contain the test suites and the results of the execution of the test cases that had faulty behaviour. Each test suites contains several test cases automatically generated for a single RTEMS manager directive. Note that there are test suites that have no test case results associated, meaning that no errors were found in their execution.

In order to completely understand the test results and its analysis, the reading of the RTEMS documentation is recommended, specially the RTEMS C User's Guide and the POSIX API Users Guide.

A.1. RTEMS Task Manager Campaigns

A.1.1. RTEMS-CMP-CL-TSK: Task Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-TSK
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the task manager.
Workload File:	rtems-cmp-cl-tsk.c
Test Suites:	1. RTEMS-TS-CL-TSKCRT 2. RTEMS-TS-CL-TSKSTR 3. RTEMS-TS-CL-TSKRST 4. RTEMS-TS-CL-TSKDLT 5. RTEMS-TS-CL-TSKRSM 6. RTEMS-TS-CL-TSKSPT 7. RTEMS-TS-CL-TSKMOD
Workload Description: This workload, composed by two tasks, evokes several directives of the RTEMS task manager. The used directives provide functionalities to: • Create/delete a task; • Obtain task identifier; • Set task priority and mode; • Start/restart a task; • Suspend/resume a task;	

A.1.1.1. RTEMS-TS-CL-TSKCRT: rtems_task_create Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TSKCRT
Purpose:	To test rtems_task_create by invoking it with the entire range of test values[1] for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-tsk.c
	Lines: [120-125]
	`returnStatus = rtems_task_create (testTaskName,`
	`testTaskInitPriority,`
	`testTaskStackSize,`
	`testTaskInitMode,`
	`testTaskAtt,`
	`&testTaskId);`

Test Item:	rtems_task_create (rtems_name name, rtems_task_priority initial_priority, rtems_unsigned32 rtems_stack_size, rtems_mode initial_modes, rtems_attribute attribute_set, rtems_id *id)
Generated Test Cases:	18

[1] Please, refer to section 4.3 in this document for details about the test values.

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-TSKCRT- 008 (same results obtained in RTEMS-TCR-CL-TSKCRT- 009)

Input Specification:
`+ testTaskStackSize = 0;`
`returnStatus = rtems_task_create (testTaskName, testTaskInitPriority, testTaskStackSize, testTaskInitMode, testTaskAtt, &testTaskId);`	

Failure Description: No error code was returned by the *rtems_task_create* function, even though *testTaskStackSize* had the value 0 (zero). As specified by RTEMS documentation, *rtems_task_create* should have returned RTEMS_INVALID_SIZE, meaning that the stack size specified was too small.	
Notes: All the assertions passed, showing that the task was created and run correctly. Further analysis of the RTEMS source code showed that STACK_MINIMUM_SIZE was used as the value for the stack size.	

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-TSKCRT-010

Input Specification:
`+ testTaskStackSize = 4294967295;`
`returnStatus = rtems_task_create (testTaskName, testTaskInitPriority, testTaskStackSize, testTaskInitMode, testTaskAtt, &testTaskId);`	

Failure Description: No error code was returned by the *rtems_task_create* function, even though *testTaskStackSize* had the value 4Gb. As specified by RTEMS documentation, *rtems_task_create* should have returned RTEMS_UNSATISFIED meaning that there was not enough memory to allocate the specified stack.	
Notes: All the assertions passed, showing that the task was created and run correctly.	

A.1.1.2. RTEMS-TS-CL-TSKSTR: rtems_task_start Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TSKSTR
Purpose:	To test rtems_task_start by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-tsk.c Lines: [163-164] `returnStatus = rtems_task_start (testTaskId, &testTask, (rtems_task_argument) testTaskArg);`
Test Item:	rtems_task_start (rtems_id id, rtems_task_entry entry_point, rtems_task_argument argument)
Generated Test Cases:	7

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-TSKSTR-021 (same results obtained in RTEMS-TCR-CL-TSKCRT-002, RTEMS-TCR-CL-TSKCRT-005, RTEMS-TCR-CL-TSKCRT-007, RTEMS-TCR-CL-TSKSRT-029)

Input Specification:
`+ testTaskId = 0;`
`returnStatus = rtems_task_start (testTaskId, &testTask,`
`(rtems_task_argument) testTaskArg);`	

Failure Description: When setting the task identifier to 0 (zero), the workload get in an infinite loop. After executing the *rtems_task_start* difective, it comes back to the *rtems_task_create* call.	
Notes:	

A.1.1.3. RTEMS-TS-CL-TSKRST: rtems_task_restart Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TSKRST
Purpose:	To test rtems_task_restart by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-tsk.c
	`Lines: [182-183]`
	`returnStatus = rtems_task_restart (testTaskId,`
	`(rtems_task_argument) testTaskArg);`

Test Item:	rtems_task_restart (rtems_id id, rtems_task_argument argument)
Generated Test Cases:	6

Test Cases Results

No faults were detected in this RTEMS task manager directive.

A.1.1.4. RTEMS-TS-CL-TSKDLT: rtems_task_delete Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TSKDLT
Purpose:	To test rtems_task_delete by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-tsk.c Lines: [304] `returnStatus = rtems_task_delete (testTaskId);`
Test Item:	rtems_task_delete (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS task manager directive.

A.1.1.5. RTEMS-TS-CL-TSKRSM: rtems_task_resume Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-TSK-CL-TSKRSM
Purpose:	To test rtems_task_resume by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-tsk.c Lines: [238] `returnStatus = rtems_task_resume (testTaskId);`
Test Item:	rtems_task_resume (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS task manager directive.

A.1.1.6. RTEMS-TS-CL-TSKSPT: rtems_task_set_priority Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TSKSPT
Purpose:	To test rtems_task_set_priority by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-tsk.c
	Lines: [255-257]
	`returnStatus = rtems_task_set_priority (testTaskId,`
	`testTaskInitPriority + 1,`
	`&testTaskOldPriority);`

Test Item:	rtems_task_set_priority (rtems_id id, rtems_task_priority new_priority, rtems_task_priority *old_priority)
Generated Test Cases:	9

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-TSKSPT-049
Input Specification:	
This test case attempts to set the priority to (4294967295 + 1) = 0.	

`+ testTaskInitPriority = 4294967295;`
`returnStatus = rtems_task_set_priority (testTaskId,`
`testTaskInitPriority + 1,`
`&testTaskOldPriority);`	

Failure Description:	
No error code was returned when attempting to set the priority of the task to 0. As specified by RTEMS documentation, *rtems_task_set_priority* should have returned RTEMS_INVALID_PRIORITY since the range of valid priorities in RTEMS is [1..255]. After trying to obtain the actual task priority, it is checked that it is not the value that was set, meaning that the priority was not changed to the specified value (i.e. 0). Further analysis showed that the priority of the task was not affected by the call.	
Notes:	

A.1.1.7. RTEMS-TS-CL-TSKMOD: rtems_task_mode Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TSKMOD
Purpose:	To test rtems_task_mode by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-tsk.c
	Lines: [280-282]
	`returnStatus = rtems_task_mode (RTEMS_NO_PREEMPT,`
	`RTEMS_PREEMPT_MASK,`
	`&testTaskOldMode);`

Test Item:	rtems_task_mode (rtems_mode mode_set, rtems_mode mask, rtems_mode *previous_mode_set)
Generated Test Cases:	9

Test Cases Results

No faults were detected in this RTEMS task manager directive.

A.2. RTEMS Interrupt Manager Campaigns

A.2.1. RTEMS-CMP-CL-INT: Interrupt Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-INT
Purpose:	To test the robustness of rtems_interrupt_catch of the interrupt manager.
Workload File:	rtems-cmp-cl-int.c
Test Suites:	1. RTEMS-TS-CL-INTCTC
Workload Description:	
This workload defines a Interrupt Service Routine (ISR) that increments a global variable each time it is called. This ISR is attached to the Illegal Instruction trap (0x02). The assembly *unimp* mnemonic is used to generate an Illegal Intruction trap.	

A.2.1.1. RTEMS-TS-CL-INTCTC: rtems_interrupt_catch Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-INTCTC
Purpose:	To test rtems_interrupt_catch by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source File: rtems-cmp-cl-int.c
	Lines: [97-99]
	`returnStatus = rtems_interrupt_catch (testISR,`
	`vecNum,`
	`&oldISR);`

Test Item:	rtems_interrupt_catch (rtems_isr_entry new_isr_handler, rtems_vector_number vector, rtems_isr_entry *old_isr_handler)
Generated Test Cases:	5

Test Cases Results

No faults were detected in this RTEMS interrupt manager directive.

A.3. RTEMS Clock Manager Campaigns

A.3.1. RTEMS-CMP-CL-CLK: Clock Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-CLK
Purpose:	To test the robustness of all RTEMS Classic APIs related to Clock manager.
Workload File:	rtems-cmp-cl-clk.c
Test Suites:	1. RTEMS-TS-CL-CLKSET 2. RTEMS-TS-CL-CLKTCK
Workload Description: The main task of this workload is initialised with the preemption disabled. This means that it will be not preempted by other tasks. The task starts by setting the time of day, using the clock manager directives. Then, it gets the time of day and verifies that its value, with the exception of the milliseconds field, is equal to the value that was set before. The task also checks the number of seconds elapsed since 1998/01/01 0h0m0s until 2001/01/01 0h0m0s. It is assumed that the body of the task executes in less than one second.	

A.3.1.1. RTEMS-TS-CL-CLKSET: rtems_clock_set Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-CLKSET
Purpose:	To test rtems_clock_set by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-clk.c Lines: [194] `returnStatus = rtems_clock_set (&newTimeOfDay);`
Test Item:	rtems_clock_set (rtems_time_of_day *time_buffer)
Generated Test Cases:	42

Test Cases Results

No faults were detected in this RTEMS clock manager directive.

A.3.1.2. RTEMS-TS-CL-CLKTCK: rtems_clock_get Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-CLKGET
Purpose:	To test rtems_clock_get by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-clk.c Lines: [109] `rtems_clock_get (RTEMS_CLOCK_GET_TIME_VALUE, &ctTimeValue);`
Test Item:	rtems_clock_tick (void)
Generated Test Cases:	26

Test Cases Results

No faults were detected in this RTEMS clock manager directive.

A.3.2. RTEMS-CMP-PX-CLK: Clock Manager POSIX API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-PX-CLK
Purpose:	To test the robustness of the selected RTEMS POSIX directives related to the clock manager.
Workload File:	rtems-cmp-px-clk.c
Test Suites:	1. RTEMS-TS-PX-CLKCST 2. RTEMS-TS-PX-CLKGET 3. RTEMS-TS-PX-CLKSLP 4. RTEMS-TS-PX-CLKNNS

Workload Description:

This workload uses several clock manager directives, namely:

- *clock_gettime* - Obtain time of day
- *clock_settime* - Set time of day
- *sleep* – Delay process execution
- *nanosleep* – Delay with high resolution.

The delay functions (*sleep* and *nanosleep*) are asserted using the time elapsed in the function call: two calls to the *clock_gettime* are done just before and immediately after the delay function call. This way, and taking into account a small overhead, it is possible to calculate the actual delay time. This time is then compared with the expected delay (which is an argument to the *sleep* and to the *nanosleep* functions).

A.3.2.1. RTEMS-TS-PX-CLKCST: clock_settime Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-CLKCST
Purpose:	To test clock_settime by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-px-clk.c
	Lines: [114 - 115]
	`result_set = clock_settime (clock_id,`
	`&tp_set);`

Test Item:	int clock_settime(clockid_t clock_id, const struct timespec *tp)
Generated Test Cases:	8

Test Cases Results

No faults were detected in this RTEMS clock manager directive.

A.3.2.2. RTEMS-TS-PX-CLKGET: clock_gettime Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-CLKGET
Purpose:	To test clock_gettime by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-px-clk.c
	Lines: [105 - 106]
	`result_get = clock_gettime (clock_id,`
	`&tp_before_set);`

Test Item:	int clock_settime(clockid_t clock_id, const struct timespec *tp)
Generated Test Cases:	8

Test Cases Results

No faults were detected in this RTEMS clock manager directive.

A.3.2.3. RTEMS-TS-PX-CLKSLP: sleep Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-CLKSLP
Purpose:	To test sleep by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-clk.c Lines: [147] `sleep_result = sleep (sleep_time);`
Test Item:	unsigned int sleep(unsigned int seconds)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS clock manager directive.

A.3.2.4. RTEMS-TS-PX-CLKNNS: nanosleep Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-CLKNNS
Purpose:	To test nanosleep by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-px-clk.c
	Lines: [181 - 182]
	`sleep_result = nanosleep (&rqtp,`
	`&rmtp);`

Test Item:	Int nanosleep(const struct timespec *rqtp, struct timespec *rmtp)
Generated Test Cases:	13

Test Cases Results

No faults were detected in this RTEMS clock manager directive.

A.4. RTEMS Timer Manager Campaigns

A.4.1. RTEMS-CMP-CL-TMR: Timer Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-TMR
Purpose:	To test the robustness of all RTEMS Classic APIs related to Timer Manager manager.
Workload File:	rtems-cmp-cl-tmr.c
Test Suites:	1. RTEMS-TS-CL-TMRCRT 2. RTEMS-TS-CL-TMRDLT 3. RTEMS-TS-CL-TMRFAF 4. RTEMS-TS-CL-TMRWHN 5. RTEMS-TS-CL-TMRCNL 6. RTEMS-TS-CL-TMRRST
Workload Description: This workload defines a Timer Service Routine that increments a counter variable each time it is called. After creating the timer, it is programed several times in order to fire after a specified number of ticks. All these timer actions are monitored from the Timer Service Routine. Another directive that is used is the one that allows the deletion of a timer.	

A.4.1.1. RTEMS-TS-CL-TMRCRT: rtems_timer_create Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TMRCRT
Purpose:	To test rtems_timer_create by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-tmr.c Lines: [115] `returnStatus = rtems_timer_create (tmrName, &tmrId);`
Test Item:	rtems_timer_create (rtems_name name, Rtems_id *id)
Generated Test Cases:	21

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-TMRCRT-005

Input Specification:
`+ pttmrId = NULL;`
`returnStatus = rtems_timer_create (tmrName, pttmrId);`	

Failure Description: When the NULL value is provided as the id parameter, a data access exception occurs.	
Notes: No check agains a NULL value on the id parameter of the *rtems_timer_create* is performed. The following output is presented by the simulator: `Unexpected trap (0x09) at address 0x0200957c data access exception at 0x00000000`	

RTEMS-TS-CL-TMRDLT: rtems_timer_delete Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TMRDLT
Purpose:	To test rtems_timer_delete by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-tmr.c Lines: [269] `returnStatus = rtems_timer_delete(tmrId);`
Test Item:	rtems_timer_delete (rtems_id id)
Generated Test Cases:	6

Test Cases Results

No faults were detected in this RTEMS timer manager directive.

A.4.1.2. RTEMS-TS-CL-TMRFAF: rtems_timer_fire_after Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TMRFAF
Purpose:	To test rtems_timer_fire_after by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-tmr.c
	Lines: [134-137]
	`returnStatus = rtems_timer_fire_after (tmrId,`
	`TICKS_TO_ACTIVATION,`
	`pttmrSrvs,`
	`ptcnt);`

Test Item:	rtems_timer_fire_after (rtems_id id, rtems_internal ticks, rtems_timer_service_routine_entry routine, void *user_data)
Generated Test Cases:	8

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-TMRFAF-037

Input Specification:
`+ pttmrSrvs = NULL;`
`returnStatus = rtems_timer_fire_after (tmrId,`
`TICKS_TO_ACTIVATION,`
`pttmrSrvs,`
`Ptcnt);`	

Failure Description: When the NULL value is provided as the routine parameter, the first time that the Timer Service routine is called a trap is generated. There is an attempt to execute the instruction at address 0x00000000.	
Notes: No check against a NULL routine parameter of the *rtems_timer_fire_after* is performed.	

The following output is presented by the simulator:

```
Unexpected trap (0x02) at address 0x00000000 illegal instruction
```

A.4.1.3. RTEMS-TS-CL-TMRWHN : rtems_timer_fire_when Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TMRWHN
Purpose:	To test rtems_timer_fire_when by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-tmr.c Lines: [174-177] `returnStatus = rtems_timer_fire_when (tmrId, &timeStruct, pttmrSrvs, ptcnt);`
Test Item:	rtems_timer_fire_when (rtems_id id, Rtems_time_of_day wall_time, Rtems_timer_service_routine_entry routine, void *user_data);
Generated Test Cases:	26

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-TMRWHN-064
Input Specification:	`+ pttmrSrvs = NULL;` `returnStatus = rtems_timer_fire_when (tmrId, &timeStruct, pttmrSrvs, ptcnt);`
Failure Description:	When the NULL value is provided, the first time that the Timer Service routine is called a trap is generated. There is an attempt to execute the instruction at address 0x00000000.
Notes:	No check against a NULL routine parameter of the *rtems_timer_fire_when* is performed. The following output is presented by the simulator: `Unexpected trap (0x02) at address 0x00000000 illegal instruction`

A.4.1.4. RTEMS-TS-CL-TMRCNL: rtems_timer_cancel Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TMRCNL
Purpose:	To test rtems_timer_cancel by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-tmr.c Lines: [211] `returnStatus = rtems_timer_cancel(tmrId);`
Test Item:	rtems_timer_cancel (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS timer manager directive.

A.4.1.5. RTEMS-TS-CL-TMRRST: rtems_timer_reset Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-TMRRST
Purpose:	To test rtems_timer_reset by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-tmr.c Lines: [244] `returnStatus = rtems_timer_reset (tmrId);`
Test Item:	rtems_timer_reset (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS timer manager directive.

A.4.2. RTEMS-CMP-PX-TMR: Timer Manager POSIX API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-PX-TMR
Purpose:	To test the robustness of the selected RTEMS POSIX APIs related to the timer manager.
Workload File:	rtems-cmp-px-tmr.c
Test Suites:	1. RTEMS-TS-PX-TMRCRT 2. RTEMS-TS-PX-TMRDLT 3. RTEMS-TS-PX-TMRSTM
Workload Description:	
This workload contains one thread, POSIX_Init, and a signal handling function. The main thread will start by creating a new timer and by setting the signal handler as the handler to the SIGALRM signal, the signal that the timer sends to an application by default. Then, it will program the timer to fire after a predefined period of time (two seconds). After this time has elapsed, an assertion is done on whether the timer has fired or not. Finally, the timer is deleted.	

A.4.2.1. RTEMS-TS-PX-TMRCRT: timer_create Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-TMRCRT
Purpose:	To test timer_create by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-tmr.c Lines: [95] `result = timer_create(clockid, &timer_event_spec, &timerid);`
Test Item:	int timer_create(clockid_t clock_id, struct sigevent *evp, timer_t *timerid)
Generated Test Cases:	18

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-TMRCRT-003 (same results obtained in RTEMS-TCR-PX-TMR-TMRCRT-004)
Input Specification:	
`+ clockid = 1;` `result = timer_create (clockid, &timer_event_spec, &timerid);`	
Failure Description:	
POSIX specification defines only the CLOCK_REALTIME value for this parameter, leaving, the possibility to the existence of other clocks in a specific system. However, RTEMS does not define any other clock. For this reason, it should return -1 in this function call if the clockid parameter is different from CLOCK_REALTIME and set the *errno* to EINVAL. What RTEMS is doing is to return an error only when this parameter is 0 (zero), accepting any positive integer value as a clock ID.	
Notes:	

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-TMRCRT-006

Input Specification:
`+ timer_event_spec.sigev_notify = SIGEV_NONE;`
`result = timer_create (clockid, &timer_event_spec, &timerid);`	

Failure Description: Although a SIGEV_NONE has been specified in the *sigevent* data structure, a signal has been delivered to the application.	
Notes: POSIX defines only two possible values to this attribute (see notes): • SIGEV_SIGNAL and • SIGEV_NONE. The SIGEV_NONE is used when the goal is not to use anything for asynchronous notification. Note that it's possible the problem is inside other function (e.g. *timer_settime*) and not in the *timer_create*.	

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-TMRCRT-010 (same results obtained in RTEMS-TCR-PX-TMR-TMRCRT-012, RTEMS-TCR-PX-TMR-TMRCRT-013, RTEMS-TCR-PX-TMR-TMRCRT-014)

Input Specification:
`+ timer_event_spec.sigev_signo = 0;`
`result = timer_create (clockid, &timer_event_spec, &timerid);`	

Failure Description: No check on the *evt* parameter.	
Notes: Although not specified by the POSIX, it is usual to return EINVAL when this structure contains a signal that does not exist in the system.	

A.4.2.2. RTEMS-TS-PX-TMRDLT: timer_delete Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-TMRDLT
Purpose:	To test timer_delete by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-tmr.c Lines: [133] `Result = timer_delete(timerid);`
Test Item:	int timer_delete(timer_t timerid)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS timer manager directive.

A.4.2.3. RTEMS-TS-PX-TMRSTM: timer_settime Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-TMRSTM
Purpose:	To test the corresponding API by invoking it with the entire range of valid parameters.
Fault Location(s):	Source file: rtems-cmp-px-tmr.c Lines: [116] `result = timer_settime(timerid, flags, &setting, &old_setting);`
Test Item:	int timer_settime(timer_t timer_id, int flags, const struct itimerspec *value, struct itimerspec *ovalue)
Generated Test Cases:	8

Test Cases Results

No faults were detected in this RTEMS timer manager directive.

A.5. RTEMS Semaphore Manager Campaigns

A.5.1. RTEMS-CMP-CL-SMP: Semaphore Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-SMP
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the semaphore manager.
Workload File:	rtems-cmp-cl-smp.c
Test Suites:	1. RTEMS-TS-CL-SMPCRT 2. RTEMS-TS-CL-SMPDLT 3. RTEMS-TS-CL-SMPOBT 4. RTEMS-TS-CL-SMPRLS 5. RTEMS-TS-CL-SMPFLS
Workload Description:	
This workload as two tasks. The Init task performs all the main operations provided by the semaphore manager regarding the semaphore, namely the operations to create, delete, obtain, release and flush semaphores.	

A.5.1.1. RTEMS-TS-CL-SMPCRT: rtems_semaphore_create Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-SMPCRT
Purpose:	To test rtems_semaphore_create by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-smp.c
	Lines: [117- 121]
	`returnStatus = rtems_semaphore_create (semName,`
	`count,`
	`semAttr,`
	`semPriority,`
	`ptsemId);`

Test Item:	rtems_semaphore_create (rtems_name name, rtems_unsigned32 count, rtems_attribute attribute_set, rtems_task_priority priority_ceiling, rtems_id *id)
Generated Test Cases:	15

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-SMPCRT-014

Input Specification:
`+ ptsemId = NULL;`
`returnStatus = rtems_semaphore_create (semName, count, semAttr, semPriority, ptsemId);`	

Failure Description: When the NULL value is provided a data access exception occurs.	
Notes: No check against a NULL id parameter of the rtems_semaphore_create is performed. The following output is presented by the simulator: `Unexpected trap (0x09) at address 0x02005248 data access exception at 0x00000000`	

A.5.1.2. RTEMS-TS-CL-SMPDLT: rtems_semaphore_delete Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-SMPDLT
Purpose:	To test rtems_semaphore_delete by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-smp.c Lines: [251] `returnStatus = rtems_semaphore_delete (semId);`
Test Item:	rtems_semaphore_delete (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS semaphore manager directive.

A.5.1.3. RTEMS-TS-CL-SMPOBT: rtems_semaphore_obtain Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-SMPOBT
Purpose:	To test rtems_semaphore_obtain by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-smp.c Lines: [141 - 143] `returnStatus = rtems_semaphore_obtain (semId, option_set, timeout);`
Test Item:	rtems_semaphore_obtain (rtems_id id, rtems_unsigned32 option_set, rtems_interval timeout)
Generated Test Cases:	9

Test Cases Results

No faults were detected in this RTEMS semaphore manager directive.

A.5.1.4. RTEMS-TS-CL-SMPRLS: rtems_semaphore_release Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-SMPRLS
Purpose:	To test rtems_semaphore_release by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-smp.c Lines: [167] `returnStatus = rtems_semaphore_release(semId);`
Test Item:	rtems_semaphore_release (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS semaphore manager directive.

A.5.1.5. RTEMS-TS-CL-SMPFLS: rtems_semaphore_flush Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-SMPFLS
Purpose:	To test rtems_semaphore_flush by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-smp.c Lines: [225] `returnStatus = rtems_semaphore_flush (semId);`
Test Item:	rtems_semaphore_flush (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS semaphore manager directive.

A.6. RTEMS Message Manager Campaigns

A.6.1. RTEMS-CMP-CL-MSG: Message Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-MSG
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the message manager.
Workload File:	rtems-cmp-cl-msg.c
Test Suites:	1. RTEMS-TS-CL-MSGCRT 2. RTEMS-TS-CL-MSGDLT 3. RTEMS-TS-CL-MSGSND 4. RTEMS-TS-CL-MSGURG 5. RTEMS-TS-CL-MSGBRD 6. RTEMS-TS-CL-MSGRCV 7. RTEMS-TS-CL-MSGFSH
Workload Description: This workload contains three tasks. The main task (Init) creates two queues and two test tasks. One of the queues is used by the main task to send messages to the test tasks and the other queue is used by the test tasks to send messages to the main task. The first task created (TA1) has a higher priority then the second task created (TA2). This workload uses the messages priority: normal and urgent messages.	

A.6.1.1. RTEMS-TS-CL-MSGCRT: rtems_message_queue_create Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-MSGCRT
Purpose:	To test rtems_message_queue_create by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-msg.c
	Lines: [188 - 192]
	`returnStatus = rtems_message_queue_create (msgQueueFromTasksName,`
	`MaxMessages,`
	`MaxMsgSize,`
	`attribute,`
	`ptmsgQueueFromTasksId);`

Test Item:	rtems_message_queue_create (rtems_name name, rtems_unsigned32 count, rtems_unsigned32 max_message_size, rtems_attribute attribute_set, rtems_id *id)
Generated Test Cases:	31

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-MSGCRT-002

Input Specification:

```
+ msgQueueToTasksName = 0;
     returnStatus = rtems_message_queue_create (msgQueueToTasksName,
                                                MaxMessages,
                                                MaxMsgSize,
                                                attribute,
                                                ptmsgQueueToTasksId);
```

Failure Description:

No error code is returned by the *rtems_message_queue_create* RTEMS directive. However, the next calls that operate on that queue fail, returning RTEMS_INVALID_ID meaning that the queue was not properly created.

Notes:

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-MSGCRT-007

Input Specification:

```
MaxMessages = 4294967295;
     returnStatus = rtems_message_queue_create (msgQueueToTasksName,
                                                MaxMessages,
                                                MaxMsgSize,
                                                attribute,
                                                ptmsgQueueToTasksId);
```

Failure Description:

Error code return by the *rtems_message_queue_create* RTEMS directive is not consistent neither with the documentation nor with the results of other generated test cases. It should return RTEMS_INVALID_NUMBER meaning that the parameter count is invalid. Instead it returns RTEMS_TOO_MANY. The later error code means that the maximum number of message queues was reached.

Notes:

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-MSGCRT-010

Input Specification:

```
MaxMsgSize = 4294967295;
     returnStatus = rtems_message_queue_create (msgQueueToTasksName,
                                                MaxMessages,
                                                MaxMsgSize,
                                                attribute,
                                                ptmsgQueueToTasksId);
```

Failure Description:

When attempting to create a message queue with a message size of 4294967295, a memory address not aligned exception occurs. Instead, the *rtems_message_queue_create* RTEMS directive should have returned RTEMS_INVALID_SIZE.

Notes:

The simulator returned the following output:

```
Unexpected trap (0x07) at address 0x02009f4c
memory address not aligned
```

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-MSGCRT-014

Input Specification:
`ptmsgQueueToTasksId = NULL;`
`returnStatus = rtems_message_queue_create (msgQueueToTasksName,`
`MaxMessages,`
`MaxMsgSize,`
`attribute,`
`ptmsgQueueToTasksId);`	

Failure Description: When a NULL pointer is given as argument a data exception occurs.	

Notes:
No check agains a NULL value on the id parameter of the *rtems_message_queue_create* RTEMS directive.
The simulator returned the following output:
`Unexpected trap (0x09) at address 0x0200336c`
`data access exception at 0x00000000`	

A.6.1.2. RTEMS-TS-CL-MSGDLT: rtems_message_queue_delete Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-MSGDLT
Purpose:	To test rtems_message_queue_delete by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-msg.c Lines: [378] `returnStatus = rtems_message_queue_delete (msgQueueFromTasksId);`
Test Item:	rtems_message_queue_delete (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS message manager directive.

A.6.1.3. RTEMS-TS-CL-MSGSND: rtems_message_queue_send Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-MSGSND
Purpose:	To test rtems_message_queue_send by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-msg.c
	Lines: [243 - 245]
	`returnStatus = rtems_message_queue_send (msgQueueToTasksId,`
	`msgSendBuf,`
	`msgSendSize);`

Test Item:	rtems_message_queue_send (rtems_id id, void *buffer, rtems_unsigned32 size)
Generated Test Cases:	7

Test Cases Results

No faults were detected in this RTEMS message manager directive.

A.6.1.4. RTEMS-TS-CL-MSGURG: rtems_message_queue_urgent Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-MSGURG
Purpose:	To test rtems_message_queue_urgent by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-msg.c Lines: [259 - 261] `returnStatus = rtems_message_queue_urgent (msgQueueToTasksId, msgSendBuf, msgSendSize);`
Test Item:	rtems_message_queue_urgent (rtems_id id, void * buffer, rtems_unsigned32 size)
Generated Test Cases:	7

Test Cases Results

No faults were detected in this RTEMS message manager directive.

A.6.1.5. RTEMS-TS-CL-MSGBRD: rtems_message_queue_broadcast Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-MSGBRD
Purpose:	To test rtems_message_queue_broadcast by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-msg.c Lines: [339 - 342] `returnStatus = rtems_message_queue_broadcast (msgQueueToTasksId, msgSendBuf, msgSendSize, ptnMessages);`
Test Item:	rtems_message_queue_broadcast (rtems_id id, void * buffer, rtems_unsigned32 size, rtems_unsigned32 *count)
Generated Test Cases:	10

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-MSGBRD-045

Input Specification:
`ptnMessages = NULL;`
`returnStatus = rtems_message_queue_broadcast (msgQueueToTasksId,`
`msgSendBuf,`
`msgSendSize,`
`ptnMessages);`	

Failure Description: When a NULL pointer is given as argument a data exception occurs.	

