

MaRTE-OS: Minimal Real-Time Operating System for Embedded Applications

FOSDEM 2009 Ada Developer Room

Miguel Telleria de Esteban
Daniel Sangorrin

Universidad de Cantabria – Computadores y Tiempo Real
<http://www.ctr.unican.es>

8 Feb 2008

Talk sponsored by the FRESCOR project (FP6/2005/IST/5-034026)
funded in part by the European Union's Sixth Framework Programme.
<http://www.frescor.org>

Copyright notice



This document, *MaRTE-OS: Minimal Real-Time Operating System for Embedded Applications* by Miguel Telleria and Daniel Sangorrin from *Computadores y Tiempo Real, Univ. de Cantabria* ^a is licensed under a **Creative Commons Attribution Sharealike 3.0 license** ^b.

^a<http://www.ctr.unican.es>

^b<http://creativecommons.org/licenses/by-sa/3.0/legalcode/>

This means that **you are free**:

- **To share**, to copy, distribute and transmit the work.
- **To remix**, to adapt the work (even for commercial purposes).

Under the following **conditions**:

- **Attribution** You must give the original author credit.
- **ShareAlike** If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

Any of the above conditions can be waived if you get permission from the copyright holder.

Original editable \LaTeX content should be available from the same source of the binary document or from the copyright holder possibly upon request.

About us and MaRTE-OS

The presents...

Miguel Telleria

- **User** of MaRTE-OS since 2006
- Devel of FRESCOR code for CPU scheduling... in C
- Firm Adaist defender since 1998

Daniel Sangorrín

- **User and devel** of MaRTE-OS since 2004 in Ada and C
- **Maintainer** of MaRTE-OS drivers and the web
- Implementor of FRESCOR code for Network scheduling in Ada
- Firm Adaist defender since 2003

... and the absents:

Mario Aldea Originator and main maintainer of MaRTE-OS

Michael González Harbour Leader of our lab for MaRTE-OS project

Outline

- 1 What is MaRTE OS
 - MaRTE-OS features and architecture
 - MaRTE-OS features and architecture
- 2 Real Time systems
 - Real Time generalities
 - Minimal Real Time POSIX 13 Subset
 - Ada 2005 Annex D. Services
 - Application Defined Scheduling
- 3 FRESCOR and flexible scheduling

Outline

- 1 What is MaRTE OS
 - MaRTE-OS features and architecture
 - MaRTE-OS features and architecture
- 2 Real Time systems
 - Real Time generalities
 - Minimal Real Time POSIX 13 Subset
 - Ada 2005 Annex D. Services
 - Application Defined Scheduling
- 3 FRESCOR and flexible scheduling

MaRTE is an Operating System

Like any other OS...It manages tasks, devices and memory

- Schedules tasks.
- Manages memory.
- Handles I/O and interrupts.

... somethings make it special ...

- Real time oriented: Time predictability.
- Has nice scheduling features.
- Easy to deploy in embedded systems.
- It is written in Ada (runtime exception checking).

MaRTE is an Operating System

Like any other OS...It manages tasks, devices and memory

- Schedules tasks.
- Manages memory.
- Handles I/O and interrupts.

... somethings make it special ...

- Real time oriented: Time predictability.
- Has nice scheduling features.
- Easy to deploy in embedded systems.
- It is written in Ada (runtime exception checking).

History and evolution

Developed at CTR, University of Cantabria

- Main maintainer: Mario Aldea (this was his Ph.D.)
- First presented in Ada Europe 2001 (Leuven)
- Contributions from other institutions: AdaCore (GNAT runtime), Valencia (TLSF), Zaragoza (Wifi and industrial drivers), York (jRate)
- Releases available (GPL) at <http://marte.unican.es>

Initial academic objective

- Provide students a Free and simple Real Time Embedded OS
- Playground for our research (scheduling policies, Ada & POSIX standards...)

