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

RAMS Call-off Order 2

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

ESTEC/Contract Nº 16582/02/NL/PA

Date: *25-11-2003*
Pages: *20*
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

RAMS Call-off Order 2

Approved Version: 1.29			
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	24-07-2003
Luís Henriques	lhenriques@criticalsoftware.com	Senior Engineer	12-07-2003
Ricardo Barbosa	rbarbosa@criticalsoftware.com	Project Engineer	12-09-2003
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	21-07-2003	First Draft	Lubomir Velkov, Ricardo Barbosa
1.0	24-07-2003	Update after internal review.	Lubomir Velkov, Ricardo Barbosa, Ricardo Maia
1.1	12-09-2003	Updated with test execution results. Some corrections on the test definition.	Luís Henriques, Ricardo Barbosa, Ricardo Maia
1.2	17-09-2003	Updated after internal review meeting. Updated test methodology. Moved test definition and test results to an annex.	Ricardo Maia
1.3	25-11-2003	Updated disclaimer. Corrected some minor bugs.	Ricardo Maia

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Objective

This document presents the results of the robustness testing performed in the Real Time Executive for Multiprocessor Systems (RTEMS) version 4.5.0. This evaluation is performed in the scope of the Call-off Order number 02 under project Software Dependability and Safety Evaluations, ESTEC/Contract Nº 16582/02/NL/PA.

The main goal of this document is to define the test cases to perform on RTEMS, as well as the results driven from these tests along with the methodology applied in the definition and execution of the tests.

1.2 Scope

This report the deliverable of the 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 and presents the results of WP310 and WP320.

1.3 Audience

This document is targeted at several groups of readers, namely:

- "Software Dependability and Safety Evaluations" team members and in particular the Call-off Order 2 team members.
- Space software staff involved in the development of RTEMS related software.
- Space software product assurance staff.
- Management and technical ESA/ESTEC staff.

1.4 Acronyms

Acronyms	Description
API	Application Programming Interface
CSW	Critical Software, S.A.
TBD	To Be Defined
TBC	To Be Confirmed
POSIX	Portable Operating System Interface
RAMS	Reliability, Availability, Maintainability and Safety
RTEMS	Real Time Executive for Multiprocessor Systems

Table 1 Definitions and Acronyms

1.5 Document Structure

This document has the following structure:

Chapter 1 Introduces the document, as well as the document scope, intended audience and a list of acronyms and references used through out the document.

Chapter 2 Presents the test methodology and the fault model used in the robustness testing of RTEMS.

Chapter 3 Presents the templates used for the definition of the test campaigns and test suits and the templates used for the presentation of the test cases results.

Chapter 4	Summarizes the results obtained in evaluation of the RTEMS 4.5.0.
Annex A	Presents the test campaigns and test suites definition as well the results of the execution of the tests. It also provides an analysis of the results of each of the test cases.
Annex B	Contains the source code of all of the workloads, including the Classic and POSIX, used in the evaluation of the RTEMS 4.5.0

Annex A and Annex B are available as a separate volume of this document.

1.6 References

[1] RTEMS 4.5.0 Evaluation Report, DL-RAMS02-01-02, CSW-RAMS-2003-RPT-1334, Ricardo Barbosa, Ricardo Maia, Luís Henriques, Lubomir Velkov, João Esteves, 17/09/2003

[2] OAR, RTEMS C User's Guide, September 2000

[3] OAR, RTEMS POSIX User's Guide, September 2000

[4] OAR, RTEMS POSIX 1003.1 Compliance Guide, September 2000

[5] ORK-ERC32-SW - Software Requirement Specification, CSW-STADY-2002-SRS-0835, Ricardo Maia, November 2002

[6] Automated Robustness Testing of Off-the-Shelf Software Components, June 1998, 28th Fault Tolerant Computing Symposium, in press, Kropp, N., Koopman, P. & Siewiorek, D.

2. Test Methodology and Fault Model

2.1 Test Methodology

The methodology used in this robustness testing of the RTEMS real-time kernel, consists in testing the RTEMS API calls using out-of-bound parameters.