Notes:
No check is performed against a NULL value on the count parameter of the rtems_message_queue_broadcast rtems directive.
The simulator returned the following output:
`Unexpected trap (0x09) at address 0x0200a234`
`Data access exception at 0x00000000`	

A.6.1.6. RTEMS-TS-CL-MSGRCV: rtems_message_queue_receive Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-MSGRCV
Purpose:	To test rtems_message_queue_receive by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-msg.c
	Lines: [273 - 277]
	`returnStatus = rtems_message_queue_receive (msgQueueFromTasksId,`
	`msgReceiveBuf,`
	`ptmsgReceiveSize,`
	`option,`
	`timeout);`

Test Item:	rtems_message_queue_receive (rtems_id id, void *buffer, rtems_unsigned32 *size, rtems_unsigned32 option_set, rtems_interval timeout)
Generated Test Cases:	13

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-MSGRCV-053
Input Specification:	

`ptmsgReceiveSize = NULL;`
`returnStatus = rtems_message_queue_receive (msgQueueFromTasksId,`
`msgReceiveBuf,`
`ptmsgReceiveSize,`
`option,`
`timeout);`	

Failure Description:	
When a NULL pointer is given as argument a data exception occurs.	
Notes:	

No check is performed against a NULL value on the count parameter of the *rtems_message_queue_broadcast* RTEMS directive.
The simulator returned the following output:
`Unexpected trap (0x09) at address 0x0200a490`
`data access exception at 0x00000000`	

A.6.1.7. RTEMS-TS-CL-MSGFSH: rtems_message_queue_flush Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-MSGFSH
Purpose:	To test rtems_message_queue_flush by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-msg.c
	Lines: [313 - 314]
	`returnStatus = rtems_message_queue_flush(msgQueueToTasksId,`
	`ptnMessages);`

Test Item:	rtems_message_queue_flush (rtems_id id, rtems_unsigned32 *count)
Generated Test Cases:	6

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-MSGFSH-073
Input Specification:	

`ptnMessages = NULL;`
`returnStatus = rtems_message_queue_flush(msgQueueToTasksId,`
`ptnMessages);`	

Failure Description:	
When a NULL pointer is given as argument a data exception occurs.	
Notes:	

No check is performed against a NULL value on the count parameter of the *rtems_message_queue_get_flush* RTEMS directive.
The simulator returned the following output:
`Unexpected trap (0x09) at address 0x02003498`
`Data access exception at 0x00000000`	

A.6.1.8. RTEMS-TS-CL-MSGGNP: rtems_message_queue_get_number_pending Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-MSGGNP
Purpose:	To test rtems_message_queue_get_number_pending by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-msg.c Lines: [294 - 295] `returnStatus = rtems_message_queue_get_number_pending(msgQueueToTasksId, ptnMessages);`
Test Item:	rtems_message_queue_create (rtems_name name, rtems_unsigned32 count, rtems_unsigned32 max_message_size, rtems_attribute attribute_set, rtems_id *id)
Generated Test Cases:	6

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-MSGGNP-066
Input Specification:	`ptnMessages = NULL;` `returnStatus = rtems_message_queue_get_number_pending(msgQueueToTasksId, ptnMessages);`
Failure Description:	When a NULL pointer is given as argument a data exception occurs.

Notes:	No check is performed against a NULL value on the count parameter of the *rtems_message_queue_get_number_pending* RTEMS directive.
	The simulator returned the following output:
	`Unexpected trap (0x09) at address 0x02003504`
	`Data access exception at 0x00000000`

A.6.2. RTEMS-CMP-PX-MSG: Message Manager POSIX API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-PX-MSG
Purpose:	To test the robustness of the selected RTEMS POSIX APIs related to the message manager.
Workload File:	rtems-cmp-px-msg.c
Test Suites:	1. RTEMS-TS-PX-MSGOPN 2. RTEMS-TS-PX-MSGCLS 3. RTEMS-TS-PX-MSGULK 4. RTEMS-TS-PX-MSGSND 5. RTEMS-TS-PX-MSGRCV 6. RTEMS-TS-PX-MSGNTF 7. RTEMS-TS-PX-MSGSAT

Workload Description:

This workload uses the selected directives from the message manager. It was designed to test the following functionalities:

- Create of a message queue;
- Send and receive a message in a message queue;
- Notify a message;
- Set message queue attributes;
- Close and unlink a message queue.

This application uses one single task and a signal handler (to the message notification).

A.6.2.1. RTEMS-TS-PX-MSGOPN: mq_open Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MSGOPN
Purpose:	To test mq_open by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-msg.c Lines: [111] mq = mq_open(QUEUE_NAME, oflag, mode, &attr);
Test Item:	mqd_t mq_open(const char *name, int oflag, mode_t mode, struct mq_attr *attr)
Generated Test Cases:	33

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-MSGOPN-009 (same results obtained in RTEMS-TCR-PX-MSGOPN-010, RTEMS-TCR-PX-MSGOPN-011, RTEMS-TCR-PX-MSGOPN-012, RTEMS-TCR-PX-MSGOPN-013)

Input Specification:
`+ mode = 0;`
`mq = mq_open (QUEUE_NAME, oflag, mode, &attr);`	

Failure Description: The *mode* argument is ignored by all the calls. If the message queue is created with these permissions (no permissions at all for every user), then it should not be possible to read or write from it.	
Notes:	

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-MSGOPN-019 (same results obtained in RTEMS-TCR-PX-MSGOPN-023)

Input Specification:
`+ attr.mq_maxmsg = 0;`
`mq = mq_open (QUEUE_NAME, oflag, mode, &attr);`	

Failure Description: If the maximum number of messages is set to 0 (zero), the call to the *mq_open* does not return the error message EINVAL.	
Notes: The expected error, according to the POSIX specification, that should be returned by this directive whenever the *mq_attr* structure has the *mq_maxmsg* attribute less than or equal to 0 is EINVAL. As a result of this, the application will block when a *mq_send* call (send a message to a message queue) is executed.	

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-MSGOPN-024 (same results obtained in RTEMS-TCR-PX-MSGOPN-025, RTEMS-TCR-PX-MSGOPN-028)

Input Specification:
`+ attr.mq_maxsize = 0;`
`mq = mq_open (QUEUE_NAME, oflag, mode, &attr);`	

Failure Description: If the messages maximum size is set to 0 (zero), the call to the *mq_open* does not return the error message EINVAL.	
Notes: The expected error, according to the POSIX specification, that should be returned by this directive whenever the *mq_attr* structure has the *mq_maxsize* attribute less than or equal to 0 is EINVAL. As a result of this, the application will block when a *mq_send* call (send a message to a message queue) is executed.	

A.6.2.2. RTEMS-TS-PX-MSGCLS: mq_close Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MSGCLS
Purpose:	To test mq_close by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-msg.c Lines: [207] `result = mq_close(mq);`
Test Item:	Int mq_close(mqd_t mqdes)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS message manager directive.

A.6.2.3. RTEMS-TS-PX-MSGULK: mq_unlink Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MSGULK
Purpose:	To test mq_unlink by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-msg.c Lines: [195] result = mq_unlink(QUEUE_NAME);
Test Item:	Int mq_unlink(const char *name)
Generated Test Cases:	2

Test Cases Results

No faults were detected in this RTEMS message manager directive.

A.6.2.4. RTEMS-TS-PX-MSGSND: mq_send Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MSGSND
Purpose:	To test mq_send by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-msg.c Lines: [140] result = mq_send(mq, msg, msg_len, msg_prio);
Test Item:	Int mq_send(mqd_t mqdes, const char *msg_ptr, size_t msg_len, unsigned int msg_prio)
Generated Test Cases:	13

Test Cases Results

No faults were detected in this RTEMS message manager directive.

A.6.2.5. RTEMS-TS-PX-MSGRCV: mq_receive Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MSGRCV
Purpose:	To test mq_receive by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-msg.c Lines: [162] result = mq_receive(mq, buffer, buf_len, &msg_prio);
Test Item:	size_t mq_receive(mqd_t mqdes, char *sg_ptr, size_t msg_len, Unsigned int *msg_prio)
Generated Test Cases:	13

Test Cases Results

No faults were detected in this RTEMS message manager directive.

A.6.2.6. RTEMS-TS-PX-MSGNTF: mq_notify Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MSGNTF
Purpose:	To test mq_notify by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-msg.c Lines: [134] result = mq_notify(mq, &notification);
Test Item:	int mq_notify(mqd_t mqdes, const struct sigevent *notification)
Generated Test Cases:	15

Test Cases Results

No faults were detected in this RTEMS message manager directive.

A.6.2.7. RTEMS-TS-PX-MSGSAT: mq_setattr Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MSGSAT
Purpose:	To test mq_setattr by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-msg.c Lines: [177] result = mq_setattr(mq, &new_attr, &old_attr);
Test Item:	int mq_setattr(mqd_t mqdes, const struct mq_attr *mqstat, struct mq_attr *omqstat)
Generated Test Cases:	43

Test Cases Results

No faults were detected in this RTEMS message manager directive.

A.7. RTEMS Event Manager Campaigns

A.7.1. RTEMS-CMP-CL-EVT: Event Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-EVT
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the event manager.
Workload File:	rtems-cmp-cl-evt.c
Test Suites:	1. RTEMS-TS-CL-EVTSND 2. RTEMS-TS-CL-EVTRCV
Workload Description: This workload has two tasks.The main task sends an event to the second task. After receiving the event, it sends the same event again to the main task.	

A.7.1.1. RTEMS-TS-CL-EVTSND: rtems_event_send Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-EVTSND
Purpose:	To test rtems_event_send by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-evt.c Lines: [167 - 168] `returnStatus = rtems_event_send (taskId, evtSet);`
Test Item:	rtems_event_send (rtems_id id, rtems_event_set event_in)
Generated Test Cases:	6

Test Cases Results

No faults were detected in this RTEMS event manager directive.

A.7.1.2. RTEMS-TS-CL-EVTRCV: rtems_event_receive Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-EVTRCV
Purpose:	To test rtems_event_receive by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-evt.c Lines: [172 - 175] `returnStatus = rtems_event_receive (evtToReceive, optionSet, timeout, ptevtReceived);`
Test Item:	rtems_event_receive (rtems_event_set event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out)
Generated Test Cases:	12

Test Cases Results

No faults were detected in this RTEMS event manager directive.

A.8. RTEMS Signal Manager Campaigns

A.8.1. RTEMS-CMP-CL-SGL: Signal Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-SGL
Purpose:	To test the robustness of all RTEMS Classic APIs related to the corresponding manager.
Workload File:	Rtems-cmp-cl-sgl.c
Test Suites:	1. RTEMS-TS-CL-SGLCTH **Test Cases Results** No faults were detected in this RTEMS signal manager directive. 2. RTEMS-TS-CL-SGLSNDRTEMS-TS-CL-SGLSND
Workload Description:	This workload contains only one task. This main task, Init, performs all tests of the signal manager. It just establishes an ASR (Asynchronous Signal Routine) and send a signal to itself.

A.8.1.1. RTEMS-TS-CL-SGLCTH: rtems_signal_catch Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-SGL-CL-SGLCTH
Purpose:	To test rtems_signal_catch by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-sgl.c Lines: [111 - 112] `returnStatus = rtems_signal_catch (ptsigHndlr, mode);`
Test Item:	rtems_signal_catch (rtems_asr_entry asr_handler, rtems_mode mode)
Generated Test Cases:	4

Test Cases Results

No faults were detected in this RTEMS signal manager directive.

A.8.1.2. RTEMS-TS-CL-SGLSND: rtems_signal_send Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-SGLSND
Purpose:	To test rtems_signal_send by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-sgl.c Lines: [135] `returnStatus = rtems_signal_send (id, signalSet);`
Test Item:	rtems_signal_send (rtems_id id, rtems_signal_set signal_set)

Test Cases Results

TEST CASE RESULT
Test case result identifier: RTEMS-TCR-CL-SGLSND-010

Input Specification:
`signalSet = 0;`
`returnStatus = rtems_signal_send (id, signalSet);`

Failure Description: No error is returned by the *rtems_signal_send* directive when sending signal 0 (zero). However no signal is received by the signal error routine.
Notes:

A.8.2. RTEMS-CMP-PX-SGL: Signal Manager POSIX API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-PX-SGL
Purpose:	To test the robustness of the selected RTEMS POSIX APIs related to the signal manager.
Workload File:	rtems-cmp-px-sgl.c
Test Suites:	1. RTEMS-TS-PX-SGLSAS 2. RTEMS-TS-PX-SGLSDS 3. RTEMS-TS-PX-SGLSFS 4. RTEMS-TS-PX-SGLSES 5. RTEMS-TS-PX-SGLSAC 6. RTEMS-TS-PX-SGLPTK 7. RTEMS-TS-PX-SGLSPM 8. RTEMS-TS-PX-SGLKIL 9. RTEMS-TS-PX-SGLSUS 10. RTEMS-TS-PX-SGLSWI 11. RTEMS-TS-PX-SGLSTO

Workload Description:

This workload is composed by one main thread (POSIX_Init), two other helper threads and signal handlers. Besides the signal set operations (fill, empty, set and delete signal set), it evokes other signal manager directives. These directives include:

- Assign signal handlers to specific signals;
- Send signals;
- Receive signals;
- Modification of the signal mask;
- Block/unblock signals;
- Suspend execution until signal arrival;
- Suspend execution until signal arrival or a predefined period of time expires.

The helper threads are used to send signals to the main thread or to receive signals from the main thread.

A.8.2.1. RTEMS-TS-PX-SGLSAS : sigaddset Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-SGLSAS
Purpose:	To test sigaddset by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [185] `Res = sigaddset (&set, signo);`
Test Item:	int sigaddset (sigset_t *seg, int signo)
Generated Test Cases:	8

Test Cases Results

No faults were detected in this RTEMS signal manager directive.

A.8.2.2. RTEMS-TS-PX-SGLSDS sigdelset Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-SGLSDS
Purpose:	To test sigdelset by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [205] `res = sigdelset (&set, signo);`
Test Item:	int sigdelset(sigset_t *set, int signo)
Generated Test Cases:	8

Test Cases Results

No faults were detected in this RTEMS signal manager directive.

A.8.2.3. RTEMS-TS-PX-SGLSFS : sigfillset Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-SGLSFS
Purpose:	To test sigfillset by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [155] `res = sigfillset (&set);`
Test Item:	int sigfillset(sigset_t *set)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS signal manager directive.

A.8.2.4. RTEMS-TS-PX-SGLSES: sigemptyset Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-SGLSES
Purpose:	To test sigemptyset by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [170] `res = sigemptyset (&set);`
Test Item:	int sigemptyset(sigset_t *set)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS signal manager directive.

A.8.2.5. RTEMS-TS-PX-SGLSAC: sigaction Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-SGLSAC
Purpose:	To test sigaction by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [228] `res = sigaction (sig, &act, &old_act);`
Test Item:	int sigaction(int sig, const struct aigaction *act, struct sigaction *oact)
Generated Test Cases:	25

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-SLGSAC-028
Input Specification:	`+ sig = 0;` `res = sigaction (sig,&act,&old_act);`
Failure Description:	When called with a signal 0 (zero), the *sigaction* function returns successfully, while POSIX specification defines that it shall return -1 and set *errno* to EINVAL if the signal number is invalid. Since there is no signal 0, the return value should be -1.
Notes:	Very similar results were obtained in the RTEMS Classic API for this manager.

A.8.2.6. RTEMS-TS-PX-SGLPTK: pthread_kill Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-SGLPTK
Purpose:	To test pthread_kill by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [353] `res = pthread_kill (thread, sig);`
Test Item:	int pthread_kill(pthread_t thread, int sig)
Generated Test Cases:	18

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-SGLPTK-057

Input Specification:
`+ sig = 0;`
`res = pthread_kill (thread, sig);`	

Failure Description: A call to the *pthread_kill* function with a signal 0 (zero) returns successfully, while POSIX specification defines that it shall return -1 if the signal number is invalid. Since there is no signal 0, the return value should be -1.	
Notes:	

A.8.2.7. RTEMS-TS-PX-SGLSPM: sigprocmask Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-SGLSPM
Purpose:	To test sigprocmask by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [276] `res = sigprocmask (how, &set, &old_set);`
Test Item:	int sigprocmask(int How, const sigset_t *set, Sigset_t *oset)
Generated Test Cases:	11

Test Cases Results

No faults were detected in this RTEMS signal manager directive.

A.8.2.8. RTEMS-TS-PX-SGLKIL: kill Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-SGLKIL
Purpose:	To test kill by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [238] `Res = kill (pid, sig);`
Test Item:	int kill(pid_t pid, int sig)
Generated Test Cases:	10

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-SGLKIL-090

Input Specification:
`+ sig = 0;`
`res = kill (pid,sig);`	

Failure Description: A call to the *kill* function with a signal 0 (zero) returns successfully, while POSIX specification defines that it shall return -1 if the signal number is invalid. Since there is no signal 0 in the RTEMS system, the return value should be -1 and the *errno* should be set to EINVAL.	
Notes:	

A.8.2.9. RTEMS-TS-PX-SGLSUS: sigsuspend Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-SGLSUS
Purpose:	To test sigsuspend by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [321] `res = sigsuspend (&set);`
Test Item:	int sigsuspend(const sigset_t *sigmask)
Generated Test Cases:	3

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-SGLSUS-001 (see all test cases)
Input Specification: `res = sigsuspend (&set);`	
Failure Description: Error in *sigsuspend* function implementation. This function, according to POSIX specification, never returns unless a signal arrives. In the manner of all blocking POSIX functions which are interrupted by signals, *sigsuspend* returns -1, with *errno* set to EINTR when it is interrupted by a signal. However, the RTEMS implementation does not return -1 but the signal number that it received.	
Notes:	

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-SGLSUS-096 (same results obtained in RTEMS-TCR-PX-SGLSUS-097, RTEMS-TCR-PX-SGLSUS-098)

Input Specification:
`+ set = 0;`
`res = sigsuspend (&set);`	

Failure Description: POSIX specifies that the *sigsuspend* function replaces the process' signal mask with the set of signals pointed to by the *set* argument and suspends the process until delivery of a signal whose action is either to execute a signal handler or to terminate the process. As the *set* parameter in the mutant has been changed to 0 (zero), the process should suspend forever (or until a signal that can not be blocked arrives, e.g. SIGKILL) and never return. However, the function is always returning after receiving a SIGUSR1 signal.	
Notes:	

A.8.2.10. RTEMS-TS-PX-SGLSWI: sigwaitinfo Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-SGL-PX-SGLSWI
Purpose:	To test sigwaitinfo by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [383] `res = sigwaitinfo (&set, &siginfo);`
Test Item:	int sigwaitinfo(const sigset_t *set, siginfo_t *info)
Generated Test Cases:	14

Test Cases Results

No faults were detected in this RTEMS signal manager directive.

A.8.2.11. RTEMS-TS-PX-SGLSTO: sigtimedwait Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-SGLSTO
Purpose:	To test sigtimedwait by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-sgn.c Lines: [406] `res = sigtimedwait (&set, &siginfo, &timeout);`
Test Item:	int sigtimedwait(const sigset_t *set, sifinfo_t *info, const struct timespec *timeout)
Generated Test Cases:	19

Test Cases Results

No faults were detected in this RTEMS signal manager directive.

A.9. RTEMS Partition Manager Campaigns

A.9.1. RTEMS-CMP-CL-PRT: Partition Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-PRT
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the partition manager.
Workload File:	Rtems-cmp-cl-prt.c
Test Suites:	1. RTEMS-TS-CL-PRTCRT 2. RTEMS-TS-CL-PRTGBF 3. RTEMS-TS-CL-PRTRBF 4. RTEMS-TS-CL-PRTDLT
Workload Description: This workload has only one main task. This task is used to perform all tests for the partition manager. These tests are: • Creation of an RTEMS partition; • Allocate buffer from partition; • Re-allocate the same buffer from partition; • Delete partition.	

A.9.1.1. RTEMS-TS-CL-PRTCRT: rtems_partition_create Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-PRTCRT
Purpose:	To test rtems_partition_create by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-prt.c
	Lines: [109 - 114]
	`returnStatus = rtems_partition_create (partitionName,`
	`startAddress,`
	`length,`
	`bufferSize,`
	`attributes,`
	`ptpartitionId);`

Test Item:	rtems_partition_create (rtems_name name, void *starting_address, rtems_unsigned32 length, rtems_unsigned32 buffer_size, rtems_attribute attribute_set, rtems_id *id)
Generated Test Cases:	16

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-PRTCRT-015
Input Specification:	

`ptpartitionId = NULL;`
`returnStatus = rtems_partition_create (partitionName,`
`startAddress,`
`length,`
`bufferSize,`
`attributes,`
`ptpartitionId);`	

Failure Description:	
When a NULL pointer is given as argument, a data exception occurs.	
Notes:	

No check is performed against a NULL value on the id parameter of the *rtems_partition_create* RTEMS directive.
The simulator returns the following output:
`Unexpected trap (0x09) at address 0x02003520`
`Data access exception at 0x00000000`	

A.9.1.2. RTEMS-TS-CL-PRTGBF: rtems_partition_get_buffer Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-PRTGBF
Purpose:	To test rtems_partition_get_buffer by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-prt.c
	Lines: [141 - 142]
	`returnStatus = rtems_partition_get_buffer(partitionId,`
	`ptfirstBuffer);`

Test Item:	rtems_partition_get_buffer (rtems_id id, void **buffer)
Generated Test Cases:	4

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-PRTGBF-022
Input Specification:	

`ptfirstBuffer = NULL;`
`returnStatus = rtems_partition_get_buffer(partitionId,`
`ptfirstBuffer);`	

Failure Description:	
When a NULL pointer is given as argument a data exception occurs.	
Notes:	

No check is performed against a NULL value on the id parameter of the rtems_partition_get_buffer rtems directive.
The simulator returns the following output:
`Unexpected trap (0x09) at address 0x020036c4`
`data access exception at 0x00000000`	

A.9.1.3. RTEMS-TS-CL-PRTRBF: rtems_partition_return_buffer Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-PRTRBF
Purpose:	To test rtems_partition_return_buffer by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-prt.c Lines: [163 - 164] `returnStatus = rtems_partition_return_buffer(partitionId, firstBuffer);`
Test Item:	rtems_partition_return_buffer (rtems_id id, void *buffer)
Generated Test Cases:	4

Test Cases Results

No faults were detected in this RTEMS partitionmanager directive.

A.9.1.4. RTEMS-TS-CL-PRTDLT: rtems_partition_delete Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-PRTDLT
Purpose:	To test rtems_partition_delete by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-prt.c Lines: [206] `returnStatus = rtems_partition_delete (partitionId);`
Test Item:	rtems_partition_delete (rtems_id id)
Generated Test Cases:	4

Test Cases Results

No faults were detected in this RTEMS partition manager directive.

A.10. RTEMS Region Manager Campaigns

A.10.1. RTEMS-CMP-CL-RGN: Region Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-RGN
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the region manager.
Workload File:	Rtems-cmp-cl-rgn.c
Test Suites:	1. RTEMS-TS-CL-RGNCRT 2. RTEMS-TS-CL-RGNGSG 3. RTEMS-TS-CL-RGNGSS 4. RTEMS-TS-CL-RGNEXT 5. RTEMS-TS-CL-RGNDLT 6. RTEMS-TS-CL-RGNRSG

Workload Description:

This workload only has one main task. This task performs all tests of the region manager . It executes the following region manager related operations:

- Create a region;
- Get a segment from region;
- Return the segment to region;
- Extend region;
- Delete region.

Important Remark:

The *rtems_region_extend* does not run successfully since the *rtems_region_extend* directive is not implemented.

A.10.1.1. RTEMS-TS-CL-RGNCRT: rtems_region_create Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-RGNCRT
Purpose:	To test rtems_region_create by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-rgn.c
	Lines: [129 - 134]
	`returnStatus = rtems_region_create (regionName,`
	`(void *)startAddress,`
	`initialMemBlockLen,`
	`page_size,`
	`attributes,`
	`ptregionId);`

Test Item:	rtems_region_create (rtems_name name, void *starting_address, rtems_unsigned32 length, rtems_unsigned32 page_size, rtems_attribute attribute_set, rtems_id *id)
Generated Test Cases:	16

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-RGNCRT-015
Input Specification:	
ptregionId = NULL; returnStatus = rtems_region_create (regionName, (void *)startAddress, initialMemBlockLen, page_size, attributes, ptregionId);	
Failure Description:	
When a NULL pointer is given as argument a data exception occurs.	
Notes:	
No check is performed against a NULL value on the id parameter of the rtems_region_create rtems directive. The simulator returns the following output: Unexpected trap (0x09) at address 0x02003520 data access exception at 0x00000000	

A.10.1.2. RTEMS-TS-CL-RGNGSG: rtems_region_get_segment Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-RGNGSG
Purpose:	To test rtems_region_get_segment by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-rgn.c Lines: [155 - 159] returnStatus = rtems_region_get_segment (regionId, requestedSize1, option, timeout, ptsegment1);
Test Item:	rtems_region_get_segment (rtems_id *id, rtems_unsigned32 size, rtems_option option_set, rtems_interval timeout, void **segment)
Generated Test Cases:	17

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-RGNGSG-022 (same results obtained in RTEMS-TCR-CL-RGNGSG-024)

Input Specification:

```
requestedSize1 = 0;
     returnStatus = rtems_region_get_segment (regionId,
                                              requestedSize1,
                                              option,
                                              timeout,
                                              ptsegment1);
```

Failure Description:

A Memory Exception occurs while attempting to retrieve a segment of size zero.The same happens when attempting to retrieve a segment of size 4294967295.

Notes:

The simulator returns the following output:

```
Memory exception at fffffffc (illegal address)
Unexpected trap (0x09) at address 0x0200aaac
Data access exception at 0xfffffffc
```

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-RGNGSG-031

Input Specification:

```
ptsegment1 = NULL;
     returnStatus = rtems_region_get_segment (regionId,
                                              requestedSize1,
                                              option,
                                              timeout,
                                              ptsegment1);
```

Failure Description:

When a NULL pointer is given as argument a data exception occurs.

Notes:

No check is performed against a NULL value on the segment parameter of the rtems_region_get_segment rtems directive.

The simulator returns the following output:

```
Unexpected trap (0x09) at address 0x02004e64
data access exception at 0x00000000
```

A.10.1.3. RTEMS-TS-CL-RGNGSS: rtems_region_get_segment_size Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-RGNGSS
Purpose:	To test rtems_region_get_segment_size by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-rgn.c
	Lines: [177 - 179]
	`returnStatus = rtems_region_get_segment_size(regionId,`
	`segment1,`
	`ptsegment1Size);`

Test Item:	rtems_region_get_segment_size (rtems_id *id, void *segment, rtems_unsigned32 *size)
Generated Test Cases:	20

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-RGNGSS-040

Input Specification:	`segment1 = NULL;`
	`returnStatus = rtems_region_get_segment_size(regionId,`
	`segment1,`
	`ptsegment1Size);`

Failure Description:	A Memory Exception occurs when NULL value is give as the address of the segment.

Notes:	This directive should return RTEMS_INVALID_ADDRESS meaning that the specified segment does not belong to the specified region.
	The following output is returned by the simulator:
	`Memory exception at fffffffc (illegal address)`
	`Unexpected trap (0x09) at address 0x0200aab4`
	`data access exception at 0xfffffffc`

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-RGNGSS-041

Input Specification:	`segment1 = 1;`
	`returnStatus = rtems_region_get_segment_size(regionId,`
	`segment1,`
	`ptsegment1Size);`

Failure Description:	A Memory Address Not Aligned Exception occurs when value 1 is give as the address of the segment.

Notes:	This directive should return RTEMS_INVALID_ADDRESS meaning that the specified segment does not belong to the specified region.
	The following output is returned by the simulator:
	`Unexpected trap (0x07) at address 0x0200aab8`
	`memory address not aligned`

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-RGNGSS-043

Input Specification:
`ptsegment1Size = NULL;`
`returnStatus = rtems_region_get_segment_size(regionId,`
`segment1,`
`ptsegment1Size);`	

Failure Description: When a NULL pointer is given as argument a data exception occurs.	

Notes:
No check is performed against a NULL value on the segment parameter of the *rtems_region_get_segment_size* rtems directive.
The following output is returned by the simulator:
`Unexpected trap (0x09) at address 0x0200aafc`
`Data access exception at 0x00000000`	

A.10.1.4. RTEMS-TS-CL-RGNEXT: rtems_region_extend Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-RGNEXT
Purpose:	To test rtems_region_extend by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-rgn.c
	Lines: [225 - 227]
	`returnStatus = rtems_region_extend (regionId,`
	`extendAddress,`
	`extentionMemBlockLen);`

Test Item:	rtems_region_extend (rtems_id id, void *starting_address, rtems_unsigned32 length)
Generated Test Cases:	7

Test Cases Results

No faults were detected in this RTEMS region manager directive.

A.10.1.5. RTEMS-TS-CL-RGNDLT: rtems_region_delete Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-RGNDLT
Purpose:	To test rtems_region_delete by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-rgn.c Lines: [263] `returnStatus = rtems_region_delete (regionId);`
Test Item:	rtems_region_delete (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS region manager directive.

A.10.1.6. RTEMS-TS-CL-RGNRSG: rtems_region_return_segment Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-RGNRSG
Purpose:	To test rtems_region_return_segment by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-rgn.c Lines: [197 - 198] `returnStatus = rtems_region_return_segment (regionId, segment1);`
Test Item:	rtems_region_return_segment (rtems_id id, void *segment)
Generated Test Cases:	4

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-RGNRSG-094
Input Specification:	
`segment = NULL;` `returnStatus = rtems_region_return_segment (regionId, segment1);`	
Failure Description:	
A Memory Exception occurs when NULL value is given as the address of a segment to return.	
Notes:	

The following output is returned by the simulator:
`Memory exception at ffffffc (illegal address)`
`Unexpected trap (0x09) at address 0x0200aaac`
`Data access exception at 0xffffffc`	

A.11. RTEMS IO Manager Campaigns

A.11.1. RTEMS-CMP-CL-IO: I/O Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-IO
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the I/O manager.
Workload File:	rtems-cmp-cl-io.c
Test Suites:	1. RTEMS-TS-CL-IOINI 2. RTEMS-TS-CL-IOREG 3. RTEMS-TS-CL-IOOPN 4. RTEMS-TS-CL-IOCLS 5. RTEMS-TS-CL-IOREAD 6. RTEMS-TS-CL-IOWRT 7. RTEMS-TS-CL-IOCTL
Workload Description: This workload uses only one task that will use the I/O manager directives. It initializes a dummy device and performs several operations over it: register name, open device, read, write and control operations and, finally, closes the device.	

A.11.1.1. RTEMS-TS-CL-IOINI: rtems_io_initialize Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-IOINI
Purpose:	To test rtems_io_initialize by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-io.c
	Lines: [177 - 179]
	`returnStatus = rtems_io_initialize (majorNum,`
	`minorNum,`
	`NULL);`

Test Item:	rtems_io_initialize (rtems_device_major_number major, rtems_device_minor_number minor, void *argument)
Generated Test Cases:	7

Test Cases Results

No faults were detected in this RTEMS IO manager directive.

