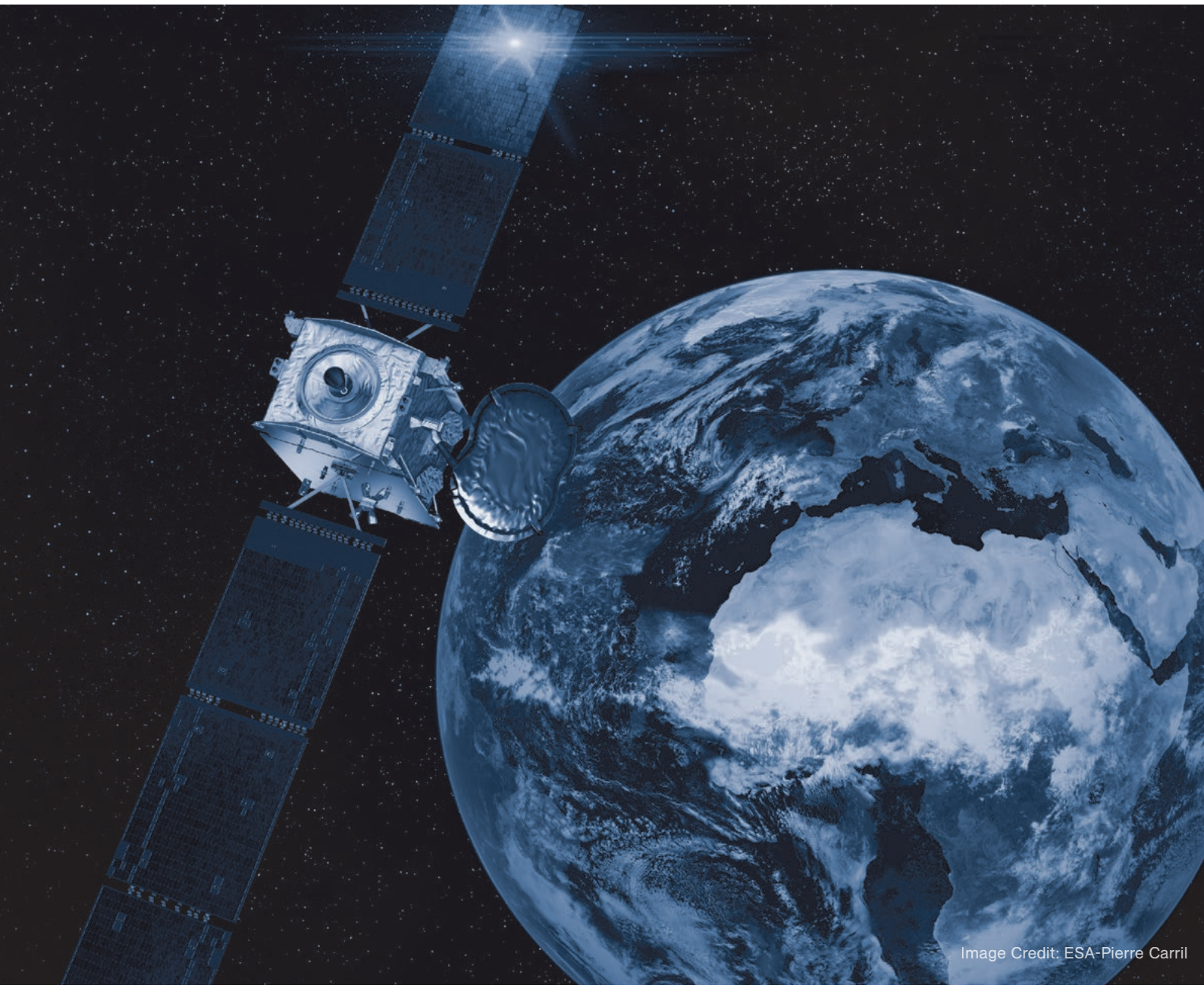


RTEMS

Information Brochure

Table of contents

RTEMS in a nutshell 1
Why RTEMS? 2
Why Open Source? 3
RTEMS features 3
RTEMS SMP features 6
ECSS Qualification 7
Our RTEMS support service 8
embedded brains driven RTEMS activities 9
embedded brains‘ cooperation partners 10

