RTEMS Improvement – Space Qualification of RTEMS Executive

Helder Silva, José Sousa, Daniel Freitas, Sergio Faustino,
Alexandre Constantino and Manuel Coutinho,

EDISOFT, Empresa de Serviços e Desenvolvimento de Software, S.A.,
Rua Quinta dos Medronheiros - Lazarim,
Apartado 2826-801 Caparica, Portugal
{Helder.Silva, José.Sousa, Daniel.Freitas, Sergio.Faustino,
Alexandre.Constantino, Manuel.Coutinho}@Edisoft.pt

Abstract. RTEMS (Real-Time Executive for Multiprocessor Systems) has been selected as a possible ESA (European Space Agency) building block for space missions. As a first step to achieve the category of a building block, RTEMS must attain TRL (Technological Readiness Level) 6, which is equivalent to a product release in software engineering. TRL6 is achieved after the successful evaluation of the RTEMS technology, in this case, after the conclusion of RTEMS Improvement project. The full qualification of the technology will be achieved after the successful run of the operating system qualification process, with the space mission hardware and the application running on top of RTEMS, like control systems, telecommand or telemetry applications.

Keywords: RTEMS, Space Qualification, Galileo Software Standards, Real-Time and Embedded Systems.

1 Introduction

This paper presents the current outputs of an industrial activity held by EDISOFT to facilitate the qualification of RTEMS (Real-Time Executive for Multiprocessing Systems) for space applications and space missions. The activity has been performed with and for ESA (European Space Agency) following the GSWS (Galileo Software Standards) for a DAL (Development Assurance Level) allocation B (software whose anomalous behaviour would cause or contribute to a failure resulting in a critical event).

This paper summarizes the presentations made in the last 3 years in DAta Systems in Aerospace conference in 2007, [1], 2008, [2] and 2009, [3], which describe the activities performed in ESA contracts [6] and [7] of the RTEMS CENTRE project (Support and Maintenance CENTRE to RTEMS Operating System), final presentation made in ESTEC (European Space Research and Technology Centre) in December 2008 [4] and RTEMS Improvement project, an on-going ESA project.

RTEMS CENTRE objectives were the design, development, maintenance and integration of tools to augment and sustain RTEMS operating system and the creation

and maintenance of technical competences and support site to RTEMS operating system in Europe. The general idea is to minimize the cost of airborne and space applications incorporation/integration.

RTEMS Improvement primary objectives are to improve RTEMS product and its documentation in order to facilitate the qualification of RTEMS for future space missions, taking into account the specific operational requirements and target environment and to provide Memory Manager for LEON hardware Memory Management Unit (MMU).

The first section makes the introduction of the paper, the rationale behind a qualification process and the description of each of the sections. Section 2 makes a brief presentation of the RTEMS executive, section 3 provides an overview of the qualification process and section 5 concludes the paper and highlights some of the future work of EDISOFT in RTEMS.

2 RTEMS Overview

Real Time Executive for Multiprocessor Systems (RTEMS) is an open source Real Time Operating System (RTOS) designed for deeply embedded systems that aim to be competitive with closed source and commercial products [9]. Currently it is maintained by the On-Line Research Corporation (OAR) albeit many of the features and platform support for it have been developed by RTEMS users. The first version, of what is today RTEMS, was released in 1988.

RTEMS supports multiple processors, including SPARC (ERC32 and LEON family), i386, Power PC and others, complies with several standards (POSIX, RTEID/ORKID, TCP/IP, µITRON, ANSI C/C++, partially with Ada95), supports basic kernel features, provides networking (FreeBSD TCP/IP stack, UDP, TCP, etc) and filesystem functionalities (IMFS, FAT 32/16/12, etc), and includes debugging features (over Ethernet and serial port). RTEMS was designed to support applications with the most inflexible time frames requirements, making it possible for the user to develop hard real time systems [10].

The RTEMS kernel supports several features. The most important are multi-tasking, networking and support of file systems.

Multi-tasking allows the software to be more responsive and modular, but brings a lot of issues that must be solved by the kernel. RTEMS makes possible to use multi-tasking, since it implements three scheduling features (Event driven, Priority driven and Rate monotonic). The kernel also allows the selection of the modules that are loaded, avoiding unnecessary delays and memory usage (footprint).