Now used in industrial applications

- Welding robot
- GPS receiver

History and evolution

Developed at CTR, University of Cantabria

- Main maintainer: Mario Aldea (this was his Ph.D.)
- First presented in Ada Europe 2001 (Leuven)
- Contributions from other institutions: AdaCore (GNAT runtime), Valencia (TLSF), Zaragoza (Wifi and industrial drivers), York (jRate)
- Releases available (GPL) at <http://marte.unican.es>

Initial academic objective

- Provide students a Free and simple Real Time Embedded OS
- Playground for our research (scheduling policies, Ada & POSIX standards...)

Now used in industrial applications

- Welding robot
- GPS receiver

History and evolution

Developed at CTR, University of Cantabria

- Main maintainer: Mario Aldea (this was his Ph.D.)
- First presented in Ada Europe 2001 (Leuven)
- Contributions from other institutions: AdaCore (GNAT runtime), Valencia (TLSF), Zaragoza (Wifi and industrial drivers), York (jRate)
- Releases available (GPL) at <http://marte.unican.es>

Initial academic objective

- Provide students a Free and simple Real Time Embedded OS
- Playground for our research (scheduling policies, Ada & POSIX standards...)

Now used in industrial applications

- Welding robot
- GPS receiver

Characteristics

Follows the Minimal Real-Time POSIX.13 subset

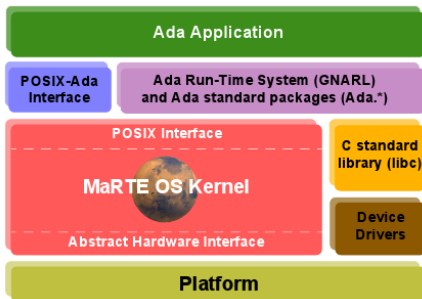
- Concurrency at thread level (all program is a single process)
- Single memory space (threads, driver and OS)
- Static link: Output is a single bootable image. No filesystem required ^a

^aAn experimental FAT filesystem is available

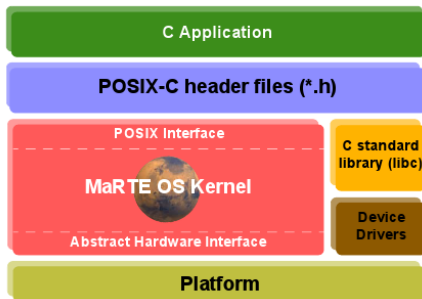
(almost) unique features

- Implements new Ada 2005 Annex D (Real Time) services.
- Implements Application Defined Scheduling proposal.

Architecture



Ada application running on
MaRTE-OS



C application running on
MaRTE-OS

Three modes

- x86_arch**
 - Standalone bootable image.
 - Access hardware with MaRTE drivers.
 - Provides full real time behaviour.
 - Debug through serial console or ethernet traces.
- linux_arch**
 - Statically linked executable file on Linux.
 - Threads use MaRTE-OS scheduler.
 - Linked exclusively with MaRTE-OS libs.
 - Functional testbench debuggable with GDB
- linux_lib**
 - Dynamically linked executable file on Linux.
 - Threads use MaRTE-OS scheduler.
 - Linked with the system's glibc (and other external libs).
 - Can use Linux filesystem.
 - Way to use scheduling features of MaRTE on Linux.

Architecture (cont'd)

GNAT run-time library has been adapted for MaRTE-OS. New RTS:marteuc

- Exceptions, run-time checks etc are handled by GNARL the usual way.
- The low level part of GNARL uses a POSIX thread interface and it has been re-routed to MaRTE kernel.
- Some low level POSIX calls are rerouted to Linux for `linux_arch` and `linux_lib`
- MaRTE RTS is developed specifically for a GNAT version. We follow AdaCore GNAT GPL editions.

Support has been added for C++ run-time.

- Constructor and destructor calls.
- uSTL library.