This methodology is composed by several phases:

- Preparation – including all the tasks needed to define the test cases.
- Test Execution – execution of the defined test cases.
- Log Analysis – analysis of the results of the test cases and identification of the RTEMS faults.

Preparation phase comprises the following tasks:

- Product Analysis and scope definition – analysis of the product under evaluation (i.e. the RTEMS 4.5.0) and selection of the API calls that will be subjected to the evaluation.
- Fault Model Definition – definition of the in-bound and out-of-bound values that will be used for each of the RTEMS data types.
- Construction of the workloads – definition and implementation of the applications that will exercise the RTEMS APIs.
- Definition of the test campaigns and test suites – definition of the test suites that will be used to automatically generate the test cases. Test suites are grouped logically in test campaigns.

The Test Execution phase follows the Preparation. During this phase test cases are executed and the results are collected in a database. This task is performed in unattended mode the by Xception.

The final phase of the robustness testing is the Log Analysis. In this phase detailed analysis of the log of each test case is performed comparing the obtained results against the expected values. This phase can be time consuming. For this reason it is important to have a concise workload output that enables the analyst to quickly find out if the result of the test case is consistent with the input parameters or not.

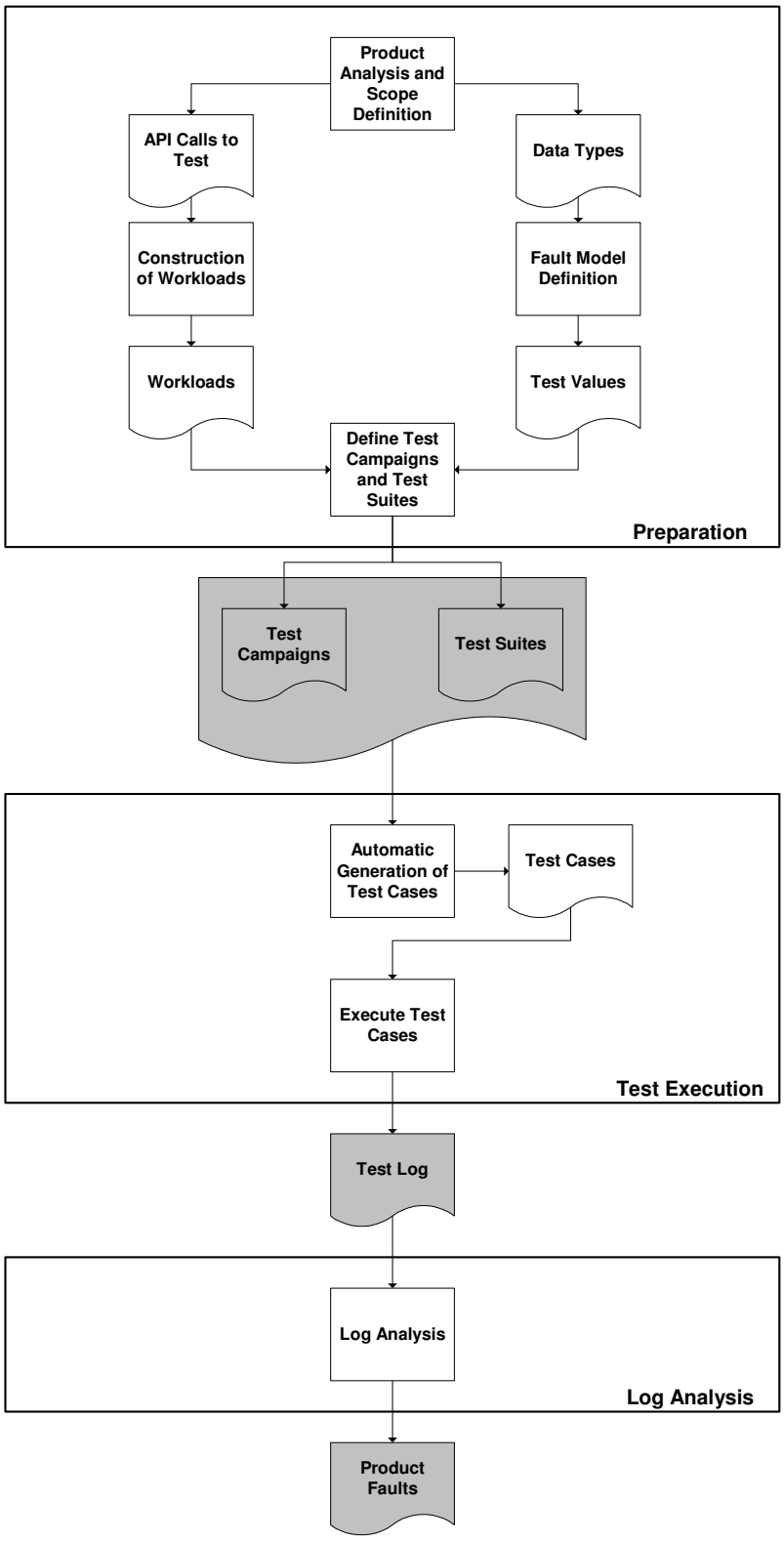

Figure 1 - Robustness Testing Methodology

The next sections provide details of the *Product Analysis and scope definition* and *Fault Model Definition* tasks performed in the evaluation of the RTEMS.

