RTEMS CENTRE – SUPPORT AND MAINTENANCE CENTRE TO RTEMS OPERATING SYSTEM

Helder Silva[(1)], **Alexandre Constantino**[(1)], **Daniel Freitas**[(1)] **and Manuel Coutinho**[(1)]
Sérgio Faustino[(1)], **Miguel Mota**[(1)], **Pedro Colaço**[(1)], **José Sousa**[(1)], **Luis Dias**[(1)], **and Bosko Damjanovic**[(1)]
Marco Zulianello[(2)]
José Rufino[(3)]

[(1)]*EDISOFT S.A*
Rua Quinta do Medronheiros, Lazarim, Apartado 382, Monte da Caparica, 2826-801 CAPARICA, Portugal
Email:{Helder.Silva, Alexandre.Constantino, Daniel.Freitas, Manuel.Coutinho, Sergio.Faustino, Miguel Mota, Pedro.Colaço, Jose.Sousa, Luis.Dias and Bosko.Damjanovic} AT Edisoft DOT pt

[(2)] *ESA-ESTEC*
Postbus 299, 2200 AG Noordwijk, The Netherlands
Email:{Marco.Zulianello} AT ESA DOT int

[(3)] FCUL - *Faculdade de Ciências da Universidade de Lisboa*
Campo Grande, 1749-016 Lisboa, Portugal
Email:{Ruf} AT DI DOT FC DOT UL DOT pt

ABSTRACT

RTEMS CENTRE – Support and Maintenance Centre to RTEMS Operating System is a joint ESA/Portuguese Task Force initiative to develop a support and maintenance centre to the Real-Time Executive for Multiprocessor Systems (RTEMS).

This paper gives a high level visibility of the progress, the results obtained and the future work in the RTEMS CENTRE [6] and in the RTEMS Improvement [7] projects.

RTEMS CENTRE started officially in November 2006, with the RTEMS 4.6.99.2 version. A full analysis of RTEMS operating system was produced. The architecture was analysed in terms of conceptual, organizational and operational concepts.

The original objectives [1] of the centre were primarily to create and maintain technical expertise and competences in this RTOS, to develop a website to provide the European Space Community an entry point for obtaining support (http://rtemscentre.edisoft.pt), to design, develop, maintain and integrate some RTEMS support tools (Timeline Tool, Configuration and Management Tools), to maintain flight libraries and Board Support Packages, to develop a strong relationship with the World RTEMS Community and finally to produce some considerations in ARINC-653, DO-178B and ECSS E-40 standards.

RTEMS Improvement is the continuation of the RTEMS CENTRE. Currently the RTEMS, version 4.8.0, is being facilitated for a future qualification. In this work, the validation material is being produced following the Galileo Software Standards Development Assurance Level B [5]. RTEMS is being completely tested, errors analysed, dead and deactivated code removed and tests produced to achieve 100% statement and decision coverage of source code [2]. The SW to exploit the LEON Memory Management Unit (MMU) hardware will be also added. A brief description of the expected implementations will be given.

1 RTEMS Operating System

RTEMS is the Real-Time Operating System for Multiprocessor Systems. It is a full featured Real Time Operating System that supports a variety of open API and interface standards. It provides a high performance environment for embedded applications including multitasking capabilities, homogeneous and heterogeneous multiprocessor systems, event-driven, priority-based, pre-emptive scheduling, optional rate monotonic scheduling, inter-task communication and synchronization, priority inheritance, responsive interrupt management, dynamic memory allocation and high level of user configurability. RTEMS is a free open source Real Time Operating System designed for deeply embedded systems and aim to be competitive with closed source and commercial products. It was developed to support applications with the most inflexible time frames requirements, making it possible for the user to develop hard real time systems. RTEMS is maintained by the On-Line Research Corporation (OAR) [8], albeit several of the features and platform support for it have been developed by RTEMS community. The first version of what is today RTEMS was released in 1988 [9].