Outline

- 1 What is MaRTE OS
 - MaRTE-OS features and architecture
 - MaRTE-OS features and architecture
- 2 Real Time systems
 - Real Time generalities
 - Minimal Real Time POSIX 13 Subset
 - Ada 2005 Annex D. Services
 - Application Defined Scheduling
- 3 FRESCOR and flexible scheduling

Concept of Real Time

Having the results **on time** is as important as the results themselves

Goal: Time predictability

- In hardware (detection of events, transmissions of commands)
- In operating system and network (context switch, timed services, network Tx/Rx)
- At the application level (through analysis techniques)

	Goal	Overload
Non Real Time	Maximum throughput	Global fair impact
Real Time	Attain all deadlines	Penalize only low priority tasks

Real Time doesn't mean maximum throughput

- Extensive computing: Overload is fairly distributed among tasks
- Real Time computing: Overload is distributed to the lowest priority

Real Time tools

How to achieve time predictability in the OS

- Strict preemptive scheduling policies (priority and/or deadline driven).
- Use $O(1)$ queueing algorithms to be independent of nr of tasks, pending timers, etc.
- Implement a mechanism to avoid priority inversion in synchronisation.
- Use real time network protocols that avoid unbounded collision time.
- Eliminate active (polling) waits.

Tools offered for the application

- Tools to implement waiting synchronisation and mutual exclusion.
- Tools to measure time (elapsed and consumed).
- Efficient mechanisms to trigger time events.
- Analysis tools exist offline (e.g. MAST) and online (FRESCOR) to ensure the schedulability of the system.

Minimal Real time POSIX 13 Subset

The whole POSIX (Portable Operating System Interface) too large for real time.

- POSIX.13 defines four real-time system subsets (profiles)

POSIX profiles

Profile	File System	Multiple Processes
Minimal	No	No
Controlled	Yes	No
Dedicated	No	Yes
Multi-purpose	Yes	Yes

Minimal Real time POSIX 13 Subset Cont'd

Functionality included in the minimal profile:

- Threads (policies FIFO, Round-Robin and Sporadic Server)
- Mutexes, condition variables, semaphores
- Signals
- Clocks and timers
 - CPU-time clocks and timers
- Thread suspension, absolute and relative delays.
- Device “files” and device I/O (`open()`, `read()`, `write()`, ...)

Available for C and Ada (Ada bindings implemented in FLORIST).
Implemented in Ada.

Ada 2005 Annex D. Services

Ada was always concerned with supporting timing aspects

- Provides native types for time: Duration and Time Span
- Defines a FIFO priority based scheduling policy (*dispatching model*).
- Provides operation for absolute and relative delays.

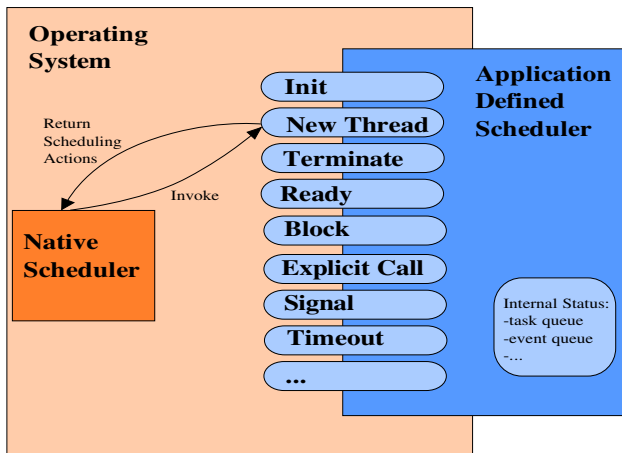
With Ada 2005 Annex D. POSIX features have been added

- New dispatching policies: Earliest Deadline First and Round-Robin
- Timing events.
- Execution-time clocks and timers (and also for task sets)
- Dynamic priorities for Protected Objects
- Priority ceilings on Protected Objects