A.11.1.2. RTEMS-TS-CL-IOREG: rtems_io_register_name Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-IOREG
Purpose:	To test rtems_io_register_name by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-io.c
	Lines: [200 - 202]
	`returnStatus = rtems_io_register_name (deviceName,`
	`majorNum,`
	`minorNum);`

Test Item:	rtems_io_register_name (char *name, rtems_device_major_number major, rtems_device_minor_number minor)
Generated Test Cases:	8

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-IOREG-012

Input Specification:	`majorNum = 0;`
	`returnStatus = rtems_io_register_name (deviceName,`
	`majorNum,`
	`minorNum);`

Failure Description:	A data access exception occurs when attempting to register a name for the driver with major number 0 (which is invalid).

Notes:	The following output is returned from the simulator:
	`Unexpected trap (0x09) at address 0x02005db4`
	`data access exception at 0x00000028`

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-IOREG-014

Input Specification:	`majorNum = 4294967295;`
	`returnStatus = rtems_io_register_name (deviceName,`
	`majorNum,`
	`minorNum);`

Failure Description:	Although an invalid value is given for the device major number parameter, no error code is return by the *rtems_io_register_name* RTEMS directive. This behaviour is not consistent with the other io directives which return RTEMS_INVALID_NUMBER error code if the value for the device major number id greater than the maximum number of drivers
Notes:	

A.11.1.3. RTEMS-TS-CL-IOOPN: rtems_io_open Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-IOOPN
Purpose:	To test rtems_io_open by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-io.c
	Lines: [229- 231]
	`returnStatus = rtems_io_open (majorNum,`
	`minorNum,`
	`NULL);`

Test Item:	rtems_io_open (rtems_device_major_number major, rtems_device_minor_number minor, void *argument)
Generated Test Cases:	7

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-IOOPN-019
Input Specification:	

`majorNum = 0;`
`returnStatus = rtems_io_open (majorNum,`
`minorNum,`
`NULL);`	

Failure Description:	
A data access exception occurs when attempting to open the driver with major number 0 (which is invalid).	
Notes:	

The following output is returned from the simulator:
`Unexpected trap (0x09) at address 0x02005db4`
`data access exception at 0x00000028`	

A.11.1.4. RTEMS-TS-CL-IOCLS: rtems_io_close Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-IOCLS
Purpose:	To test rtems_io_close by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-io.c
	Lines: [313 - 315]
	`returnStatus = rtems_io_close (majorNum,`
	`minorNum,`
	`NULL);`

Test Item:	rtems_io_close (rtems_device_major_number major, rtems_device_minor_number minor, void *argument)
Generated Test Cases:	7

Test Cases Results

No faults were detected in this RTEMS IO manager directive.

A.11.1.5. RTEMS-TS-CL-IOREAD: rtems_io_read Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-IOREAD
Purpose:	To test rtems_io_read by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-io.c
	Lines: [267 - 269]
	`returnStatus = rtems_io_read(majorNum,`
	`minorNum,`
	`(void *) &testData);`

Test Item:	rtems_io_read (rtems_device_major_number major, rtems_device_minor_number minor, void *argument)
Generated Test Cases:	7

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-IOREAD-035
Input Specification:	

`majorNum = 0;`
`returnStatus = rtems_io_read(majorNum,`
`minorNum,`
`(void *) &testData);`	

Failure Description:	
A data access exception occurs when attempting to read from the driver with major number 0 (which is invalid).	
Notes:	

The following output is returned from the simulator:
`Unexpected trap (0x09) at address 0x0200611c`
`data access exception at 0x0000000c`	

A.11.1.6. RTEMS-TS-CL-IOWRT : rtems_io_write Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-IOWRT
Purpose:	To test rtems_io_write by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-io.c
	Lines: [253 - 255]
	`returnStatus = rtems_io_write (majorNum,`
	`minorNum,`
	`(void *) &testData);`

Test Item:	rtems_io_write (rtems_device_major_number major, rtems_device_minor_number minor, void *argument)
Generated Test Cases:	7

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-IOWRT-043
Input Specification:	

`majorNum = 0;`
`returnStatus = rtems_io_write (majorNum,`
`minorNum,`
`(void *) &testData);`	

Failure Description:	
A data access exception occurs when attempting to write to the driver with major number 0 (which is invalid).	
Notes:	

The following output is returned from the simulator:
`Unexpected trap (0x09) at address 0x0200611c`
`data access exception at 0x0000000c`	

A.11.1.7. RTEMS-TS-CL-IOCTL: rtems_io_control Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-IOCTL
Purpose:	To test rtems_io_control by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-io.c
	Lines: [289 - 291]
	`returnStatus = rtems_io_control (majorNum,`
	`minorNum,`
	`NULL);`

Test Item:	rtems_io_control (rtems_device_major_number major, rtems_device_minor_number minor, void *argument)
Generated Test Cases:	7

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-IOCTL-051
Input Specification:	

`majorNum = 0;`
`returnStatus = rtems_io_control (majorNum,`
`minorNum,`
`NULL);`	

Failure Description:	
A data access exception occurs when attempting to control the driver with major number 0 (which is invalid).	
Notes:	

The following output is returned from the simulator:
`Unexpected trap (0x09) at address 0x0200611c`
`Data access exception at 0x0000000c`	

A.12. RTEMS Fatal Error Manager Campaigns

A.12.1. RTEMS-CMP-CL-FER: Fatal Error Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-FER
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the fatal error manager.
Workload File:	rtems-cmp-cl-fer.c
Test Suites:	1. RTEMS-TS-CL-FEROCC
Workload Description:	
This workload has only one task. This task creates a fatal extension and calls the fatal error directive to test, namely, fatal_error_occured. It is then verified that the error source, location and code are the expected ones.	

A.12.1.1. RTEMS-TS-CL-FEROCC: rtems_fatal_error_occured Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-FEROCC
Purpose:	To test rtems_fatal_error_occured by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-fer.c Lines: [146] `rtems_fatal_error_occurred(error);`
Test Item:	rtems_fatal_error_occured (rtems_unsigned32 the_error)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS fatal error manager directive.

A.13. RTEMS Rate Monotonic Manager Campaigns

A.13.1. RTEMS-CMP-CL-RMT: Rate Monotonic Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-RMT
Purpose:	To test the robustness of all RTEMS Classic APIs related to the corresponding manager.
Workload File:	Rtems-cmp-cl-rmt.c
Test Suites:	1. RTEMS-TS-CL-RMTCRT 2. RTEMS-TS-CL-RMTDLT 3. RTEMS-TS-CL-RMTCNL 4. RTEMS-TS-CL-RMTPRD
Workload Description: This workload contains five tasks, one main task, the Init task, and four secondary tasks. The Init task creates and starts all four tasks. The secondary tasks create and manage their own rate monotonic period. Each of these tasks wait for their individual period to elapse and increment the corresponding global counter. The Init task also has its own rate monotonic period, which is greater than the periods of the secondary tasks. Each time its period elapses it gets the values of the global counters and check them against predefined values.	

A.13.1.1. RTEMS-TS-CL-RMTCRT: rtems_rate_monotonic_create Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-RMTCRT
Purpose:	To test rtems_rate_monotonic_create by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-rmt.c Lines: [215 - 216] `returnStatus = rtems_rate_monotonic_create( rmname, ptrmid );`
Test Item:	rtems_rate_monotonic_create (rtems_name name, rtems_id id)
Generated Test Cases:	6

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-RMTCRT-005

Input Specification:
`ptrmid = NULL;`
`returnStatus = rtems_rate_monotonic_create( rmname, ptrmid );`	

Failure Description: When a NULL pointer is given as argument a data exception occurs.	

Notes:
No check is performed against a NULL value on the id parameter of the rtems_rate_monotonic_create rtems directive.
The following output is returned by the simulator:
`Unexpected trap (0x09) at address 0x02004bb0`
`data access exception at 0x00000000`	

A.13.1.2. RTEMS-TS-CL-RMTDLT: rtems_rate_monotonic_delete Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-RMTDLT
Purpose:	To test rtems_rate_monotonic_delete by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-rmt.c Lines: [295] `returnStatus = rtems_rate_monotonic_delete (rmid);`
Test Item:	rtems_rate_monotonic_delete (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS rate monotonic manager directive.

A.13.1.3. RTEMS-TS-CL-RMTCNL: rtems_rate_monotonic_cancel Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS--RMTCNL
Purpose:	To test rtems_rate_monotonic_cancel by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-rmt.c Lines: [281] `returnStatus = rtems_rate_monotonic_cancel (rmid);`
Test Item:	rtems_rate_monotonic_cancel (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS rate monotonic manager directive.

A.13.1.4. RTEMS-TS-CL-RMTPRD: rtems_rate_monotonic_period Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-RMTPRD
Purpose:	To test rtems_rate_monotonic_period by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-rmt.c
	Lines: [239 - 240]
	`returnStatus = rtems_rate_monotonic_period (rmid,`
	`Periods[INIT_TASK] );`

Test Item:	rtems_rate_monotonic_period (rtems_id id, rtems_interval length)
Generated Test Cases:	12

Test Cases Results

No faults were detected in this RTEMS rate monotonic manager directive.

A.14. RTEMS User Extensions Manager Campaigns

A.14.1. RTEMS-CMP-CL-UEX: User Extensions Manager Classic API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-UEX
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the user extensions manager.
Workload File:	Rtems-cmp-cl-uex.c
Test Suites:	1. RTEMS-TS-CL-UEXCRT 2. RTEMS-TS-CL-UEXDLT
Workload Description: This workload defines a table that points to the routines that will extend the RTEMS critical sections such as the task creation, task deletion, etc. Then, it will create a new task, with a low priority that will be used to test the new user extensions: when the task is created, for instance, a global variable will be set to the value CREATION by the task creation extension. All the other extensions will be tested in a similar way. The following extensions are used: • Task creation extension; • Task start extension; • Task delete extension; • Task begin extension.	

A.14.1.1. RTEMS-TS-CL-UEXCRT: rtems_extension_create Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-UEXCRT
Purpose:	To test rtems_extension_create by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-cl-uex.c
	Lines: [181 - 183]
	`returnStatus = rtems_extension_create (tblName,`
	`ptUser_Extensions,`
	`pttblId);`

Test Item:	rtems_extension_create (rtems_name name, rtems_extensions_table *table, rtems_id *id)

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-UEXCRT-013
Input Specification:	

`pttblId = NULL;`
`returnStatus = rtems_extension_create (tblName,`
`ptUser_Extensions,`
`pttblId);`	

Failure Description:	
When a NULL pointer is given as argument a data exception occurs.	
Notes:	

No check is performed against a NULL value on the id parameter of the rtems_extension_create rtems directive.
The following output is returned by the simulator:
`Unexpected trap (0x09) at address 0x0200267c`
`data access exception at 0x00000000`	

A.14.1.2. RTEMS-TS-CL-UEXDLT : rtems_extension_delete Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-UEXDLT
Purpose:	To test rtems_extension_delete by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-cl-uex.c Lines: [289] `returnStatus = rtems_extension_delete (tblId);`
Test Item:	rtems_extension_delete (rtems_id id)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS user extensions manager directive.

A.15. RTEMS Mutex Manager Campaigns

A.15.1. RTEMS-CMP-PX-MTX: Mutex Manager POSIX API Interface Test Campaign

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-PX-MTX
Purpose:	To test the robustness of the selected RTEMS POSIX APIs related to the mutex manager.
Workload File:	rtems-cmp-px-mtx.c
Test Suites:	1. RTEMS-TS-PX-MTXMAI 2. RTEMS-TS-PX-MTXMAD 3. RTEMS-TS-PX-MTXAPT 4. RTEMS-TS-PX-MTXACL 5. RTEMS-TS-PX-MTXASH 6. RTEMS-TS-PX-MTXINI 7. RTEMS-TS-PX-MTXDTR 8. RTEMS-TS-PX-MTXTLK 9. RTEMS-TS-PX-MTXTML 10. RTEMS-TS-PX-MTXULK 11. RTEMS-TS-PX-MTXCEI

Workload Description:

This workload uses the mutex manager directives that handle several functionalities of this type of objects. These are:

- Mutex attributes initialisation/destruction;
- Changing the mutex attributes;
- Mutex creation/destruction;
- Mutex locking/unlocking;
- Try to lock a mutex until a specified timeout expires

The workload has two threads: the main thread where the main operations and assertions are executed and another thread to help in the mutex operations.

A.15.1.1. RTEMS-TS-PX-MTXMAI: pthread_mutexattr_init Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXMAI
Purpose:	To test pthread_mutexattr_init by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-mtx.c Lines: [121] `result = pthread_mutexattr_init(&attr);`
Test Item:	int pthread_mutexattr_init(pthread_mutexattr_t *attr)
Generated Test Cases:	25

Test Cases Results

No faults were detected in this RTEMS mutex manager directive.

A.15.1.2. RTEMS-TS-PX-MTXMAD: pthread_mutexattr_destroy Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXMAD
Purpose:	To test pthread_mutexattr_destroy by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-mtx.c Lines: [143] `Result = pthread_mutexattr_destroy(&attr);`
Test Item:	int pthread_mutexattr_destroy(pthread_mutexattr_t *attr)
Generated Test Cases:	25

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-MTXMAD-029 (same results obtained in RTEMS-TCR-PX-TMX- MTXMAD-030. RTEMS-TCR-PX-TMX- MTXMAD-031, RTEMS-TCR-PX-TMX- MTXMAD-032)

Input Specification:
`+ attr.is_initialized = -1;`
`result = pthread_mutexattr_destroy(&attr);`	

Failure Description: A call to the *pthread_mutexattr_destroy* always returns successfully as long as the *is_initialized* attribute of the parameter is different from 0 (zero); it seems that no other verification is done.	
Notes:	

A.15.1.3. RTEMS-TS-PX-MTXAPT : pthread_mutexattr_setprotocol Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXAPT
Purpose:	To test the corresponding API by invoking it with the entire range of valid parameters.

Fault Location(s):	Source file: rtems-cmp-px-mtx.c
	Lines: [170-171]
	`result = pthread_mutexattr_setprotocol(&attr,`
	`protocol1);`

Test Item:	int pthread_mutexattr_setprotocol(pthread_mutexattr_t *attr, int protocol)
Generated Test Cases:	30

Test Cases Results

No faults were detected in this RTEMS mutex manager directive.

A.15.1.4. RTEMS-TS-PX-MTXACL: pthread_mutexattr_setprioceiling Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXACL
Purpose:	To test pthread_mutexattr_setprioceiling by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-mtx.c Lines: [201-202] `result = pthread_mutexattr_setprioceiling(&attr, protocol1);`
Test Item:	int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *attr, int prioceiling)
Generated Test Cases:	30

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-MTXACL-208 (same results obtained in RTEMS-TCR-PX-MTX- MTXACL-209, RTEMS-TCR-PX-MTX- MTXACL-210)

Input Specification:
`+ attr.is_initialized = -1;`
`result = pthread_mutexattr_setprioceiling(&attr, protocol1);`	

Failure Description: Setting the *is_initialized* attribute of the *pthread_mutexattr_t* parameter with a value different of 0 (zero) causes the function *pthread_mutexattr_setprioceiling* to accept the mutex as valid – no other validation is done in the parameter.	
Notes:	

A.15.1.5. RTEMS-TS-PX-MTXASH: pthread_mutexattr_setpshared Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXASH
Purpose:	To test pthread_mutexattr_setpshared by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-mtx.c Lines: [237-238] `result = pthread_mutexattr_setpshared(&attr, pshared1);`
Test Item:	int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr, int pshared)
Generated Test Cases:	30

Test Cases Results

No faults were detected in this RTEMS mutex manager directive.

A.15.1.6. RTEMS-TS-PX-MTXINI: pthread_mutex_init Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXINI
Purpose:	To test pthread_mutex_init by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-px-mtx.c
	Lines: [292 - 293]
	`result = pthread_mutex_init(&mutex,`
	`&attr);`

Test Item:	int pthread_mutex_init(pthread_mutex_t *mutex, pthread_mutexattr_t *attr)
Generated Test Cases:	28

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-MTXINI-161 (same results obtained in RTEMS-TCR-PX-MTXINI-162, RTEMS-TCR-PX-MTXINI-163, RTEMS-TCR-PX-MTXINI-164)

Input Specification:
`+ attr.process_shared = 1;`
`pthread_mutex_init(&mutex,`
`&attr);`	

Failure Description:
When the *process_shared* attribute is different from 0 (PTHREAD_PROCESS_PRIVATE), the application ends execution and there is a RTEMS kernel assertion. The output for this mutant was:
`assertion "the_attr->process_shared == PTHREAD_PROCESS_PRIVATE" failed: file "../../../../../../../rtems-4.5.0/c/src/exec/posix/src/mutexinit.c", line 96`
`pthread_mutexattr_init(): 0; Assertion-01: success; pthread_mutexattr_destroy(): 0;Assertion-02: success; pthread_mutexattr_setprotocol(): 0; Assertion-03: success; pthread_mutexattr_setprioceiling(): 0; Assertion-04: success; pthread_mutexattr_setpshared(): 0; Assertion-05: success; pthread_mutex_init(): 0; pthread_mutex_destroy(): 0;Assertion-06: success;`	

Notes:	

A.15.1.7. RTEMS-TS-PX-MTXDTR: pthread_mutex_destroy Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXDTR
Purpose:	To test pthread_mutex_destroy by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-mtx.c Lines: [264] `Result = pthread_mutex_destroy(&mutex);`
Test Item:	int pthread_mutex_destroy(pthread_mutex_t *mutex)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS mutex manager directive.

A.15.1.8. RTEMS-TS-PX-MTXLCK: pthread_mutex_lock Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXLCK
Purpose:	To test pthread_mutex_lock by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-mtx.c Lines: [295] `result = pthread_mutex_lock(&mutex);`
Test Item:	int pthread_mutex_lock(pthread_mutex_t *mutex)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS mutex manager directive.

A.15.1.9. RTEMS-TS-PX-MTXTLK: pthread_mutex_trylock Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXTLK
Purpose:	To test pthread_mutex_trylock by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-mtx.c Lines: [319] `result = pthread_mutex_trylock(&mutex);`
Test Item:	int pthread_mutex_trylock(pthread_mutex_t *mutex)
Generated Test Cases:	3

Test Cases Results

No faults were detected in this RTEMS mutex manager directive.

A.15.1.10. RTEMS-TS-PX-MTXTML: pthread_mutex_timedlock Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXTML
Purpose:	To test pthread_mutex_timedlock by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-px-mtx.c
	Lines: [351-352]
	`result = pthread_mutex_timedlock(&mutex,`
	`&timeout);`

Test Item:	int pthread_mutex_timedlock(pthread_mutex_t *mutex, const struct timespec *timeout)
Generated Test Cases:	8

Test Cases Results

No faults were detected in this RTEMS mutex manager directive.

A.15.1.11. RTEMS-TS-PX-MTXULK: pthread_mutex_unlock Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXULK
Purpose:	To test pthread_mutex_unlock by invoking it with the entire range of test values for each of its parameters.
Fault Location(s):	Source file: rtems-cmp-px-mtx.c Lines: [377] `Result = pthread_mutex_unlock(&mutex);`
Test Item:	int pthread_mutex_unlock(pthread_mutex_t *mutex)
Generated Test Cases:	3

Test Cases Results

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-PX-MTXULK-204

Input Specification:
`+ mutex = 18446744073709551615;`
`result = pthread_mutex_unlock(&mutex);`	

Failure Description: All the test values used in the *pthread_mutex_unlock* function caused an EINVAL error code. However, with this test value, the error code was EPERM (error code number 1 - *Not super-user*).	
Notes:	

A.15.1.12. RTEMS-TS-PX-MTXCEI: pthread_mutex_setprioceiling Test Suite

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-PX-MTXCEI
Purpose:	To test pthread_mutex_setprioceiling by invoking it with the entire range of test values for each of its parameters.

Fault Location(s):	Source file: rtems-cmp-px-mtx.c
	Lines: [401-403]
	`result = pthread_mutex_setprioceiling(&mutex,`
	`prioceiling,`
	`&oprioceiling);`

Test Item:	Int pthread_mutex_setprioceiling(Pthread_mutex_t *mutex, Int prioceiling, Int *oldceiling)
Generated Test Cases:	35

Test Cases Results

No faults were detected in this RTEMS mutex manager directive.

Annex B. RTEMS workloads

B.1. RTEMS Classic API Workloads

The following sections present the workloads that were used to build the test cases of the RTEMS Classic API.

B.1.1. RTEMS-CMP-CL-TSK.C

B.1.1.1. Description

This workload is composed by two tasks, the Init task and one that it creates, TA1. The Init task is initialised to be preempted and with priority CURRENT_TASK_LOW_PRIORITY = 15. The TA1 is created with prority TEST_TASK_PRIORITY = 10. A global counter is incremented in the body of the TA1 task each time it runs.

The main task (Init) does the following:

1. Create a task (TA1)
2. **Assertion 1:** Check with rtems_task_ident that the100 task ID was successfully assigned
3. Start TA1 to increment a global variable
4. **Assertion 2:** Global variable is incremented
5. Restart TA1
6. **Assertion 3:** Global variable is incremented
7. Set RTEMS task priority to CURRENT_TASK_HIGH_PRIORITY;
8. Restart task;
9. Suspend RTEMS task
10. Set RTEMS task priority
11. **Assertion 4:** Global variable was not incremented because TA1 is suspended
12. Resume RTEMS task
13. **Assertion 5:** Global variable is incremented
14. Set TA1 task priority
15. **Assertion 6:** Task priority set successful
16. Set RTEMS task mode
17. **Assertion 7:** Task mode set successful

B.1.1.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Task Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
   testing/implementation/classic-workloads/rtems-cmp-cl-tsk/rtems-cmp-cl-tsk.c,v $
6    $Id: rtems-cmp-cl-tsk.c,v 1.5 2003/09/02 14:38:05 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
```

```
11
12  SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14  OS-TYPE : RTEMS 4.5.0
15
16  AUTHOR : lvelkov
17
18  KEYWORDS : ----
19  PURPOSE : Test the functionality of all RTEMS APIs related to the Task
20            Manager.
21
22  CREATED ON : 17-07-2003
23  CHANGED ON : $Date: 2003/09/02 14:38:05 $
24  CHANGED BY : $Author: rmaia $
25
26  $Revision: 1.5 $
27  STICKY TAG : $Name:  $
28
29  INSPECTED ON: 24-07-2003
30  MODERATOR : rmaia
31
32  TABLES : none.
33
34  HISTORY
35  $Log: rtems-cmp-cl-tsk.c,v $
36  Revision 1.5  2003/09/02 14:38:05  rmaia
37  Commited after correcting a comentary
38
39  Revision 1.4  2003/08/21 15:29:33  rmaia
40  Added test to rtems_task_delete.
41
42  Revision 1.3  2003/08/21 14:29:35  rmaia
43  Compiled and tested with success.
44
45  Revision 1.2  2003/08/05 16:17:12  rbarbosa
46  Updated after review
47
48  Revision 1.1  2003/07/29 13:26:29  rmaia
49  Compile for the RTEMS 4.5.0. Added configuration and replace main by init.
50
51  Revision 1.3  2003/07/24 23:31:33  rmaia
52  Updated header.
53
54  Revision 1.2  2003/07/24 23:27:51  rmaia
55  Revision.
56  Revision 1.1  2003/07/24 17:46:56  lvelkov
57  Created and added to cvs. First draft version to be review. Not yet compiled.
58
59 ******************************************************************************/
60
61
62 /*#include <tasks.h>*/
63
64 #include <bsp.h>
65 #include <stdio.h>
66
67 int cnt;
68
```

```
69  const rtems_task_priority TEST_TASK_PRIORITY = 10;
70
71  const rtems_task_priority CURRENT_TASK_HIGH_PRIORITY = 5;
72  const rtems_task_priority CURRENT_TASK_LOW_PRIORITY = 15;
73
74  rtems_task testTask (rtems_task_argument ptcounter) {
75     (*(int *)ptcounter)++;
76
77
78     /* To be able to restart the task it must stay on a state other than dormant */
79     while( 1 ) {
80        rtems_task_wake_after( 50 );
81     }
82
83  }
84
85  /*
86   * INIT TASK BODY
87   */
88
89  rtems_task Init(rtems_task_argument ignored){
90
91
92     int *testTaskArg = &cnt;
93
94     rtems_status_code returnStatus = 0;
95
96     rtems_name testTaskName = rtems_build_name( 'T', 'A', '1', ' ' );
97     rtems_task_priority testTaskInitPriority = TEST_TASK_PRIORITY;
98     rtems_unsigned32 testTaskStackSize = RTEMS_MINIMUM_STACK_SIZE;
99     rtems_mode testTaskInitMode = RTEMS_DEFAULT_MODES;
100    rtems_mode testTaskOldMode = RTEMS_DEFAULT_MODES;
101    rtems_mode testTaskCurrentMode = RTEMS_DEFAULT_MODES;
102    rtems_attribute testTaskAtt = RTEMS_DEFAULT_ATTRIBUTES;
103    rtems_id testTaskId = 0; /* Users Guide: "... task ID is stored in a user provided
    location." */
104
105    rtems_id assertionTaskId = 0;
106
107    rtems_task_priority currentTaskOldPriority = 0;
108
109    rtems_task_priority testTaskOldPriority = 0;
110    rtems_task_priority testTaskCurrentPriority = 0;
111
112    /* Initialize global counter */
113    cnt = 0;
114
115    /*
116     * TEST TASK_CREATE
117     */
118
119    /*Create a RTEMS task*/
120    returnStatus = rtems_task_create (testTaskName,
121                                      testTaskInitPriority,
122                                      testTaskStackSize,
123                                      testTaskInitMode,
124                                      testTaskAtt,
125                                      &testTaskId);
```

```
126
127    printf ("rtems_task_create(): %d;", returnStatus);
128
129    /*Assertion check*/
130    returnStatus = rtems_task_ident(testTaskName,
131                                    RTEMS_SEARCH_ALL_NODES,
132                                    &assertionTaskId);
133
134    printf ("rtems_task_ident(): %d;", returnStatus);
135
136    if (testTaskId == assertionTaskId) {
137       printf ("Assertion-01: success;");
138    }
139    else {
140       printf ("Assertion-01: failure;");
141    }
142
143
144    /*
145     * TEST START_TASK
146     */
147
148    /* The priority of the current task must be lower than the testTask
149     * in order to be preempted. */
150    returnStatus = rtems_task_set_priority (RTEMS_SELF,
151                                            CURRENT_TASK_LOW_PRIORITY,
152                                            &currentTaskOldPriority);
153
154    printf ("rtems_task_set_priority(): %d;", returnStatus);
155
156    returnStatus = rtems_task_mode (RTEMS_PREEMPT,
157                                    RTEMS_PREEMPT_MASK,
158                                    &testTaskOldMode);
159
160    printf ("rtems_task_mode(): %d;", returnStatus);
161
162    /*Start RTEMS task*/
163    returnStatus =  rtems_task_start (testTaskId, &testTask,
164                                      (rtems_task_argument) testTaskArg);
165    printf ("rtems_task_start(): %d;", returnStatus);
166
167    /*Assertion check*/
168    if (cnt == 1) {
169       printf ("Assertion-02: success;");
170    }
171    else {
172       printf ("Assertion-02: failure;");
173    }
174
175
176    /*
177     * TEST RESTART_TASK
178     */
179
180    /*Restart RTEMS task*/
181
182    returnStatus = rtems_task_restart (testTaskId,
183                                       (rtems_task_argument) testTaskArg);
```

```
184   printf ("rtems_task_restart(): %d;", returnStatus);
185
186   /*Assertion check*/
187   if (cnt == 2) {
188      printf ("Assertion-03: success;");
189   }
190   else {
191      printf ("Assertion-03: failure;");
192   }
193
194   /*
195    * TEST TASK_SUSPEND
196    */
197
198   /* The priority of the current task must be higher than the testTask
199      in order to not be preempted between restart and suspend. */
200   returnStatus = rtems_task_set_priority (RTEMS_SELF,
201                                           CURRENT_TASK_HIGH_PRIORITY,
202                                           &currentTaskOldPriority);
203
204   printf ("rtems_task_set_priority(): %d;", returnStatus);
205
206   /*Restart RTEMS task*/
207   returnStatus = rtems_task_restart (testTaskId,
208                                      (rtems_task_argument) testTaskArg);
209   printf ("rtems_task_restart(): %d;", returnStatus);
210
211   /*Suspend RTEMS task*/
212   returnStatus = rtems_task_suspend (testTaskId);
213   printf ("rtems_task_suspend(): %d;", returnStatus);
214
215   /* Lower the priority of the current task in order to be preempted by the
216      test task. */
217   returnStatus = rtems_task_set_priority (RTEMS_SELF,
218                                           CURRENT_TASK_LOW_PRIORITY,
219                                           &currentTaskOldPriority);
220
221   printf ("rtems_task_set_priority(): %d;", returnStatus);
222
223   /*Assertion check*/
224   /* The value is not incremented beacause the testTask is suspended */
225   if (cnt == 2) {
226      printf ("Assertion-04: success;");
227   }
228   else {
229      printf ("Assertion-04: failure;");
230   }
231
232
233   /*
234    * TEST TASK_RESUME
235    */
236
237   /*Resume RTEMS task*/
238   returnStatus = rtems_task_resume (testTaskId);
239   printf ("rtems_task_resume(): %d;", returnStatus);
240
241   /*Assertion check*/
```

```
242    if (cnt == 3) {
243       printf ("Assertion-05: success;");
244    }
245    else {
246       printf ("Assertion-05: failure;");
247    }
248
249
250    /*
251     * TEST TASK_SET_PRIORITY
252     */
253
254    /*Set RTEMS task priority*/
255    returnStatus = rtems_task_set_priority (testTaskId,
256                                            testTaskInitPriority + 1,
257                                            &testTaskOldPriority);
258
259    printf ("rtems_task_set_priority(): %d;", returnStatus);
260
261    /*Assertion check*/
262    /*Get task current priority */
263    returnStatus = rtems_task_set_priority (testTaskId,
264                                            RTEMS_CURRENT_PRIORITY,
265                                            &testTaskCurrentPriority);
266    printf ("rtems_task_set_priority(): %d;", returnStatus);
267
268    if (testTaskCurrentPriority == testTaskInitPriority + 1) {
269       printf ("Assertion-06: success;");
270    }
271    else {
272       printf ("Assertion-06: failure;");
273    }
274
275    /*
276     * TEST TASK_MODE
277     */
278
279    /*Set RTEMS task mode*/
280    returnStatus = rtems_task_mode (RTEMS_NO_PREEMPT,
281                                    RTEMS_PREEMPT_MASK,
282                                    &testTaskOldMode);
283
284    printf ("rtems_task_set_mode(): %d;", returnStatus);
285
286    /*Assertion check*/
287    /*Get current testTask mode*/
288    returnStatus = rtems_task_mode (RTEMS_NO_PREEMPT,
289                                    RTEMS_CURRENT_MODE,
290                                    &testTaskCurrentMode);
291
292    if ((testTaskCurrentMode & RTEMS_PREEMPT_MASK) == RTEMS_NO_PREEMPT) {
293       printf ("Assertion-07: success;");
294    }
295    else {
296       printf ("Assertion-07: failure;");
297    }
298
299    /*
```

