
The MaRTE OS run-time as a support platform for real-time programming in Ada 2005

Michael González Harbour (mgh@unican.es)
Mario Aldea Rivas (aldeam@unican.es)
Universidad de Cantabria

*Ada UK Conference
Manchester, September 2007*

Introduction

Some of the most interesting Ada 2005 innovations are those targeting the real-time community

- new capabilities neither addressed by other programming languages nor supported by most execution platforms

The most important of these new services are:

- timing events
- execution-time clocks and timers
- group execution-time budgets
- dynamic priorities for protected objects
- immediate priority changes
- new scheduling and task dispatching mechanisms

We need run-time systems that provide these new services

Objectives of this presentation

Present the current efforts to implement the new real-time services defined in *Ada 2005*

- Introduce the new real-time services
- Present the architecture of the MaRTE OS run-time
- Talk about the implementation status
- Discuss the advantages for real-time programmers

MaRTE OS as a support platform for real-time programming in Ada 2005

1. Introduction
2. MaRTE OS run-time system for GNAT
3. Timing events
4. Execution time clocks and timers
5. Group execution-time budgets
6. Dynamic priorities for PO's
7. New scheduling policies
8. Conclusion and Future work

2. MaRTE OS: Minimal Real-Time OS for Embedded Applications

Follows the Minimal Real-Time POSIX.13 subset

- Concurrence at thread level



All services with a time-bounded response

- also time-bounded interrupt latency

Single address space shared by kernel and application

Monolithic kernel

Written in Ada (some C and assembler code)

Can execute concurrent Ada and C applications (C++, RT Java)

Portable to different platforms via abstract hardware interface

Functionality provided to applications

Functionality included in the minimal profile:

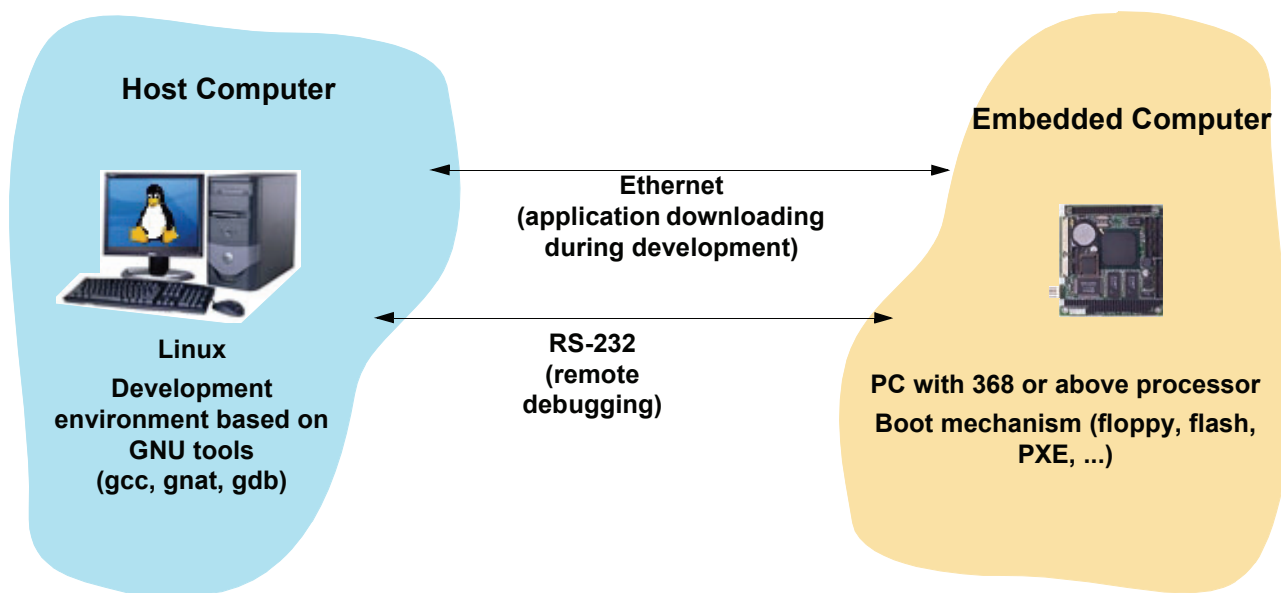
- Threads and real-time scheduling
- Mutexes, condition variables, semaphores
- Signals
- Clocks and timers
 - CPU-time clocks and timers
- Absolute and relative delays.
- Device “files” and device I/O (`open()`, `read()`, `write()`, ...)