Networking features are also quite valuable, since RTEMS implements several useful networking protocols. This allows loading a kernel or executive from a network, using BOOTP (Bootstrap Protocol). This avoids the need to have a complete operating system running in the target just to load a new executive since it's loaded from the network. Other protocols are useful for the application itself, like TCP, UDP, ICMP, RARP and DHCP. There are also some servers implemented as FTP server, HTTP server that allows file transfer in an easy manner. The PPP server allows the connection via dial-up modems and Telnet to control and configure the target system.

RTEMS filesystem provides features like UNIX, e.g., mountable systems, hierarchical system directory structure, POSIX compliant set of routines for the manipulation of files and directories.

RTEMS remote debugging is very important, since debugging on site is difficult due to board constraints. For development itself, RTEMS can build applications in C and C++, following the ISO standards, and ADA95 (limited). POSIX and µITRON standards are implemented in the system API's.

RTEMS can be characterized into three distinct levels [10], the operational, organizational and conceptual levels.

The operational level defines the relationship between RTEMS and the user application. The RTEMS installation process ends with two major libraries (librtemsbsp.a and librtemscpu.a) that are linked with the user application.

The organization level is the approach taken by RTEMS developers to organize and structure the source code of the operating system. Fig. 1 presents the top level RTEMS directory structure.

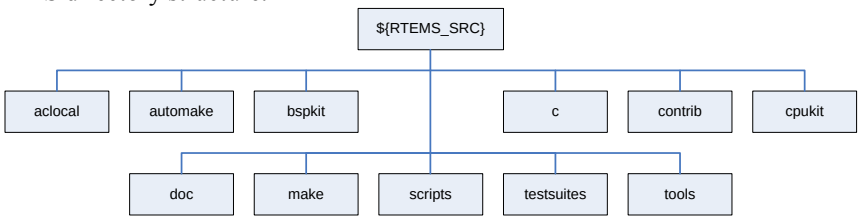

Fig. 1. RTEMS Directory Structure

The conceptual level is divided in three different layers, the hardware support, the kernel and the application interface. Applications can be developed in C, C++ and Ada95 (support is limited for Ada95) using different APIs such as Ada, POSIX, µITRON and RTEMS' own API set (based on the RTEID/ORKID standard). All APIs use RTEMS supercore, except for the Ada API which uses directly the RTEMS' API as an abstraction layer. The hardware support layer encompasses the processor and board dependent files as well as a common hardware library. The kernel layer is the heart of RTEMS and encompasses the super core, the super API and several portable support libraries. Fig. 2 presents a schematic of the RTEMS conceptual level architecture.

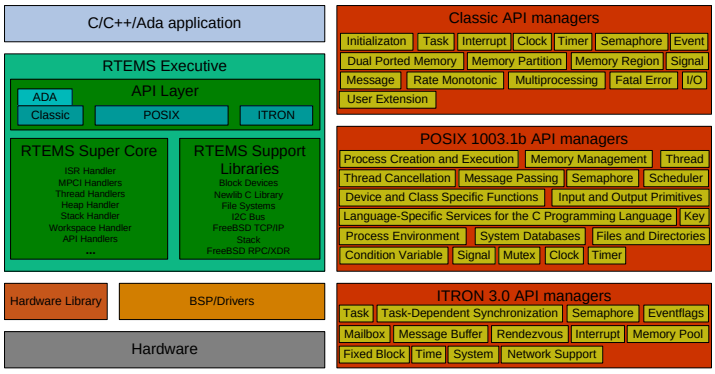

Fig. 2. RTEMS Architecture

3 Qualification Process

The qualification process aims to increase the quality of any product and its associated documentation. This prevents the occurrence of future problems with the software. The qualification deliverables were produced for RTEMS version 4.8.0 and for ERC32 (SPARC V7 hardware architecture), LEON2 and LEON3 (SPARC V8 architecture), the hardware boards used and produced by ESA for the space missions. The qualification process can be made using some of the software engineering standards like DO-178B, ECSS-E40, ECSS-Q80 and GSWS. Since it was predicted the usage of RTEMS in Galileo program and since the GSWS is the most complete standard known in terms of qualification, ESA and EDISOFT decided the usage of GSWS.