```
300     * TEST TASK_DELETE
301     */
302
303    /*Delete a RTEMS task*/
304    returnStatus = rtems_task_delete (testTaskId);
305
306    printf ("rtems_task_delete(): %d;", returnStatus);
307
308    /*Assertion check*/
309    returnStatus = rtems_task_ident(testTaskName,
310                                    RTEMS_SEARCH_ALL_NODES,
311                                    &assertionTaskId);
312
313    if (returnStatus == RTEMS_INVALID_NAME) {
314       printf ("Assertion-08: success;");
315    }
316    else {
317       printf ("Assertion-08: failure;");
318    }
319
320
321    printf ("\n");
322
323    exit(0);
324 }
325
326 /* configuration information */
327
328 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
329
330 #define CONFIGURE_MAXIMUM_TASKS 2
331 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
332
333 #define CONFIGURE_INIT
334
335 #include <confdefs.h>
336
337 /* end of file */
```

B.1.2. RTEMS-CMP-CL-CLK.C

B.1.2.1. Description

The main task of this workload is initialized with the preemption disabled. This means that it will be not preempted by other tasks. It is assumed that the body of the task executes in less than one second.

1. The main task (Init) does the following:Set RTEMS time time of day (TOD) to 2001/01/01 0h0m0s0ms
2. Get RTEMS time TOD
3. **Assertion 1:** The returned time is the same (with exception of the milliseconds field)
4. Get clock time value (seconds elapsed since 1998/01/01 0h0m0s0ms)
5. **Assertion 2:** The returned value corresponds to the number of seconds elaped since 1998/01/01 0h0m0s until 2001/01/01 0h0m0s.

B.1.2.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Clock Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-clk/rtems-cmp-cl-clk.c,v $
6    $Id: rtems-cmp-cl-clk.c,v 1.6 2003/09/05 11:19:47 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Clock
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/05 11:19:47 $
24   CHANGED BY : $Author: rmaia $
25
26   $Revision: 1.6 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 25-07-2003
30   MODERATOR : rmaia
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-clk.c,v $
36   Revision 1.6  2003/09/05 11:19:47  rmaia
37   Added test for rtems_clock_set
38
39   Revision 1.5  2003/09/05 10:56:36  rmaia
40   Replaced &variable by a pointer pt variare
41
42   Revision 1.4  2003/09/04 17:58:45  rmaia
43   Use a variable to pass the options to the get command
44
45   Revision 1.3  2003/08/22 13:14:28  rmaia
46   Compiled and tested with success.
47
48   Revision 1.2  2003/08/05 16:19:23  rbarbosa
49   Updated after review
50
51   Revision 1.1  2003/08/05 11:44:19  rbarbosa
52   Updated after reviewed. Added header and RTEMS specific code lines.
53   Ready for compilation
54
55   Revision 1.2  2003/07/28 10:18:33  rmaia
```

```
56   Revision
57
58
59   ***************************************************************************/
60
61  #include <bsp.h>
62  #include <rtems/system.h>
63  #include <rtems/rtems/status.h>
64  #include <rtems/rtems/clock.h>
65  #include <stdio.h>
66
67  /*
68   * Init task is initialized with  RTEMS_NO_PREEMPT mode.
69   * This means that no other task will be scheduled while
70   * this task is executing.
71   * It is assumed that this task executes in less than one second.
72   */
73
74
75  rtems_task Init(rtems_task_argument ignored)
76  {
77
78     /* Initialized return value with some error code */
79     rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
80
81     rtems_time_of_day newTimeOfDay;
82     rtems_time_of_day ctTimeOfDay;
83
84     rtems_time_of_day *ptnewTimeOfDay;
85
86     rtems_clock_time_value ctTimeValue;
87
88     rtems_clock_get_options option;
89
90     newTimeOfDay.year = 2001;
91     newTimeOfDay.month = 1;
92     newTimeOfDay.day = 1;
93     newTimeOfDay.hour = 0;
94     newTimeOfDay.minute = 0;
95     newTimeOfDay.second = 0;
96     newTimeOfDay.ticks = 0;
97
98     /*
99      * TEST CLOCK_SET and CLOCK_GET
100     */
101
102    /*Set RTEMS time*/
103    ptnewTimeOfDay = &newTimeOfDay;
104    returnStatus = rtems_clock_set (ptnewTimeOfDay);
105    printf ("rtems_clock_set(): %d;", returnStatus);
106
107    /*Get RTEMS time*/
108    option = RTEMS_CLOCK_GET_TOD;
109    rtems_clock_get (option, &ctTimeOfDay);
110    printf ("rtems_clock_get(): %d;", returnStatus);
111
112    /*Assertion check*/
113    if ((ctTimeOfDay.year == newTimeOfDay.year)     &&
```

```
114         (ctTimeOfDay.month == newTimeOfDay.month)   &&
115         (ctTimeOfDay.day == newTimeOfDay.day)       &&
116         (ctTimeOfDay.hour == newTimeOfDay.hour)     &&
117         (ctTimeOfDay.minute == newTimeOfDay.minute) &&
118         (ctTimeOfDay.second == newTimeOfDay.second)){
119
120       printf ("Assertion-01: success;");
121    }
122    else {
123       printf ("Assertion-01: failure;");
124    }
125
126    /*Set RTEMS time*/
127    returnStatus = rtems_clock_set (&newTimeOfDay);
128    printf ("rtems_clock_set(): %d;", returnStatus);
129
130    /*Get RTEMS time*/
131    option = RTEMS_CLOCK_GET_TOD;
132    rtems_clock_get (option, &ctTimeOfDay);
133    printf ("rtems_clock_get(): %d;", returnStatus);
134
135    /*Assertion check*/
136    if ((ctTimeOfDay.year == newTimeOfDay.year)     &&
137        (ctTimeOfDay.month == newTimeOfDay.month)   &&
138        (ctTimeOfDay.day == newTimeOfDay.day)       &&
139        (ctTimeOfDay.hour == newTimeOfDay.hour)     &&
140        (ctTimeOfDay.minute == newTimeOfDay.minute) &&
141        (ctTimeOfDay.second == newTimeOfDay.second)){
142
143       printf ("Assertion-02: success;");
144    }
145    else {
146       printf ("Assertion-02: failure;");
147    }
148
149
150    /*
151     * TEST CLOCK_GET (remaining modes)
152     */
153
154    /*Get RTEMS time*/
155    option = RTEMS_CLOCK_GET_TIME_VALUE;
156    rtems_clock_get (option, &ctTimeValue);
157    printf ("rtems_clock_get(): %d;", returnStatus);
158
159
160    /*Assertion check: It is assumed that less than one second elapsed since the set*/
161    /* Seconds elapsed since 01/01/1988: days * hours * minites * seconds.
162    */
163    if (ctTimeValue.seconds == 410313600){
164       printf ("Assertion-03: success;");
165    }
166    else {
167       printf ("Assertion-03: failure;");
168    }
169
170
171    printf ("\n");
```

```
172
173    exit(0);
174 }
175
176 /* configuration information */
177
178 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
179
180 #define CONFIGURE_MAXIMUM_TASKS 4
181 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
182
183 #define CONFIGURE_INIT
184
185 #include <confdefs.h>
186
187 /* end of file */
```

B.1.3. RTEMS-CMP-CL-EVT.C

B.1.3.1. Description

This manager has two tasks.The main task sends an event to the second task and this one by its turn sends the same event back to the main task. The main task does the following:

1. Create task
2. Start task
3. Send event to test task
4. Receive event from the test task
5. **Assertion 1:** The event received is the same as the event sent

B.1.3.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Event Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-evt/rtems-cmp-cl-evt.c,v $
6    $Id: rtems-cmp-cl-evt.c,v 1.5 2003/09/08 16:16:14 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Event
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/08 16:16:14 $
```

```
24   CHANGED BY : $Author: rmaia $
25
26   $Revision: 1.5 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 31-07-2003
30   MODERATOR : rbarbosa
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-evt.c,v $
36   Revision 1.5  2003/09/08 16:16:14  rmaia
37   *** empty log message ***
38
39   Revision 1.4  2003/08/27 12:03:18  rmaia
40   Some changes on the algorithm of the workload. Compiled and tested with success
41
42   Revision 1.3  2003/08/27 09:54:09  rmaia
43   compiled and tested
44
45   Revision 1.2  2003/08/05 15:23:36  rbarbosa
46   Update after reviewing file format
47
48   Revision 1.1  2003/08/05 11:47:17  rbarbosa
49   Updated after reviewed. Added header and RTEMS specific code lines.
50   Ready for compilation
51
52
53   ******************************************************************************/
54
55  #include <bsp.h>
56  #include <stdio.h>
57
58
59  rtems_id InitTaskId = 0;
60
61
62  /*
63   * This task is used to perform tests on the rtems_event_receive
64   */
65  rtems_task testTask(rtems_task_argument ptTaskId)
66  {
67      rtems_event_set evtToReceive = RTEMS_EVENT_8;
68      rtems_event_set evtReceived  = RTEMS_EVENT_0;
69      rtems_option optionSet       = RTEMS_WAIT | RTEMS_EVENT_ANY;
70      rtems_interval timeout       = RTEMS_NO_TIMEOUT;
71
72      rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
73
74     /*
75      * TEST RTEMS_EVENT_RECEIVE
76      */
77
78     returnStatus = rtems_event_receive (evtToReceive,
79                                         optionSet,
80                                         timeout,
81                                         &evtReceived);
```

```
82
83     /* Print results for rtems_event_receive */
84     printf ("rtems_event_receive(): %d;", returnStatus);
85
86     returnStatus = rtems_event_send (*((rtems_id *)ptTaskId),
87                                      evtReceived);
88
89     printf ("rtems_event_send(): %d;", returnStatus);
90
91  }
92
93  /*
94   * Init task is the main task and is used to create the task
95   * that tests the rtems_event_send directive and to test the
96   * rtems_event_receive
97   */
98  rtems_task Init(rtems_task_argument ignored)
99  {
100    rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
101
102
103    rtems_event_set evtSent      = RTEMS_EVENT_8;
104    rtems_event_set evtToReceive = RTEMS_EVENT_8;
105    rtems_event_set evtReceived  = RTEMS_EVENT_0;
106    rtems_event_set *ptevtReceived  = &evtReceived;
107    rtems_option optionSet       = RTEMS_WAIT | RTEMS_EVENT_ANY;
108    rtems_interval timeout       = RTEMS_NO_TIMEOUT;
109
110
111    rtems_name taskName;
112    rtems_task_priority taskPriority = 5;                          /* Set test task with a
    priority */
113    rtems_unsigned32 taskStackSize = RTEMS_MINIMUM_STACK_SIZE;     /* higher then the init
    task      */
114    rtems_mode taskMode = RTEMS_PREEMPT;
115    rtems_attribute taskAtt = RTEMS_DEFAULT_ATTRIBUTES;
116    rtems_id taskId = 0;
117    rtems_task_argument taskArg = (rtems_task_argument) &InitTaskId;
118
119    rtems_task_priority initTaskPriority = 15;
120    rtems_task_priority oldPri;
121    rtems_mode oldMode;
122
123    taskName = rtems_build_name('T','A','1',' ');
124
125    /* Create task that will send the event */
126    returnStatus = rtems_task_create (taskName,
127                                      taskPriority,
128                                      taskStackSize,
129                                      taskMode,
130                                      taskAtt,
131                                      &taskId);
132
133    printf ("rtems_task_create(): %d;", returnStatus);
134
135    returnStatus = rtems_task_ident(RTEMS_SELF,
136                                    RTEMS_SEARCH_ALL_NODES,
137                                    &InitTaskId);
```

```
138
139    printf ("rtems_task_ident(): %d;", returnStatus);
140
141    /* Start the task that will send the event*/
142    returnStatus =  rtems_task_start (taskId,
143                                      testTask,
144                                      taskArg);
145
146
147
148    returnStatus = rtems_task_set_priority (RTEMS_SELF,
149                                            initTaskPriority,
150                                            &oldPri);
151
152    printf ("rtems_task_set_priority(): %d;", returnStatus);
153
154
155    returnStatus = rtems_task_mode (RTEMS_PREEMPT,
156                                    RTEMS_PREEMPT_MASK,
157                                    &oldMode);
158
159    printf ("rtems_task_mode(): %d;", returnStatus);
160
161
162    /*
163     * TEST RTEMS_EVENT_SEND
164     */
165
166    /* Send an RTEMS event */
167    returnStatus = rtems_event_send (taskId,
168                                     evtSent);
169
170    printf ("rtems_event_send(): %d;", returnStatus);
171
172    returnStatus = rtems_event_receive (evtToReceive,
173                                        optionSet,
174                                        timeout,
175                                        ptevtReceived);
176
177    /* Assertion check */
178    if (evtSent == evtReceived)
179    {
180       printf ("Assertion-01: success;");
181    }
182    else
183    {
184       printf ("Assertion-01: failure;");
185    }
186
187    printf ("\n");
188
189    exit(0);
190 }
191
192 /* configuration information */
193
194 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
195
```

```
196 #define CONFIGURE_MAXIMUM_TASKS 2
197 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
198
199 /* This definition makes the 'init' task preemptable */
200 /*#define CONFIGURE_INIT_TASK_INITIAL_MODES RTEMS_PREEMPT*/
201
202 /* This definition sets the 'init' task as the higher priority task,
203    in comparison with the test task */
204 /*#define CONFIGURE_RTEMS_INIT_TASK_PRIORITY 15*/
205
206 #define CONFIGURE_INIT
207
208 #include <confdefs.h>
209
210 /* end of file */
```

B.1.4. RTEMS-CMP-CL-FER.C

B.1.4.1. Description

This workload has only one task. This task creates an extension and calls the directive to test, namely, fatal_error_occured.

1. Create fatal error extension
2. Call fatal error directive
3. **Assertion 1:** RTEMS error source is the expected one
4. **Assertion 2:** RTEMS error location is the expected one
5. **Assertion 3:** RTEMS error code is the expected one

B.1.4.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Fatal Error Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-fer/rtems-cmp-cl-fer.c,v $
6    $Id: rtems-cmp-cl-fer.c,v 1.4 2003/08/27 12:27:18 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Fatal Error
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/08/27 12:27:18 $
24   CHANGED BY : $Author: rmaia $
```

```
25
26   $Revision: 1.4 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 25-07-2003
30   MODERATOR : rbarbosa
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-fer.c,v $
36   Revision 1.4  2003/08/27 12:27:18  rmaia
37   removed some redundant output and add assertion for is_internal and the_source
38
39   Revision 1.3  2003/08/27 09:54:17  rmaia
40   compiled and tested
41
42   Revision 1.2  2003/08/05 16:33:24  rbarbosa
43   Updated after review
44
45   Revision 1.1  2003/07/29 14:27:48  rbarbosa
46   Ready to compile in RTEMS cross-compiling system.
47   Added configuration and replace main by init on code.
48
49   Makefile for fatal error manager robustness test workload
50
51   Revision 1.3  2003/07/25 16:57:12  rbarbosa
52   Update after code changes
53
54   Revision 1.2  2003/07/25 15:32:53  rbarbosa
55   Revision and addition of header
56
57
58   ****************************************************************************/
59
60  #include <bsp.h>
61  #include <stdio.h>
62  #include <rtems/score/interr.h> /* for INTERNAL_ERROR_RTEMS_API */
63
64  /* Fatal Error User Extension - This extension is called whenever
65   * fatal_error_occured is called */
66  rtems_extension fatal_extension(the_source,
67                                  is_internal,
68                                  the_error)
69  {
70      /* Assertion 1 */
71      /* The error generated by the fatal_error_occured directive is the RTEMS_UNSATISFIED
    */
72      /* If it is catched here, the routine is working properly */
73
74      if(the_source == INTERNAL_ERROR_RTEMS_API)
75      {
76          printf("Assertion-01: success; ");
77      }
78      else
79      {
80          printf("Assertion-01: failure; ");
81      }
```

```
82
83      if(is_internal == FALSE)
84      {
85          printf("Assertion-02: success; ");
86      }
87      else
88      {
89          printf("Assertion-02: failure; ");
90      }
91
92      if(the_error == RTEMS_UNSATISFIED)
93      {
94          printf("Assertion-03: success; ");
95      }
96      else
97      {
98          printf("Assertion-03: failure; ");
99      }
100
101     exit(0);
102 }
103
104 /* This structure defines the user extensions entry points */
105 rtems_extensions_table User_Extensions =
106 {
107     NULL, /* task creation extension */
108     NULL, /* task start extension */
109     NULL, /* task restart extension */
110     NULL, /* task delete extension */
111     NULL, /* task switch extension */
112     NULL, /* task begin extension */
113     NULL, /* task exitted extension */
114     fatal_extension  /* fatal error extension */
115 };
116
117
118 /*
119  * Init task is the only task of this workload.
120  */
121
122 rtems_task Init(rtems_task_argument ignored)
123 {
124    rtems_name tblName;
125    rtems_id tblId;
126
127    /* Error code to be triggered */
128    rtems_status_code error = RTEMS_UNSATISFIED;
129
130    /* Name given to the user extension table */
131    tblName = rtems_build_name('T','B','1',' ');
132
133
134    /*
135     * TEST FATAL_ERROR_OCCURED
136     */
137
138
139    /* Create Extension */
```

```
140    rtems_extension_create (tblName,
141                            &User_Extensions,
142                            &tblId);
143
144    /*Create RTEMS error*/
145    /* This function has a void return value */
146    rtems_fatal_error_occurred(error);
147
148    /* The Assertion are made in the routines called by the rtems_fatal_error_occured
149       namelly fatal extension */
150
151    printf("Assertion-04: failure; ");
152
153    return;
154 }
155
156 /* configuration information */
157
158 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
159 #define CONFIGURE_MAXIMUM_TASKS 1
160 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
161 #define CONFIGURE_MAXIMUM_USER_EXTENSIONS 2
162 #define CONFIGURE_INIT
163
164 #include <confdefs.h>
165
166 /* end of file */
```

B.1.5. RTEMS-CMP-CL-INT.C

B.1.5.1. Description

This workload defines a Interrupt Service Routine (ISR) that increments a global variable each time it is called. This ISR is attached to the Ilegal Instruction trap (0x02). The assembly *unimp* mnemonic is used in order to generate a Ilegal Intructions trap.

The main task (Init) does the following:

1. Attach RTEMS interrupt handler to interrupt vector twice.
2. Generate an Ilegal Instruction trap.
3. **Assertion 1:** Global variable was incremented.
4. **Assertion 2:** The old_isr_handler returned by the second call is the same attached on the first call.

B.1.5.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Interrupt Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-int/rtems-cmp-cl-int.c,v $
6    $Id: rtems-cmp-cl-int.c,v 1.2 2003/08/22 12:47:04 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
```

```
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Interrupt
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/08/22 12:47:04 $
24   CHANGED BY : $Author: rmaia $
25
26   $Revision: 1.2 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 31-07-2003
30   MODERATOR : rbarbosa
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-int.c,v $
36   Revision 1.2  2003/08/22 12:47:04  rmaia
37   Compiled and tested with success.
38
39   Revision 1.1  2003/08/05 11:47:34  rbarbosa
40   Updated after reviewed. Added header and RTEMS specific code lines.
41   Ready for compilation
42
43
44   ******************************************************************************/
45
46  #include <bsp.h>
47  #include <rtems/rtems/intr.h>
48  #include <stdio.h>
49
50  int cnt = 0;
51
52  /*
53   * Ilegal Instruction handler
54   * It just increments a global variable
55   */
56
57  rtems_isr testISR (rtems_vector_number ignored)
58  {
59     cnt++;
60     return;
61  }
62
63
64  /*
65   * Init task called by the executive.
66   */
67
68  rtems_task Init(rtems_task_argument ignored)
69  {
```

```
70
71
72      /* RTEMS determines if a trap is sychronous or asynchronous
73       * by adding 0x100 to the sychronous traps.
74       */
75      rtems_vector_number vecNum = 0x102; /* Ilegal instruction */
76
77      rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
78      rtems_isr_entry oldISR = NULL;
79
80      /* Install the handler twice
81       * At the second time the oldISR will contains the address
82       * of the testISR
83       */
84      returnStatus = rtems_interrupt_catch (testISR,
85                                            vecNum,
86                                            &oldISR);
87
88
89      printf ("rtems_interrupt_catch(): %d;", returnStatus);
90
91
92      /*
93       * TEST RTEMS_INTERRUPT_CATCH
94       */
95
96      /*Catch Ilegal Intruction traps*/
97      returnStatus = rtems_interrupt_catch (testISR,
98                                            vecNum,
99                                            &oldISR);
100
101
102     printf ("rtems_interrupt_catch(): %d;", returnStatus);
103
104     /* Ilegal Instruction - execute trap handler */
105     asm("unimp");
106
107
108     /* Assertion: check if the cnt was incremented by the ISR*/
109     if (cnt == 1)
110     {
111        printf ("Assertion-01: success;");
112     }
113     else
114     {
115        printf ("Assertion-01: failure;");
116     }
117
118     /* Assertion: check if oldISR was testISR*/
119     if (oldISR == testISR)
120     {
121        printf ("Assertion-02: success;");
122     }
123     else
124     {
125        printf ("Assertion-02: failure;");
126     }
127
```

```
128    exit(0);
129 }
130
131 /* configuration information */
132
133 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
134
135 #define CONFIGURE_MAXIMUM_TASKS 4
136 #define CONFIGURE_INIT_TASK_ATTRIBUTES RTEMS_FLOATING_POINT
137 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
138
139 #define CONFIGURE_INIT
140
141 #include <confdefs.h>
142
143 /* end of file */
```

B.1.6. RTEMS-CMP-CL-IO.C

B.1.6.1. Description

This workload uses only one task. This main task, Init, is used to perform all tests to the io manager. It initializes a dummy device and performs the register name, open device, read, write and control operations and them closes the device.

1. Initialize RTEMS IO
2. **Assertion 1:** Global variable is correctly set
3. Register RTEMS IO test name
4. Lookup RTEMS name
5. **Assertion 2:** RTEMS IO name correctly registed, values for major and minor number are correct
6. Open RTEMS IO
7. **Assertion 3:** Global variable is correctly set
8. Write test data to RTEMS IO
9. Read test data from the RTEMS IO device
10. **Assertion 4:** Global variable is correctly set
11. Control function of RTEMS IO device
12. **Assertion 5:** Global variable is correctly set
13. Close RTEMS IO
14. **Assertion 6:** Global variable is correctly set

B.1.6.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : IO Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-io/rtems-cmp-cl-io.c,v $
6    $Id: rtems-cmp-cl-io.c,v 1.4 2003/09/02 09:06:49 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
```

```
14  OS-TYPE : RTEMS 4.5.0
15
16  AUTHOR : lvelkov
17
18  KEYWORDS : ----
19  PURPOSE : Test the functionality of all RTEMS APIs related to the IO
20            Manager.
21
22  CREATED ON : 17-07-2003
23  CHANGED ON : $Date: 2003/09/02 09:06:49 $
24  CHANGED BY : $Author: rmaia $
25
26  $Revision: 1.4 $
27  STICKY TAG : $Name:  $
28
29  INSPECTED ON: 25-07-2003
30  MODERATOR : rbarbosa
31
32  TABLES : none.
33
34  HISTORY
35  $Log: rtems-cmp-cl-io.c,v $
36  Revision 1.4  2003/09/02 09:06:49  rmaia
37  Added some comments
38
39  Revision 1.2  2003/08/05 16:32:45  rbarbosa
40  Updated after review
41
42  Revision 1.1  2003/07/29 14:28:05  rbarbosa
43  Ready to compile in RTEMS cross-compiling system.
44  Added configuration and replace main by init on code.
45
46  Makefile for IO manager robustness test workload
47
48  Revision 1.4  2003/07/25 17:16:04  rbarbosa
49  Updated the returnStatus variable initialisation scheme
50
51  Revision 1.3  2003/07/25 16:44:37  rbarbosa
52  Update after code changes
53
54  Revision 1.2  2003/07/25 15:20:10  rbarbosa
55  Revision and addition of header
56
57
58  ****************************************************************************/
59
60 #include <bsp.h>
61 #include <rtems.h>
62 #include <stdio.h>
63
64 /*** Definition of the test device ***/
65
66 #define DEVICE_NOT_INIT -1
67 #define DEVICE_INIT 0
68 #define DEVICE_OPENED 1
69 #define DEVICE_CLOSED 2
70
71 /* Global variable used in the tests */
```

```
72  int deviceStatus = DEVICE_NOT_INIT;
73  int deviceValue = -1;
74
75  /* Routine that emulates device driver initialisation */
76  rtems_device_driver dev_init(
77    rtems_device_major_number major,
78    rtems_device_minor_number minor,
79    void                    * arg
80  )
81  {
82      deviceStatus = DEVICE_INIT;
83      return RTEMS_SUCCESSFUL;
84  }
85
86  /* Routine that emulates device driver opening */
87  rtems_device_driver dev_open(
88    rtems_device_major_number major,
89    rtems_device_minor_number minor,
90    void                    * arg
91  )
92  {
93      deviceStatus = DEVICE_OPENED;
94      return RTEMS_SUCCESSFUL;
95  }
96
97
98  /* Routine that emulates the write operation in the device */
99  rtems_device_driver dev_write(
100   rtems_device_major_number major,
101   rtems_device_minor_number minor,
102   void                    * arg
103 )
104 {
105     deviceValue = *((int *) arg);
106     return RTEMS_SUCCESSFUL;
107 }
108
109 /* Routine that emulates the read operation in the device */
110 rtems_device_driver dev_read(
111   rtems_device_major_number major,
112   rtems_device_minor_number minor,
113   void                    * arg
114 )
115 {
116     *((int *) arg) = deviceValue;
117     return RTEMS_SUCCESSFUL;
118 }
119
120 /* Routine that emulates a specific functionality operation of the device */
121 rtems_device_driver dev_control(
122   rtems_device_major_number major,
123   rtems_device_minor_number minor,
124   void                    * arg
125 )
126 {
127     deviceStatus = DEVICE_NOT_INIT;
128     return RTEMS_SUCCESSFUL;
129 }
```

```
130
131
132 /* Routine that emulates the close operation in the device */
133 rtems_device_driver dev_close(
134   rtems_device_major_number major,
135   rtems_device_minor_number minor,
136   void                    * arg
137 )
138 {
139     deviceStatus = DEVICE_CLOSED;
140     return RTEMS_SUCCESSFUL;
141 }
142
143 /*** End of the definition of the test device ***/
144
145 /* Init Task uses the driver */
146
147 #define TEST_VALUE 0xF0F0
148
149
150 /*
151  * The Init task is the only task intiated
152  * and performs all testing operations
153  */
154
155
156 rtems_task Init(rtems_task_argument ignored)
157 {
158    char deviceName[16] = "testDevice";
159
160    rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
161
162    /* The test driver is in the fourth entry (index = 3) of the drivers table*/
163    rtems_device_major_number majorNum = 0x03;
164    rtems_device_minor_number minorNum = 0x00;
165
166    rtems_driver_name_t *deviceInfo;
167
168    int testData = TEST_VALUE;
169
170
171    /*
172     * TEST RTEMS_IO_INITIALIZE
173     */
174
175
176    /*Initialize RTEMS IO*/
177    returnStatus = rtems_io_initialize (majorNum,
178                                        minorNum,
179                                        NULL);
180
181    printf ("rtems_io_initialize(): %d;", returnStatus);
182
183    /* Assertion 1 for rtems_io_initialize */
184    if (deviceStatus == DEVICE_INIT)
185    {
186       printf ("Assertion-01: success;");
187    }
```

```
188    else
189    {
190       printf ("Assertion-01: failure;");
191    }
192
193
194
195    /*
196     * TEST RTEMS_IO_REGISTER_NAME & RTEMS_IO_LOOKUP_NAME
197     */
198
199    /* Register RTEMS IO name */
200    returnStatus = rtems_io_register_name (deviceName,
201                                           majorNum,
202                                           minorNum);
203
204    printf ("rtems_io_register_name(): %d;", returnStatus);
205
206    /* Assertion check */
207    /* If the name is registed, then the directive for look up on the device name
208       must be successfull */
209
210    rtems_io_lookup_name (deviceName,
211                          &deviceInfo);
212
213    if ((deviceInfo->major == majorNum) && (deviceInfo->minor == minorNum))
214    {
215       printf ("Assertion-02: success;");
216    }
217    else
218    {
219       printf ("Assertion-02: failure;");
220    }
221
222
223    /*
224     * TEST RTEMS_IO_OPEN
225     */
226
227
228    /*Open RTEMS IO*/
229    returnStatus = rtems_io_open (majorNum,
230                                  minorNum,
231                                  NULL);
232
233
234    printf ("rtems_io_open(): %d;", returnStatus);
235
236    /* Assertion 3 for rtems_io_open */
237    if (deviceStatus == DEVICE_OPENED)
238    {
239       printf ("Assertion-03: success;");
240    }
241    else
242    {
243       printf ("Assertion-03: failure;");
244    }
245
```

```
246
247    /*
248     * TEST RTEMS_IO_WRITE & TEST RTEMS_IO_READ
249     */
250
251
252    /*Write to RTEMS IO*/
253    returnStatus = rtems_io_write (majorNum,
254                                   minorNum,
255                                   (void *) &testData);
256
257
258    printf ("rtems_io_write(): %d;", returnStatus);
259
260    /* Assertion check */
261    /* If we have write on the test device a number, we chould be able to
262       read the same number back */
263
264    /* Set variable to zero in order to collect data from device */
265    testData = 0;
266
267    returnStatus = rtems_io_read(majorNum,
268                                 minorNum,
269                                 (void *) &testData);
270
271    printf("rtems_io_read(): %d;",returnStatus);
272
273    if (testData == TEST_VALUE)
274    {
275       printf ("Assertion-04: success;");
276    }
277    else
278    {
279       printf ("Assertion-04: failure;");
280    }
281
282
283    /*
284     * TEST RTEMS_IO_CONTROL
285     */
286
287
288    /*Control RTEMS IO*/
289    returnStatus = rtems_io_control (majorNum,
290                                     minorNum,
291                                     NULL);
292
293    printf ("rtems_io_control(): %d;", returnStatus);
294
295    /* Assertion 6 for rtems_io_control */
296    if (deviceStatus == DEVICE_NOT_INIT)
297    {
298       printf ("Assertion-05: success;");
299    }
300    else
301    {
302       printf ("Assertion-05: failure;");
303    }
```

```
304
305
306    /*
307     * TEST RTEMS_IO_CLOSE
308     */
309
310
311    /*Close RTEMS IO*/
312
313    returnStatus = rtems_io_close (majorNum,
314                                   minorNum,
315                                   NULL);
316
317    printf ("rtems_io_close(): %d;", returnStatus);
318
319    /* Assertion 7 for rtems_io_open */
320    if (deviceStatus == DEVICE_CLOSED)
321    {
322       printf ("Assertion-06: success;");
323    }
324    else
325    {
326       printf ("Assertion-06: failure;");
327    }
328
329
330    printf ("\n");
331
332    exit(0);
333 }
334
335 /* configuration information */
336
337 #define CONFIGURE_HAS_OWN_DEVICE_DRIVER_TABLE
338
339 #include <console.h> /* for CONSOLE_DRIVER_TABLE_ENTRY */
340 #include <clockdrv.h> /* for CLOCK_DRIVER_TABLE_ENTRY */
341
342 rtems_driver_address_table Device_drivers[] = {
343   CONSOLE_DRIVER_TABLE_ENTRY,
344   CLOCK_DRIVER_TABLE_ENTRY,
345   { NULL, NULL, NULL, NULL, NULL, NULL },
346   { dev_init, dev_open, dev_close, dev_read, dev_write, dev_control }
347 };
348
349
350 #define CONFIGURE_MAXIMUM_TASKS 1
351 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
352 #define CONFIGURE_INIT
353
354 #include <confdefs.h>
355
356 /* end of file */
```

B.1.7. RTEMS-CMP-CL-MSG.C

B.1.7.1. Description

This workload contains three tasks. The main task (Init) creates two queues and two test tasks. One of the queues is used by the main task to send messages to the test tasks and the other queue is used by the test tasks to send messages to the main task. The first task created (TA1) has a higher priority then the second task created (TA2).