The definition of the test campaigns and test suites as well as the analysis of the test cases execution is presented in the annex A of this document.

The source code of the workloads can be found in annex B.

2.2 Product Analysis and Scope Definition

The Real Time Executive for Multiprocessor Systems (RTEMS) is a real time executive that provides a high performance environment for embedded critical and military applications including the following features:

- Multitasking capabilities;
- Homogeneous and heterogeneous multiprocessor systems support;
- Event-driven, priority based, pre-emptive scheduling;
- Optional rate monotonic scheduling;
- Intertask communication and synchronisation;
- Priority Inheritance mechanisms;
- Responsive interrupt management;
- Dynamic memory allocation;
- High level of user configurability.

The internal architecture for RTEMS can be viewed as a set of layers that work closely with each other to provide the set of services to the real time applications. The executive interface presented to the application is formed by directives (RTEMS API Calls) grouped into logical sets called resource managers.

RTEMS 4.5.0 provides several APIs for real time application programming. Two of this APIs were subject to this evaluation: the Classic API and the POSIX API (see Figure 2 and Figure 3). The Classic is the native and older RTEMS API. The POSIX API is intended for compliance of the kernel with the IEEE Std 1003 POSIX standard.

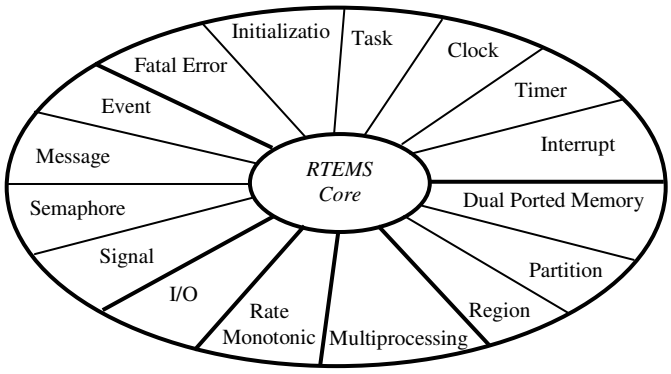

Figure 2. RTEMS Classic API Internal Architecture

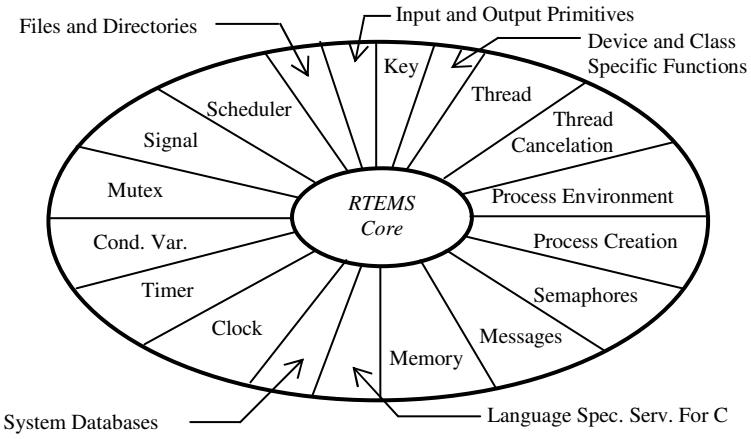

Figure 3 - RTEMS POSIX API Internal Architecture

Although almost all of the directives of the RTEMS Classic API were subject to this evaluation the same was not done for the POSIX API. For further details on the scope definition please refer to [1] chapter 3.