RTEMS in a nutshell

Open Source

- Code transparency
- Independent in use
- No royalities

Safety Qualifiable

- ECSS Space qualified (Cat.C, tailored Cat.B)
- Automated Test Suite
- 100% code and branch coverage

Well established

- Continuously developed for over 30 years
- Broad range of BSPs, interfaces and drivers
- Used in various industries

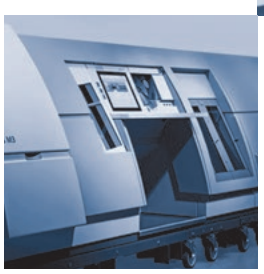
Multicore Performance

- Symmetrical Multiprocessing (SMP) using 2 to 24 cores
- High performance
- OS operating with less than 100KB of memory

NASA Perseverance Mars Rover

E&K Automated Guided Vehicles

BMW high speed data logger



G+D high-performance banknote processing system

Why RTEMS

RTEMS is a commercial grade Open-Source „hard“ real-time OS with high flexibility, permitting minimal resource demands and maximum performance, particularly on small and medium-size systems. It is available for a broad range of processors and provides all common interfaces and drivers for embedded requirements. Originally designed more than 25 years ago for military purposes, the single-core version was replaced with a SMP multi-core-version (using Symmetric Multiprocessing) in 2015.

- Minimal resource demand
- A safety-qualified multicore realtime OS that can be operated with 50KB of memory
- BSPs are available for most 16-, 32- and 64-bit embedded processors (incl 12/24-core QorIQ T4240 and RISC-V architecture)
- Advanced features: POSIX and Classic API interface, OpenMP support, C11/C++11 threading and synchronization support, Flattened Device Tree (FDT) support and 10 Gbit/s Ethernet code quality
- The SMP core code and selected features are safety qualified for the Space Domain
- Independency, sustainability, maintenance and compatibility provided by a long-standing user community, while users enjoy freedom of choice and liberal license conditions without royalties

Why choose carefully?

There are numerous RTOS available. Apart from the decision between commercial and open-source software you should consider further aspects.

- Choose the right „sized“ OS: A simple OS will be fast to get into but may reach its limits soon. This often becomes a blocking point in a progressed state of a project. Choosing a very extensive OS may avoid this problem but will require many years of expertise (and possibly consume excessive system resources).
  Hence, a scalable OS with many options may be the best choice. For safety critical applications it is important to consider that the qualification effort increases exponentially with code complexity. In case software packages need to be qualified it is vital to keep the code size as small as possible.

- Consider your time investment: Getting highly experienced with an OS requires a lot of time. A scalable OS that can be used for many different projects is way more efficient compared to using a best-fit OS for each individual project. This is true for RTEMS that can be assembled for tiny systems with some 10kB of RAM as well as for massive 24-core systems with 10GB Ethernet etc.

Why Open Source

Advantages

- No royalties
- Getting easily started without paperwork
- 100% transparent source code for debugging and interfacing
- Product support alternatively provided by supportive community or commercial providers
- No dependency on a supplier‘s business model
- No forced updates driven by a supplier

Implications

- The roadmap is dependent on software development volunteers, hence it may become difficult to address strategic investments and due dates
- No active marketing is provided, hence customer contact and product information is not as intrusive as for commercial software
- License conditions may imply restrictions for own source code and for re-use of shared code

Challenges

- The effort required for creating new functions or interfaces is widely underestimated, in particular for complex software packages
- The creation of a „private“ version will become outdated as the public code will evolve over time. Re-merging the private code version into the public code will become increasingly expensive
- The effort needed for code maintenance of the community version is often underestimated, in particular for large software packages with dozens of patches being published daily

Benefits of professional support

- Minimization of risks and optimization of maintenance support
- Synchronization with recent fixes and improvements of the main line
- Focus on own development