1. Create RTEMS message queue with access policy FIFO, named 'Q1'
2. Create RTEMS message queue with access policy PRIORITY, named 'Q2'
3. **Assertion 1:** Q2 message queue ID successfuly assigned
4. Create and start the two tasks, named 'TA1' and 'TA2'
5. Send a normal and urgent message to the 'Q1'
6. 'TA1' receives the urgent message and sends it back to the Init task through 'Q2'
7. Init task receives a message
8. **Assertion 2:** The message was send by task 'TA1' and it is an urgent message
9. Get number of pending messages
10. **Assertion 3:** The number of messages is correct
11. Flush queue 'Q1'
12. **Assertion 4:** The number of messages flushed from the queue is correct
13. Broadcast a normal message to queue 'Q1'
14. 'TA1' receives the normal message and sends it back to the Init task through 'Q2'
15. **Assertion 5:** The message was send by task 'TA1' and it is a normal message
16. 'TA2' receives the normal message and sends it back to the Init task through 'Q2'
17. **Assertion 6:** The message was send by task 'TA2' and it is a normal message
18. Delete RTEMS message queue 'Q1'
19. Delete RTEMS message queue 'Q2'
20. Get 'Q2' message queue identifier
21. **Assertion 7:** Q2 Message queue identifier not available

B.1.7.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Message Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-msg/rtems-cmp-cl-msg.c,v $
6    $Id: rtems-cmp-cl-msg.c,v 1.7 2003/09/12 14:56:46 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Message
20             Manager.
21
22   CREATED ON : 17-07-2003
```

```
23   CHANGED ON : $Date: 2003/09/12 14:56:46 $
24   CHANGED BY : $Author: rmaia $
25
26   $Revision: 1.7 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 30-07-2003
30   MODERATOR : rbarbosa
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-msg.c,v $
36   Revision 1.7  2003/09/12 14:56:46  rmaia
37   Replaced constants in the rtems call by variables
38
39   Revision 1.6  2003/09/08 16:17:49  rmaia
40   *** empty log message ***
41
42   Revision 1.5  2003/08/28 16:30:20  rmaia
43   Changed the algorithm of the worload.
44   Compiled and tested with success.
45
46   Revision 1.4  2003/08/05 16:26:11  rbarbosa
47   Updated after review
48
49   Revision 1.3  2003/08/05 14:48:28  rbarbosa
50   Update after reviewing code
51
52   Revision 1.2  2003/08/05 14:47:04  rbarbosa
53   Update after formating file
54
55   Revision 1.1  2003/08/05 11:49:59  rbarbosa
56   Updated after reviewed. Added header and RTEMS specific code lines.
57   Ready for compilation
58
59
60   ******************************************************************************/
61
62  #include <bsp.h>
63  #include <stdio.h>
64
65  #define MAX_MSG_SIZE 20
66  #define MAX_MESSAGES 20
67
68  rtems_name msgQueueToTasksName;
69  rtems_id msgQueueToTasksId = 0;
70  rtems_id *ptmsgQueueToTasksId = &msgQueueToTasksId;
71
72  rtems_name msgQueueFromTasksName;
73  rtems_id msgQueueFromTasksId = 0;
74  rtems_id *ptmsgQueueFromTasksId = &msgQueueFromTasksId;
75
76  #define TASKS_BASE_PRIORITY 10
77  #define TICKS_TO_BLOCK 200
78
79
80  enum testTasks {
```

```
81      INIT_TASK = -1,
82      TASK1,
83      TASK2,
84      N_TASKS
85  };
86
87  enum msgTypes{
88      NORMAL = 15,
89      URGENT,
90  };
91
92
93  /*
94   * Test Tasks body
95   * It simply receives a message one one queue and sent it
96   * back to InitTask using the other queue.
97   * It changes also the id of the sender in the message
98   */
99  rtems_task testTask(rtems_task_argument taskSenderId)
100 {
101    int msgBuf[MAX_MSG_SIZE];
102    rtems_unsigned32 msgSize;
103
104    rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
105
106    while(1){
107         /*Receive on RTEMS message queue*/
108         returnStatus = rtems_message_queue_receive (msgQueueToTasksId,
109                                                     msgBuf,
110                                                     &msgSize,
111                                                     RTEMS_WAIT,
112                                                     RTEMS_NO_TIMEOUT);
113
114
115         msgBuf[0]=taskSenderId;
116
117         /*Send on RTEMS message queue*/
118         returnStatus = rtems_message_queue_send (msgQueueFromTasksId,
119                                                  msgBuf,
120                                                  msgSize);
121
122     }
123 }
124
125 /*
126  * INIT TASK
127  *
128  * It creates the test Task and start them.
129  * It sends a normal message and an urgent message to queue and then it wait
130  * for an ack from the first task (the test task with higher priority )on
131  * the other queue. Then it flushes the queue and send a broacast message to
132  * the test tasks. Finally it receives the ack from both tasks and deletes the
133  * message queues.
134  */
135
136 rtems_task Init(rtems_task_argument ignored)
137 {
138
```

```
139   rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
140
141   /* Message Queue Id retrived by rtems_message_queue_ident*/
142   rtems_id assertionMsgQueueId = 0;
143
144   /* Buffer to Store the messages to send */
145   int msgSendBuf[2];
146   /* Size of the messages sent */
147   int msgSendSize = sizeof(msgSendBuf);
148
149   /* Buffer to Store the messages received */
150   int msgReceiveBuf[MAX_MSG_SIZE];
151   /* Size of the message received */
152   int msgReceiveSize;
153   int *ptmsgReceiveSize = &msgReceiveSize;
154
155   rtems_unsigned32 MaxMessages  = MAX_MESSAGES;
156   rtems_unsigned32 MaxMsgSize   = MAX_MSG_SIZE;
157   rtems_attribute  attribute    = RTEMS_FIFO | RTEMS_LOCAL;
158   rtems_unsigned32 option       = RTEMS_WAIT;
159   rtems_interval   timeout      = RTEMS_NO_TIMEOUT;
160   rtems_unsigned32 nMessages    = 0;
161   rtems_unsigned32 *ptnMessages = &nMessages;
162
163   int i;
164
165   rtems_name          taskNames[N_TASKS];
166   rtems_id            taskIds[N_TASKS];
167   rtems_task_priority taskPriorities[N_TASKS];
168
169   msgQueueToTasksName   = rtems_build_name('Q','1',' ',' ');
170   msgQueueFromTasksName = rtems_build_name('Q','2',' ',' ');
171
172   for(i = TASK1; i < N_TASKS; i++){
173        taskPriorities[i] = TASKS_BASE_PRIORITY + i;
174        taskNames[i] = rtems_build_name('T','A','0' + i,' ');
175   }
176
177   /*Create RTEMS message queue*/
178   returnStatus = rtems_message_queue_create (msgQueueToTasksName,
179                                              MaxMessages,
180                                              MaxMsgSize,
181                                              attribute,
182                                              ptmsgQueueToTasksId);
183   /*
184    * TEST RTEMS_MESSAGE_QUEUE_CREATE
185    */
186
187   /*Create RTEMS message queue*/
188   returnStatus = rtems_message_queue_create (msgQueueFromTasksName,
189                                              MaxMessages,
190                                              MaxMsgSize,
191                                              attribute,
192                                              ptmsgQueueFromTasksId);
193
194   printf ("rtems_message_queue_create(): %d;", returnStatus);
195
196   returnStatus = rtems_message_queue_ident (msgQueueFromTasksName,
```

```
197                                          RTEMS_SEARCH_ALL_NODES,
198                                          &assertionMsgQueueId);
199
200    printf ("rtems_message_queue_ident(): %d;", returnStatus);
201
202    /* Assertion check */
203    if (msgQueueFromTasksId == assertionMsgQueueId)
204    {
205       printf ("Assertion-01: success;");
206    }
207    else
208    {
209       printf ("Assertion-01: failure;");
210    }
211
212
213    /*
214     * CREATE & START TASKS
215     */
216
217    for(i = TASK1; i < N_TASKS; i++){
218
219         returnStatus = rtems_task_create (taskNames[i],
220                                           taskPriorities[i],
221                                           RTEMS_MINIMUM_STACK_SIZE,
222                                           RTEMS_DEFAULT_MODES,
223                                           RTEMS_DEFAULT_ATTRIBUTES,
224                                           &taskIds[i]);
225
226         printf ("rtems_task_create(): %d;", returnStatus);
227
228         returnStatus =  rtems_task_start (taskIds[i],
229                                           testTask,
230                                           (rtems_task_argument) i);
231
232         printf ("rtems_task_start(): %d;", returnStatus);
233    }
234
235    msgSendBuf[0] = INIT_TASK;
236    msgSendBuf[1] = NORMAL;
237
238    /*
239     * TEST RTEMS_MESSAGE_QUEUE_SEND
240     */
241
242    /* Send message */
243    returnStatus = rtems_message_queue_send (msgQueueToTasksId,
244                                             msgSendBuf,
245                                             msgSendSize);
246
247    printf ("rtems_message_queue_send(): %d;", returnStatus);
248
249    msgSendBuf[0] = INIT_TASK;
250    msgSendBuf[1] = URGENT;
251
252
253    /*
254     * TEST RTEMS_MESSAGE_QUEUE_URGENT
```

```
255     */
256
257
258    /* Send urgent message */
259    returnStatus = rtems_message_queue_urgent (msgQueueToTasksId,
260                                               msgSendBuf,
261                                               msgSendSize);
262
263    printf ("rtems_message_queue_urgent(): %d;", returnStatus);
264
265    /*
266     * TEST RTEMS_MESSAGE_QUEUE_RECEIVE
267     */
268
269    /*
270     * The first task (the one with higher priority) will receive
271     * the urgent message and ack
272     */
273    returnStatus = rtems_message_queue_receive (msgQueueFromTasksId,
274                                                msgReceiveBuf,
275                                                ptmsgReceiveSize,
276                                                option,
277                                                timeout);
278
279    printf ("rtems_message_queue_receive(): %d;", returnStatus);
280
281    /* Assertion check */
282    if ((msgReceiveBuf[0] == TASK1) && (msgReceiveBuf[1] == URGENT))
283    {
284       printf ("Assertion-02: success;");
285    }
286    else
287    {
288       printf ("Assertion-02: failure;");
289    }
290
291    /*
292     * TEST RTEMS_MESSAGE_QUEUE_GET_NUMBER_PENDING
293     */
294    returnStatus = rtems_message_queue_get_number_pending(msgQueueToTasksId,
295                                                          ptnMessages);
296
297    printf ("rtems_message_queue_get_number_pending(): %d;", returnStatus);
298
299    /* Assertion check */
300    if (nMessages == 1)
301    {
302       printf ("Assertion-03: success;");
303    }
304    else
305    {
306       printf ("Assertion-03: failure;");
307    }
308
309    /*
310     * TEST RTEMS_MESSAGE_QUEUE_FLUSH
311     */
312
```

```
313    returnStatus = rtems_message_queue_flush(msgQueueToTasksId,
314                                             ptnMessages);
315
316    printf ("rtems_message_queue_flush(): %d;", returnStatus);
317
318    /* Assertion check */
319    if (nMessages == 1)
320    {
321       printf ("Assertion-04: success;");
322    }
323    else
324    {
325       printf ("Assertion-04: failure;");
326    }
327
328    /*
329     * TEST RTEMS_MESSAGE_QUEUE_BROADCAST
330     */
331    msgSendBuf[0] = INIT_TASK;
332    msgSendBuf[1] = NORMAL;
333
334    returnStatus = rtems_task_wake_after(TICKS_TO_BLOCK);
335    printf ("rtems_task_wake_after(): %d;", returnStatus);
336
337
338    /* Send message */
339    returnStatus = rtems_message_queue_broadcast (msgQueueToTasksId,
340                                                  msgSendBuf,
341                                                  msgSendSize,
342                                                  ptnMessages);
343
344    printf ("rtems_message_queue_broadcast(): %d;", returnStatus);
345
346    /* Receive ack from all tasks */
347    /* The first message will be from task 1, second from task 2, etc ..*/
348    for(i = TASK1; i < N_TASKS ; i++)
349    {
350
351         returnStatus = rtems_message_queue_receive (msgQueueFromTasksId,
352                                                     msgReceiveBuf,
353                                                     ptmsgReceiveSize,
354                                                     RTEMS_WAIT,
355                                                     RTEMS_NO_TIMEOUT);
356
357         /* Assertion check */
358         if ((msgReceiveBuf[0] == i) && (msgReceiveBuf[1] == NORMAL))
359         {
360             printf ("Assertion-0%d: success;", 5 + (i - TASK1));
361         }
362         else
363         {
364             printf ("Assertion-0%d: failure;", 5 + (i - TASK1));
365         }
366    }
367
368    returnStatus = rtems_message_queue_delete (msgQueueToTasksId);
369
370    printf ("rtems_message_queue_delete(): %d;", returnStatus);
```

```
371
372
373    /*
374     * TEST RTEMS_MESSAGE_QUEUE_DELETE
375     */
376
377
378    returnStatus = rtems_message_queue_delete (msgQueueFromTasksId);
379
380    printf ("rtems_message_queue_delete(): %d;", returnStatus);
381
382    returnStatus = rtems_message_queue_ident (msgQueueFromTasksName,
383                                              RTEMS_SEARCH_ALL_NODES,
384                                              &assertionMsgQueueId);
385
386    /* Assertion check */
387    if (returnStatus == RTEMS_INVALID_NAME)
388    {
389       printf ("Assertion-07: success;");
390    }
391    else
392    {
393       printf ("Assertion-07: failure;");
394    }
395
396
397    printf ("\n");
398
399    exit(0);
400 }
401
402 /* configuration information */
403
404 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
405 #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
406
407 #define CONFIGURE_MAXIMUM_TASKS 3
408 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
409 #define CONFIGURE_MAXIMUM_MESSAGE_QUEUES 2
410
411 #define CONFIGURE_INIT_TASK_INITIAL_MODES RTEMS_PREEMPT
412 #define CONFIGURE_INIT
413
414 #include <confdefs.h>
415
416 /* end of file */
```

B.1.8. RTEMS-CMP-CL-PRT.C

B.1.8.1. Description

This workload has only one main task. This task is used to perform all tests for the partition manager.

This task (Init) does the following:

1. Create RTEMS partition
2. **Assertion 1:** RTEMS partition ID successfuly assigned
3. Retreive a buffer from a partition
4. **Assertion 2:** The buffer was successfully allocated at the beginning of the partition
5. Return the allocated buffer to the partition
6. Retreive the same buffer from the partition
7. **Assertion 3:** The buffer was successfully allocated at the beginning of the partition
8. Delete RTEMS partition
9. Get partition identifier
10. **Assertion 4:** RTEMS partition identifier not available

B.1.8.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Partition Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-prt/rtems-cmp-cl-prt.c,v $
6    $Id: rtems-cmp-cl-prt.c,v 1.4 2003/09/09 17:31:51 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Partition
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/09 17:31:51 $
24   CHANGED BY : $Author: rmaia $
25
26   $Revision: 1.4 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 25-07-2003
30   MODERATOR : rbarbosa
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-prt.c,v $
36   Revision 1.4  2003/09/09 17:31:51  rmaia
37   *** empty log message ***
38
39   Revision 1.3  2003/08/28 18:05:45  rmaia
40   Compiled and tested with success.
41
42   Revision 1.2  2003/08/05 16:28:58  rbarbosa
43   Updated after review
44
```

```
45   Revision 1.1  2003/07/29 14:28:21  rbarbosa
46   Ready to compile in RTEMS cross-compiling system.
47   Added configuration and replace main by init on code.
48
49   Makefile for partition manager robustness test workload
50
51   Revision 1.4  2003/07/25 17:16:29  rbarbosa
52   Updated the returnStatus variable initialisation scheme
53
54   Revision 1.3  2003/07/25 16:13:10  rbarbosa
55   Update after code changes
56
57   Revision 1.2  2003/07/25 14:39:38  rbarbosa
58   First revision and added header
59
60
61   ***************************************************************************/
62
63
64  #include <bsp.h>
65  #include <stdio.h>
66
67
68  /*
69   * Init task is the only task of this workload.
70   */
71
72  rtems_task Init(rtems_task_argument ignored)
73  {
74     /* Partition Name */
75     rtems_name partitionName;
76     /* Partition ID */
77     rtems_id partitionId = 0;
78     rtems_id *ptpartitionId = &partitionId;
79
80     /* Partition ID returned by ident */
81     rtems_id assertionPartitionId = 0;
82
83
84     rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
85
86     /* Only two slots are available for allocating buffers*/
87     rtems_unsigned32 startAddress[1024] CPU_STRUCTURE_ALIGNMENT;
88     rtems_unsigned32 length = 1024;
89     rtems_unsigned32 bufferSize = 512;
90
91     rtems_attribute attributes = RTEMS_DEFAULT_ATTRIBUTES;
92
93     void *firstBuffer = NULL;
94     void **ptfirstBuffer = &firstBuffer;
95
96     void *secondBuffer = NULL;
97     void **ptsecondBuffer = &secondBuffer;
98
99     partitionName = rtems_build_name('P','T','1',' ');
100
101
102    /*
```

```
103     * TEST RTEMS_PARTITION_CREATE
104     */
105
106
107    /*Create RTEMS partition*/
108
109    returnStatus = rtems_partition_create (partitionName,
110                                           startAddress,
111                                           length,
112                                           bufferSize,
113                                           attributes,
114                                           ptpartitionId);
115
116    printf ("rtems_partition_create(): %d;", returnStatus);
117
118
119    returnStatus = rtems_partition_ident (partitionName,
120                                          RTEMS_SEARCH_ALL_NODES,
121                                          &assertionPartitionId);
122    /*Assertion check*/
123    /* If the prtId is assigned a value diferent than NULL, then the
124       directive is working properly */
125
126    if (partitionId == assertionPartitionId)
127    {
128       printf ("Assertion-01: success;");
129    }
130    else
131    {
132       printf ("Assertion-01: failure;");
133    }
134
135
136    /*
137     * TEST RTEMS_PARTITION_GET_BUFFER
138     */
139
140
141    returnStatus = rtems_partition_get_buffer(partitionId,
142                                              ptfirstBuffer);
143
144    printf ("rtems_partition_get_buffer(): %d;", returnStatus);
145
146
147    /*The first buffer is allocated on the start of the region*/
148    if (firstBuffer == startAddress)
149    {
150       printf ("Assertion-02: success;");
151    }
152    else
153    {
154       printf ("Assertion-02: failure;");
155    }
156
157
158    /*
159     * TEST RTEMS_PARTITION_RETURN_BUFFER
160     */
```

```
161
162
163    returnStatus = rtems_partition_return_buffer(partitionId,
164                                                 firstBuffer);
165
166    printf ("rtems_partition_return_buffer(): %d;", returnStatus);
167
168    /* Allocates a buffer on the second slot */
169    returnStatus = rtems_partition_get_buffer(partitionId,
170                                              ptfirstBuffer);
171
172    printf ("rtems_partition_get_buffer(): %d;", returnStatus);
173
174    /* Allocates again a buffer on the first slot*/
175    returnStatus = rtems_partition_get_buffer(partitionId,
176                                              ptsecondBuffer);
177
178    printf ("rtems_partition_get_buffer(): %d;", returnStatus);
179
180    /*The first buffer is allocated on the start of the region*/
181    if (secondBuffer == startAddress)
182    {
183       printf ("Assertion-03: success;");
184    }
185    else
186    {
187       printf ("Assertion-03: failure;");
188    }
189
190    /* Buffer must be returned before deleting partition */
191    returnStatus = rtems_partition_return_buffer(partitionId,
192                                                 firstBuffer);
193    printf ("rtems_partition_return_buffer(): %d;", returnStatus);
194
195    returnStatus = rtems_partition_return_buffer(partitionId,
196                                                 secondBuffer);
197    printf ("rtems_partition_return_buffer(): %d;", returnStatus);
198
199
200    /*
201     * TEST RTEMS_PARTITION_DELETE
202     */
203
204
205    /*Delete RTEMS partition*/
206    returnStatus = rtems_partition_delete (partitionId);
207
208    printf ("rtems_partition_delete(): %d;", returnStatus);
209
210    /*Assertion check*/
211    /* It shouldn't be possible to access any information regarding to the
212       deleted partition */
213
214    assertionPartitionId = 0;
215    returnStatus = rtems_partition_ident(partitionName,
216                                         RTEMS_SEARCH_ALL_NODES,
217                                         &assertionPartitionId);
218
```

```
219    if (returnStatus == RTEMS_INVALID_NAME)
220    {
221       printf ("Assertion-04: success;");
222    }
223    else
224    {
225       printf ("Assertion-04: failure;");
226    }
227
228    printf ("\n");
229
230    exit(0);
231 }
232
233 /* configuration information */
234
235 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
236 #define CONFIGURE_MAXIMUM_TASKS 1
237 #define CONFIGURE_MAXIMUM_PARTITIONS 1
238 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
239 #define CONFIGURE_INIT
240
241 #include <confdefs.h>
242
243 /* end of file */
```

B.1.9. RTEMS-CMP-CL-RGN.C

B.1.9.1. Description

This workload only has one main task. This main task, Init, performs all tests of the region manager . It starts by ceating a region, gets a segment, extends and deletes the region.

This task (Init) does the following:

1. Create RTEMS region
2. **Assertion 1:** RTEMS region ID successfuly assigned
3. Get a predefined segment from region
4. **Assertion 2:** Segment is located within the expected limits
5. Get the retrieved segment size
6. **Assertion 2:** The size of the retrieved segment is correct
7. Return the segment rto the region
8. Get a new segment from region
9. **Assertion 3:** The new segment was retrieved from the region successfully
10. Extend RTEMS region
11. Get another segment from the region
12. **Assertion 3:** The new segment was retrieved from the region successfully
13. Delete RTEMS region
14. Get region identifier
15. **Assertion 4:** RTEMS region identifier is not available

Important Remark: Action 11 does not run successfully since the rtems_region_extend directive is not implemented.

B.1.9.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Region Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-rgn/rtems-cmp-cl-rgn.c,v $
6    $Id: rtems-cmp-cl-rgn.c,v 1.5 2003/09/11 09:39:08 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Region
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/11 09:39:08 $
24   CHANGED BY : $Author: rmaia $
25
26   $Revision: 1.5 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 25-07-2003
30   MODERATOR : rbarbosa
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-rgn.c,v $
36   Revision 1.5  2003/09/11 09:39:08  rmaia
37   *** empty log message ***
38
39   Revision 1.4  2003/09/10 18:53:16  rmaia
40   *** empty log message ***
41
42   Revision 1.3  2003/08/29 13:39:44  rmaia
43   Compiled and tested with success.
44
45   Revision 1.2  2003/08/05 16:30:02  rbarbosa
46   Updated after review
47
48   Revision 1.1  2003/07/29 14:28:34  rbarbosa
49   Ready to compile in RTEMS cross-compiling system.
50   Added configuration and replace main by init on code.
51
52   Makefile for region manager robustness test workload
53
54   Revision 1.4  2003/07/25 17:16:55  rbarbosa
55   Updated the returnStatus variable initialisation scheme
```

```
56
57   Revision 1.3  2003/07/25 16:18:53  rbarbosa
58   Update after code changes
59
60   Revision 1.2  2003/07/25 15:02:59  rbarbosa
61   Revision and addition of header
62
63
64   ****************************************************************************/
65
66
67  #include <bsp.h>
68  #include <stdio.h>
69  #include <rtems.h>
70
71  #define INITIAL_MEM_BLOCK_LEN   4096
72  #define EXTENTION_MEM_BLOCK_LEN 8192
73  #define PAGE_SIZE               128
74
75  #define REQUESTED_SIZE_1        2000
76  #define REQUESTED_SIZE_2        3072
77
78
79  /*
80   * Init task is the only task of this workload..
81   */
82  rtems_task Init(rtems_task_argument ignored)
83  {
84     /* Name of the Memory region */
85     rtems_name regionName;
86     /* Id of the memory region */
87     rtems_id regionId = 0;
88     rtems_id *ptregionId = &regionId;
89
90     /* Region Id return by rtems_region_ident */
91     rtems_name assertionRegionId;
92
93     /* status code return by rtems directives */
94     rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
95
96     /* block of memory to be used by the memory region */
97     rtems_unsigned8 startAddress[INITIAL_MEM_BLOCK_LEN] CPU_STRUCTURE_ALIGNMENT;
98     /* Another block of memory to extend the memory region */
99     rtems_unsigned8 extendAddress[EXTENTION_MEM_BLOCK_LEN] CPU_STRUCTURE_ALIGNMENT;
100
101    /* Pointers to the requested segments */
102    void *segment1 = NULL;
103    void **ptsegment1 = &segment1;
104
105    void *segment2 = NULL;
106    void **ptsegment2 = &segment2;
107
108    rtems_unsigned32 segment1Size = 0;
109    rtems_unsigned32 *ptsegment1Size = &segment1Size;
110
111    rtems_unsigned32 initialMemBlockLen = INITIAL_MEM_BLOCK_LEN;
112    rtems_unsigned32 extentionMemBlockLen = EXTENTION_MEM_BLOCK_LEN;
113    rtems_unsigned32 page_size = PAGE_SIZE;
```

```
114    rtems_attribute  attributes = RTEMS_DEFAULT_ATTRIBUTES;
115
116    rtems_unsigned32 requestedSize1 = REQUESTED_SIZE_1;
117    rtems_unsigned32 requestedSize2 = REQUESTED_SIZE_2;
118    rtems_option option = RTEMS_NO_WAIT;
119    rtems_interval timeout = RTEMS_NO_TIMEOUT;
120
121    regionName = rtems_build_name('R','N','1',' ');
122
123
124    /*
125     * TEST RTEMS_REGION_CREATE
126     */
127
128    /*Create RTEMS region*/
129    returnStatus = rtems_region_create (regionName,
130                                        (void *)startAddress,
131                                        initialMemBlockLen,
132                                        page_size,
133                                        attributes,
134                                        ptregionId);
135
136    printf ("rtems_region_create(): %d;", returnStatus);
137
138    returnStatus = rtems_region_ident (regionName,
139                                       &assertionRegionId);
140    /* Assertion check */
141    if (regionId == assertionRegionId)
142    {
143       printf ("Assertion-01: success;");
144    }
145    else
146    {
147       printf ("Assertion-01: failure;");
148    }
149
150
151    /*
152     * TEST RTEMS_REGION_GET_SEGMENT
153     */
154
155    returnStatus = rtems_region_get_segment (regionId,
156                                             requestedSize1,
157                                             option,
158                                             timeout,
159                                             ptsegment1);
160
161    printf ("rtems_region_get_segment(): %d;", returnStatus);
162
163    if ((segment1 > (void *) startAddress) &&
164        (segment1 < ((void *)startAddress + initialMemBlockLen - requestedSize1)))
165    {
166       printf ("Assertion-02: success;");
167    }
168    else
169    {
170       printf ("Assertion-02: failure;");
171    }
```

```
172
173    /*
174     * TEST RTEMS_REGION_GET_SEGMENT_SIZE
175     */
176
177    returnStatus = rtems_region_get_segment_size(regionId,
178                                                 segment1,
179                                                 ptsegment1Size);
180
181    printf ("rtems_region_get_segment_size(): %d;", returnStatus);
182
183    if (segment1Size >= REQUESTED_SIZE_1)
184    {
185       printf ("Assertion-03: success;");
186    }
187    else
188    {
189       printf ("Assertion-03: failure;");
190    }
191
192
193    /*
194     * TEST RTEMS_REGION_RETURN_SEGMENT
195     */
196
197    returnStatus = rtems_region_return_segment (regionId,
198                                                segment1);
199
200    printf ("rtems_region_return_segment(): %d;", returnStatus);
201
202    /* If segment1 was returned with success there is room for
203     * segment2 allocation */
204    returnStatus = rtems_region_get_segment (regionId,
205                                             requestedSize2,
206                                             option,
207                                             timeout,
208                                             ptsegment2);
209
210    if (returnStatus == RTEMS_SUCCESSFUL)
211    {
212       printf ("Assertion-04: success;");
213    }
214    else
215    {
216       printf ("Assertion-04: failure;");
217    }
218
219
220    /*
221     * TEST RTEMS_REGION_EXTEND
222     */
223
224
225    returnStatus = rtems_region_extend (regionId,
226                                        extendAddress,
227                                        extentionMemBlockLen);
228
229    printf ("rtems_region_extend(): %d;", returnStatus);
```