2.3 Fault Model Definition

The test values for the basic data types used in this evaluation were defined taking into account previous experience from the application of the robustness testing methodology at Critial Software SA and elsewhere namely from Ballista project [6].

Table 2 shows the test values used for basic data types.

Type Name	Test Values[1]
char	0, 255
signed char	0, -128, 127
int	0, 1, -1, 2147483647, -2147483648
unsigned int	0, 1, 18446744073709551615
short int	0, 1, -1, 32767, -32768
unsigned short int	0, 1, 65535
long	0, 1, -1, 9223372036854775807, -9223372036854775808
unsigned long	0, 1, 18446744073709551615
pointers	NULL

Table 2 - Test Values for Basic Data Types

Note that all RTEMS data types derive from these basic data types. Thus, the test values used for each of the RTEMS data types are the ones presented for the corresponding basic data type.

For complete list of the RTEMS data types and the respective test values please refer to [1].

2.4 Workloads

In order to perform the evaluation it is required to have some application that exercises the RTEMS directives under test. These applications are called the workloads.

[1] Test Values typicaly include MAXIMUM and MINIMUM values of the corresponding type.

For the evaluation of the RTEMS 4.5.0 it was decided to have one workload for each of the resource managers under test. Each of these workloads exercises the directives of the corresponding manager.

A total of 19 workloads were constructed during the preparation phase of this evaluation, 14 corresponding to the Classic API and 5 to the POSIX API.

API	Workloads
Classic	14
POSIX	5
Total	**19**

Table 3 - Workloads Defined

The source code of the defined workloads can be found in Annex B.

2.5 Definition of Test Campaigns and Test Suites

During the preparation phase of this evaluation test campaigns and test suites were defined. One campaign was defined for each of the RTEMS resource managers under test. Each campaign includes one test suite for each directive of the corresponding manager.

A total of 19 campaigns were defined, 14 for the Classic resource managers and 5 for the POSIX resource managers.

Table 4 shows the number of tests suites defined for both Classic and POSIX RTEMS APIs.

API	Number of Test Suites
Classic	56
POSIX	37
Total	**93**

Table 4 - Number of Test Suites Defined

3. Documentation Templates

The following templates are used throughout the document in the definition of the test campaigns, test suites and in the presentation of the test cases results.

3.1 Test Campaign Definition Template

The following table is used to specify the characteristics of a test campaign.

Test Campaign Definition	
Campaign Identifier:	
Purpose:	
Workload File:	
Test Suites:	**1.**
Workload Description:	

Table 5 Test Campaign Definition Template

3.1.1 Campaign Identifier

This field identifies the test campaign name. The following naming convention is used:

RTEMS-CMP-<API INTERFACE ID>-<RTEMS MANAGER ID>

(e.g. RTEMS-CMP-CL-TSK)

- RTEMS – The acronym of the product under evaluation
- CMP – stands for campaign
- <API INTERFACE ID> - Identifier of the API to be tested by the campaign (refer to Table 6)
- <RTEMS MANAGER ID> - Identifier of the resource manager to be tested by the campaign (refer to Table 7).

API Interface Abbreviation	Description
CL	Classic
PX	POSIX

Table 6 API Interface Identifiers

RTEMS Manager Abbreviation	Description	API Interface(s)
TSK	Task Manager	CL
INT	Interrupt Manager	CL
CLK	Clock Manager	CL, PX
TMR	Timer Manager	CL, PX
SMP	Semaphore Manager	CL
MSG	Message Manager	CL, PX

EVT	Event Manager	CL
SGL	Signal Manager	CL, PX
PRT	Partition Manager	CL
RGN	Region Manager	CL
IO	Input Output Manager	CL
FER	Fatal Error Manager	CL
RMT	Rate Monotonic Manager	CL
UEX	User Extensions Manager	CL
MTX	Mutex Manager	PX

Table 7 RTEMS Manager Identifiers

3.1.2 Purpose

Describes the purpose of this test campaign.