RTEMS features

Supported Architectures

- ARMv7-ARM (with SMP support)
- Xilinx Zynq
- Altera/Intel Cyclone/Arria
- NXP i.MX7
- STMicroelectronics STM32
- NXP LPC
- Atmel/Microchip SAM E70/S70/V70/V71
- Raspberry Pi
- Texas Instruments TMS570
- ARMv8-AR (with SMP support)

- Xilinx UltraScale+
- PowerPC 32 and 64 bit (with SMP support)
- NXP QorIQ and many more
- SPARC/LEON (with SMP support)
- Gaisler, for example GR712RC and GR740 (with SMP support)
- RISC-V 32 and 64 bit (with SMP support)
- Xilinx MirocBlaze
- Altera/Intel Nios II
- and several legacy architectures

Features

- OpenMP (via the GCC provided libgomp)
- LibBSD provides networking stack, USB, SD, WLAN, IPSec support as well as additional drivers
- Dynamic Host Configuration Protocol (DHCP) daemon
- File Transfer Protocol (FTP) as file system client and daemon
- Trivial File Transfer Protocol (TFTP) file system client
- Telnet daemon
- PCIe
- NVMe
- Very small memory footprint (20-100KB for core services)
- Peripheral Component Interconnect (PCI) bus support
- Thread synchronization and communication

File systems

- IMFS
- FAT
- RFS
- NFSv2
- JFFS2 (NOR flashes)
- YAFFS2 (NAND flashes, GPL or commercial license required)

Device Drivers

- Termios (serial interfaces)
- I2C (Linux user-space API compatible)
- SPI (Linux user-space API compatible)
- Network stacks (legacy, libbsd, lwIP)
- USB stack (libbsd)
- SD/MMC card stack (libbsd)
- Framebuffer (Linux user-space API compatible, Qt)

Thread synchronization and communication

- Mutexes with and without locking protocols
- Counting semaphores
- Binary semaphores
- Condition variables
- Events
- Message queues
- Barriers
- Futex (used by OpenMP barriers)
- Epoch Based Reclamation (libbsd)

Locking Protocols

- Immediate Ceiling Priority Protocol (ICPP)
- Priority Inheritance Protocol
- Multiprocessor Resource Sharing Protocol (MrsP)
- O(m) Independence-Preserving Protocol (OMIP)

Uniprocessor Schedulers

- Deterministic Priority Scheduler
- Simple Priority Scheduler
- Earliest Deadline First Scheduler
- Constant Bandwidth Server Scheduling (CBS)

Clustered Schedulers (SMP feature)

- Flexible link-time configuration
- Job-level fixed-priority scheduler (EDF) with support for one-to-one and one-to-all thread to processor affinities (default SMP scheduler)
- Fixed Priority SMP Scheduler
- Arbitrary Processor Affinity Priority SMP Scheduler (proof of concept)

Supported Languages

- ADA (not on lm32 nor sh)
- C (GCC) / C++ (GNU C++)
- Erlang
- Fortran (not on i386, lm32, m68k, mips, riscv, sh, sparc nor sparc64)
- OpenMP 4.5
- Python and MicroPython

API Support

- RTEMS Classic API
- POSIX API
- High Performance API

RTEMS Classic API

- Chains
- Tasks
- Semaphores
- Message Queues
- Events
- Barriers
- Interrupts
- Red-Black Trees
- Signals
- Time/Clock
- Timers
- Rate Monotonic Periods
- Fixed Allocation Memory Pools
- Variable Allocation Memory Pools
- Various Locking Protocols
- Various Schedulers

Development Platforms

- GNU/Linux distributions
- FreeBSD and NetBSD
- Windows
- Mac OS

RTEMS SMP features

SMP machines consist of a set of processors (players) attached to a common memory (field).