```
230
231    /* Now there should be room for both segments */
232    returnStatus = rtems_region_get_segment (regionId,
233                                             requestedSize1,
234                                             option,
235                                             timeout,
236                                             ptsegment1);
237
238    if (returnStatus == RTEMS_SUCCESSFUL)
239    {
240       printf ("Assertion-05: success;");
241    }
242    else
243    {
244       printf ("Assertion-05: failure;");
245    }
246
247
248    /* Segment 1 is not being allocated because rtems_region_extend
249    is not implemented.*/
250
251    /*returnStatus = rtems_region_return_segment (regionId,
252                                                  segment1);
253    printf ("rtems_region_return_segment(): %d;", returnStatus);*/
254
255    returnStatus = rtems_region_return_segment (regionId,
256                                                segment2);
257    printf ("rtems_region_return_segment(): %d;", returnStatus);
258
259    /*
260     * TEST RTEMS_REGION_DELETE
261     */
262    /*Delete RTEMS region*/
263    returnStatus = rtems_region_delete (regionId);
264
265    printf ("rtems_region_delete(): %d;", returnStatus);
266
267    /* Assertion check */
268    returnStatus = rtems_region_ident (regionName,
269                                       &assertionRegionId);
270
271
272    if (returnStatus == RTEMS_INVALID_NAME)
273    {
274       printf ("Assertion-06: success;");
275    }
276    else
277    {
278       printf ("Assertion-06: failure;");
279    }
280
281    printf ("\n");
282
283    exit(0);
284 }
285
286 /* configuration information */
287
```

```
288 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
289 #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
290 #define CONFIGURE_MAXIMUM_TASKS 1
291 #define CONFIGURE_MAXIMUM_REGIONS 10
292 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
293 #define CONFIGURE_INIT
294
295 #include <confdefs.h>
296
297 /* end of file */
```

B.1.10. RTEMS-CMP-CL-RMT.C

B.1.10.1. Description

This workload contains five tasks, one main task, the Init task, and four secondary tasks. The Init task creates and starts all four tasks. The secondary tasks create and manage their own rate monotonic period. Each of these task wait for their individual period to elapse and increment the corresponding global counter.

The Init task also has is own rate monotonic period, which is greater than the periods of the secondary tasks. Each time its periods elapses it gets the values of the global counters and check then against predefined values.

The main task (Init) performs the following actions:

1. Create and start all secondary tasks
2. Create RTEMS rate monotonic
3. **Assertion 1:** RTEMS rate monotonic ID successfuly assigned
4. Set its own rate monotonic period
5. Wait for a period to elapse
6. **Assertion 2:** Task 1 counter is correct.
7. **Assertion 3:** Task 2 counter is correct.
8. **Assertion 4:** Task 3 counter is correct.
9. **Assertion 5:** Task 4 counter is correct.
10. Cancel RTEMS rate monotonic
11. Set new rate monotonic period
12. **Assertion 6:** Task 1 counter is correct.
13. **Assertion 7:** Task 2 counter is correct.
14. **Assertion 8:** Task 3 counter is correct.
15. **Assertion 9:** Task 4 counter is correct.
16. Delete RTEMS rate monotonic
17. Get RTEMS rate monotonic identifier
18. **Assertion 10:** RTEMS rate monotonic identifier unavailable

B.1.10.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Rate Monotonic Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
   testing/implementation/classic-workloads/rtems-cmp-cl-rmt/rtems-cmp-cl-rmt.c,v $
6    $Id: rtems-cmp-cl-rmt.c,v 1.7 2003/09/11 13:23:08 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
```

```
 9
10  Copyright (c) Critical Software (www.criticalsoftware.com)
11
12  SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14  OS-TYPE : RTEMS 4.5.0
15
16  AUTHOR : lvelkov
17
18  KEYWORDS : ----
19  PURPOSE : Test the functionality of all RTEMS APIs related to the Rate
20            Monotonic Manager.
21
22  CREATED ON : 17-07-2003
23  CHANGED ON : $Date: 2003/09/11 13:23:08 $
24  CHANGED BY : $Author: rmaia $
25
26  $Revision: 1.7 $
27  STICKY TAG : $Name:  $
28
29  INSPECTED ON: 25-07-2003
30  MODERATOR : rbarbosa
31
32  TABLES : none.
33
34  HISTORY
35  $Log: rtems-cmp-cl-rmt.c,v $
36  Revision 1.7  2003/09/11 13:23:08  rmaia
37  *** empty log message ***
38
39  Revision 1.6  2003/09/02 11:01:10  rmaia
40  Commited after correcting issue related with number of assertions
41
42  Revision 1.5  2003/09/02 10:49:48  rmaia
43  Commited after correcting some commentaries
44
45  Revision 1.4  2003/09/02 10:20:19  rmaia
46  Commited after deleting unused variables and addin
47
48  Revision 1.3  2003/09/02 09:27:37  rmaia
49  Commited after changing tasks periods and adding commentaries
50
51  Revision 1.2  2003/09/01 19:14:01  rmaia
52  Compiled and tested with success.
53
54  Revision 1.1  2003/07/29 14:28:48  rbarbosa
55  Ready to compile in RTEMS cross-compiling system.
56  Added configuration and replace main by init on code.
57
58  Makefile for rate monotonic perido robustness test workload
59
60  Revision 1.4  2003/07/25 17:17:13  rbarbosa
61  Updated the returnStatus variable initialisation scheme
62
63  Revision 1.3  2003/07/25 17:05:40  rbarbosa
64  Update after code changes
65
66  Revision 1.2  2003/07/25 15:42:40  rbarbosa
```

```
67   Revision and addition of header
68
69
70   ***************************************************************************/
71
72  #include <bsp.h>
73  #include <rtems.h>
74  #include <stdio.h>
75
76  enum tasks{
77    INIT_TASK,
78    TASK1,
79    TASK2,
80    TASK3,
81    TASK4,
82    N_TASKS
83  };
84
85  struct counters {
86    rtems_unsigned32 Count[N_TASKS];
87  };
88
89  rtems_unsigned32    Periods[N_TASKS]    = { 200, 4,   5,  10,  20 };
90  rtems_unsigned32    Iterations[N_TASKS] = { 1,  50,  40,  20,  10 };
91  rtems_task_priority Priorities[N_TASKS] = { 5,   1,   2,   3,   4 };
92  rtems_name          TaskNames[N_TASKS];
93  rtems_id            TaskIds[N_TASKS];
94
95  struct counters     Count;
96  struct counters     TempCount;
97
98  /* void GetCounters(void)
99   *
100  * Description: Disables preemption and stores the Counter global variable
101  *              on a temporary location.
102  *              Reset Counters for TASK2..N_TASK - 1;
103  *              Finally it enables preemption again
104  */
105
106 void GetCounters(void){
107   int index;
108   rtems_mode        oldTaskMode;
109
110   /* return status return by the rtems directives */
111   rtems_status_code returnStatus;
112
113   /* Set init task mode to be not preemptable in order to access counters */
114   returnStatus = rtems_task_mode (RTEMS_NO_PREEMPT,
115                                   RTEMS_PREEMPT_MASK,
116                                   &oldTaskMode);
117   TempCount = Count;
118
119   for (index = TASK1; index < N_TASKS; index ++){
120     Count.Count[index] = 0;
121   }
122    /* Restore task mode */
123   returnStatus = rtems_task_mode (RTEMS_PREEMPT,
124                                   RTEMS_PREEMPT_MASK,
```

```
125                                  &oldTaskMode);
126
127 }
128
129 /*
130  * This test task creates its own rate monotonic
131  * period and manages it. It also increments a global
132  * variable that is  used for assertion purposes
133  */
134
135 rtems_task testTasks(
136   rtems_unsigned32 argument
137 )
138 {
139   /* Id of the Ratemonotonic Period */
140   rtems_id          rmid;
141
142   /* Name of the Ratemonotonic Period */
143   rtems_name        rmname = rtems_build_name('R','M','1' + argument,' ');
144
145   /* return status return by the rtems directives */
146   rtems_status_code returnStatus;
147
148   returnStatus = rtems_rate_monotonic_create(rmname,
149                                              &rmid );
150
151   /* First time rtems_rate_monotonic_period is called is sets the
152    * rate monotonic period */
153
154   returnStatus = rtems_rate_monotonic_period (rmid,
155                                               Periods[argument] );
156
157   while ( 1 ) {
158     returnStatus = rtems_rate_monotonic_period(rmid,
159                                                Periods[ argument ] );
160     Count.Count[argument]++;
161   }
162 }
163
164
165 /*
166  * Init task is the first task to execute
167  */
168
169
170 rtems_task Init(rtems_task_argument ignored)
171 {
172   int               i;
173   rtems_unsigned32  index;
174
175   /* Id of the Ratemonotonic Period */
176   rtems_id          rmid;
177   rtems_id         *ptrmid = &rmid;
178
179   /* Name of the Ratemonotonic Period */
180   rtems_name        rmname = rtems_build_name('R','M','1', ' ');
181
182   /* Id of the Ratemonotonic Period return by rtems_rate_monotonic_ident */
```

```
183  rtems_id          test_rmid;
184
185  /* return status return by the rtems directives */
186  rtems_status_code returnStatus;
187
188  /* Create Test Tasks */
189  for ( index = TASK1 ; index < N_TASKS ; index++ ) {
190
191    TaskNames[ index ] =  rtems_build_name( 'T', 'A', '1' + index , ' ' );
192
193    returnStatus = rtems_task_create (TaskNames[ index ],
194                                      Priorities[ index ],
195                                      RTEMS_MINIMUM_STACK_SIZE,
196                                      RTEMS_DEFAULT_MODES,
197                                      RTEMS_DEFAULT_ATTRIBUTES,
198                                      &TaskIds[ index ]);
199
200    Count.Count[index] = 0;
201  }
202
203
204  /* Start Test Tasks */
205  for ( index = TASK1 ; index < N_TASKS ; index++ ) {
206    returnStatus = rtems_task_start( TaskIds[ index ],
207                                     testTasks,
208                                     index );
209  }
210
211  /*
212   * TEST RTEMS_RATE_MONOTONIC_CREATE
213   */
214
215  returnStatus = rtems_rate_monotonic_create( rmname,
216                                              ptrmid );
217
218  printf ("rtems_rate_monotonic_create(): %d;", returnStatus);
219
220  /* Assertion */
221  returnStatus = rtems_rate_monotonic_ident( rmname,
222                                             &test_rmid );
223
224  printf ("rtems_rate_monotonic_ident(): %d;", returnStatus);
225
226  if (rmid == test_rmid)
227  {
228     printf ("Assertion-01: success;");
229  }
230  else
231  {
232     printf ("Assertion-01: failure;");
233  }
234
235  /*
236   * TEST RTEMS_RATE_MONOTONIC_PERIOD
237   */
238
239  returnStatus = rtems_rate_monotonic_period (rmid,
240                                              Periods[INIT_TASK] );
```

```
241
242   printf ("rtems_rate_monotonic_period(): %d;", returnStatus);
243
244   for(i = 0; i < 2; i++){
245       /* Wait for the next period */
246       returnStatus = rtems_rate_monotonic_period (rmid,
247                                                   Periods[INIT_TASK] );
248
249       printf ("rtems_rate_monotonic_period(): %d;", returnStatus);
250
251       /* Increments the number of times INIT_TASK executed
252        * This counter is not reset
253        */
254       Count.Count[INIT_TASK]++;
255
256       GetCounters();
257
258       /* Assertion check */
259       for(index = TASK1; index < N_TASKS; index++){
260         /*printf("Counter %d: %d;", index, TempCount.Count[index]);*/
261         if ( TempCount.Count[index] == Iterations[index] ){
262           printf ("Assertion-0%d: success;", i * (N_TASKS - TASK1) + index + 1);
263         }else
264         {
265           printf ("Assertion-0%d: failure;", i * (N_TASKS - TASK1) + index + 1);
266         }
267       }
268
269       if( i < 1){
270
271         Periods[INIT_TASK] = Periods[INIT_TASK] / 2;
272
273         for ( index = TASK1; index < N_TASKS; index++){
274           Iterations[index] = Iterations[index] / 2;
275         }
276
277         /*
278          * TEST RTEMS_RATE_MONOTONIC_CANCEL
279          */
280
281         returnStatus = rtems_rate_monotonic_cancel (rmid);
282
283         printf ("rtems_rate_monotonic_cancel(): %d;", returnStatus);
284
285         returnStatus = rtems_rate_monotonic_period (rmid,
286                                                     Periods[INIT_TASK] );
287       }
288       else
289       {
290
291         /*
292          * TEST RTEMS_RATE_MONOTONIC_DELETE
293          */
294
295         returnStatus = rtems_rate_monotonic_delete (rmid);
296
297         printf ("rtems_rate_monotonic_delete(): %d;", returnStatus);
298
```

```
299         returnStatus = rtems_rate_monotonic_ident(rmname,
300                                                   &test_rmid );
301
302         if (returnStatus == RTEMS_INVALID_NAME)
303         {
304           printf ("Assertion-10: success;");
305         }
306         else
307         {
308           printf ("Assertion-10: failure;");
309         }
310
311         printf("\n");
312
313         exit(0);
314       }
315     }
316 }
317
318
319
320 /* configuration information */
321
322 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
323 #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
324
325 #define CONFIGURE_EXTRA_TASK_STACKS         (15 * RTEMS_MINIMUM_STACK_SIZE)
326
327 #define CONFIGURE_MAXIMUM_TASKS 6
328 #define CONFIGURE_MAXIMUM_PERIODS 10
329 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
330 #define CONFIGURE_INIT_TASK_PRIORITY 5
331 #define CONFIGURE_INIT
332
333 #include <confdefs.h>
334
335 /* end of file */
```

B.1.11. RTEMS-CMP-CL-SGL.C

B.1.11.1. Description

This workload contains only one task. This main task, Init, performs all tests of the signal manager. It uses a function to handle the signal catching.

The main(Init) task does the following:

1. Establish an ASR to the task
2. **Assertion 1:** Handler successfully assigned
3. Send a RTEMS signal to self
4. **Assertion 2:** RTEMS signal sent successfully

B.1.11.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Signal Manager Workload
3    MODULE-VERS : N/A
```

```
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-sgl/rtems-cmp-cl-sgl.c,v $
6    $Id: rtems-cmp-cl-sgl.c,v 1.4 2003/09/09 13:42:21 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Signal
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/09 13:42:21 $
24   CHANGED BY : $Author: rmaia $
25
26   $Revision: 1.4 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 25-07-2003
30   MODERATOR : rbarbosa
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-sgl.c,v $
36   Revision 1.4  2003/09/09 13:42:21  rmaia
37   *** empty log message ***
38
39   Revision 1.3  2003/08/29 14:26:08  rmaia
40   Compiled and tested with success.
41
42   Revision 1.2  2003/08/05 16:28:28  rbarbosa
43   Updated after review
44
45   Revision 1.1  2003/07/29 14:29:02  rbarbosa
46   Ready to compile in RTEMS cross-compiling system.
47   Added configuration and replace main by init on code.
48
49   Makefile for signal manager robustness test workload
50
51   Revision 1.4  2003/07/25 17:17:40  rbarbosa
52   Updated the returnStatus variable initialisation scheme
53
54   Revision 1.3  2003/07/25 16:07:41  rbarbosa
55   Update after code changes
56
57   Revision 1.2  2003/07/25 14:19:57  rbarbosa
58   First revision and added header
59
60
```

```
61   ****************************************************************************/
62
63
64  #include <bsp.h>
65  #include <rtems.h>
66  #include <stdio.h>
67
68  /* Signal set received by the ASR */
69  rtems_signal_set testValue = 0;
70
71  /* Number of times ASR was called */
72  int cnt = 0;
73
74  /*
75   * This function is used as a signal handler
76   * It update testValue according to the set of signals received.
77   */
78
79  rtems_asr sigHndlr (rtems_signal_set signals)
80  {
81     testValue = signals;
82     cnt++;
83     return;
84  }
85
86  /*
87   * Init task is the only task of this workload.
88   */
89
90  rtems_task Init(rtems_task_argument ignored)
91  {
92
93     /* return status of rtems directives*/
94     rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
95
96     /* Mode of the ASR */
97     rtems_mode mode = RTEMS_NO_PREEMPT;
98
99     rtems_asr *ptsigHndlr = sigHndlr;
100    rtems_id id = RTEMS_SELF;
101    rtems_signal_set signalSet = RTEMS_SIGNAL_1;
102    cnt = 0;
103
104
105    /*
106     * TEST RTEMS_SIGNAL_CATCH
107     */
108
109
110    /*This directive establishes an ASR for the specified signal set*/
111    returnStatus = rtems_signal_catch (ptsigHndlr,
112                                       mode);
113
114    printf ("rtems_signal_catch(): %d;", returnStatus);
115
116    /*Assertion check*/
117    /*ASR was not called yet*/
118    if (cnt == 0)
```

```
119    {
120       printf ("Assertion-01: success;");
121    }
122    else
123    {
124       printf ("Assertion-01: failure;");
125    }
126
127
128    /*
129     * TEST RTEMS_SIGNAL_SEND
130     */
131
132
133    /* This directive sends a RTEMS signal to RTEMS_SELF, namely
134       to the 'Init' task */
135    returnStatus = rtems_signal_send (id, signalSet);
136
137    printf ("rtems_signal_send(): %d;", returnStatus);
138
139    /*Assertion check*/
140    if ((testValue == signalSet) && (cnt == 1))
141    {
142       printf ("Assertion-02: success;");
143    }
144    else
145    {
146       printf ("Assertion-02: failure;");
147    }
148
149
150    printf ("\n");
151
152    exit(0);
153 }
154
155 /* configuration information */
156
157 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
158 #define CONFIGURE_MAXIMUM_TASKS 4
159 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
160 #define CONFIGURE_INIT
161
162 #include <confdefs.h>
163
164 /* end of file */
```

B.1.12. RTEMS-CMP-CL-SMP.C

B.1.12.1. Description

This workload as two tasks. The main task performs all the operations regarding the semaphore test campaign, namely, create, delete, obtain, release and flush semaphores. The second task is used to aid the test of the flush semaphore directive.

1. Create a binary RTEMS semaphore
2. **Assertion 1:** RTEMS semaphore ID successfuly assigned
3. Obtain ownership of the RTEMS semaphore

4. Try to obtain ownership of the same semaphore
5. **Assertion 2:** The RTEMS semaphore could not be obtained
6. Release RTEMS semaphore in new task
7. Try to obtain the semaphore
8. **Assertion 3:** The semaphore was obtained correctly since it was released with success
9. Create a task
10. Block task while attemping to obtain a used semaphore
11. Flush RTEMS semaphore
12. **Assertion 4:** The semaphore was obtained by the task successfully.
13. Release semaphore
14. Delete RTEMS Semaphore
15. Try to obtain the identifier of the deleted semaphore
16. **Assertion 5:** Identifier not available

B.1.12.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Semaphore Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-smp/rtems-cmp-cl-smp.c,v $
6    $Id: rtems-cmp-cl-smp.c,v 1.5 2003/09/08 16:18:07 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Semaphore
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/08 16:18:07 $
24   CHANGED BY : $Author: rmaia $
25
26   $Revision: 1.5 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 31-07-2003
30   MODERATOR : rbarbosa
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-smp.c,v $
36   Revision 1.5  2003/09/08 16:18:07  rmaia
37   *** empty log message ***
38
39   Revision 1.4  2003/09/02 13:28:10  rmaia
40   Commited after fixing workload error and changing some comentaries
```

```
41
42   Revision 1.3  2003/09/02 12:35:05  rmaia
43    Compiled and tested with success.
44
45   Revision 1.2  2003/08/05 16:47:49  rbarbosa
46   Updated after review
47
48   Revision 1.1  2003/08/05 11:50:20  rbarbosa
49   Updated after reviewed. Added header and RTEMS specific code lines.
50   Ready for compilation
51
52
53   ****************************************************************************/
54
55  #include <bsp.h>
56  #include <stdio.h>
57
58  rtems_id semId = 0;
59  rtems_id *ptsemId = &semId;
60  rtems_name semName;
61  rtems_task_priority semPriority = RTEMS_NO_PRIORITY;
62  rtems_attribute semAttr = RTEMS_DEFAULT_ATTRIBUTES;
63  rtems_status_code testreturnStatus = RTEMS_NOT_DEFINED;
64
65
66  /*
67   * This task is used to test the rtems_semaphore_flush directive.
68   */
69
70
71  rtems_task testTask(rtems_task_argument testTaskArg)
72  {
73     rtems_unsigned32 test_option_set = RTEMS_WAIT;
74     rtems_interval test_timeout = RTEMS_NO_TIMEOUT;
75
76     testreturnStatus = rtems_semaphore_obtain(semId,
77                                               test_option_set,
78                                               test_timeout);
79  }
80
81
82  /*
83   * Init task is the first to be executed.
84   */
85
86
87  rtems_task Init(rtems_task_argument ignored)
88  {
89     rtems_unsigned32 count = 1;
90     rtems_status_code returnStatus = RTEMS_NOT_DEFINED;
91     rtems_unsigned32 option_set = RTEMS_WAIT;
92     rtems_interval timeout = 10;            /* ten ticks timeout */
93     rtems_unsigned32 node = RTEMS_SEARCH_ALL_NODES;
94
95     /* Test task related data structures */
96     rtems_name testTaskName;
97     rtems_task_priority testTaskPriority = 2;
98     rtems_unsigned32 testTaskStackSize = RTEMS_MINIMUM_STACK_SIZE;
```

```
99     rtems_mode testTaskMode = RTEMS_DEFAULT_MODES;
100    rtems_attribute testTaskAtt = RTEMS_DEFAULT_ATTRIBUTES;
101    rtems_id testTaskId = 0;
102
103    rtems_task_priority initTaskPriority = 15;
104    rtems_task_priority oldPrio = 0;
105    rtems_mode oldMode = 0;
106
107    semName = rtems_build_name('S','M','1',' ');
108    testTaskName = rtems_build_name( 'T', 'A', '1', ' ' );
109
110
111    /*
112     * TEST RTEMS_SEMAPHORE_CREATE
113     */
114
115
116    /*Create RTEMS semaphore*/
117    returnStatus = rtems_semaphore_create (semName,
118                                           count,
119                                           semAttr,
120                                           semPriority,
121                                           ptsemId);
122
123    printf ("rtems_semaphore_create(): %d;", returnStatus);
124
125    /*Assertion check*/
126    if (semId != 0)
127    {
128       printf ("Assertion-01: success;");
129    }
130    else
131    {
132       printf ("Assertion-01: failure;");
133    }
134
135
136    /*
137     * TEST RTEMS_SEMAPHORE_OBTAIN
138     */
139
140
141    returnStatus = rtems_semaphore_obtain (semId,
142                                           option_set,
143                                           timeout);
144
145    printf ("rtems_semaphore_obtain(): %d;", returnStatus);
146
147    /*Assertion check*/
148    returnStatus = rtems_semaphore_obtain (semId,
149                                           RTEMS_NO_WAIT,
150                                           timeout);
151
152    if (returnStatus == RTEMS_UNSATISFIED)
153    {
154       printf ("Assertion-02: success;");
155    }
156    else
```

```
157    {
158       printf ("Assertion-02: failure;");
159    }
160
161
162    /*
163     * TEST RTEMS_SEMAPHORE_RELEASE
164     */
165
166
167    returnStatus = rtems_semaphore_release(semId);
168
169    printf ("rtems_semaphore_release(): %d;", returnStatus);
170
171    /* Assertion check */
172    /* We should be able to obtain a semaphore that was released */
173    returnStatus = rtems_semaphore_obtain(semId,
174                                          option_set,
175                                          timeout);
176
177    if (returnStatus == 0)
178    {
179       printf ("Assertion-03: success;");
180    }
181    else
182    {
183       printf ("Assertion-03: failure;");
184    }
185
186
187    /*
188     * TEST RTEMS_SEMAPHORE_FLUSH
189     */
190
191
192    /* Create a task */
193    returnStatus = rtems_task_create (testTaskName,
194                                      testTaskPriority,
195                                      testTaskStackSize,
196                                      testTaskMode,
197                                      testTaskAtt,
198                                      &testTaskId);
199
200    printf ("rtems_task_create(): %d;", returnStatus);
201
202    /* Start the task */
203    returnStatus =  rtems_task_start (testTaskId,
204                                      &testTask,
205                                      semId);
206
207    printf ("rtems_task_start(): %d;", returnStatus);
208
209    /* Change Init task priority */
210    returnStatus = rtems_task_set_priority (RTEMS_SELF,
211                                            initTaskPriority,
212                                            &oldPrio);
213
214    printf ("rtems_task_set_priority(): %d;", returnStatus);
```

```
215
216    /* Change Init task mode */
217    returnStatus = rtems_task_mode (RTEMS_PREEMPT,
218                                    RTEMS_PREEMPT_MASK,
219                                    &oldMode);
220
221    printf ("rtems_task_mode(): %d;", returnStatus);
222
223
224    /* Flush RTEMS semaphore */
225    returnStatus = rtems_semaphore_flush (semId);
226
227    printf ("rtems_semaphore_flush(): %d;", returnStatus);
228
229    /* Assertion check */
230    if(testreturnStatus == RTEMS_UNSATISFIED)
231    {
232      printf("Assertion-04:success; ");
233    }
234    else
235    {
236      printf("Assertion-04:failure; ");
237    }
238
239
240    /*
241     * TEST RTEMS_SEMAPHORE_DELETE
242     */
243
244
245    /* Delete RTEMS semaphore */
246    /* To delete it, it is necessary to release it first */
247    returnStatus = rtems_semaphore_release(semId);
248
249    printf("rtems_semaphore_release: %d;",returnStatus);
250
251    returnStatus = rtems_semaphore_delete (semId);
252
253    printf ("rtems_semaphore_delete(): %d;", returnStatus);
254
255    /* Assertion check */
256    /* We should be able to obtain a semaphore that was released */
257    returnStatus = rtems_semaphore_ident(semName,
258                                         node,
259                                         ptsemId);
260
261    if (returnStatus == RTEMS_INVALID_NAME)
262    {
263       printf ("Assertion-05: success;");
264    }
265    else
266    {
267       printf ("Assertion-05: failure;");
268    }
269
270    printf ("\n");
271
272    exit(0);
```

```
273
274 }
275
276 /* configuration information */
277
278 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
279
280 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
281
282 #define CONFIGURE_MAXIMUM_TASKS 3
283
284 #define CONFIGURE_MAXIMUM_SEMAPHORES 10
285
286 #define CONFIGURE_INIT
287
288 #include <confdefs.h>
289
290 /* end of file */
```

B.1.13. RTEMS-CMP-CL-TMR.C

B.1.13.1. Description

This workload defines a Timer Service Routine that increments a counter variable each time it is called.

1. Create RTEMS timer
2. **Assertion 1:** Check with rtems_timer_ident that the timer ID was successfuly assigned
3. Program the timer with rtems_fire_after to fire after TICKS_TO_ACTIVATION
4. **Assertion 2:** Global variable was not incremented
5. Wake task after TICKS_TO_ACTIVATION
6. **Assertion 3:** Global variable was incremented
7. Program the timer with rtems_fire_when to fire in the instant (Current Time + TICKS_TO_ACTIVATION)
8. **Assertion 4:** Global variable was not incremented
9. Wake task after RTEMS timer reaches predefined time
10. **Assertion 5:** Global variable has been incremented
11. Program the timer with rtems_fire_after to fire after TICKS_TO_ACTIVATION
12. Cancel timer
13. Delay task to see if timer is activated
14. **Assertion 6:** Global variable was not incremented
15. Program the timer with rtems_fire_after to fire after TICKS_TO_ACTIVATION
16. Wake task after (TICKS_TO_ACTIVATION – TICKS_TO_RESET)
17. Reset timer
18. Wake task after TICKS_TO_RESET
19. **Assertion 7:** Global variable was not incrementedWake task after (TICKS_TO_ACTIVATION – TICKS_TO_RESET)
20. **Assertion 8:** Global variable was incremented
21. Delete the timer
22. Get timer identifier with rtems_timer_ident
23. **Assertion 9:** Identifier unavailable