2 RTEMS CENTRE

2.1 RTEMS Constraints

The first work produced in RTEMS CENTRE was to identify the RTEMS constraints. They were separated into two areas.

1. Runtime constraints:
   - Reduced scheduling methods implemented;
   - System is unable to read CD-ROM images (ISO 9960 file systems);
   - Inability to create a custom file system;
   - Lack of advanced security features;
   - No support for ARINC 653 (work is being developed [10] by Skysoft [15] and FCUL [16]);
   - No full support for Ada applications;
   - No full support for POSIX applications;
   - GJC (GNU Java Compiler) not working with RTEMS.

2. Development constraints:
   - No IDE incorporated in the RTEMS package (eclipse could be a candidate);
   - No polished tool for monitoring the system or the applications that run in it (RTEMS CENTRE develop such a tool, see section below);
   - No timeline visualization tool (RTEMS CENTRE develop such a tool, see section below);
   - No certification with DO-178B and ECSS-E40 standards (RTEMS Improvement is facilitating such a qualification for Galileo Software Standards);
   - No installation/configuration tool (RTEMS CENTRE develop such a tool, see section below).

2.2 RTEMS Architecture

From an operational point of view RTEMS can be described as a set of libraries that are with the application. The installation process of RTEMS produces several system libraries, in which most of them end up wrapped in two main ones:

- *librtemsbsp.a*: includes the processor, the board and the peripheral devices dependent code. The source code files are located in the ${RTEMS_SRC }/c/src directory;
- *librtemscpu.a*: includes the kernel itself and various multi platform support libraries; there is however an small part of the code that is target dependent. This library can be mapped to the ${RTEMS_SRC}/cpukit directory.

The RTEMS organization is structured to meet the following requirements:

- Encourage development of modular components;
- Isolate processor and target dependent code, while allowing as much common source code as possible to be shared across multiple processors and target boards;
- Allow multiple RTEMS users to perform simultaneous compilation of RTEMS and its support facilities for different processors and targets.
-

On a conceptual level RTEMS can be characterized by three layers: hardware support, kernel and APIs. The user then develops the application by using the available APIs. The next image illustrates the perspective of the RTEMS conceptual architecture.

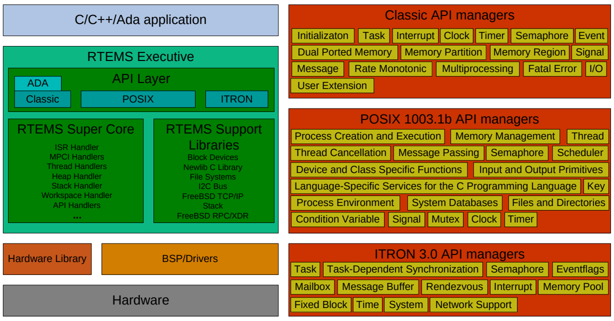


Figure 1: RTEMS Conceptual Architecture

The RTEMS Classical Application API is divided into Managers. We have performed a deeper analysis of each manager. Additionally, we have performed the reverse engineering of each manager. The RTEMS SuperCore handlers study included the analysis of functions, variables, headers, implementations, inlines and macros. The following figure presents a sample of the full reverse engineering performed to the RTEMS SuperCore (in this case the class diagram of Core Semaphore Handler).