Clustered Scheduling

- Independent scheduler instances for processor subsets (cache topology)
- Flexible link-time configuration
- Fixed-priority scheduler
- Job-level fixed-priority scheduler (EDF)

Locking Protocols for Mutual Exclusion

- Fine grained locking (Big Kernel Lock removed)
- Transitive priority inheritance
- O(m) Independence-Preserving Protocol (OMIP)
- Priority ceiling
- Multiprocessor Resource-Sharing Protocol (MrsP)

Lock-Free Timestamps

- Scalable timer implementation e.g. based on red-black trees
- Thread operation timeouts use current processor
- Timer use dedicated processor set during timer creation
- Each processor has its own data set to maintain timers

BSPs supporting SMP

- SPARC (1 to 4 cores): GR712C and GR740
- PowerPC (1 to 24 cores): QorIQ (e.g. P1020, P2020, T2080, T4240)
- ARMv7-A (1 to 4 cores): Altera Cyclone V, Xilinx Zynq, Raspberry Pi2
- RISC-V

ECSS Qualification

Qualification of Open Source software

RTEMS code is Open- Source, maintained by a community of volunteers:
- many configurations
- many patches / updates

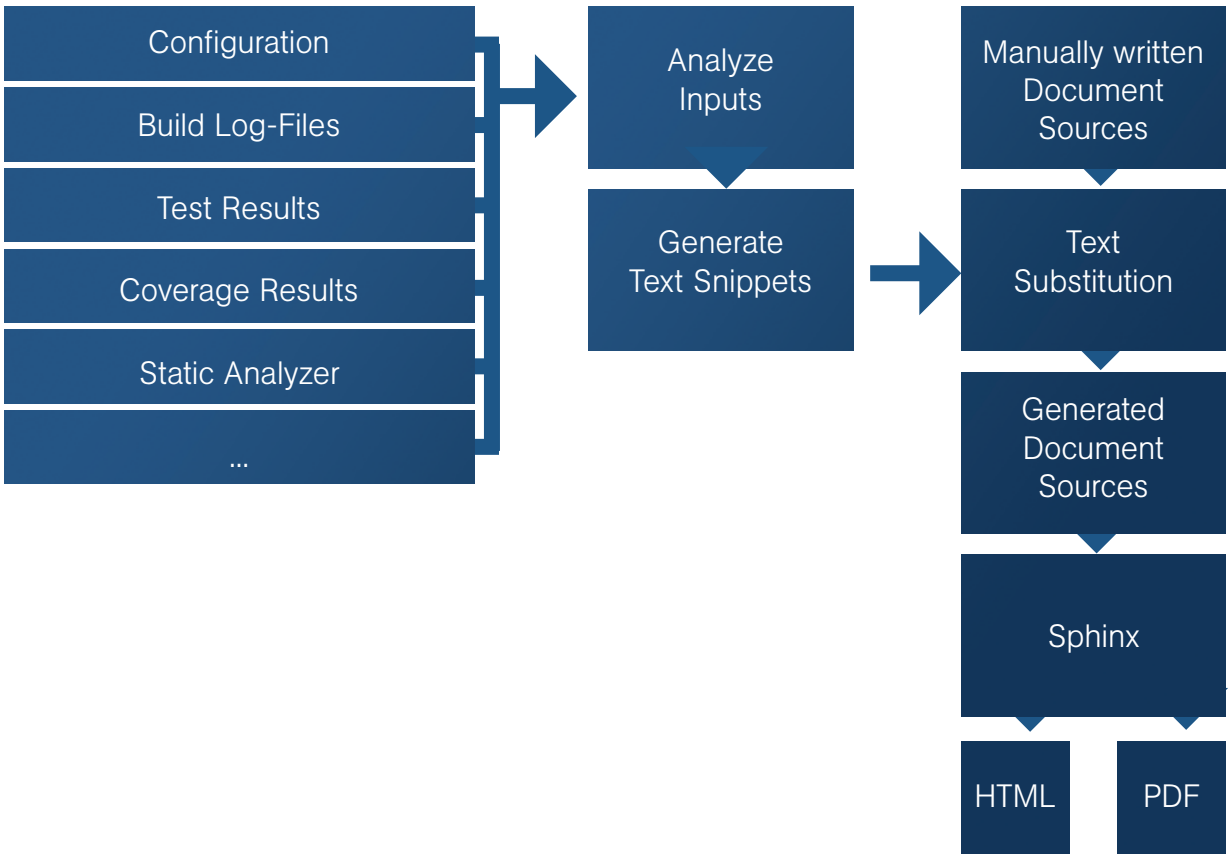
=> very dynamic

Qualification is a labor-intensive process to be integrally performed for a set software package.