Extra functionality

- Hardware interrupt management
- Application-defined scheduling

Cross development environment (x86 architecture)



Successfully used in industry and academia



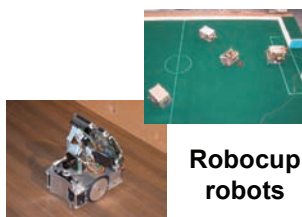
Industrial robot controllers



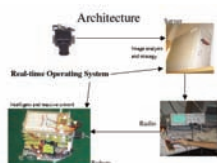
Welding machines



Satellite antenna for high-speed train



Robocup robots



Platform video control



Mobile robot

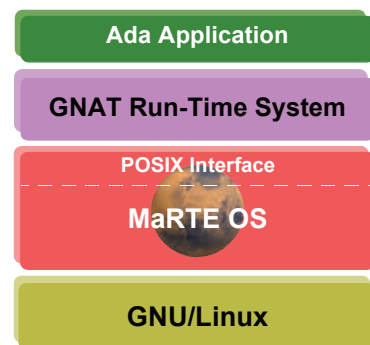
MaRTE OS run-time system for GNAT

MaRTE OS has been adapted to behave as a POSIX-threads library for the GNU/Linux operating system

- concurrence at library level (not using Linux Threads)

The GNAT run-time library has been adapted to run on top of MaRTE OS

- using a specialized GNULL layer
- maps Ada tasking constructs to the MaRTE POSIX interface
- joint effort by *AdaCore* and *UC*



MaRTE OS run-time library is included in current versions of the GNAT compiler

3. Timing events

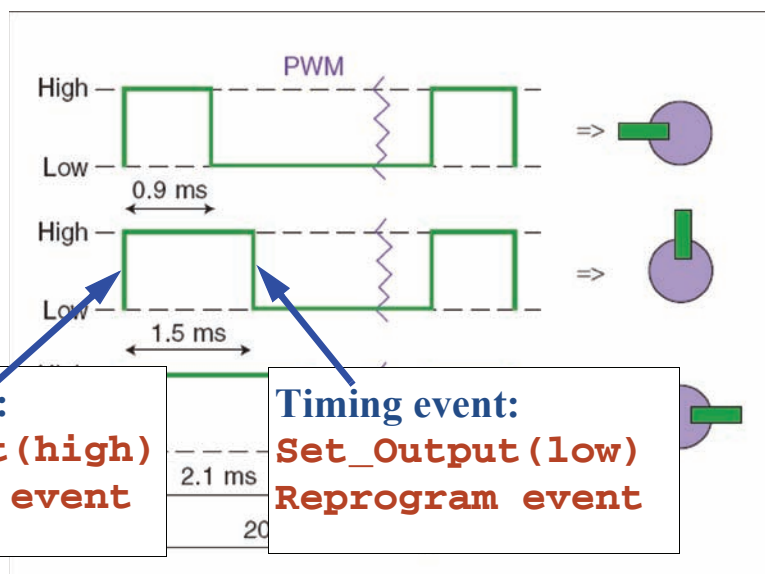
Mechanism to execute user-defined procedures at a specified time *without the need of an auxiliary task* or a delay statement

- the procedure may be executed directly in the context of the clock interrupt handler
- included as a new language-defined package:
`Ada.Real_Time.Timing_Events`

Useful to program:

- short time-triggered procedures
- scheduling algorithms that require programming actions to be executed at a future time

Example: pulsed width modulation



Performance of timing events

Comparison with auxiliary thread:

Metric	Time (μ s) (using timer and auxiliary thread)	Time (μ s) (using timed handlers)
From user's thread to handler	1.1	0.4
From handler to user's thread	0.8	0.7
Total time:	1.9	1.1

4. Execution-time clocks and timers

CPU-time clocks measure the time spent by the system executing each task

- including run-time or system services executed on its behalf

Included in the Ada 2005 standard as a new language-defined package: `Ada.Execution_Time`

CPU-time timers allow calling a handler when a task has used a defined amount of CPU time

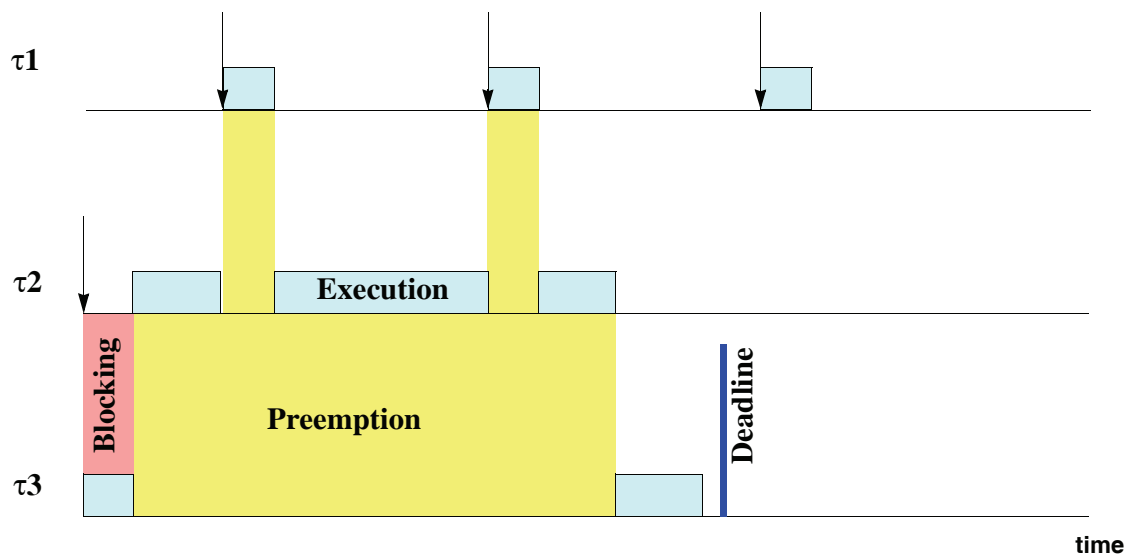
- included in the standard as a new language-defined package (`Ada.Execution_Time.Timer`)

Execution-time clocks and timers (cont'd)

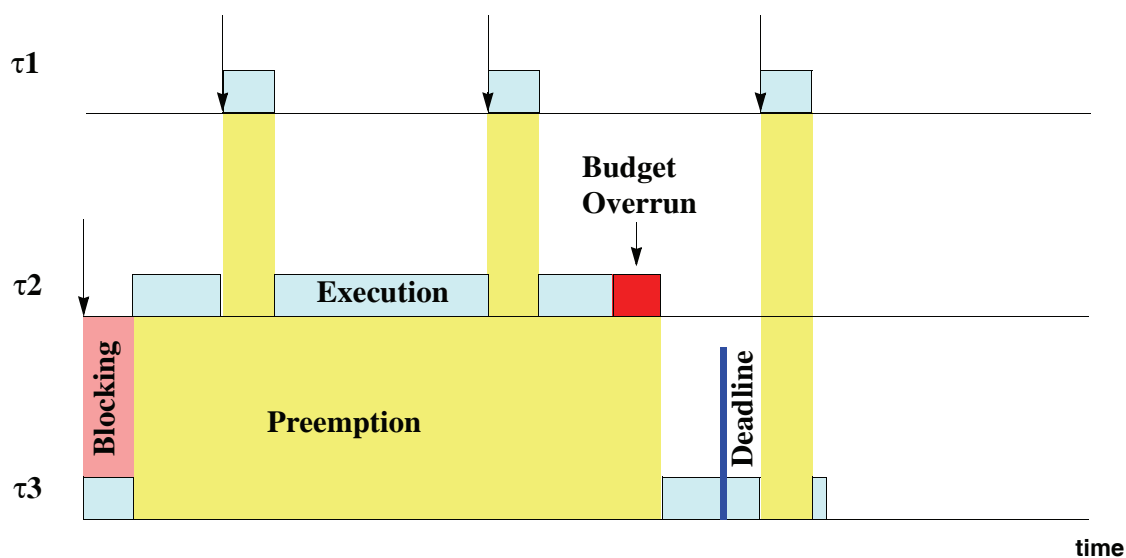
Useful for:

- Run-time monitoring of processor usage
 - detection of and reaction to wrong WCET estimations in a controlled manner
- Implementing some real-time scheduling policies
 - aperiodic servers
 - any other policy that makes scheduling decisions based on CPU time consumption

Example: budget overrun detection



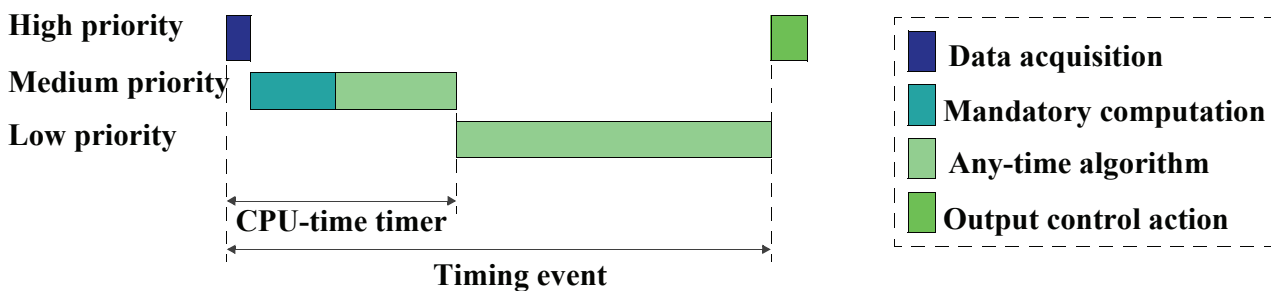
Example: budget overrun detection



Example: scheduling policy for control applications

In this policy, control activities are divided into four parts:

- data acquisition, mandatory computation, improvement using any-time algorithm and output of the control action
- in order to minimize jitter, “data acquisition” and “output control action” parts are executed at high priority



Example: scheduling policy for control applications (cont'd)

This policy is based on the one presented by Alfons Crespo et al. in Ada-Europe 2001

- four Ada tasks and three protected objects were used to implement every control activity using Ada 95 constructs
- with Ada 2005 only one task and one PO are required
 - and execution time measurement is more precise and has less restrictions

Performance of execution-time clocks and timers

Execution time accounting introduces a small overhead

- enabling this service in MaRTE OS increments the context switch time by less than 5%

CPU-time timers have same performance as timing events:

Metric	Time (μ s) (using timer and auxiliary thread)	Time (μ s) (using timed handlers)
From user's thread to handler	1.1	0.4
From handler to user's thread	0.8	0.7
Total time:	1.9	1.1

5. Group execution-time budgets

To assign execution time budgets to groups of tasks

- Execution of any member of the group of tasks results in the budget counting down
 - when the budget becomes exhausted a handler is executed

There is no similar functionality in POSIX

In MaRTE OS we have developed the primitives to support Ada 2005 “group execution-time budgets” service:

- threads sets
- group clocks
- and “timed handlers” based on group clocks

Overhead introduced by thread groups

Group execution time accounting introduces some overhead

- increments the context switch time by 9%
 - including 5% caused by general CPU-time accounting

For a regular x86 platform, with 1KHz tasks

Case	Context switch overhead
No CPU-Time accounting	0.1%
General CPU-time accounting	0.105%
Group Budgets	0.109%

6. Dynamic priorities for protected objects

In Ada 95 it was possible to dynamically change the priority of a task

- however it was not possible to dynamically change the *priority ceiling* of a protected object