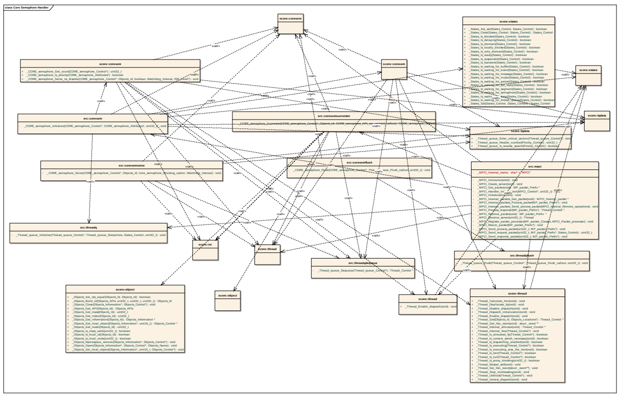


Figure 2: Core Semaphore Handler Class Diagram

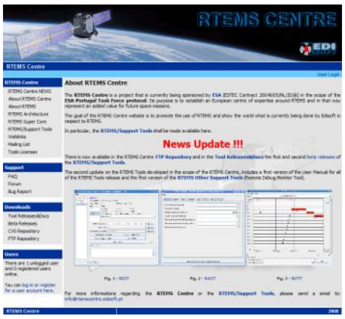
The POSIX layer was also analysed and cross-checked with the POSIX standard [11]. The following items summarize the POSIX features partially available (RTEMS 4.6.99.3):

- Process primitives family: this is due to the fact that RTEMS supports a single process, multithreaded POSIX environment. Moreover, RTEMS supports a number of POSIX process, user, and group oriented routines in a "SUSP" (Single-User, Single Process) manner;
- dup(): the current implementation is insufficient;
- pipe(): the current implementation is a dummy;
- mkfifo(): the current implementation is untested because none of the available filesystems support FIFOs;
- asynchronous I/O family;
- flockfile() family;
- getc/putc unlocked family;
- shared memory: it is unclear what level of support is appropriate and possible for RTEMS;
- mapped memory: it is unclear what level of support is appropriate and possible for RTEMS;
- functional testing for the POSIX functionality implemented in Newlib. There is functional testing only for the functionality implemented in RTEMS.

RTEMS classifies target dependent code based upon its dependencies into one of the following categories: CPU dependent, board dependent and peripheral dependent. The CPU dependent code is organized into executive files and support files. The board dependent code encompasses the BSP and associated Device Drivers and provides the most specific glue between RTEMS and a particular board. The peripheral dependent class of code can be described as a collection of reusable peripheral device drivers which can be tailored easily to a particular board. The source code can be found in ${RTEMS_SRC}/cpukit/score/cpu, ${RTEMS_SRC}/c/src/lib/libcpu, ${RTEMS_SRC}/c/src/lib/libbsp and ${RTEMS_SRC}/c/src/lib/libchip.

2.3 Support Site

RTEMS CENTRE Support Platform (http://rtemscentre.edisoft.pt) [4] was developed in order to complement the RTEMS official website. The aim of such platform is to make public the technical know-how and developments produced of the RTEMS CENTRE.



RTEMS CENTRE

RTEMS Centre

About RTEMS Centre

The **RTEMS Centre** is a project that is currently being sponsored by **ESA** (ESTEC Contract 20048/05/NL/JD(sk)) in the scope of the **ESA-Portugal Task Force protocol**. Its purpose is to establish an European centre of expertise around RTEMS and in that way represent an added value for future space missions.

The goal of the RTEMS Centre website is to promote the use of RTEMS and show the world what is currently being done by Edisoft in respect to RTEMS.

In particular, the **RTEMS/Support Tools** shall be made available here.

News Update !!!

There is now available in the RTEMS Centre **FTP Repository** and in the **Tool Releases&Docs** the first and second beta releases of the **RTEMS/Support Tools**.

The second update on the RTEMS Tools developed in the scope of the RTEMS Centre, includes a first version of the User Manual for all of the RTEMS Tools release and the first version of the **RTEMS Other Support Tools** (Remote Debug Monitor Tool).

Fig. 1 - RCIT   Fig. 2 - RACT   Fig. 3 - RCTT

For more informations regarding the **RTEMS Centre** or the **RTEMS/Support Tools**, please send a email to: info@rtemscentre.edisoft.pt

Figure 3: RTEMS CENTRE Support Site

The website includes:

- A mailing lists for technical support;
- User forums for extended/dedicated technical support and debate of ideas
- FAQs for the frequently asked questions;
- Software repository for the software produced;
- Bug reporting tools and links for the RTEMS community websites.