3.1 Galileo Software Standards

The Galileo Software Standard (GSWS) [5] sets requirements to be followed for the management, evaluation, procurement, development, production, verification, operation and maintenance of all software products.

It defines procedures to be followed for software engineering, software product assurance and software configuration management. Several reference life cycles for the SW Components are defined. Table 1 presents the generic software waterfall life cycle used for RTEMS Improvement.

Table 1. Generic software waterfall life cycle phases and reviews

Phase	SW Review
SW Planning Phase	Software Requirements Review
SW Specification Phase	Preliminary Design review
SW Design Phase	Detailed Design Review
SW Implementation Phase	Test Readiness Review
SW Integration and TS-Validation Phase	Critical Design Review
SW RB-Validation Phase	Qualification Review
SW Acceptance Phase	Acceptance Review
SW Maintenance Phase	--
SW Operation Phase	--

The software is then classified with a DAL (Development Assurance Level) that is dependent of the criticality usage of the software. The levels go from Level A, the highest level of criticality, to Level E, the lowest level of criticality. RTEMS was classified as DAL B, since its anomalous behaviour can cause a failure resulting in a critical event. Based in the DAL selection of the software, the development process, the deliverables and tests to be performed are fully defined, meaning that each Level has its own development life-cycle, document deliverables, phases and document versions. Table 2 makes an overview of GSWS software life cycles, type of software and DAL.

Table 2. GSWS Life Cycles vs. DAL and Types of Software

Type Of Software	DAL A	DAL B	DAL C	DAL D	DAL E
Generic	Waterfall	Waterfall	Waterfall	Waterfall Incremental	Waterfall Incremental
Databases	Waterfall	Waterfall	Waterfall	Waterfall	Waterfall
MMI	Evolutionary	Evolutionary	Evolutionary	Evolutionary	Evolutionary
Test Software	--	--	--	--	Waterfall Incremental
Simulators	Waterfall	Waterfall	Waterfall	Waterfall	Waterfall
Algorithm Prototypes	--	--	--	--	Evolutionary

The following figures (Fig. 3, Fig. 4 and Fig. 5) present a sample of the different life cycles used in GSWS. In the waterfall life cycle, the one used in RTEMS Improvement, the phases are produced continuously and each phase is marked by a review meeting.

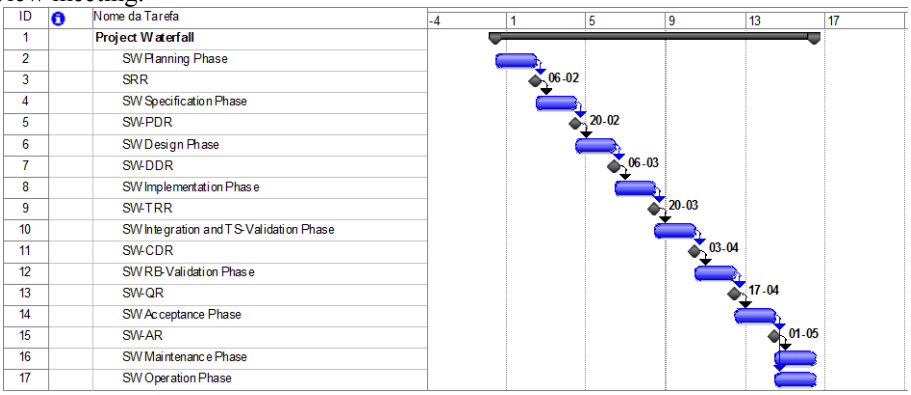

Fig. 3. Waterfall Project

In the evolutionary life-cycle, the software is built in different builds of design, implementation, integration and validation, ending by a QR and followed by the acceptance phase.

