

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 cancel
	rtems task variable add		rtems rate monotonic delet
	rtems task variable get		rtems rate monotonic perio
	rtems_task_variable_delete		rtems_rate_monotonic_get_s tatus
	Interrupt	All directives	User Extensions

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).

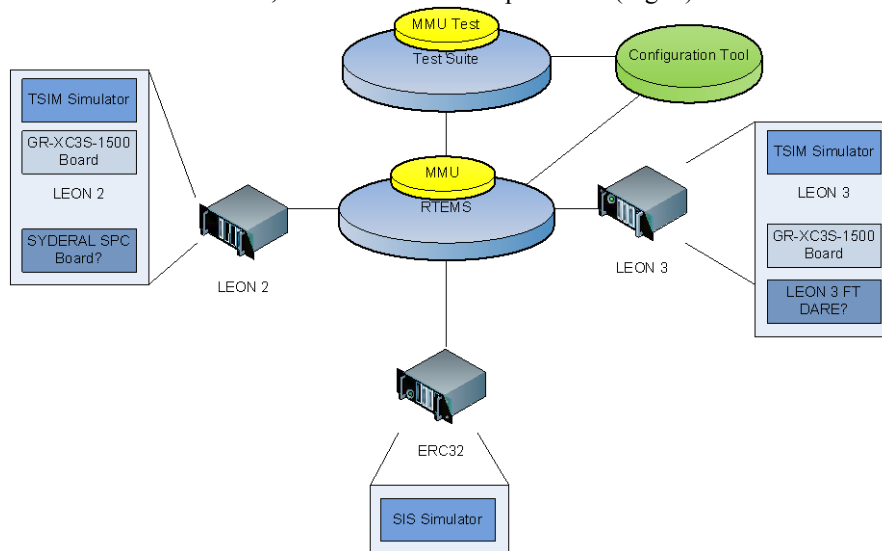


Fig. 7. RTEMS Engineering and Testing Overview

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, DAData 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, DAData 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, DAData 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.