This deficiency was a limitation for some purposes:

- “mode changes”
- libraries that include protected objects

Ada 2005 introduces the attribute `'Priority` for POs

- can be read or set inside protected actions
- when set, the ceiling priority is updated just before the end of the protected action

Dynamic priorities for POs (cont'd)

Changing the ceiling of a PO is an inexpensive operation because:

- `pthread_mutex_setprioceiling_locked()` only changes the ceiling of the mutex
 - without affecting the active priority of the task that is locking the mutex

7. New scheduling policies

Ada 95 defined just one standard policy

- `FIFO_Within_Priorities`

Ada 2005 defines new scheduling policies

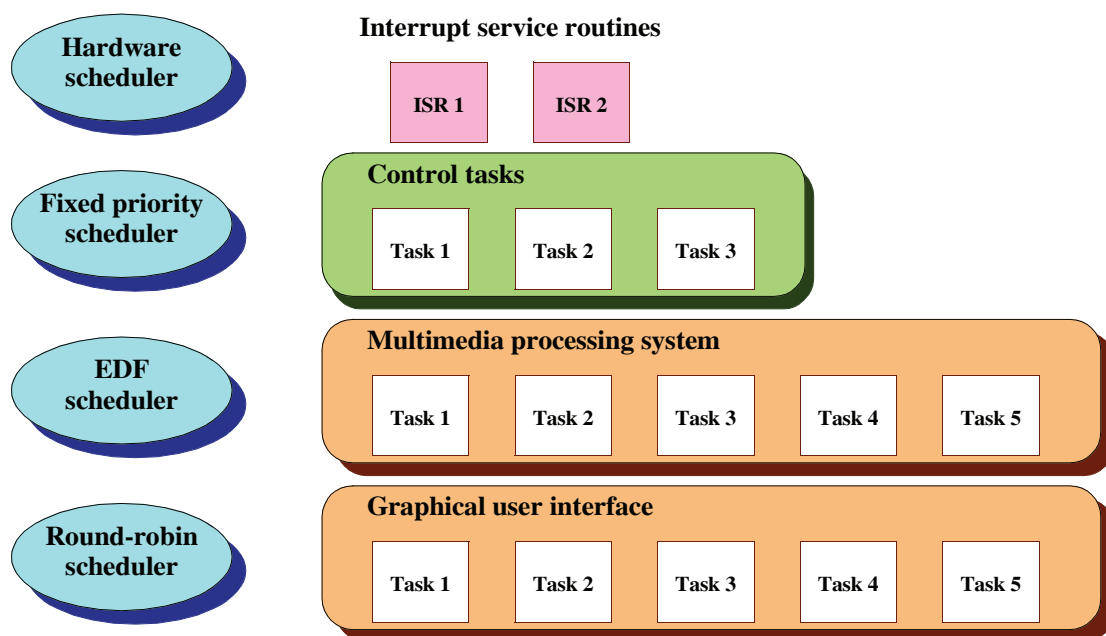
- `Non_Preemptive_FIFO_Within_Priorities`
- `Round_Robin_Within_Priorities`
- `EDF_Across_Priorities`

They can be mixed together in priority bands

- `Priority_Specific_Dispatching`

For the new policies, additional *priority ceiling* rules allow POs free of unbounded priority inversion

Mixing EDF and fixed priorities



8. Conclusion

In GNAT/MaRTE OS for GNU/Linux we have implemented four of the new Ada 2005 real-time services:

- execution time clocks
- execution time timers
- timing events
- dynamic priorities for protected objects

Available in current versions of GNAT (GNAT GPL 2007)

Low-level support for group execution-time budgets is available

These new services can be implemented in an easy and efficient way

Future work

Together with *AdaCore* we plan to complete support for Annex D of Ada 2005:

- complete the “Group Execution Time Budgets”
- implement the new scheduling policies
 - EDF
 - round-robin
 - non-preemptive
 - Priority Specific Dispatching
- provide all these services for bare machines

MaRTE OS is free software (GPL)

<http://martec.unican.es>