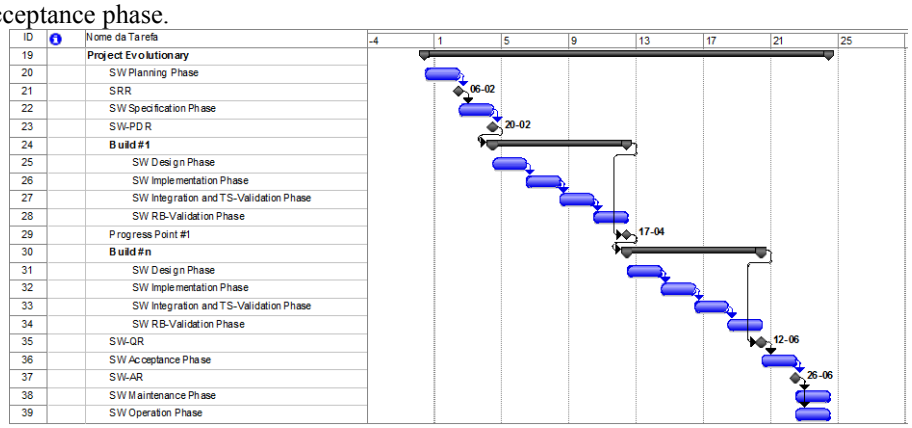

Fig. 4. Evolutionary Project

The incremental life-cycle is also built in different builds of just design, implementation, integration and a pre-validation. A formal validation of all the builds is necessary.

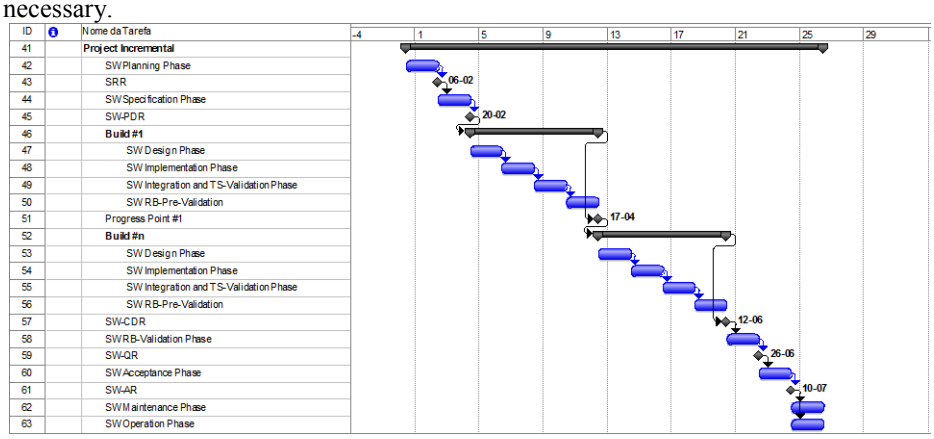

Fig. 5. Incremental Project

3.2 RTEMS Qualification Overview

The main objective is to provide a *qualifiable* version of RTEMS. The RTEMS version 4.8.0 was selected as baseline for the project because it was the latest version of RTEMS and because it added and fixed important features, like support for time granularity in nanoseconds, fixed problems on thread's priority, reduced the footprint for a more compact executable and presented a set of new drivers for LEON2 and LEON3 processors.

The user is capable of downloading RTEMS and with a configuration tool, is capable of filtering relevant RTEMS features and removing dead code (applying patch to the original RTEMS). At the same time, the same tool builds a tailored test suite that is merged and executed with the tailored version of RTEMS to produce the final system (operating system sub-set and test case battery) that facilitates the qualification of the operating system software. Fig. 6 presents a general overview of final product of the RTEMS Improvement.

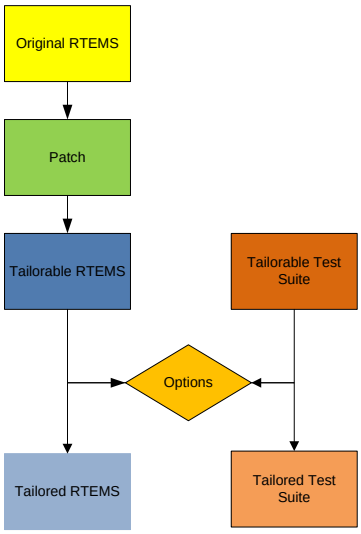

Fig. 6. RTEMS Tailoring Road Map

Documentation of RTEMS was revised and used as input to create the RTEMS requirements [16], to reverse engineer RTEMS and to build its architecture [15]. The design of RTEMS was addressed and constitutes the base for the development and design of the test battery. Table 3 presents a list of the RTEMS documentation produced so far by the RTEMS Improvement.