2.4 Timeline Tool

The RTEMS Timeline Tool aims to support the application developer by giving important information of the system runtime context. It monitors data into the following categories:

- Execution Time;
- Stack and memory usage;
- RTEMS API calls;
- Configuration Tables;
- Interrupt Generation.

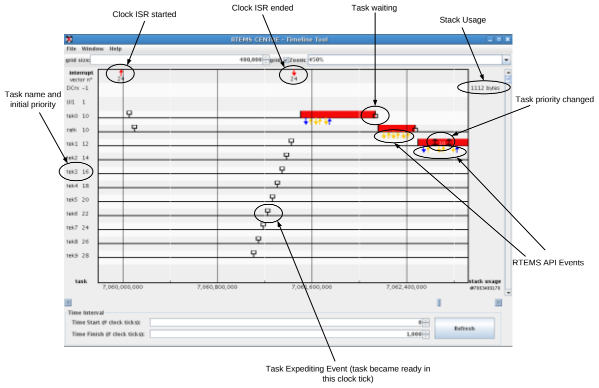
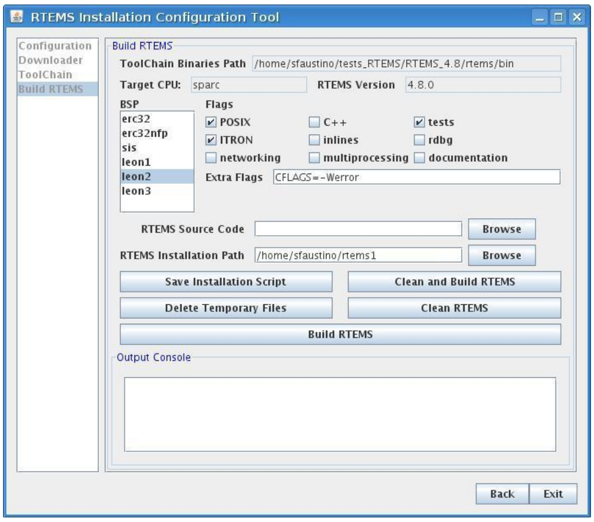


Figure 4: Timeline Tool

2.5 RTEMS Configuration and Installation Tool

The RTEMS Configuration and Installation Tool is a graphical application that is accessible by either the command line or the Eclipse IDE. The user interacts with the application to set options and parameters of RTEMS. At the end of the configuration process, the application uses the provided values to deploy the RTEMS toolchain.



ToolChain Binaries Path: /home/sfaustino/tests_RTEMS/RTEMS_4.8/rtems/bin

Target CPU: sparc

RTEMS Version: 4.8.0

BSP: erc32, erc32nfp, sis, leon1, leon2 (selected), leon3

Flags:
☑ POSIX
□ C++
☑ tests
☑ ITRON
□ inlines
□ rdbg
□ networking
□ multiprocessing
□ documentation

Extra Flags: CFLAGS=-Werror

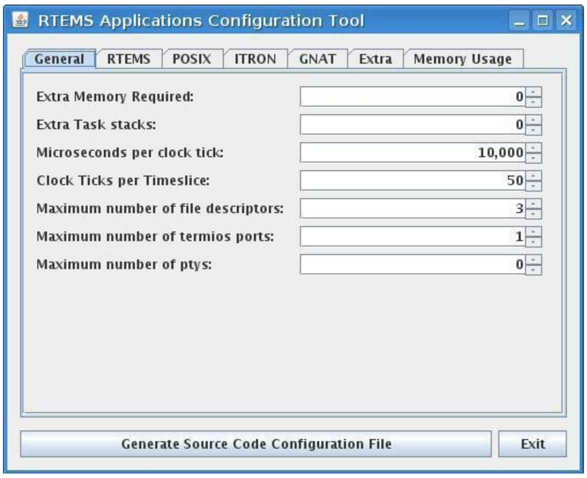
RTEMS Source Code:

RTEMS Installation Path: /home/sfaustino/rtems1

Figure 5: RTEMS Configuration and Installation Tool

2.6 RTEMS Application Configuration Tool

The RTEMS Applications Configuration Tool is a graphical application that is accessible by either the command line or the Eclipse IDE. The graphical interface interacts with the user to obtain the necessary application dependent configurable parameters (e.g. number of tasks, etc). The configuration component generates a C source file containing the configured parameters.