B.1.13.2. Source Code

```
1   /*************************************************************************
2    SRC-MODULE : Timer Manager Workload.
3    MODULE-VERS : N/A
4
```

```
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-tmr/rtems-cmp-cl-tmr.c,v $
6    $Id: rtems-cmp-cl-tmr.c,v 1.5 2003/09/05 17:26:48 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Timer
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/05 17:26:48 $
24   CHANGED BY : $Author: rmaia $
25
26   $Revision: 1.5 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 28-07-2003
30   MODERATOR : rmaia
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-tmr.c,v $
36   Revision 1.5  2003/09/05 17:26:48  rmaia
37   Correct a bug on printf
38
39   Revision 1.4  2003/09/05 13:55:26  rmaia
40   Replaced variables addresses (&variable) by pointers (ptvariable).
41
42   Revision 1.3  2003/08/25 15:05:04  rmaia
43   Compiled and tested with success.
44
45   Revision 1.2  2003/08/05 16:23:00  rbarbosa
46   Updated after review
47
48   Revision 1.1  2003/08/05 11:50:33  rbarbosa
49   Updated after reviewed. Added header and RTEMS specific code lines.
50   Ready for compilation
51
52   Revision 1.2  2003/07/28 13:20:55  rmaia
53   Revision
54
55
56   Revision 1.1  2003/07/24 17:46:56  lvelkov
57   Created and added to cvs. First draft version to be review. Not yet compiled.
58
59   ******************************************************************************/
60
61
```

```
62  #include <stdio.h>
63  #include <bsp.h>
64
65  /*
66   * Timer Service Routine. Increments a counter variable.
67   */
68  rtems_timer_service_routine tmrSrvs (rtems_id timer_id, void *user_data)
69  {
70     (*((int *)user_data))++;
71     return;
72  }
73
74  #define TICKS_TO_ACTIVATION 200
75  #define TICKS_TO_RESET 20
76  #define TICKS_DELAY TICKS_TO_ACTIVATION + 20
77
78  /*
79   * Init task is the only task of this workload. This means that
80   * it will run as soon as it gets to the ready queue.
81   */
82
83  rtems_task Init(rtems_task_argument ignored)
84  {
85     int cnt = 0;
86     void * ptcnt = &cnt;
87
88     rtems_name tmrName = rtems_build_name( 'T', 'M', '1', ' ' );
89
90     rtems_id tmrId = 0;
91     rtems_id *pttmrId = &tmrId;
92
93     rtems_id assertionTmrId = 0;
94     rtems_timer_service_routine * pttmrSrvs = &tmrSrvs;
95     /*initialize return code with some error code*/
96     rtems_status_code returnStatus = RTEMS_INVALID_ID;
97
98     rtems_interval ticksPerSecond;
99
100    rtems_time_of_day timeStruct;
101
102    timeStruct.year = 2001;
103    timeStruct.month = 1;
104    timeStruct.day = 1;
105    timeStruct.hour = 0;
106    timeStruct.minute = 0;
107    timeStruct.second = 0;
108    timeStruct.ticks = 0;
109
110    /*
111     * TEST TIMER_CREATE
112     */
113
114    /*Create RTEMS timer*/
115    returnStatus = rtems_timer_create (tmrName, pttmrId);
116    printf ("rtems_timer_create(): %d;", returnStatus);
117
118    /*Assertion check*/
119    returnStatus = rtems_timer_ident(tmrName, &assertionTmrId);
```

```
120    printf ("rtems_timer_ident(): %d;", returnStatus);
121
122    if (tmrId == assertionTmrId) {
123       printf ("Assertion-01: success;");
124    }
125    else {
126       printf ("Assertion-01: failure;");
127    }
128
129    /*
130     * TEST TIMER_FIRE_AFTER
131     */
132
133    /*Fire RTEMS timer after TICKS_TO_ACTIVATION*/
134    returnStatus = rtems_timer_fire_after (tmrId,
135                                           TICKS_TO_ACTIVATION,
136                                           pttmrSrvs,
137                                           ptcnt);
138
139    printf ("rtems_timer_fire_after(): %d;", returnStatus);
140
141    /*Assertion check*/
142    if (cnt == 0) {
143       printf ("Assertion-02: success;");
144    }
145    else {
146       printf ("Assertion-02: failure;");
147    }
148
149    /* Wait for the activation of the Timer */
150    returnStatus = rtems_task_wake_after(TICKS_TO_ACTIVATION);
151    printf ("rtems_task_wake_after(): %d;", returnStatus);
152
153    if (cnt == 1) {
154       printf ("Assertion-03: success;");
155    }
156    else {
157       printf ("Assertion-03: failure;");
158    }
159
160    /*
161     * TEST TIMER_FIRE_WHEN
162     */
163
164    /*Set time*/
165    returnStatus = rtems_clock_set (&timeStruct);
166    printf ("rtems_clock_set(): %d;", returnStatus);
167
168    rtems_clock_get (RTEMS_CLOCK_GET_TICKS_PER_SECOND, &ticksPerSecond);
169
170    /* NOTE: seconds do not overflow in this computation */
171    timeStruct.second += TICKS_TO_ACTIVATION / ticksPerSecond;
172    timeStruct.ticks += TICKS_TO_ACTIVATION % ticksPerSecond;
173
174    returnStatus = rtems_timer_fire_when (tmrId,
175                                          &timeStruct,
176                                          pttmrSrvs,
177                                          ptcnt);
```

```
178
179    printf ("rtems_timer_fire_when(): %d;", returnStatus);
180
181    if (cnt == 1) {
182       printf ("Assertion-04: success;");
183    }
184    else {
185       printf ("Assertion-04: failure;");
186    }
187
188    /* Wait for the activation of the Timer */
189    rtems_task_wake_after(TICKS_DELAY);
190    printf ("rtems_task_wake_after(): %d;", returnStatus);
191
192    if (cnt == 2) {
193       printf ("Assertion-05: success;");
194    }
195    else {
196       printf ("Assertion-05: failure;");
197    }
198
199    /*
200     * TEST TIMER_CANCEL
201     */
202
203    /*Fire RTEMS timer after TICKS_TO_ACTIVATION*/
204    returnStatus = rtems_timer_fire_after (tmrId,
205                                           TICKS_TO_ACTIVATION,
206                                           pttmrSrvs,
207                                           ptcnt);
208
209    printf ("rtems_timer_fire_after(): %d;", returnStatus);
210
211    returnStatus = rtems_timer_cancel(tmrId);
212    printf ("rtems_timer_cancel(): %d;", returnStatus);
213
214    /*Assertion check*/
215
216    /*Delay to check if the timer is activated*/
217    returnStatus = rtems_task_wake_after(TICKS_TO_ACTIVATION);
218    printf ("rtems_task_wake_after(): %d;", returnStatus);
219
220    if (cnt == 2) {
221       printf ("Assertion-06: success;");
222    }
223    else {
224       printf ("Assertion-06: failure;");
225    }
226
227    /*
228     * TEST TIMER_RESET
229     */
230
231    /*Fire RTEMS timer after TICKS_TO_ACTIVATION*/
232    returnStatus = rtems_timer_fire_after (tmrId,
233                                           TICKS_TO_ACTIVATION,
234                                           pttmrSrvs,
235                                           ptcnt);
```

```
236
237    printf ("rtems_timer_fire_after(): %d;", returnStatus);
238
239
240    /* Wait for while and then reset the timer */
241    returnStatus = rtems_task_wake_after(TICKS_TO_ACTIVATION - TICKS_TO_RESET);
242    printf ("rtems_task_wake_after(): %d;", returnStatus);
243
244    returnStatus = rtems_timer_reset (tmrId);
245    printf ("rtems_timer_reset(): %d;", returnStatus);
246
247    /* Wait the remaining time */
248    returnStatus = rtems_task_wake_after(TICKS_TO_RESET);
249    printf ("rtems_task_wake_after(): %d;", returnStatus);
250
251    if (cnt == 2) {
252       printf ("Assertion-07: success;");
253    }
254    else {
255       printf ("Assertion-07: failure;");
256    }
257
258    /* Wait for the activation of the Timer */
259    returnStatus = rtems_task_wake_after(TICKS_TO_ACTIVATION - TICKS_TO_RESET);
260    printf ("rtems_task_wake_after(): %d;", returnStatus);
261
262    if (cnt == 3) {
263       printf ("Assertion-08: success;");
264    }
265    else {
266       printf ("Assertion-08: failure;");
267    }
268
269    returnStatus = rtems_timer_delete(tmrId);
270    printf ("rtems_timer_delay(): %d;", returnStatus);
271
272    /*Assertion check*/
273    returnStatus = rtems_timer_ident(tmrName, &assertionTmrId);
274
275    if (returnStatus == RTEMS_INVALID_NAME) {
276       printf ("Assertion-09: success;");
277    }
278    else {
279       printf ("Assertion-09: failure;");
280    }
281
282    printf ("\n");
283
284    exit(0);
285 }
286
287 /* configuration information */
288
289 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
290 #define CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER
291 #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
292 #define CONFIGURE_MAXIMUM_TIMERS 1
293
```

```
294
295 #define CONFIGURE_MAXIMUM_TASKS 1
296 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
297
298 #define CONFIGURE_INIT
299
300 #include <confdefs.h>
301
302 /* end of file */
```

B.1.14. RTEMS-CMP-CL-UEX.C

B.1.14.1. Description

This workload contains two tasks. The main task, Init, creates another task, testTask, who's only function is to lower its own priority, in order to be preempted by the Init task.

The main task (Init) performs the following actions:

1. Create RTEMS user extension table
2. **Assertion 1:** RTEMS user extension table ID successfuly assigned
3. Create test task
4. **Assertion 2:** Global variable set successfully by the user extension
5. Start test task
6. **Assertion 3:** Global variable set successfully by the user extension
7. Set Init task priority lower than the test task
8. In the test task: Set test task priority lower than the Init task
9. **Assertion 4:** Global variable set successfully by the user extension
10. Delete RTEMS user extension table
11. Get user extension identifier
12. **Assertion 5:** RTEMS user extension identifier is unavailable

B.1.14.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : User Extensions Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/classic-workloads/rtems-cmp-cl-uex/rtems-cmp-cl-uex.c,v $
6    $Id: rtems-cmp-cl-uex.c,v 1.3 2003/09/11 14:33:28 rmaia Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : lvelkov
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the User
20             Extensions Manager.
21
22   CREATED ON : 17-07-2003
```

```
23   CHANGED ON : $Date: 2003/09/11 14:33:28 $
24   CHANGED BY : $Author: rmaia $
25
26   $Revision: 1.3 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 25-07-2003
30   MODERATOR : rbarbosa
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-cl-uex.c,v $
36   Revision 1.3  2003/09/11 14:33:28  rmaia
37   *** empty log message ***
38
39   Revision 1.2  2003/09/02 15:32:03  rmaia
40   Commited after changes on workload
41
42   Revision 1.1  2003/07/29 14:29:16  rbarbosa
43   Ready to compile in RTEMS cross-compiling system.
44   Added configuration and replace main by init on code.
45
46   Makefile for user extensions manager robustness test workload
47
48   Revision 1.4  2003/07/25 17:14:16  rbarbosa
49   Updated the returnStatus variable initialisation scheme
50
51   Revision 1.3  2003/07/25 17:11:43  rbarbosa
52   Update after code changes
53
54   Revision 1.2  2003/07/25 15:53:17  rbarbosa
55   Revision and addition of header
56
57   ******************************************************************************/
58
59  #include <bsp.h>
60  #include <rtems.h>
61  #include <stdio.h>
62
63  enum location
64  {
65     CREATION = 1,
66     START    = 2,
67     BEGIN    = 3,
68     DELETE   = 4
69  };
70
71  int testVariable = 0;
72  rtems_task_priority currentTaskOldPriority = 15;
73  rtems_mode testTaskOldMode                 = RTEMS_DEFAULT_MODES;
74  rtems_status_code returnStatus             = RTEMS_NOT_DEFINED;
75
76
77  /*
78   * This function is the extension to the default action of
79   * the task creation action
80   */
```

```
81
82  rtems_extension creation_extension(rtems_tcb *current_task,
83                                     rtems_tcb *new_task)
84  {
85          testVariable = CREATION;
86  }
87
88  /*
89   * This function is the extension to the default action of
90   * the task start action
91   */
92
93  rtems_extension start_extension(rtems_tcb *current_task,
94                                  rtems_tcb *started_task)
95  {
96          testVariable = START;
97  }
98
99  /*
100  * This function is the extension to the default action of
101  * the task begin action
102  */
103
104 rtems_extension begin_extension(rtems_tcb *current_task)
105 {
106         testVariable = BEGIN;
107 }
108
109 /*
110  * This function is the extension to the default action of
111  * the task delete action
112  */
113
114 rtems_extension delete_extension(rtems_tcb *current_task,
115                                  rtems_tcb *deleted_task)
116 {
117         testVariable = DELETE;
118 }
119
120
121 /*
122  * Test Task
123  */
124
125
126 rtems_task testTask(rtems_task_argument testTaskArgument)
127 {
128    /* The priority of the current task must be lower than the Init task
129     * in order to be preempted. */
130    returnStatus = rtems_task_set_priority (RTEMS_SELF,
131                                            20,
132                                            &currentTaskOldPriority);
133 }
134
135
136 /*
137  * Init task
138  */
```

```
139
140
141 rtems_task Init(rtems_task_argument ignored)
142 {
143    rtems_name tblName;
144    rtems_id tblId = 0;
145    rtems_id *pttblId = &tblId;
146
147    /* Test task related data structures */
148    rtems_name testTaskName                    = rtems_build_name('T','A','1',' ');
149    rtems_task_priority testTaskInitPriority   = 15;
150    rtems_unsigned32 testTaskStackSize         = RTEMS_MINIMUM_STACK_SIZE;
151    rtems_mode testTaskInitMode                = RTEMS_DEFAULT_MODES;
152    rtems_attribute testTaskAtt                = RTEMS_DEFAULT_ATTRIBUTES;
153    rtems_id testTaskId                        = 0;
154    void * testTaskArg                         = NULL;
155
156  /* This defines a table that pointes to the routines that extend RTEMS critical
157     sections such as tqask creationm task deletion, etc. */
158
159     rtems_extensions_table User_Extensions = {
160         creation_extension, /* task creation extension */
161         start_extension,    /* task start extension */
162         NULL,               /* task restart extension */
163         delete_extension,   /* task delete extension */
164         NULL,               /* task switch extension */
165         begin_extension,    /* task begin extension */
166         NULL,               /* task exitted extension */
167         NULL                /* fatal error extension */
168         };
169
170    rtems_extensions_table *ptUser_Extensions = &User_Extensions;
171
172    tblName = rtems_build_name('T','B','1',' ');
173
174
175    /*
176     * TEST RTEMS_EXTENSION_CREATE
177     */
178
179
180    /*Create the user extension table*/
181    returnStatus = rtems_extension_create (tblName,
182                                           ptUser_Extensions,
183                                           pttblId);
184
185    /* Print Results */
186    printf ("rtems_extension_create(): %d;", returnStatus);
187
188    /* Assertion check */
189    /* If tblId is assigned a value diferent than NULL, then this directive is working
190       properly */
191
192    if (tblId != 0)
193    {
194       printf ("Assertion-01: success;");
195    }
196    else
```

```
197    {
198       printf ("Assertion-01: failure;");
199    }
200
201    /***** BEGIN TESTING EXTENSIONS *****/
202
203    /*Create test task*/
204    returnStatus = rtems_task_create (testTaskName,
205                                      testTaskInitPriority,
206                                      testTaskStackSize,
207                                      testTaskInitMode,
208                                      testTaskAtt,
209                                      &testTaskId);
210
211    printf ("rtems_task_create(): %d;", returnStatus);
212
213    /* Assertion */
214    if(testVariable == CREATION)
215    {
216       printf("Assertion-02: success; ");
217    }
218    else
219    {
220       printf("Assertion-02: failure; ");
221    }
222
223
224    /*Start test task*/
225    returnStatus =  rtems_task_start (testTaskId,
226                                      &testTask,
227                                      (rtems_task_argument) testTaskArg);
228
229    printf ("rtems_task_start(): %d;", returnStatus);
230
231    /* Assertion */
232    if(testVariable == START)
233    {
234       printf("Assertion-03: success; ");
235    }
236    else
237    {
238       printf("Assertion-03: failure; ");
239    }
240
241    /* Preempt current task:
242     * The priority of the current task must be lower than the testTask
243     * in order to be preempted. */
244    returnStatus = rtems_task_set_priority (RTEMS_SELF,
245                                            16,
246                                            &currentTaskOldPriority);
247
248    printf ("rtems_task_set_priority(): %d;", returnStatus);
249
250    returnStatus = rtems_task_mode (RTEMS_PREEMPT,
251                                    RTEMS_PREEMPT_MASK,
252                                    &testTaskOldMode);
253
254    printf ("rtems_task_mode(): %d;", returnStatus);
```

```
255
256    /* Assertion */
257    if(testVariable == BEGIN)
258    {
259       printf("Assertion-04: success; ");
260    }
261    else
262    {
263       printf("Assertion-04: failure; ");
264    }
265
266    /* Delete test task */
267    returnStatus = rtems_task_delete (testTaskId);
268
269    printf ("rtems_task_delete(): %d;", returnStatus);
270
271    /* Assertion */
272    if(testVariable == DELETE)
273    {
274       printf("Assertion-05: success; ");
275    }
276    else
277    {
278       printf("Assertion-05: failure; ");
279    }
280
281    /***** END TESTING EXTENSIONS *****/
282
283    /*
284     * TEST RTEMS_EXTENSION_DELETE
285     */
286
287
288    /*Delete RTEMS user extension table*/
289    returnStatus = rtems_extension_delete (tblId);
290
291    /* Print Results */
292    printf ("rtems_extension_delete(): %d;", returnStatus);
293
294    /* Assertion check */
295    /* If the user extension was deleted, then no information regarding this extension
296       should be available */
297    returnStatus = rtems_extension_ident(tblName,
298                                         &tblId);
299
300    if (returnStatus == RTEMS_INVALID_NAME)
301    {
302       printf ("Assertion-02: success;");
303    }
304    else
305    {
306       printf ("Assertion-02: failure;");
307    }
308
309    printf ("\n");
310
311    exit(0);
312 }
```

```
313
314 /* configuration information */
315
316 #define CONFIGURE_TEST_NEEDS_CONSOLE_DRIVER
317
318 #define CONFIGURE_MAXIMUM_TASKS 4
319
320 #define CONFIGURE_MAXIMUM_USER_EXTENSIONS 10
321
322 #define CONFIGURE_RTEMS_INIT_TASKS_TABLE
323
324 #define CONFIGURE_INIT
325
326 #include <confdefs.h>
327
328 /* end of file */
```

B.2. RTEMS POSIX API Workloads

The following sections present the workloads that were used to build the test cases of the RTEMS POSIX API.

B.2.1. RTEMS-CMP-PX-CLK.C

B.2.1.1. Description

This workload has only one task, POSIX_Init, and uses two macros to perform the testing, namely, TIMER_ADD and TIMER_LESS_THEN.

1. Get clock time
2. Set new clock time
3. Get new clock time
4. **Assertion 1:** The new clock time is bigger than the first clock time mesaurement
5. Get clock time
6. Sleep a predefined number of seconds
7. Get new clock time
8. **Assertion 2:** The new clock time is bigger than the first clock time measurement
9. Get clock time
10. Sleep a predefined number of nanoseconds
11. Get new clock time
12. **Assertion 3:** The new clock time is bigger than the first clock time measurement

B.2.1.2. Source Code

```
1  /*****************************************************************************
2   SRC-MODULE : Clock Manager Workload
3   MODULE-VERS : N/A
4
5   $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
   testing/implementation/posix-workloads/rtems-cmp-px-clk/rtems-cmp-px-clk.c,v $
6   $Id: rtems-cmp-px-clk.c,v 1.6 2003/09/11 15:11:12 lhenriques Exp $
7   $State: Exp $
```

```
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : rbarbosa
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Clock
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/11 15:11:12 $
24   CHANGED BY : $Author: lhenriques $
25
26   $Revision: 1.6 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 25-07-2003
30   MODERATOR : lhenriques
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-px-clk.c,v $
36   Revision 1.6  2003/09/11 15:11:12  lhenriques
37   Changed sleep time and other small details.
38
39   Revision 1.5  2003/09/01 16:51:33  rmaia
40   Commited after adding commentaries to workload
41
42   Revision 1.3  2003/09/01 15:16:52  rmaia
43   Updated the clock manager workload after testing it with the GDB built-in simulator.
44
45   Revision 1.2  2003/08/18 15:20:09  lvelkov
46   Made a minor correction in header.
47
48   Revision 1.1  2003/07/31 11:32:37  rbarbosa
49   Added the corresponding makefiles to each workload
50
51   Revision 1.3  2003/07/30 15:17:34  rbarbosa
52   Replaced main with init. Added configuration settings needed to compile
53   in RTEMS CCS. Added header to file.
54
55   Revision 1.2  2003/07/28 16:29:08  lhenriques
56   Several changes have been done and one of the workloads (for the scheduler manager) has
     been removed.
57
58   ***************************************************************************/
59
60 #include <bsp.h>
61 #include <stdio.h>
62 #include <unistd.h>
63 #include <time.h>
64
```

```
65  # define TIMER_ADD(a, b, result)                                          \
66    do {                                                                   \
67      (result)->tv_sec = (a)->tv_sec + (b)->tv_sec;                        \
68      (result)->tv_nsec = (a)->tv_nsec + (b)->tv_nsec;                     \
69      if ((result)->tv_nsec >= 1000000000)                                 \
70        {                                                                  \
71          ++(result)->tv_sec;                                            \
72          (result)->tv_nsec -= 1000000000;                               \
73        }                                                                  \
74    } while (0)
75
76  # define TIMER_LESS_THAN(a, b)                                           \
77    (((a)->tv_sec == (b)->tv_sec) ?                                        \
78     ((a)->tv_nsec < (b)->tv_nsec) :                                       \
79     ((a)->tv_sec < (b)->tv_sec))
80
81  /*
82   * POSIX Init task is the first task to execute
83   * on the system
84   */
85
86  void * POSIX_Init(void * ignored)
87  {
88      clockid_t clock_id = CLOCK_REALTIME;
89      struct timespec tp_delay, tp_result;
90      unsigned int sleep_result = 0;
91      unsigned int sleep_time = 1; /* 1 s */
92      struct timespec tp_set, tp_before_set, tp_after_set;
93      struct timespec tp_before_sleep, tp_after_sleep;
94      struct timespec rqtp, rmtp;
95
96      int result_set = 0;
97      int result_get = 0;
98
99      tp_delay.tv_sec = 0;
100     tp_delay.tv_nsec = 999999999;
101
102     /*
103      * TEST clock_gettime/clock_settime
104      */
105     result_get = clock_gettime (clock_id,
106                                 &tp_before_set);
107
108     printf ("clock_gettime(): %d;", result_get);
109
110     /* Set time structure */
111     tp_set.tv_sec = tp_before_set.tv_sec + 10;
112     tp_set.tv_nsec = 0;
113
114     result_set = clock_settime (clock_id,
115                                 &tp_set);
116
117     printf ("clock_settime(): %d; ", result_set);
118
119     /* Get a time from the system to use in this test */
120     result_get = clock_gettime (clock_id,
121                                 &tp_after_set);
122
```

```
123     printf ("clock_gettime(): %d;", result_get);
124
125     /* Assertion Check */
126     TIMER_ADD (&tp_before_set,
127                &tp_delay,
128                &tp_result);
129
130     if (TIMER_LESS_THAN (&tp_result,
131                          &tp_after_set))
132     {
133          printf ("Assertion-01: success; ");
134     }
135     else
136     {
137          printf ("Assertion-01: failure; ");
138     }
139
140     /*
141      * TEST sleep
142      */
143
144     clock_gettime (clock_id,
145                    &tp_before_sleep);
146
147     sleep_result = sleep (sleep_time);
148
149     clock_gettime (clock_id,
150                    &tp_after_sleep);
151
152     printf ("sleep(): %d; ", sleep_result);
153
154     /* Assertion Check */
155     TIMER_ADD (&tp_before_sleep,
156                &tp_delay,
157                &tp_result);
158
159     if (TIMER_LESS_THAN (&tp_result,
160                          &tp_after_sleep))
161     {
162          printf ("Asseriton-02: success; ");
163     }
164     else
165     {
166          printf ("Assertion-02: failure; ");
167     }
168
169    /*
170     * TEST nanosleep
171     */
172
173     /* Set data structures */
174     rqtp.tv_sec = sleep_time; /* 1 s */
175     rqtp.tv_nsec = 200; /* 200 ns */
176
177     /* Get clock time */
178     clock_gettime (clock_id,
179                    &tp_before_sleep);
180
```

```
181     sleep_result = nanosleep (&rqtp,
182                               &rmtp);
183
184     clock_gettime (clock_id,
185                    &tp_after_sleep);
186
187     printf ("nanosleep(): %d; ", sleep_result);
188
189     /* Assertion Check */
190     TIMER_ADD (&tp_before_sleep,
191                &tp_delay,
192                &tp_result);
193
194     if (TIMER_LESS_THAN (&tp_result,
195                          &tp_after_sleep))
196     {
197          printf ("Assertion-03: success; ");
198     }
199     else
200     {
201          printf ("Assertion-03: failure; ");
202     }
203
204     printf ("\n");
205
206     exit (0);
207 }
208
209 /* configuration information */
210
211 #define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
212 #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
213 #define CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER
214 #define CONFIGURE_MAXIMUM_TIMERS 1
215 #define CONFIGURE_POSIX_INIT_THREAD_TABLE
216 #define CONFIGURE_MAXIMUM_POSIX_THREADS 1
217 #define CONFIGURE_POSIX_INIT_TASKS_TABLE
218 #define CONFIGURE_POSIX_INIT_THREAD_TABLE
219
220 #define CONFIGURE_INIT
221
222 #include <confdefs.h>
223
224 /* end of file */
```

B.2.2. RTEMS-CMP-PX-MTX.C

B.2.2.1. Description

This workload has two functions, one being the main function, POSIX_Init, and a secondary function, lockFunc. The main function performs all tests of the mutex manager. It uses the secondary function as an aid to the pthread_mutex_timedlock() test.

1. Initialise a mutex attributes object;
2. **Assertion 1:** Directive returned success, attributes initialised;
3. Destroy mutex attributes object;
4. Try to destroy the same attributes again;

5. **Assertion 2:** The attributes were unavailable;
6. Set protocol attribute of the mutex attribute object;
7. Get protocol attribute of the mutex attribute object;
8. **Assertion 3:** protocol set;
9. Set priority ceiling attribute of a mutex attribute object;
10. Get priority ceiling attribute of a mutex attribute object;
11. **Assertion 4:** The protocol retrieved is equal to the one set;;
12. Set shared attribute of the mutex attribute object;
13. Get shared attribute of the mutex attribute object;
14. **Assertion 5:** The process share value is equal to the retrieved one;
15. Initialise a mutex object;
16. Destroy mutex;
17. **Assertion 6:** Mutex destroyed;
18. Lock a mutex object;
19. Try to lock the same object;
20. **Assertion 7:** The mutex cannot be locked again;
21. Unlock mutex;
22. Try to lock the mutex object;
23. Try to lock the same mutex;
24. **Assertion 8:** The mutex cannot be locked again;
25. Create thread that will lock a mutex;
26. Try to lock the mutex object without blocking the thread;
27. **Assertion 9:** Mutex object already locked, thread didn't block;
28. Unlock a mutex object;
29. Lock mutex object;
30. **Assertion 10:** Mutex object successfully locked;
31. Set ceiling priority of the mutex object;
32. Get ceiling priority of the mutex object;
33. **Assertion 11:** The priority retrieved is equal tohte one set;

B.2.2.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Mutex Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/posix-workloads/rtems-cmp-px-mtx/rtems-cmp-px-mtx.c,v $
6    $Id: rtems-cmp-px-mtx.c,v 1.8 2003/09/11 10:59:29 lhenriques Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : rbarbosa
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Mutex
20             Manager.
```

```
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : 25-07-2003
24   CHANGED BY : lvelkov
25
26   $Revision: 1.8 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 24-07-2003
30   MODERATOR : rmaia
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-px-mtx.c,v $
36   Revision 1.8  2003/09/11 10:59:29  lhenriques
37   Changed configuration sets to the workload.
38
39   Revision 1.7  2003/09/08 12:40:18  rmaia
40   Commited after reviewing workload
41
42   Revision 1.6  2003/09/08 12:16:59  rmaia
43   Commited after changing code details
44
45   Revision 1.5  2003/09/08 11:11:57  rmaia
46   Commited after changing some commentaries
47
48   Revision 1.4  2003/09/08 10:58:07  rmaia
49   Commited after correcting assertion
50
51   Revision 1.3  2003/09/05 15:29:19  rmaia
52   Changes done by rbarbosa.
53
54   Revision 1.2  2003/08/18 16:00:27  lvelkov
55   Removed superfluous assertion.
56
57   Revision 1.1  2003/07/31 11:33:47  rbarbosa
58   Added the corresponding makefiles to each workload
59
60   Revision 1.3  2003/07/30 15:17:34  rbarbosa
61   Replaced main with init. Added configuration settings needed to compile
62   in RTEMS CCS. Added header to file.
63
64   Revision 1.2  2003/07/28 09:44:31  lvelkov
65   Made chages after review.
66
67   ****************************************************************************/
68
69
70  #include <bsp.h>
71  #include <pthread.h>
72  #include <stdio.h>
73  #include <time.h>
74  #include <errno.h>
75  #include <unistd.h>
76
77  pthread_mutex_t mutex;
78
```

```
79
80  /*
81   * This function is used to perform a lock and unlock
82   * on a mutex
83   */
84
85
86  void *lockFunc (void * dummy) {
87
88     pthread_mutex_lock (&mutex);
89     sleep (2);
90     pthread_mutex_unlock (&mutex);
91     return NULL;
92
93  }
94
95
96  /*
97   * This is the main function of this workload
98   */
99
100
101 void * POSIX_Init(void * ignored)
102 {
103         int result = 0;
104         pthread_mutexattr_t attr;
105         int protocol1;
106         int protocol2;
107         int pshared1;
108         int pshared2;
109         struct timespec timeout;
110         int prioceiling;
111         int oprioceiling;
112         pthread_t thread;
113         void * dummy = NULL;
114
115
116 /*
117  * POSIX Function pthread_mutexattr_init()
118  */
119
120
121         result = pthread_mutexattr_init(&attr);
122
123         /* Print Results */
124         printf("pthread_mutexattr_init(): %d; ", result);
125
126         /* Assertion 1 */
127         if(result == 0)
128         {
129                 printf("Assertion-01: success; ");
130         }
131         else
132         {
133                 printf("Assertion-01: failure; ");
134         }
135
136
```

```
137 /*
138  * POSIX Function pthread_mutexattr_destroy()
139  */
140
141
142         /* Create a mutexattr in order to destroy it */
143         result = pthread_mutexattr_destroy(&attr);
144
145         /* Print Results */
146         printf("pthread_mutexattr_destroy(): %d;", result);
147
148         /* Assertion 1 */
149         result = pthread_mutexattr_destroy(&attr);
150         if(result == EINVAL)
151         {
152                 printf("Assertion-02: success; ");
153         }
154         else
155         {
156                 printf("Assertion-02: failure; ");
157         }
158
159 /*
160  * POSIX Function pthread_mutexattr_setprotocol()
161  */
162
163
164         /* Create a mutexattr in order to destroy it */
165         pthread_mutexattr_init(&attr);
166
167         /* Protocol to set */
168         protocol1 = PTHREAD_PRIO_NONE;
169
170         result = pthread_mutexattr_setprotocol(&attr,
171                                                 protocol1);
172
173         /* Print Results */
174         printf("pthread_mutexattr_setprotocol(): %d; ", result);
175
176         /* Assertion 1 */
177         pthread_mutexattr_getprotocol(&attr,
178                                       &protocol2);
179         if(protocol1 == protocol2)
180         {
181                 printf("Assertion-03: success; ");
182         }
183         else
184         {
185                 printf("Assertion-03: failure; ");
186         }
187         pthread_mutexattr_destroy(&attr);
188
189
190 /*
191  * POSIX Function pthread_mutexattr_setprioceiling()
192  */
193
194
```

```
195         /* Create a mutexattr in order to destroy it */
196         pthread_mutexattr_init(&attr);
197
198         /* Protocol to set */
199         protocol1 = PTHREAD_PRIO_PROTECT;
200
201         result = pthread_mutexattr_setprioceiling(&attr,
202                                                   protocol1);
203         /* Print Results */
204         printf("pthread_mutexattr_setprioceiling(): %d; ", result);
205         /* Assertion 1 */
206         pthread_mutexattr_getprioceiling(&attr,
207                                          &protocol2);
208         if(protocol1 == protocol2)
209         {
210                 printf("Assertion-04: success; ");
211         }
212         else
213         {
214                 printf("Assertion-04: failure; ");
215         }
216         pthread_mutexattr_destroy(&attr);
217
218
219 /*
220  * POSIX Function pthread_mutexattr_setpshared()
221  */
222
223
224         /* Create a mutexattr in order to destroy it */
225         pthread_mutexattr_init(&attr);
226
227         /* Protocol to set */
228         pshared1 = PTHREAD_PROCESS_PRIVATE;
229
230         result = pthread_mutexattr_setpshared(&attr,
231                                               pshared1);
232
233         /* Print Results */
234         printf("pthread_mutexattr_setpshared(): %d; ", result);
235
236         /* Assertion 1 */
237         pthread_mutexattr_getpshared(&attr,
238                                      &pshared2);
239         if(pshared1 == pshared2)
240         {
241                 printf("Assertion-05: success; ");
242         }
243         else
244         {
245                 printf("Assertion-05: failure; ");
246         }
247         pthread_mutexattr_destroy(&attr);
248
249
250 /*
251  * POSIX Function pthread_mutex_destroy()
252  */
```

```
253
254
255         /* Create a mutexattr in order to build a mutex */
256         pthread_mutexattr_init(&attr);
257
258         /* Initialise mutex */
259         result = pthread_mutex_init(&mutex,
260                                     &attr);
261
262         printf("pthread_mutex_init(): %d; ", result);
263
264         result = pthread_mutex_destroy(&mutex);
265
266         /* Print Results */
267         printf("pthread_mutex_destroy(): %d;", result);
268
269         /* Assertion 1 */
270         /* If we try to lock an non existente mutex, it should fail */
271         result = pthread_mutex_lock(&mutex);
272         if(result == EINVAL)
273         {
274                 printf("Assertion-06: success; ");
275         }
276         else
277         {
278                 printf("Assertion-06: failure; ");
279         }
280         pthread_mutexattr_destroy(&attr);
281
282
283 /*
284  * POSIX Function pthread_mutex_lock()
285  */
286
287
288         /* Create a mutexattr in order to build a mutex */
289         pthread_mutexattr_init(&attr);
290
291         /* Initialise mutex */
292         pthread_mutex_init(&mutex,
293                            &attr);
294
295         result = pthread_mutex_lock(&mutex);
296
297         /* Print Results */
298         printf("pthread_mutex_lock(): %d;", result);
299
300         /* Assertion 1 */
301         /* If we try to get the mutex again, a EDEADLK is raised
302            if the function is working properly */
303         if(pthread_mutex_lock(&mutex) == EDEADLK)
304         {
305                 printf("Assertion-07: success; ");
306         }
307         else
308         {
309                 printf("Assertion-07: failure; ");
310         }
```

```
311         pthread_mutex_unlock(&mutex);
312
313
314 /*
315  * POSIX Function pthread_mutex_trylock()
316  */
317
318
319         result = pthread_mutex_trylock(&mutex);
320
321         /* Print Results */
322         printf("pthread_mutex_trylock(): %d; ", result);
323
324         /* Assertion 1 */
325         /* if we lock the mutex and try to lock it, it should return a EDEADLK */
326         if(pthread_mutex_lock(&mutex) == EDEADLK)
327         {
328                 printf("Assertion-08: success; ");
329         }
330         else
331         {
332                 printf("Assertion-08: failure; ");
333         }
334         pthread_mutex_unlock(&mutex);
335
336
337 /*
338  * POSIX Function pthread_mutex_timedlock()
339  */
340
341
342         pthread_create (&thread,
343                         NULL,
344                         lockFunc,
345                         dummy);
346
347         pthread_join (thread,
348                       NULL);
349
350         timeout.tv_sec = 5;
351         result = pthread_mutex_timedlock(&mutex,
352                                          &timeout);
353
354         /* Print Results */
355         printf("pthread_mutex_timedlock(): %d; ", result);
356         /* By now, the mutex is locked by the timedlock function */
357
358         /* Assertion 1 */
359         /* If we try to lock a locked mutex, a EDEADLK is returned */
360         if(pthread_mutex_lock(&mutex) == EDEADLK)
361         {
362                 printf("Assertion-9: success; ");
363         }
364         else
365         {
366                 printf("Assertion-9: failure; ");
367         }
368         pthread_mutex_unlock(&mutex);
```

```
369
370
371 /*
372  * POSIX Function pthread_mutex_unlock()
373  */
374
375
376         pthread_mutex_lock(&mutex);
377         result = pthread_mutex_unlock(&mutex);
378
379         /* Print Results */
380         printf("pthread_mutex_unlock(): %d; ", result);
381
382         /* Assertion 1 */
383         /* it must be possible to lock it again without returning a EDEADLK */
384         if(pthread_mutex_lock(&mutex) != EDEADLK)
385         {
386                 printf("Assertion-10: success; ");
387         }
388         else
389         {
390                 printf("Assertion-10: failure; ");
391         }
392         pthread_mutex_unlock(&mutex);
393
394
395 /*
396  * POSIX Function pthread_mutex_setprioceiling()
397  */
398         /* Set priority variable */
399         prioceiling = 4;
400
401         result = pthread_mutex_setprioceiling(&mutex,
402                                               prioceiling,
403                                               &oprioceiling);
404
405         /* Print Results */
406         printf ("pthread_mutex_prioceiling(): %d; ", result);
407
408         /* Assertion 1 */
409         pthread_mutex_getprioceiling(&mutex,
410                                      &oprioceiling);
411
412         if(prioceiling == oprioceiling)
413         {
414                 printf("Assertion-11: success; ");
415         }
416         else
417         {
418                 printf("Assertion-11: failure; ");
419         }
420
421         pthread_mutex_destroy(&mutex);
422
423         printf ("\n");
424
425       exit(0);
426 }
```

```
427
428 /* configuration information */
429
430 #define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
431 #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
432 #define CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER
433 #define CONFIGURE_MAXIMUM_TIMERS 1
434 #define CONFIGURE_POSIX_INIT_THREAD_TABLE
435 #define CONFIGURE_MAXIMUM_POSIX_THREADS 1
436 #define CONFIGURE_POSIX_INIT_TASKS_TABLE
437 #define CONFIGURE_POSIX_INIT_THREAD_TABLE
438 #define CONFIGURE_MAXIMUM_POSIX_MUTEXES 4
439
440 #define CONFIGURE_INIT
441
442 #include <confdefs.h>
443
444 /* end of file */
```