Table 3. List of RTEMS produced Documentation

RTEMS Improvement Data Pack
RTEMS managers candidate evaluation report
RTEMS Improvement Requirement Document
RTEMS Improvement User Manual and Design Notes
RTEMS Improvement Verification Report
Software Budget Report
Product Software Justification File
RTEMS Improvement Design Document
RTEMS Improvement Configuration File
RTEMS Improvement Integration Test Plan
RTEMS Improvement Unit Test Plan
RTEMS Improvement Validation Testing Specification – Technical Specification
RTEMS Tailoring Plan
RTEMS Improvement Generic Test Report
RTEMS Test Suite
RTEMS Improvement Acceptance Test Plan
RTEMS Improvement Maintenance Plan
RTEMS Improvement Installation Report
RTEMS Improvement Acceptance Data Package
RTEMS Tailored
Software Development Plan
Review Plan
Final Report

RTEMS Improvement Product Assurance Plan
RTEMS Improvement Product Assurance Report
RTEMS Improvement Configuration Management Plan
RTEMS Improvement SOC with GSWS
RTEMS Improvement Preliminary Software Criticality Analysis Report

3.3 RTEMS Candidate Managers

Surveys [11] were performed near European Space users (SAAB, OHB and ESA) and along with the EDISOFT assessment, candidate RTEMS managers were selected. Table 4 provides the list of the managers currently being used in some space projects by SAAB and OHB. This information provides RTEMS Improvement guides for the facilitation of qualification.

Table 4. Space Users Survey Results

RTEMS Managers	SAAB Survey	OHB Survey	ESA SoW
Initialization Manager	Yes	Yes	Yes
Task Manager	Yes	Yes	Yes
Interrupt Manager	Yes	Yes	Yes
Clock Manager	Yes	Yes	Yes
Timer Manager	Yes	Yes	Yes
Semaphore Manager	Yes	Yes	Yes
Message Manager	Yes	Maybe	Yes
Event Manager	Yes	Maybe	Yes
Signal Manager	No	Maybe	Yes
Partition Manager	Yes	Maybe	No
Region Manager	No	Maybe	No
Dual-Ported Memory Manager	No	No	No
I/O Manager	No	Yes	Yes
Fatal Error Manager	Yes	Yes	Yes
Rate Monotonic Manager	Yes	Yes	Yes
Barrier Manager	No	Maybe	No
User Extensions Manager	No	No	Yes
Multiprocessing Manager	No	No	Yes
Stack Bounds Checker	No	No	No
CPU Usage Statistics	No	No	No

Based in the survey conducted and a deep analysis of the RTEMS Classic API Managers and its dependencies, it was possible to select the candidate managers and primitives to be included in the work. Table 5 displays the results.

Table 5. Selected RTEMS Managers

RTEMS Manager	RTEMS Primitive	RTEMS Manager	RTEMS Primitive
Initialization	All directives	Clock	All directives
Task	rtems_task_create	Timer	All directives
	rtems_task_ident	Semaphore	All directives
	rtems_task_start	Message Queue	All directives
	rtems_task_restart	Event	All directives
	rtems_task_delete	I/O	rtems_io_initialize
	rtems_task_suspend		rtems_io_open
	rtems_task_resume		rtems_io_close
	rtems_task_is_suspended		rtems_io_read
	rtems_task_set_priority		rtems_io_write
	rtems_task_mode		rtems_io_control
	rtems_task_get_note	Fatal Error	All directives
	rtems_task_set_note	Rate Monotonic	rtems_rate_monotonic_creat
	rtems_task_wake_after		rtems_rate_monotonic_ident
	rtems_task_wake_when		rtems_rate_monotonic_cance
	rtems_task_variable_add		rtems_rate_monotonic_delet
	rtems_task_variable_get		rtems_rate_monotonic_perio
	rtems_task_variable_delete		rtems_rate_monotonic_get_status
Interrupt	All directives	User Extensions	All directives

3.4 RTEMS Engineering and Testing

The original version of RTEMS was truncated and several files were removed because of two main reasons, they were considered unnecessary and they were dead code. As part of the main goal, one of the project outputs is to provide means to achieve a RTEMS tailored version starting from the RTEMS original version. The version shall run in LEON2, LEON3 and ERC32 platforms (Fig. 7).

