The Ravenscar Profile

Alan Burns
Real-Time Systems Research Group
Department of Computer Science
University of York, UK
burns@cs.york.ac.uk

Abstract

The Ravenscar profile is described. All its features and restrictions are noted. Also, the means of designating the profile is presented. Detailed motivations for the profile are not given. The aim of the paper is to summaries the outcome of deliberations at the 8th, 9th and 10th International Workshops on Real-Time Ada Issues.

1 Introduction

One of the significant outputs of the 8th International Real-Time Ada Workshop (IRTAW) was the definition of a restricted tasking profile for use in high-integrity efficient real-time systems [2]. The *Ravenscar Profile*, as it is now designated, has been the subject of considerable attention since that workshop and at least two full implementations are now available. Other studies have also been undertaken [9, 8, 7]. In March 1999 the 9th IRTAW was held and the profile was again examined with the result that a few minor alterations and clarifications were agreed [1]. A similar exercise took place at the 10th IRTAW in Spetember 2000.

The profile now has a stable definition, and although it is only a *de facto* standard it has been referenced in the ISO report 'Use of Ada in High-Integrity Systems' [10] (an early version of this report is available via Ada Letters [3]).

The purpose of this paper is to describe, in one document, the Ravenscar Profile as defined by the 8th, 9th and 10th IRTAW. The paper is a revision of an early one that appeared in Ada Letters[4]. Modifications arising from the 10th IRTAW are designated by a † symbol. It is not intended to give a full motivation for the profile; those issues have been addressed elsewhere [2, 6, 5].

2 The Profile

The Ravenscar profile is defined by noting which features are allowed, which are disallowed, what dynamic semantics are required and how the restrictions can be represented. Consider first, the **forbidden** features:

- Task types and object declarations other than at the library level. Thus, there is no hierarchy of tasks.
- Dynamic allocation and unchecked deallocation of protected and task objects.
- Requeue
- ATC (asynchronous transfer of control via the asynchronous_select statement)
- Abort statements including `Abort_Task` in package `Ada.Task_Identification`
- Task entries
- Dynamic priorities
- Calendar package
- Relative delays
- Protected types and object declarations other than at the library level
- Protected types with more than one entry
- Protected entries with barriers other than a single boolean variable declared within the same protected type
- An entry call to a protected entry with a call already queued
- Asynchronous task control
- All forms of select statement
- User-defined task attributes
- Dynamic interrupt handler attachments †

In addition to these restrictions an implementation can make the assumption that none of the program's tasks will terminate.

Tasking features that are supported by the profile are as follows:

- Task objects, restricted as above
- Protected objects, restricted as above
- Atomic and Volatile pragmas
- 'Delay until' statements
- Ceiling Locking policy and FIFO within priority dispatching
- Count Attribute (but not within entry barriers)
- Task identifiers, e.g. `T'Identity, E'Caller`
- Synchronous task control
- Task discriminants
- 'Real-Time' package
- Protected procedures as (statically bound) interrupt handlers

3 Dynamic Semantics

Three aspects of the profile require their dynamic semantics to be defined.

1. If an entry call is made on an entry that already has a queued call (i.e. the queue length would become 2) then `Program_Error` is raised.

2. If a task attempts to terminate, this is classified as a bounded error (i.e. there is a documentation requirement on the implementation to define its effect) - one allowed outcome being the permanent suspension of the task.

3. If a task executes a potentially blocking operation from within a protected object then `Program_Error` *must* be raised (unless a subprogram call is made to a foreign language domain)† - the full language defines this as a bounded error, `Program_Error` being just one of the allowed outcomes.

Note the use of `Program_Error` in (1) is consistent with its use in the definition of synchronous suspension objects [ARM D.10(10)].

4 Denoting the Restrictions

Many of the restrictions implied by the Ravenscar profile can be designated by the existing definition of pragma `Restrictions`. The following identifiers apply:

```
No_Task_Hierarchy
No_Abort_Statements
No_Task_Allocators
No_Dynamic_Priorities
No_Asynchronous_Control
Max_Task_Entries => 0
Max_Protected_Entries => 1
Max_Asynchronous_Select_Nesting => 0
Max_Tasks => N -- fixed by the application
```

To give a complete definition, however, requires the use of ten new restriction identifiers:

```
Simple_Barrier_Variables
Max_Entry_Queue_Depth => 1 -- in general, N
No_Calendar
No_Relative_Delay
No_Protected_Type_Allocators
No_Local_Protected_Objects
No_Requeue
No_Select_Statements
No_Task_Attributes
No_Task_Termination
No_Dynamic_Interrupt_Handlers
```

The meaning of these restrictions are straightforward and follow directly from the definition of the profile above. Hence `Simple_Barrier_Variables` requires all entry barriers to consist of single boolean variables, and `Max_Entry_Queue_Depth` restricts the maximum length of an entry queue. For the profile this to set to one. The last restriction† implies that none of the subprograms defined in `Ada.Interrupts` will be called by the application (hence `Interrupt_Id` may still be used).

5 Conclusion

The industrial interest currently being shown in the Ravenscar Profile is an indication of its growing importance. Although the 9th and 10th IRTAW made some minor changes to the profile, the overall conclusion of these workshops was a confirmation of the definition produced at the 8th IRTAW. The profile now represents key Ada Technology.

Acknowledgements

The Ravenscar profile was defined by the collective consideration of all those at the 8th, 9th and 10th IRTAW.

References

[1] L. Asplund, B. Johnson, and K. Lundqvist. Session summary: The Ravenscar profile and implementation issues. In A. Burns, editor, *Proceedings of the 9th International Real-Time Ada Workshop*, volume XIX(2), pages 12–14. ACM Ada Letters, June 1999.

[2] T. Baker and T. Vardanega. Session summary: Tasking profiles. In A. Wellings, editor, *Proceedings of the 8th International Real-Time Ada Workshop*, pages 5–7. ACM Ada Letters, 1997.

[3] P. Bhansali*et al*. Guidance for the use of the Ada programming language in high-integrity systems. *ACM Ada Letters*, 1998.

[4] A. Burns. The Ravenscar Profile. *ACM Ada Letters*, XIX(4):49–52, Dec 1999.

[5] A. Burns, B. Dobbing, and G. Romanski. The Ravenscar tasking profile for high integrity real-time programs. In *Reliable Software Technologies, Proceedings of the Ada Europe Conference, Uppsala*, pages 263 – 275. Springer Verlag, 1998.

[6] A. Burns and A. Welling. Restricted tasking models. In A. Wellings, editor, *Proceedings of the 8th International Real-Time Ada Workshop*, pages 27–32. ACM Ada Letters, 1997.

[7] M. Kamrad and B. Spinney. An Ada runtime system implementation of the Ravenscar profile for high speed application-layer data switch. In *Reliable Software Technologies, Proceedings of the Ada Europe Conference, Santander*, pages 26–38. Springer Verlag, 1999.

[8] K. Lundqvist, L. Asplund, and S. Michell. A formal model of the Ada Ravenscar tasking profile; protected objects. In *Reliable Software Technologies, Proceedings of the Ada Europe Conference, Santander*, pages 12–25. Springer Verlag, 1999.

[9] D. Naydich and D. Guaspari. Analyzing ravenscar profile tasks by model checking. Technical report, Odyssey Reseach Associates, TM-98-0034, 1988.

[10] B. Wichmann(Editor). The use of Ada in high-integrity systems – final draft. Technical report, ISO, 1999.