RTEMS Applications Configuration Tool
General
RTEMS
POSIX
ITRON
GNAT
Extra
Memory Usage
Generate Source Code Configuration File
Exit

Extra Memory Required: 0

Extra Task stacks: 0

Microseconds per clock tick: 10,000

Clock Ticks per Timeslice: 50

Maximum number of file descriptors: 3

Maximum number of termios ports: 1

Maximum number of ptys: 0

Figure 6: RTEMS Application Configuration Tool

3 RTEMS IMPROVEMENT

The RTEMS Improvement aims to increase quality of RTEMS and documentation in order to facilitate, with a small effort, the qualification for a defined space mission. The work is being developed with LEON2, LEON3 and ERC32 processors. A secondary objective is to provide the Memory Management Unit (MMU) support.

3.1 RTEMS Facilitation Qualification

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

Surveys [12] were performed near European Space users (SAAB [17], OHB [18] and ESA [19]) and along with the EDISOFT assessment, candidate RTEMS managers were selected. The following table provides the list of the managers currently being used by the some space projects by SAAB and OHB. This information provides RTEMS Improvement guides for the facilitation of qualification to be done.

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

Figure 7: Space Users Survey Results

Based on 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. The following table displays the results taken by EDISOFT team.

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

Table 1: RTEMS Managers Evaluation Report

The original version of RTEMS was truncated and several files were removed because of two major reasons, because they were considered unnecessary and because they consist of 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.

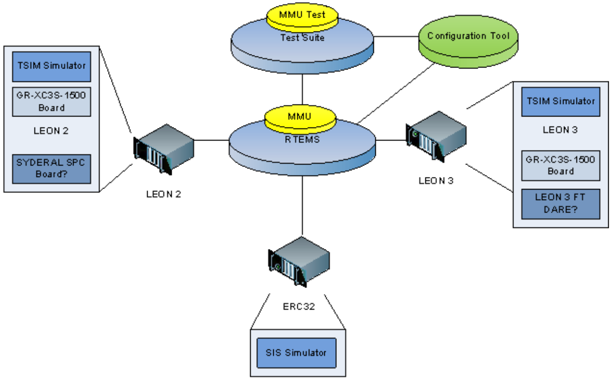


Figure 8: RTEMS Improvement Overview

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.

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 behavior.

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 tests to validate the correctness of RTEMS behavior. It also assesses code coverage and decision coverage of the code and check the robustness.

To comply with Galileo Software Standards, a set of documentation was produced for RTEMS to fulfill the validation requirements, like for example, the complete reverse engineer of RTEMS version 4.8.0. The following items show the set of documentation produced for the facilitation of qualification of the RTEMS operating system:

- RTEMS managers candidate evaluation report;
- Software Requirement Document;
- User Manual and Design Notes;
- Software Budget Report;
- Procured Software Justification File;
- Software Design Document;
- Software Integration Test Plan;
- Software Unit Test Plan;
- Software Validation Testing Specification;
- RTEMS Tailoring Plan;
- Generic Test Report;
- RTEMS Test Suite;
- Acceptance Test Plan;
- Software Maintenance Plan;
- Installation Report;
- Software Acceptance Data Package;
- RTEMS Tailored (the new version);
- SOC with GSWS;
- Software Criticality Analysis Report.

The work described above has been performed in a way that would permit a faster adaptation to future RTEMS versions (v4.9 and above).

3.2 RTEMS Memory Management Unit

In the native RTEMS implementation, a task, possibly exhibiting a faulty behavior, is not prevented from accessing (and writing) the address space of other tasks or even accessing the RTEMS kernel itself. To prevent the uncontrolled access to private or privileged address spaces, the use of memory protection mechanisms is mandatory. To be effective, these mechanisms should be supported directly by hardware address validation mechanisms, typically implemented in a Memory Management Unit (MMU).