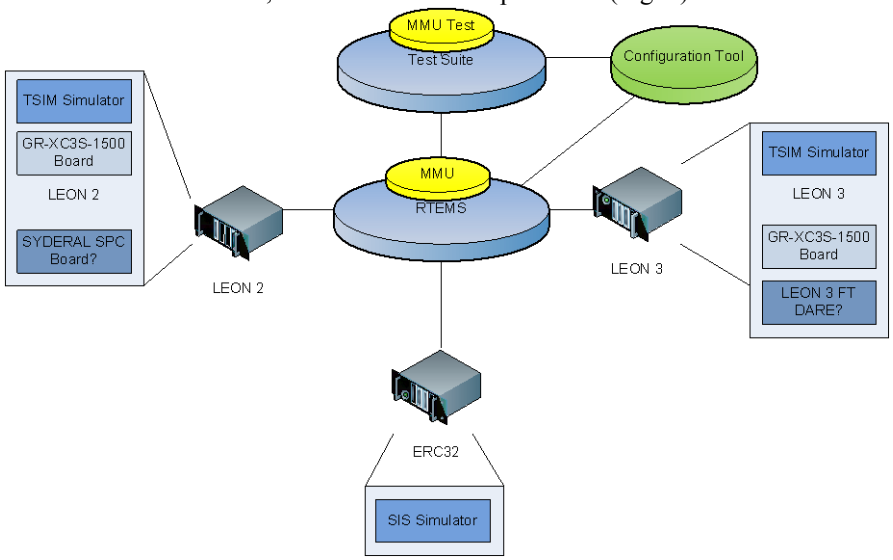

Fig. 7. RTEMS Engineering and Testing Overview

The tailored RTEMS version consists of patches and scripts that, if applied to the original RTEMS source code, will remove the unnecessary managers, files, dead code and bugs. It also adds new files and code, making all necessary code adjustments to produce the RTEMS tailored version (qualifiable). This version intends to achieve the Galileo Software Standards Development Assurance Level (DAL) B requirements. According to the standard, the structural coverage for a DAL-B qualification shall achieve 100% statement and decision coverage for the source code. Based in these requirements, the source code cannot contain dead or unused code. The current coverage of the tests are 86,1% of statement coverage and 3.228 LOC (lines of code) (API, supercore, RTEMS API and Super API) [12]. The original version of RTEMS, counting with comments and headers is around 450.000 LOCs.

In the Statement coverage testing, the code is executed in such a manner that every statement in the code is executed at least once. Branch or Decision Coverage testing helps to validate all branches in the code and also validates that no branching leads to abnormal software behaviour.

At a first phase, the code removal was a very sensible operation, since it included the removal of unselected RTEMS Managers and code shared between RTEMS Managers.

In the current phase of the project the development team is producing unit and integration tests to validate the correctness of RTEMS behaviour.

3.5 RTEMS Budget

Software budget analysis [13] was performed to RTEMS. A comparison between the original RTEMS version and RTEMS tailored was made for all hardware architectures (ERC32, LEON2 and LEON3) of interest. The measurements taken were related with interrupt latency timing, interrupt exit timing, context switch timing, maximum CPU usage, RTEMS API directives timing and memory footprint. The measurements were based in ESA requirements and the intent was to make a comparison between the original RTEMS and the RTEMS Tailored for the most useful characteristics of a real-time operating system. Table 6 presents a sample of context switch analysis budget.

Table 6. Timing Analysis for Context Switch

	Maximum Time (microseconds)					
Section	TargetSimERC32		TargetSimLeon2		TargetSimLeon3	
RTEMS Version	Original	Tailored	Original	Tailored	Original	Tailored
Context Switch without FPU	48	48	19	19	19	19
Context Switch with FPU	68	68	20	20	20	20

3.6 RTEMS Criticality Analysis

SW-FMECA (Software Failure Modes Effects and Criticality Analysis) [14] analysis is a bottom-up technique that identifies modes, causes, effects and criticalities of failures on end item (software) performance and their external interfaces. The SW-FMECA main objective is to perform the DAL assessment of each RTEMS Improvement Component/Sub-Component. The impact of a failure can be characterized by a severity classification. Each failure mode was analysed to estimate the severity level.