3.1.3 Workload File

Provides the name and reference to the related workload implementation file. The workloads source code can be found in the Annex B of this document.

3.1.4 Test Suites

Lists the test suites of this specific test campaign.

3.1.5 Workload Description

Provides details regarding to workload implementation.

3.2 Test Suite Definition Template

The following table is used in the definition of a test suite.

Test Suite Definition
Test Suite Identifier:
Purpose:
Fault Location(s):
Test Item:
Generated Test Cases:

Table 8 Test Suite Definition Template

3.2.1 Test Suite Identifier

Uniquely identifies the test suite.

The identifier used follows the naming convention:

RTEMS-TS-<API INTERFACE ID>-<RTEMS DIRECTIVE>

(e.g. RTEMS-TS-CL-TSKCRT)

- RTEMS – The acronym of the product under evaluation
- TS – stands for Test Suite

- <API INTERFACE ID> - Identifier of the API to be tested by the campaign (refer to Table 6)

- <RTEMS DIRECTIVE> - Identifier of the RTEMS directive (e.g API call) to be tested by the test suite (refer to Table 9).

RTEMS API Abbreviation	API Name	API Interface(s)
TSKCRT	rtems_task_create	CL
TSKSTR	rtems_task_start	CL
TSKRST	rtems_task_restart	CL
TSKDLT	rtems_task_delete	CL
TSKRSM	rtems_task_resume	CL
TSKSPT	rtems_task_set_priority	CL
TSKMOD	rtems_task_mode	CL
CLKSET	rtems_clock_set	CL
CLKTCK	rtems_clock_tick	CL
TMRCRT	rtems_timer_create	CL
TMRDLT	rtems_timer_delete	CL
TMRFAF	rtems_timer_fire_after	CL
TMRWHN	rtems_timer_fire_when	CL
TMRRST	rtems_timer_reset	CL
INTCTC	rtems_interrupt_catch	CL
PRTCRT	rtems_partition_create	CL
PRTGBF	rtems_partition_get_buffer	CL
PRTRBF	rtems_partition_return_buffer	CL
PRTDLT	rtems_partition_delete	CL
UEXCRT	rtems_extension_create	CL
UEXDLT	rtems_extension_delete	CL
SMPCRT	rtems_semaphore_create	CL
SMPDLT	rtems_semaphore_delete	CL
SMPOBT	rtems_semaphore_obtain	CL
SMPRLS	rtems_semaphore_release	CL
SMPFLS	rtems_semaphore_flush	CL
SGLCTC	rtems_signal_catch	CL
SGLSND	rtems_signal_send	CL
RMTCRT	rtems_rate_monotonic_create	CL
RMTDLT	rtems_rate_monotonic_delete	CL
RMTCNL	rtems_rate_monotonic_cancel	CL
RMTPRD	rtems_rate_monotonic_period	CL
RGNCRT	rtems_region_create	CL
RGNGSG	rtems_region_get_segment	CL
RGNGSS	rtems_region_get_segment_size	CL
RGNDLT	rtems_region_delete	CL
RGNEXT	rtems_region_extend	CL
MSGCRT	rtems_message_queue_create	CL
MSGDLT	rtems_message_queue_delete	CL
MSGSND	rtems_message_queue_send	CL
MSGURG	rtems_message_queue_urgent	CL
MSGBRD	rtems_message_queue_broadcast	CL