The Memory Management Unit described in the SPARC V8 specification [13] and optionally implemented in the SPARC LEON architectures makes use of a three-level paging scheme for translating a 32-bit virtual page address into 36-bit physical page address. The page size is 4 KiB. The operation of this mechanism, to be used in the improvement of RTEMS, is illustrated in the next figure.

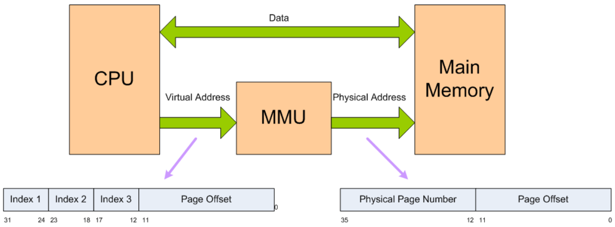


Figure 9 – The SPARC V8 MMU Architecture

The MMU Page Table Descriptors/Entries will be used in order to locate RTEMS application tasks and kernel space into the physical memory and to:

- Enable access to the data section of a task (or set of tasks sharing a given address space);
- Enable access to the stack section of each task;
- Enable the access to the kernel address space upon issuing of RTEMS system calls.
- Prevent access to kernel addressing spaces upon exiting of a system call;
- Prevent access to the data and stack sections of non-running tasks.

This implies that Page Table Entries need to be updated when a system call is invoked and upon a task switch. All the relevant (i.e. both accessible and non-accessible) address spaces need to be specified at each time in order to secure continuous protection of the entire physical memory. Each virtual section (e.g. task code, data, and stack) needs to be aligned to the boundary specified by the corresponding Page Table Entry. Memory alignment can be enforced simply by making use of a specialized parameterization of the standard GNU build tools [14]. No need to develop specific tools for this purpose is envisaged.

4 CONCLUSIONS

This paper provided a brief view of the RTEMS CENTRE's activities, the current technical expertise of the team and the developments expected to be taken in the RTEMS Improvement. The activities were centred in the acquisition of know-how, the development of RTEMS supporting tools and recently the facilitation of RTEMS qualification. The future will reserve us the development of the Memory Management Unit (MMU) for the RTEMS OS. The outputs of the work can be accessed through RTEMS Support Platform (http://rtemscentre.edisoft.pt).

5 ACKNOWLEDGMENTS

The RTEMS CENTRE team would like to acknowledge the ESA-Portugal Task Force, OAR Company, SAAB, OHB, SciSys [20], FCUL and RTEMS community for the remarkable comments and support to yours RTEMS CENTRE.

6 REFERENCES

[1] Silva, *RTEMS CENTRE Final Presentation.*
[2] GSWS Study Team, *Galileo Software Standards.*
[3] Cardoso, Faustino, Coutinho, Freitas, Mota, Constantino, Silva, *RTEMS CENTRE Final Report.*
[4] RTEMS CENTRE website, http://rtemscentre.edisoft.pt.
[5] SOC to GSWS
[6] ESA/ESTEC Contract number 20049/05/NL/JD/jk
[7] ESA/ESTEC Contract number 21141/07/NL/JD
[8] On-Line Research Corporation website, http://www.oarcorp.com.
[9] RTEMS website, http://www.rtems.com.
[10] AIR2 Project Contract number 21217/07/NL/CB
[11] Constantino, Freitas, Mota and Silva, RTEMS CENTRE Software System Specification.
[12] RTEMS Improvement RTEMS managers candidate evaluation report.
[13] SPARC International, The SPARC Architecture Manual Version 8.
[14] GNU website, www.gnu.org.
[15] Skysoft website, www.skysoft.pt.
[16] FCUL website, http://www.fc.ul.pt/.
[17] SAAB website, http://www.saab.com.
[18] OHB website, http://www.ohb-system.de.
[19] ESA website, http://www.esa.int.
[20] SciSys website, http://www.scisys.co.uk/.