The SW-FMECA analysis was performed and it was possible to find recommendations to RTEMS Operating System. The following bullets present the major recommendations found:

- When a "Fatal Error" (in any state) occurs, the RTEMS Operating System, before the User Application starts the system, shall switch to a "SAFER" state (APP_SAFE_STATE). The transition from the "APP_SAFE_STATE" to the "BEFORE_INIT" state shall be done through the Operating System re-initialization (performed by the User Application).
- An Error/Event Handler (Log Manager) to receive all errors from the RTEMS Operating System and HW devices shall be foreseen in the next version of RTEMS Operating System, to improve the management of all errors. The User Application shall access to this "Log Manager" to have the RTEMS Operating System and HW devices errors.
- The Rate Monotonic Manager shall define the deadline for each thread on the System, in order to avoid that:
  - One thread blocks the execution of other threads (low priority threads could not be executed and miss their deadlines).
  - The execution of one faulty thread (running in loop not programmed) uses all the resources of the system.
  - The system enters in degraded mode.
- The use of Dynamic Memory by the "Heap Handler" during the Initialization of the RTEMS components like, semaphores, Threads, Message Queues, timers shall be avoided (Rule 20.4 of [17]).
- RTEMS Operating System shall provide the capability to perform PBIT (Power On Built-In-Tests) when the system is initiated, in the HW devices supported in the scope of the project (Clock device and Processor).

4 Conclusions

This paper provided a brief view of the RTEMS Improvement's project activities. The activities were centred in the acquisition of know-how and recently the facilitation of RTEMS qualification. The qualification process is about to be concluded and a new version of RTEMS will be produced. This new version is based in 4.8.0 of RTEMS. In a recent future EDISOFT will develop the Memory Manager for the RTEMS OS for the LEON architecture Memory Management Unit (MMU). The outputs of the work can be accessed through RTEMS Support Platform [8] and the *qualified* version and tools are distributed as open source free package.

References

1. Constantino, A., Silva, H., Mota, M., Zulianello, M.: RTEMS CENTRE – Support and Maintenance CENTRE to RTEMS Operating System, DAta Systems in Aerospace (2007)
2. Silva, H., Constantino, A., Coutinho, M., Freitas, D., Faustino, S., Mota, M., Colaço, P., Zulianello, M.: RTEMS CENTRE – Support and Maintenance CENTRE to RTEMS Operating System, DAta Systems in Aerospace (2008)
3. Silva, H., Constantino, A., Coutinho, M., Freitas, D., Faustino, S., Mota, M., Colaço, P., Sousa, J., Dias, L., Damjanovic, B., Zulianello, M., Rufino, J.: RTEMS CENTRE – Support and Maintenance CENTRE to RTEMS Operating System, DAta Systems in Aerospace (2009)
4. Silva, H., RTEMS CENTRE Final presentation and Final Report, ESTEC (European Space Research and Technology Centre), Noordwijk - Netherlands (2008)
5. GSWS study team: Galileo Software Standards, GAL-SPE-GLI-SYST-A/0092 (2004)
6. EDISOFT: ESA/ESTEC Contract number 20049/05/NL/JD/jk
7. EDISOFT: ESA/ESTEC Contract number 21141/07/NL/JD
8. RTEMS CENTRE website: http://rtemscentre.edisoft.pt
9. RTEMS website: http://www.rtems.com
10. Constantino, A., Freitas, D., Mota, M., Silva, H.: RTEMS CENTRE Software System Specification, RTEMS CENTRE project (2008)
11. Coutinho, M.: RTEMS Managers Candidate Evaluation Report, RTEMS Improvement project (2009)
12. Coutinho, M.: RTEMS Improvement Generic Test Report, RTEMS Improvement project (2009)
13. Freitas, D.: RTEMS Improvement Software Budget Report, RTEMS Improvement project (2009)
14. Dias, L.: RTEMS Improvement Preliminary Software Criticality Analysis Report, RTEMS Improvement project (2009)
15. Colaço, P., Coutinho, M.: RTEMS Improvement Software Design Document, RTEMS Improvement project (2009)
16. Coutinho, M.: RTEMS Improvement Software Requirements Document, RTEMS Improvement project (2009)
17. MIRA Limited: MISRA-C: 2004 Guidelines for the use of C language in critical systems.