MSGRCV	rtems_message_queue_receive	CL
MSGFSH	rtems_message_queue_flush	CL
MSGGNP	rtems_message_queue_get_message_pending	CL
PRTCRT	rtems_port_create	CL
PRTDLT	rtems_port_delete	CL
IOINI	rtems_io_initialize	CL
IOREG	rtems_io_register_name	CL
IOOPN	rtems_io_open	CL
IOCLS	rtems_io_close	CL
IOREAD	rtems_io_read	CL
IOWRT	rtems_io_write	CL
IOCTL	rtems_io_control	CL
FEROCC	rtems_fatal_error_occured	CL
EVTSND	rtems_event_send	CL
EVTRCV	rtems_event_receive	CL
SGLSAS	sigaddset	PX
SGLSDS	sigdelset	PX
SGLSFS	sigfillset	PX
SGLSES	sigemptyset	PX
SGLSAC	sigaction	PX
SGLPTK	pthread_kill	PX
SGLSPM	sigprocmask	PX
SGLKIL	kill	PX
SGLSUS	sigsuspend	PX
SGLSWI	sigwaitinfo	PX
SGLSTO	sigtimedout	PX
MTXMAI	pthread_mutexattr_init	PX
MTXMAD	pthread_mutexattr_destroy	PX
MTXAPT	pthread_mutexattr_setprotocol	PX
MTXACL	pthread_mutexattr_setprioceiling	PX
MTXASH	pthread_mutexattr_setpshared	PX
MTXINI	pthread_mutex_init	PX
MTXDTR	pthread_mutex_destroy	PX
MTXLCK	pthread_mutex_lock	PX
MTXTLK	pthread_mutex_trylock	PX
MTXTML	pthread_mutex_timedlock	PX
MTXULK	pthread_mutex_unlock	PX
MTXCEI	pthread_mutex_setprioceiling	PX
CLKCST	clock_settime	PX
CLKSLP	sleep	PX
CLKNNS	nanosleep	PX
CLKGET	clock_gettime	PX
TMRCRT	timer_create	PX
TMRDLT	timer_delete	PX
TMRSTM	timer_settime	PX
MSGOPN	mq_open	PX
MSGCLS	mq_close	PX

MSGULK	mq_unlink	PX
MSGSND	mq_send	PX
MSGRCV	mq_receive	PX
MSGNTF	mq_notify	PX
MSGSAT	mq_setattr	PX

Table 9 - RTEMS Directives Identifiers

3.2.2 Purpose

Describes the purpose of the test suite.

3.2.3 Fault Location(s)

Identifies the location of the API call that will be used to perform the test. It includes the filename of the workload and the line(s) number(s) in which the call is performed.

3.2.4 Test Item

Signature of RTEMS API under test (e.g. `rtems_task_delete (rtems_id id)`).

3.2.5 Generated Test Cases

Number of test cases automatically generated by Xception.

3.3 Test Case Result Template

Each of the test cases in which a fault on the RTEMS is uncovered will be presented in this document using the template described in this section.

TEST CASE RESULT
Test case result identifier:
Input Specification:
Fault Description:
Notes:

Table 10 Test Case Result Template

3.3.1 Test Case Result Identifier

Uniquely identifies the test case result.

The identifier used follows the naming convention:

RTEMS-TCR-<API INTERFACE ID>-<RTEMS DIRECTIVE>

(e.g. RTEMS-TCR-CL-TSKCRT)

- RTEMS – The acronym of the product under evaluation
- TCR – stands for Test Case Result

- <API INTERFACE ID> - Identifier of the API tested by the test case (refer to Table 6)
- <RTEMS DIRECTIVE> - Identifier of the RTEMS directive (e.g API call) tested by the test case (refer to Table 9).

3.3.2 Input Specification

Specifies the parameter changed and the test value used in the test case.

3.3.3 Fault Description

Describes the fault uncovered in the execution of the test case.

3.3.4 Notes

Provides further information regarding to the test case.

4. Results Summary

Two different APIs of the RTEMS 4.5.0 were subject to this evaluation:

- RTEMS Classic API and
- RTEMS POSIX API.

During the evaluation of the Classic API 527 test cases were defined. The execution of these test cases uncovers 34 faults. Table 11 shows the distribution of the test cases and faults among the several RTEMS managers and directives.