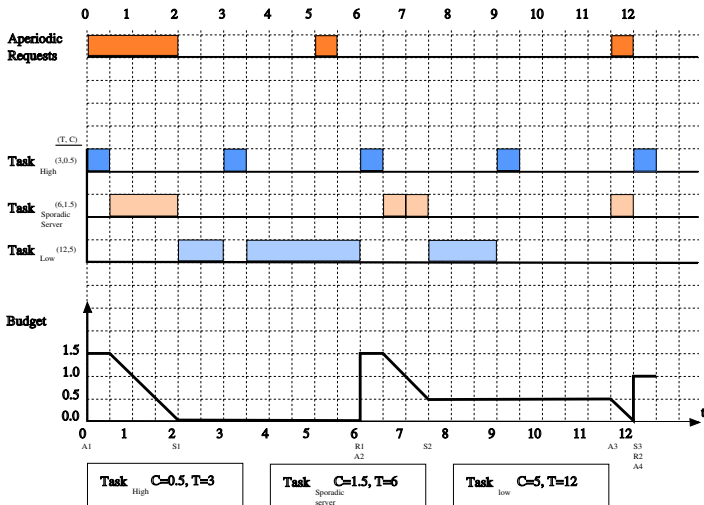
MaRTE OS is the first platform to support all these new Ada 2005 additions.

Application Defined Scheduling

We **interfere** with the scheduler to produce a new policy.



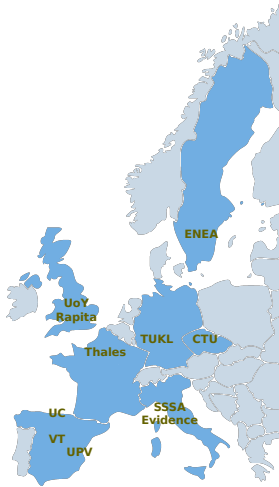
Example: Sporadic server



Outline

- 1 What is MaRTE OS
 - MaRTE-OS features and architecture
 - MaRTE-OS features and architecture
- 2 Real Time systems
 - Real Time generalities
 - Minimal Real Time POSIX 13 Subset
 - Ada 2005 Annex D. Services
 - Application Defined Scheduling
- 3 FRESCOR and flexible scheduling

FRESCOR: Framework for Real-time Embedded Systems based on COntRacts



Goal: Facilitate the adoption of flexible real time scheduling.

Hard real time vs Flexible real time

Traditional real-time

- **Worst case response.**
- Static resource allocation.
 - Single mode
- No time protection.
- No adaptation to load change.

Flexible real-time

- **Worst case response + QoS**
- Dynamic resource allocation
 - Multiple mode
- Time protection.
- Load change adaptation:
 - **Spare capacity:** (mode change)
 - **Dynamic reclamation:** (execution)

Benefits of Flexible Scheduling

- Maximise resource usage.
- Integration of heterogeneous resource requirements.
- Real time theory implicitly integrated in system.

Hard real time vs Flexible real time

Traditional real-time

- Worst case response.
- **Static resource allocation.**
 - Single mode
- No time protection.
- No adaptation to load change.

Flexible real-time

- Worst case response + QoS
- **Dynamic resource allocation**
 - Multiple mode
- Time protection.
- Load change adaptation:
 - **Spare capacity:** (mode change)
 - **Dynamic reclamation:** (execution)

Benefits of Flexible Scheduling

- Maximise resource usage.
- Integration of heterogeneous resource requirements.
- Real time theory implicitly integrated in system.

Hard real time vs Flexible real time

Traditional real-time

- Worst case response.
- Static resource allocation.
 - Single mode
- **No time protection.**
- No adaptation to load change.

Flexible real-time

- Worst case response + QoS
- Dynamic resource allocation
 - Multiple mode
- **Time protection.**
- Load change adaptation:
 - **Spare capacity:** (mode change)
 - **Dynamic reclamation:** (execution)

Benefits of Flexible Scheduling

- Maximise resource usage.
- Integration of heterogeneous resource requirements.
- Real time theory implicitly integrated in system.

Hard real time vs Flexible real time

Traditional real-time

- Worst case response.
- Static resource allocation.
 - Single mode
- No time protection.
- No adaptation to load change.

Flexible real-time

- Worst case response + QoS
- Dynamic resource allocation
 - Multiple mode
- Time protection.
- Load change adaptation:
 - **Spare capacity:** (mode change)
 - **Dynamic reclamation:** (execution)

Benefits of Flexible Scheduling

- Maximise resource usage.
- Integration of heterogeneous resource requirements.
- Real time theory implicitly integrated in system.

Hard real time vs Flexible real time

Traditional real-time

- Worst case response.
- Static resource allocation.
 - Single mode
- No time protection.
- No adaptation to load change.

