CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Technical Report

Session Summary: Clock Issues

Kristoffer Nyborg Gregertsen
Luis Miguel Pinho

CISTER-TR-181204

Session Summary: Clock Issues

Kristoffer Nyborg Gregertsen, Luis Miguel Pinho

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail:

http://www.cister.isep.ipp.pt

Abstract

Session Summary: Clock Issues

Kristoffer Nyborg Gregertsen (Chair)
SINTEF Digital, Trondheim, Norway
kristoffer.gregertsen@sintef.no

Luis Miguel Pinho (Rapporteur)
CISTER/ISEP, Portugal
lmp@isep.ipp.pt

1 Introduction

The session was based on a position paper by Kristoffer Nyborg Gregertsen [1], on clock support in Ada. Kristoffer started by providing a brief overview of time and clocks, presenting issues such as resolution, precision, accuracy and drift. The presentation continued with some examples, which led to motivate the need to support high-precision distributed time:

- Time synchronized events in distributed control systems;
- Sensors with high-precision timestamps for monitoring physical processes;
- Applications in robotics, process automation, smart grid applications, etc.

Kristoffer then summarized the support for calendar and clock in Ada

- The Calendar package, implemented by the system clock;
- Then real-time clocks, which are required to be monotonic with documented drift;
- And execution-time clocks and timers, for CPU time, tasks and interrupts.

At this point, Kristoffer raised the issue that there is no support to dynamic clock rates or different rates on CPU cores. Michael González Harbour noted that they had already faced this problem and had to disable dynamic clock rates, so a solution for this issue would be important. Michael also noted that Ada allows implementations to add additional time types as an extension that can be used in delay statements.

2 Issues Discussed

Kristoffer presented a few difficulties with clocks and timers in Ada, for which he would like to propose changes:

- There is no standard way of acquiring high-precision timestamps;
- It is not possible to relate the real-time monotonic clock to UTC/TAI;
- The Calendar package is not synchronized with UTC;
- Calendar is not allowed in Ravenscar systems;
- There are no explicit clock types with common interface, only clock functions for different time types;
- Timer are defined as tagged types, but have no common interface and have subtle differences (e.g. timing event, timer and group budgets).

Michael noted that implementation can actually implement a real-time clock that synchronizes to TAI and a calendar that synchronizes with UTC, although not forced to. Then, Andy Wellings put forward that

applications can also do this synchronization with a high-priority task doing time synchronization. Nevertheless, Kristoffer considers that there should be a standard way, even to know the discrepancy between clocks.

Concerning Ravenscar, Calendar is not allowed, which means that in Ravenscar it is not possible to reason on wall time. Kristoffer provided an example of a smart grid system, where it could be required to open a switch at a specific time instance in the day.

There was also some discussion on the fact that there are no explicit clock types and the interfaces of timers are different, which makes it difficult to provide a common hierarchy. Time could be a root type, where other time types derive (they are private). But the finer details still need to be looked at (e.g. Calendar Duration and real-time Time_Span are different types for referring to a time interval).

Afterwards, the workshop discussed the issue of execution-time timers for interrupts, a theme recurrent from previous workshops. The existing possibilities were reviewed, but no further solutions were considered. There was also some discussion on execution time and parallelism: having the possibility to know post-fact the execution time of parallel computation could be interesting. Not so much, the ability to fire event handlers on execution-time overruns.

At this point, Kristoffer proposed to have coherent clock and timers in Ada, where clocks could be defined as explicit tagged types with a shared interface. This would make it easier to understand rather than having subtle differences, other clocks could be defined with special properties such as UTC or TAI and tailored clocks for particular systems or applications. Kristoffer also compared with the support to clocks and timers in C++11 and 20 and RTSJ 2.0, noting the advances provided in these languages.

In the discussion it was also raised that it would be important at least to be able to detect loss of synchronization with a time source (e.g. NTP or GPS), either with an interface where this could be queried or the possibility to raise an exception if the program tried to use a clock which has lost synchronization.

At the end of the session it was generally agreed that it would be important for Ada to have more advanced support for dealing with time, as the premier real-time language. An argument was made that if Ada lags behind in important things where other languages are advancing it will be difficult to attract programmers.

Nevertheless, due to the time-lag in standardization, having a non-standard peer-reviewed library would be interesting as an incubator for new ideas before going to the Ada Rapporteur Group (ARG). It was generally agreed that the proposed extensions should be provided first under the “XAda” package hierarchy that the workshop decided to endorse in the Time Triggered Scheduling session [2].

References

[1] K. N. Gregertsen. Position paper: Clock support in Ada. This Issue.
[2] J. Real, B. Moore, “Session Summary: Time Triggered Scheduling in Ravenscar”. This Issue.