Manager	Directive	Test Cases	Faults
Task	rtems_task_create	18	2
	rtems_task_start	7	1
	rtems_task_restart	6	0
	rtems_task_delete	3	0
	rtems_task_resume	3	0
	rtems_task_set_priority	9	1
	rtems_task_set_mode	9	0
Interrupt	rtems_interrupt_catch	5	0
Clock	rtems_clock_set	42	0
	rtems_clock_get	26	0
Timer	rtems_timer_create	21	1
	rtems_timer_delete	6	0
	rtems_timer_fire_after	8	1
	rtems_timer_fire_when	26	1
	rtems_timer_fire_cancel	3	0
	rtems_timer_fire_reset	3	0
Semaphore	rtems_semaphore_create	15	1
	rtems_semaphore_delete	3	0
	rtems_semaphore_obtain	9	0
	rtems_semaphore_release	3	0
	rtems_semaphore_flush	3	0
Message	rtems_message_queue_create	31	4
	rtems_message_queue_delete	3	0
	rtems_message_queue_send	7	0
	rtems_message_queue_urgent	7	0
	rtems_message_queue_broadcast	10	1
	rtems_message_queue_receive	13	1
	rtems_message_queue_flush	6	1
	rtems_message_queue_get_number_pending	6	1
Event	rtems_event_send	6	0
	rtems_event_receive	12	0
Signal	rtems_signal_catch	4	0
	rtems_signal_send	6	1
Partition	rtems_partition_create	16	1
	rtems_partition_get_buffer	4	1
	rtems_partition_return_buffer	4	0
	rtems_partition_delete	3	0
Region	rtems_region_create	16	1

	rtems_region_get_segment	17	2
	rtems_region_get_segment_size	20	3
	rtems_region_extend	7	0
	rtems_region_delete	3	0
	rtems_region_return_segment	4	1
IO	rtems_io_initialize	7	0
	rtems_io_register_name	8	2
	rtems_io_open	7	1
	rtems_io_close	7	0
	rtems_io_read	7	1
	rtems_io_write	7	1
	rtems_io_control	7	1
Fatal Error	rtems_fatal_error_occured	3	0
Rate Monotonic	rtems_rate_monotonic_create	6	1
	rtems_rate_monotonic_delete	3	0
	rtems_rate_monotonic_cancel	3	0
	rtems_rate_monotonic_period	12	0
User Extensions	rtems_extension_create	14	1
	rtems_extension_delete	3	0
Total		**527**	**34**

Table 11 - Summary results of the Classic API Evaluation

During the evaluation of the POSIX API 528 test cases were defined. The execution of these test cases uncover 16 faults. Table 12 shows the distribution of the test cases and faults among the several RTEMS managers and directives.

Manager	Directive	Test Cases	Faults
Clock	clock_settime	8	0
	clock_gettime	8	0
	sleep	3	0
	nanosleep	13	0
Timer	timer_create	18	3
	timer_delete	3	0
	timer_settime	8	0
Message	mq_open	33	3
	mq_close	3	0
	mq_unlink	2	0
	mq_send	13	0
	mq_receive	13	0
	mq_notify	15	0
	mq_setattr	43	0
Signal	sigaddset	8	0
	sigdelset	8	0
	sigfillset	3	0
	sigemptyset	3	0
	sigaction	25	1
	pthread_kill	18	1
	sigprocmask	11	0
	kill	10	1

	sigsuspend	3	2
	sigwaitinfo	14	0
	sigtimedwait	19	0
Mutex	pthread_mutexattr_init	25	0
	pthread_mutexattr_destroy	25	1
	pthread_mutexattr_setprotocol	30	0
	pthread_mutexattr_setprioceiling	30	1
	pthread_mutexattr_setpshared	30	0
	pthread_mutex_init	28	1
	pthread_mutex_destroy	3	0
	pthread_mutex_lock	3	0
	pthread_mutex_trylock	3	0
	pthread_mutex_timedlock	8	0
	pthread_mutex_unlock	3	1
	pthread_mutex_setprioceiling	35	0
Total		**528**	**15**

Table 12 - Summary Resulta of the POSIX API Evaluation

The overall number of test cases and faults is shown in Table 13.

API	Test Cases	Faults
Classic	527	34
POSIX	528	15
Total	**1055**	**49**

Table 13 - Overall Results