Flexible real-time

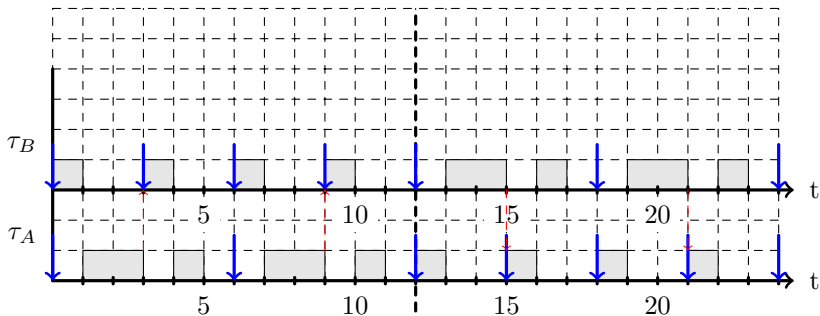
- Worst case response + QoS
- Dynamic resource allocation
 - Multiple mode
- Time protection.
- Load change adaptation:
 - **Spare capacity:** (mode change)
 - **Dynamic reclamation:** (execution)

Benefits of Flexible Scheduling

- Maximise resource usage.
- Integration of heterogeneous resource requirements.
- Real time theory implicitly integrated in system.

Flexible Scheduling Execution

- **Mode 1:** $C_A = 3$ $T_A = 6$ $C_B = 1$ $T_B = 3$
- **Mode 2:** $C_A = 1$ $T_A = 3$ $C_B = 3$ $T_B = 6$



Contract model

Requirements specified in contract

- Min-max budget, period and deadline.
- Task model: Job-based, continuous, background.
- Importance and weight as criteria for spare capacity distribution.
- Critical sections (with their WCET) on shared objects.

Negotiation and binding



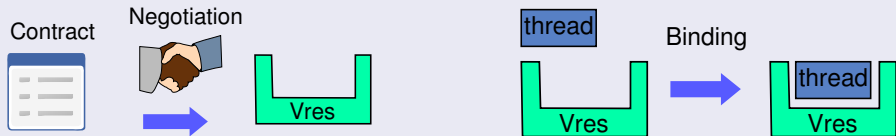
FRESOR ensures that no task goes over its budget. It can also perform an acceptance analysis

Contract model

Requirements specified in contract

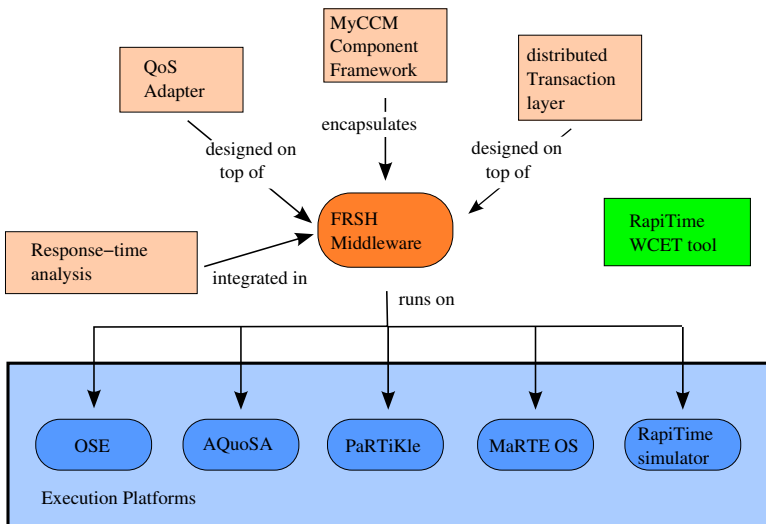
- Min-max budget, period and deadline.
- Task model: Job-based, continuous, background.
- Importance and weight as criteria for spare capacity distribution.
- Critical sections (with their WCET) on shared objects.

Negotiation and binding



FRESCOR ensures that no task goes over its budget. It can also perform an acceptance analysis

FRESCOR project



FSF evolves into FRSH

and now some hands-on work...