B.2.3. RTEMS-CMP-PX-SGN.C

B.2.3.1. Description

This workload presents three threads, namely, the main thread, Init, a second thread used to send signals, send_sgnal_thread, and a third thread used as a thread to be suspended, suspended_thread. The main function calls all the signal manager directives selected for testing.

1. Fill a set of signals
2. **Assertion 1:** signal filled with the expected values;
3. Empty the signal set;
4. **Assertion 2:** signal set is empty;
5. Add signal to a set;
6. **Assertion 3:** The added signal is equal to the one retrieved from the set;
7. Delete signal from set;
8. **Assertion 4:** The deleted signal is not on the set;
9. Assign an action to a signal;
10. Send signal to self;
11. **Assertion 5:** Signal received;
12. Change signal mask to block signal SIGUSR1;
13. Send signal to self;
14. Change signal mask to unblock signal SIGUSR1;
15. **Assertion 6:** Signal SIGUSR1 received;
16. Assign an action to a signal;
17. Create a thread that will send a signal;
18. Suspend execution until signal is received;
19. **Assertion 7:** signal arrived, directive returned;
20. Create thread;
21. Send a predefined signal to thread;
22. **Assertion 8:** Thread received signal;
23. Create thread that will send a signal;
24. Wait for the predefined signal;

25. **Assertion 9:** Expected signal has arrived;
26. Wait for predefined signal during 10ms;
27. **Assertion 10:** signal did not arrived in the predefined time;

B.2.3.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Signal Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/posix-workloads/rtems-cmp-px-sgn/rtems-cmp-px-sgn.c,v $
6    $Id: rtems-cmp-px-sgn.c,v 1.4 2003/09/11 15:13:31 lhenriques Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : rbarbosa
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Signal
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/11 15:13:31 $
24   CHANGED BY : $Author: lhenriques $
25
26   $Revision: 1.4 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 25-07-2003
30   MODERATOR : lhenriques
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-px-sgn.c,v $
36   Revision 1.4  2003/09/11 15:13:31  lhenriques
37   Removed messages from threads/signal handlers.
38
39   Revision 1.3  2003/09/04 16:46:04  lhenriques
40   Finished the review of this workload.
41
42   Revision 1.2  2003/08/18 16:18:04  lvelkov
43   Minor update in header.
44
45   Revision 1.1  2003/07/31 11:34:09  rbarbosa
46   Added the corresponding makefiles to each workload
47
48   Revision 1.3  2003/07/30 15:17:34  rbarbosa
49   Replaced main with init. Added configuration settings needed to compile
50   in RTEMS CCS. Added header to file.
```

```
51
52   Revision 1.2  2003/07/28 16:29:08  lhenriques
53   Several changes have been done and one of the workloads (for the scheduler manager) has
     been removed.
54
55
56  ****************************************************************************/
57
58  #include <bsp.h>
59  #include <time.h>
60  #include <unistd.h>
61  #include <stdio.h>
62  #include <signal.h>
63  #include <pthread.h>
64  #include <sys/types.h>
65  #include <unistd.h>
66
67  /* Macro to convert a signal number to a mask */
68  #define SIG2MASK( sig ) (1 << ((sig) - 1))
69
70  int test = 0;
71  int check;
72
73  /* Signal handlers */
74
75  /* SIGUSR1 signal handler */
76  void sigusr_signal_handler(int sig_num)
77  {
78  }
79
80
81  /* SIGSEGV signal handler */
82  void sigsegv_signal_handler(int sig_num)
83  {
84    check = 1;
85  }
86
87  void resume_signal_handler(int signum)
88  {
89    check = check + 1;
90
91    return;
92  }
93
94  /* Threads */
95  void * send_signal_thread (void * arg)
96  {
97    int result;
98    pthread_t tid = (pthread_t)arg;
99    int sigusr = SIGUSR1;
100   int sleep_time = 1;
101
102   /* Wait 2 seconds... */
103   sleep (sleep_time);
104
105   result = pthread_kill (tid, sigusr);
106
107   return (void *)0;
```

```
108 }
109
110 void * suspended_thread (void * arg)
111 {
112   struct sigaction act, act_old;
113   int sig = (int)arg;
114   int result;
115   pthread_t tid;
116
117   tid = pthread_self ();
118
119   act.sa_flags = 0;
120   act.sa_handler = resume_signal_handler;
121
122   sigemptyset (&act.sa_mask);
123   sigaddset (&act.sa_mask, sig);
124
125   result = sigaction (sig, &act, &act_old);
126
127   result = pause ();
128
129   check = check + 1;
130
131   test++;
132   return (void *)0;
133 }
134
135
136 void * POSIX_Init (void * ignored)
137 {
138   struct timespec timeout;
139   struct sigaction act, old_act;
140   siginfo_t siginfo;
141   sigset_t set, old_set;
142   pid_t pid;
143   int signo = SIGSEGV;
144   int sig = 0;
145   pthread_t thread;
146   pthread_t tid;
147   pthread_attr_t thread_attr;
148   int how;
149   int res;
150   int sleep_time = 1; /* 1 s */
151
152   /*
153    * TEST sigfillset
154    */
155   res = sigfillset (&set);
156   printf ("sigfillset(): %d; ", res);
157
158   if ((res == 0) && (set == 0xffffffff))
159     {
160       printf ("Assertion-01: success; ");
161     }
162   else
163     {
164       printf ("Assertion-01: failure; ");
165     }
```

```
166
167   /*
168    * TEST sigemptyset
169    */
170   res = sigemptyset (&set);
171   printf ("sigemptyset(): %d; ", res);
172
173   if ((res == 0) && (set == 0))
174     {
175       printf ("Assertion-02: success; ");
176     }
177   else
178     {
179       printf ("Assertion-02: failure; ");
180     }
181
182   /*
183    * TEST sigaddset
184    */
185   res = sigaddset (&set, signo);
186   printf ("sigaddset(): %d; ", res);
187
188   if ((res == 0) && (set & SIG2MASK (signo)))
189     {
190       printf ("Assertion-03: success; ");
191     }
192   else
193     {
194       printf ("Assertion-03: failure; ");
195     }
196
197   /*
198    * TEST sigdelset
199    */
200
201   /* Set all the signals */
202   res = sigfillset (&set);
203   printf ("sigfillset(): %d; ", res);
204
205   res = sigdelset (&set, signo);
206   printf ("sigdelset(): %d; ", res);
207
208   if ((res == 0) && (!(set & SIG2MASK (signo))))
209     {
210       printf ("Assertion-04: success; ");
211     }
212   else
213     {
214       printf ("Assertion-04: failure; ");
215     }
216
217   /*
218    * TEST sigaction
219    */
220
221   /* First, set structure */
222   sigemptyset (&(act.sa_mask));
223   act.sa_flags = 0;
```

```
224   act.sa_handler = sigsegv_signal_handler;
225
226   sig = SIGUSR1;
227   /* Set the action for that specific signal */
228   res = sigaction (sig, &act, &old_act);
229   printf ("sigaction(): %d; ", res);
230
231   /* Get PID */
232   pid = getpid ();
233   printf ("getpid(): %d; ", pid);
234
235   check = 0;
236
237   /* Send signal */
238   res = kill (pid, sig);
239   printf ("kill(): %d; ", res);
240
241   /* Wait for signal handler... */
242   sleep (sleep_time);
243
244   if (check == 1)
245     {
246       printf ("Assertion-05: success; ");
247     }
248   else
249     {
250       printf ("Assertion-05: failure; ");
251     }
252
253   /*
254    * TEST sigprocmask
255    */
256   how = SIG_BLOCK;
257   check = 0;
258
259   /* Initialize set of signals */
260   sigemptyset (&set);
261   sigemptyset (&old_set);
262   sigaddset (&set, sig);
263
264   /* Block signal SIGSEGV */
265   res = sigprocmask (how, &set, &old_set);
266   printf ("sigprocmask(): %d; ", res);
267
268   res = kill (pid, sig);
269   printf ("kill(): %d; ", res);
270
271   /* Wait a while... */
272   sleep (sleep_time);
273
274   how = SIG_UNBLOCK;
275   /* Unblock signal SIGSEGV */
276   res = sigprocmask (how, &set, &old_set);
277   printf ("sigprocmask(): %d; ", res);
278
279   /* Wait a signal handler... */
280   sleep (sleep_time);
281
```

```
282   if (check == 1)
283     {
284       printf ("Assertion-06: success; ");
285     }
286   else
287     {
288       printf ("Assertion-06: failure; ");
289     }
290
291   /*
292    * TEST sigsuspend, pthread_kill and pthread_create
293    */
294   sig = SIGUSR1;
295
296   /* First, set structure */
297   act.sa_flags = 0;
298   act.sa_handler = sigusr_signal_handler;
299
300   /* Set the action for that specific signal */
301   res = sigaction (sig, &act, &old_act);
302   printf ("sigaction(): %d; ", res);
303
304   /* Get thread ID */
305   tid = pthread_self ();
306   printf("pthread_self(): %ul; ", tid);
307
308   res = pthread_attr_init (&thread_attr);
309   printf ("pthread_attr_init(): %d; ", res);
310
311   /* Create a thread. This thread is only used to send the signal
312    * SIGUSR1 to THIS thread
313    */
314   res = pthread_create(&thread, &thread_attr, send_signal_thread, (void *)tid);
315   printf ("pthread_create(): %d; ", res);
316
317   /* Block all signals except SIGUSR2 while suspended */
318   sigfillset (&set);
319   sigdelset (&set, sig);
320
321   res = sigsuspend (&set);
322   printf ("sigsuspend(): %d; ", res);
323
324   /*
325    * Error in POSIX implementation! The actual return value of this function
326    * in RTEMS is the signal received, which means that this test will always
327    * fail.
328    */
329   if (res == -1)
330     {
331       printf ("Assertion-07: success; ");
332     }
333   else
334     {
335       printf ("Assertion-07: failure; ");
336     }
337
338   /*
339    * TEST
```

```
340    */
341   sig = SIGUSR2;
342   check = 0;
343
344   res = pthread_attr_init (&thread_attr);
345   printf ("pthread_attr_init: %d; ", res);
346
347   res = pthread_create (&thread, &thread_attr, suspended_thread, (void *)sig);
348   printf ("pthread_create(): %d; ", res);
349
350   /* wait 2 seconds */
351   sleep(sleep_time * 2);
352
353   res = pthread_kill (thread, sig);
354   printf ("pthread_kill(): %d; ", res);
355
356   sleep(sleep_time * 2);
357
358   if (check == 2)
359     {
360       printf ("Assertion-08: success; ");
361     }
362   else
363     {
364       printf ("Assertion-08: failure; ");
365     }
366
367   /*
368    * TEST
369    */
370   sig = SIGUSR1;
371
372   sigemptyset (&set);
373   sigaddset (&set, sig);
374
375   tid = pthread_self ();
376
377   res = pthread_attr_init (&thread_attr);
378   printf ("pthread_attr_init(): %d; ", res);
379
380   res = pthread_create (&thread, &thread_attr, send_signal_thread, (void *)tid);
381   printf ("pthread_create(): %d; ", res);
382
383   res = sigwaitinfo (&set, &siginfo);
384   printf ("sigwaitinfo(): %d; ", res);
385
386   if (res == SIGUSR1)
387     {
388       printf ("Assertion-09: success; ");
389     }
390   else
391     {
392       printf ("Assertion-09: failure; ");
393     }
394
395   /*
396    * TEST
397    */
```

```
398   sig = SIGUSR1;
399
400   timeout.tv_sec = 0;
401   timeout.tv_nsec = 10000;
402
403   sigemptyset (&set);
404   sigaddset (&set, sig);
405
406   res = sigtimedwait (&set, &siginfo, &timeout);
407   printf ("sigtimedwait(): %d; ", res);
408
409   if (res == -1)
410     {
411       printf ("Assertion-10: success; ");
412     }
413   else
414     {
415       printf ("Assertion-10: failure; ");
416     }
417
418   printf("\n");
419
420   exit(0);
421 }
422
423 /* configuration information */
424
425 #define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
426 #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
427 #define CONFIGURE_MAXIMUM_POSIX_QUEUED_SIGNALS 5
428 #define CONFIGURE_POSIX_INIT_THREAD_TABLE
429 #define CONFIGURE_MAXIMUM_POSIX_THREADS 3
430 #define CONFIGURE_POSIX_INIT_TASKS_TABLE
431 #define CONFIGURE_POSIX_INIT_THREAD_TABLE
432 #define CONFIGURE_INIT
433
434 #include <confdefs.h>
435
436 /* end of file */
```

B.2.4. RTEMS-CMP-PX-TMR.C

B.2.4.1. Description

This workload contains one thread, Init, and a signal handling function. The main thread, Init, calls all the directives from the timer manager selected for testing.

1. Assign action to the SIGALRM signal
2. Create a timer;
3. **Assertion 1:** Timer id assigned successfully;
4. Set timer value to fire in 2 seconds;
5. Sleep during 2 seconds;
6. **Assertion 2:** The signal handler was called;
7. Delete timer;
8. **Assertion 3:** The timer is unavailable

B.2.4.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Timer Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/posix-workloads/rtems-cmp-px-tmr/rtems-cmp-px-tmr.c,v $
6    $Id: rtems-cmp-px-tmr.c,v 1.3 2003/09/05 15:23:31 lhenriques Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : rbarbosa
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Timer
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/05 15:23:31 $
24   CHANGED BY : $Author: lhenriques $
25
26   $Revision: 1.3 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 25-07-2003
30   MODERATOR : lhenriques
31
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-px-tmr.c,v $
36   Revision 1.3  2003/09/05 15:23:31  lhenriques
37   Reviewed the RTEMS POSIX timer manager.
38
39   Revision 1.2  2003/08/18 16:50:42  lvelkov
40   Updated the header.
41
42   Revision 1.1  2003/07/31 11:35:03  rbarbosa
43   Added the corresponding makefiles to each workload
44
45   Revision 1.3  2003/07/30 15:17:34  rbarbosa
46   Replaced main with init. Added configuration settings needed to compile
47   in RTEMS CCS. Added header to file.
48
49   Revision 1.2  2003/07/28 16:29:08  lhenriques
50   Several changes have been done and one of the workloads (for the scheduler manager) has
    been removed.
51
52
53   *****************************************************************************/
54
```

```
55  /* POSIX Timer Manager */
56  #include <bsp.h>
57  #include <unistd.h>
58  #include <time.h>
59  #include <signal.h>
60  #include <stdio.h>
61
62  int check = 0;
63
64  void signal_handler(int sig_num)
65  {
66    check = 1;
67  }
68
69  void * POSIX_Init(void * ignored)
70  {
71    int result = 0;
72    timer_t timerid;
73    clock_t clockid = CLOCK_REALTIME;
74    int flags = 0;
75    int sig = SIGALRM;
76    struct itimerspec setting, old_setting;
77    struct sigaction act, act_old;
78    struct sigevent timer_event_spec;
79
80    /* Add SIGALRM signal handler */
81    sigemptyset (&(act.sa_mask));
82    act.sa_handler = signal_handler;
83    act.sa_flags = 0;
84
85    result = sigaction(sig, &act, &act_old);
86    printf("sigaction: %d; ", result);
87
88    /*
89     * TEST timer_create()
90     */
91
92    timer_event_spec.sigev_notify = SIGEV_SIGNAL;
93    timer_event_spec.sigev_signo = sig;
94
95    result = timer_create(clockid, &timer_event_spec, &timerid);
96    printf("timer_create(): %d; ", result);
97    if(result == 0)
98      {
99        printf("Assertion-01: success; ");
100     }
101   else
102     {
103       printf("Assertion-01: failure; ");
104     }
105
106   /*
107    * TEST timer_settime()
108    */
109
110   /* Fire only once */
111   setting.it_interval.tv_sec = 0;
112   setting.it_interval.tv_nsec = 0;
```

```
113   setting.it_value.tv_sec = 2;
114   setting.it_value.tv_nsec = 100000;
115
116   result = timer_settime(timerid, flags, &setting, &old_setting);
117   printf("timer_settime(): %d; ", result);
118
119   /* wait for signal... */
120   sleep(2);
121
122   if (check == 1)
123     {
124       printf("Assertion-02: success; ");
125     }
126   else
127     {
128       printf("Assertion-02: failure; ");
129     }
130
131
132   /* POSIX Function timer_delete() */
133   result = timer_delete(timerid);
134   printf("timer_delete(): %d; ", result);
135
136   if(result == 0)
137     {
138       printf("Assertion-03: success; ");
139     }
140   else
141     {
142       printf("Assertion-03: failure; ");
143     }
144
145   printf("\n");
146
147   exit(0);
148 }
149 /* configuration information */
150
151 #define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
152 #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
153
154 #define CONFIGURE_POSIX_INIT_THREAD_TABLE
155
156 #define CONFIGURE_MAXIMUM_POSIX_THREADS 4
157 #define CONFIGURE_MAXIMUM_POSIX_TIMERS 4
158 #define CONFIGURE_MAXIMUM_TIMERS 4
159 #define CONFIGURE_POSIX_INIT_THREAD_TABLE
160 #define CONFIGURE_POSIX_INIT_TASKS_TABLE
161 #define CONFIGURE_INIT
162
163 #include <confdefs.h>
164
165 /* end of file */
```

B.2.5. RTEMS-CMP-PX-MSG.C

B.2.5.1. Description

1. Create / Open message queue;
2. **Assertion 1:** Message queue created;
3. Register notification handler for message;
4. Send message notification;
5. Send message to queue;
6. **Assertion 2:** Message send to queue;
7. **Assertion 3:** Notification received;
8. Receive message from queue;
9. **Assertion 4:** Message received from queue;
10. Set message queue attributes;
11. **Assertion 5:** Message queue attributes set;
12. Unlink message queue;
13. **Assertion 6:** Message queue unlincked ;
14. Close message queue;
15. **Assertion 7:** Message queue closed ;

B.2.5.2. Source Code

```
1   /*****************************************************************************
2    SRC-MODULE : Message Manager Workload
3    MODULE-VERS : N/A
4
5    $Source: /home/cvscritical/esa/rams/services/rams02/services/robustness-
    testing/implementation/posix-workloads/rtems-cmp-px-msg/rtems-cmp-px-msg.c,v $
6    $Id: rtems-cmp-px-msg.c,v 1.3 2003/09/09 12:34:10 lhenriques Exp $
7    $State: Exp $
8    $Locker:  $
9
10   Copyright (c) Critical Software (www.criticalsoftware.com)
11
12   SPEC-NO : CSW-RAMS-2003-RPT-1335
13
14   OS-TYPE : RTEMS 4.5.0
15
16   AUTHOR : rbarbosa
17
18   KEYWORDS : ----
19   PURPOSE : Test the functionality of all RTEMS APIs related to the Message
20             Manager.
21
22   CREATED ON : 17-07-2003
23   CHANGED ON : $Date: 2003/09/09 12:34:10 $
24   CHANGED BY : $Author: lhenriques $
25
26   $Revision: 1.3 $
27   STICKY TAG : $Name:  $
28
29   INSPECTED ON: 25-07-2003
30   MODERATOR : lhenriques
31
```

```
32   TABLES : none.
33
34   HISTORY
35   $Log: rtems-cmp-px-msg.c,v $
36   Revision 1.3  2003/09/09 12:34:10  lhenriques
37   Removed "printf" from signal handler.
38
39   Revision 1.2  2003/09/04 09:59:08  lhenriques
40   Repository : :pserver:lhenriques@cvs.critical.pt:/home/cvscritical
41   Module     : esa/rams/services/rams02/services/robustness-testing/implementation/posix-
     workloads/rtems-cmp-px-msg
42   Working dir: ~/posix-implementation/rtems-cmp-px-msg/
43
44
45
46   In directory .:
47                Unknown                 o-optimize
48                Modified                rtems-cmp-px-msg.c
49
50   ---------------------- End ----------------------
51   -- last cmd: cvs -f -n update -d -P --
52
53   Revision 1.1  2003/07/31 11:33:32  rbarbosa
54   Added the corresponding makefiles to each workload
55
56   Revision 1.3  2003/07/30 15:17:34  rbarbosa
57   Replaced main with init. Added configuration settings needed to compile
58   in RTEMS CCS. Added header to file.
59
60   Revision 1.2  2003/07/28 16:29:08  lhenriques
61   Several changes have been done and one of the workloads (for the scheduler manager) has
     been removed.
62
63 ****************************************************************************/
64
65 #include <bsp.h>
66 #include <signal.h>
67 #include <mqueue.h>
68 #include <stdio.h>
69 #include <sys/fcntl.h>
70 #include <string.h>
71 #include <unistd.h>
72
73 #define MAX_MSG_IN_QUEUE 10
74 #define MAX_MSG_SIZE 128
75
76 char * QUEUE_NAME = "test_msg_queue";
77 mqd_t mq;
78 int notif = 0;
79
80 void sigusr_signal_handler(int sig_num)
81 {
82   notif++;
83 }
84
85 void * POSIX_Init(void * ignored)
86 {
87   int result;
```

```
88    struct mq_attr attr;
89    struct mq_attr new_attr, old_attr, test_attr;
90    int oflag = O_RDWR | O_CREAT;
91    mode_t mode = S_IRWXU | S_IRWXG | S_IRWXO;
92    unsigned int msg_prio;
93    unsigned int sleep_time = 1;
94    int sig = SIGUSR1;
95    struct sigaction act, old_act;
96    struct sigevent notification;
97    char * msg = "RTEMS Message";
98    size_t msg_len;
99    size_t buf_len = MAX_MSG_SIZE;
100   char buffer[buf_len];
101
102   /* Message queue attributes: */
103   /* Flags */
104   attr.mq_flags = O_RDWR | O_CREAT;
105   /* Maximum number of messages */
106   attr.mq_maxmsg = MAX_MSG_IN_QUEUE;
107   /* Maximum message size */
108   attr.mq_msgsize = MAX_MSG_SIZE;
109
110   /* POSIX Function mq_open() */
111   mq = mq_open(QUEUE_NAME, oflag, mode, &attr);
112   printf("mq_open(): %d; ", mq);
113
114   if(mq != -1)
115     {
116       printf("Assertion-01: success; ");
117     }
118   else
119     {
120       printf("Assertion-01: failure; ");
121     }
122
123   /* POSIX Function mq_notify() */
124   /* First, set structure */
125   act.sa_flags = 0;
126   act.sa_handler = sigusr_signal_handler;
127   /* Set the action for the SIGUSR1 signal */
128   result = sigaction(sig, &act, &old_act);
129   printf("sigaction(): %d; ", result);
130
131   notification.sigev_notify = SIGEV_SIGNAL;
132   notification.sigev_signo = SIGUSR1;
133
134   result = mq_notify(mq, &notification);
135   printf("mq_notify(): %d; ", result);
136
137   msg_len = strlen(msg);
138   /* msg_prio = MQ_PRIO_MAX , so, not to make the highest priority, we will do msgprio -
    1; */
139   msg_prio = MQ_PRIO_MAX - 1;
140   result = mq_send(mq, msg, msg_len, msg_prio);
141   printf("mq_send(): %d; ", result);
142
143   if (result == 0)
144     {
```

```
145       printf("Assertion-02: success; ");
146     }
147   else
148     {
149       printf("Assertion-02: failure; ");
150     }
151
152   /* Wait for signal handler... */
153   sleep (sleep_time);
154   if (notif != 0)
155     {
156       printf("Assertion-03: success; ");
157     }
158   else
159     {
160       printf("Assertion-03: failure; ");
161     }
162   result = mq_receive(mq, buffer, buf_len, &msg_prio);
163   printf("mq_receive(): %d; ", result);
164
165   if ((result == msg_len) && (!strcmp(buffer, msg)))
166     {
167       printf("Assertion-04: success; ");
168     }
169   else
170     {
171       printf("Assertion-04: failure; ");
172     }
173
174   new_attr.mq_msgsize = 256;
175   new_attr.mq_maxmsg = 10;
176   new_attr.mq_flags = O_RDONLY | O_NONBLOCK;
177   result = mq_setattr(mq, &new_attr, &old_attr);
178   printf("mq_setattr(): %d; ", result);
179
180   result = mq_getattr (mq, &test_attr);
181
182   /* NOTE: only the mq_flags can be changed! */
183   if ((result == 0) &&
184       (test_attr.mq_flags == new_attr.mq_flags) &&
185       (test_attr.mq_maxmsg == old_attr.mq_maxmsg) &&
186       (test_attr.mq_msgsize == old_attr.mq_msgsize))
187     {
188       printf("Assertion-05: success; ");
189     }
190   else
191     {
192       printf("Assertion-05: failure; ");
193     }
194
195   result = mq_unlink(QUEUE_NAME);
196   printf("mq_unlink(): %d; ", result);
197
198   if (result == 0)
199     {
200       printf("Assertion-06: success; ");
201     }
202   else
```

```
203     {
204       printf("Assertion-06: failure; ");
205     }
206
207   result = mq_close(mq);
208   printf("mq_close(): %d; ", result);
209
210   if(result == 0)
211
212     {
213       printf("Assertion-07: success; ");
214     }
215   else
216     {
217       printf("Assertion-07: failure; ");
218     }
219
220   printf("\n");
221
222   exit(0);
223 }
224
225 /* configuration information */
226
227 #define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
228 #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
229 #define CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER
230 #define CONFIGURE_MAXIMUM_POSIX_MESSAGE_QUEUES 4
231 #define CONFIGURE_POSIX_INIT_THREAD_TABLE
232 #define CONFIGURE_MAXIMUM_POSIX_THREADS 1
233 #define CONFIGURE_POSIX_INIT_TASKS_TABLE
234 #define CONFIGURE_POSIX_INIT_THREAD_TABLE
235 #define CONFIGURE_INIT
236
237 #include <confdefs.h>
238
239 /* end of file */
```