=> very static

Re-qualification is required for any code change
- Most flexible and efficient qualification process
- Automated testing and document generation (as far as feasible)
- Code for qualification is based on RTEMS main line

Automated Document generation



Used Technologies

- Debian 10 – Build and usage environment for QDPs
- Python 3 – For Qualification Tool Chain and all scripts
- Pytest – Test framework for Python code
- C, Assember and WAF as build system – To build RTEMS from sources
- GCC tools – Development tools (compiler, linker, …)
- RTEMS Source Builder – To build cross compiler, linker, etc.
- RTEMS Test Framework – For unit-, validation- and performance tests
- YAML – For configuration files and other input data
- Doxygen – For source code documentation and “tags” as input for traceability
- Sphinx & LaTeX – Documentation generation
- Git/GitLab – Source code management
- Coverity, CppCheck, CLANG – Static code analyzers
- GCOV/GCOVR – Coverage analyzer
- Metrix++ – Code metrics analyzer

Verification activities performed

- Static code checker results form Coverity, CLANG, and CppCheck
- Product and process metrics
- Assessment of testing and validation activities
- Problem and none-conformance reports
- Code & branch coverage 100%

→ ECSS Pre-Qualification for Cat. C using API Subset according to „Space Profile“ (Cat. B pending)

Our RTEMS support services

Working with RTEMS since 1997 we provide efficient and professional services to our customers. Using our deep experience with RTEMS will save you time and help you focussing on your development.

Training

- RTEMS Application Development
- RTEMS Symmetrical Multiprocessing (SMP)
- RTEMS Safety Qualification

Software Engineering (fixed price)

- Board-Support-Packages (BSPs)
- Drivers
- Implementation of compilers and developments tools
- Modules and application software

Support Packages (hourly based)

- Answer to upcoming questions,
- Identify bugs and work out patches,
- Help out with tooling and configuration,
- Support programming, particularly for RTEMS interfacing

Qualification Data Packages for different processors

- Gaisler GR712RC and Gr740 (SPARC V8)
- DAHLIA (ARM Cortex-R52)
- Xilinx Zynq (ARM Cortex-A9)
- Gaisler Noel-V (RISC-V)
- Gaisler LEON5 (SPARC V8)

Extensions of Qualification Data Packages

- POSIX API (mutex, semaphore, condition variables, threads, message queue)
- OpenMP
- lwIP
- High-level Device Drivers (e.g. CAN, SpaceWire, MIL-STD-1553)
- NASA cFS
- Scheduling Tools (Tracing, Analysis)

Qualification Support for application software

- Testing and preparing qualification documents for ECSS

Independent Software Verification and Validation (ISVV)

- Required for ECSS Cat. B and Cat. A qualification

embedded brains driven RTEMS activities

- 1995: first system development based on RTEMS
- 2005: first RTEMS class in munich
- Since 2005: Adaption of RTEMS to many architectures and controllers
- Since 2006: member of the RTEMS steering committee
- 2012: Integration of USB support
- 2014: Integration of improved network stack with IPv6 support etc.
- 2015: Development of RTEMS SMP support (for ESA multicore SPARC)
- 2017: Hypervisor concepts and testing for space industry
- 2021: Space qualification for RTEMS SMP (ECSS)

embedded brains cooperation partners

embedded brains GmbH & Co. KG
Dornierstr. 4
82178 Puchheim
Germany

+49-(0)89-189 47 41-00
rtems@embedded-brains.de
www.embedded-brains.de