Space &Time Partitioning with ARINC 653 and `pragma Profile`

Joyce L Tokar, PhD
Pyrrhus Software
PO Box 1352
Phoenix, AZ 85001-1352
USA
tokar@pyrrhusoft.com

Abstract

The development of embedded applications is entering into a new domain with the availability of new high-speed processors and low cost on-chip memory. As the result of these new developments in hardware, there is an interest in enabling multiple applications to share a single processor and memory. To facilitate such a model the execution time and memory space of each application must be protected from other applications in the system.

The ARINC Specification 653[1] provides the definition of an APplication EXecutive (APEX) that supports space and time partitioning of applications. ARINC 653 systems are composed of software partitions. Each partition is a separate application and there is dedicated memory space for each partition thereby providing space partitioning. Similarly, the APEX provides a dedicated time slice for each partition to support time partitioning. ARINC 653 supports a multi-tasking environment in each partition through a well-defined set of process communication mechanisms and preemptive priority based scheduling.

The availability of a multi-tasking environment within each ARINC 653 partition reduces the need for the use of the Ada process model in the development of applications to execute on top of the ARINC 653 APEX. This paper will presents a proposal for a new profile definition, ARINC_653_Processes, for consideration using the ***`pragma`*** `Profile` *capability of Ada.*

1 Introduction

The ARINC Specification 653 [1] defines an APplication EXecutive (APEX) interface to be used in Integrated Modular Avionics (IMA) applications. The APEX interface provides a common logical environment that enables the execution of multiple independently developed applications to execute together on the same hardware. The flexibility of APEX together with the standardized definition of space and time partitioning has led to the proliferation of the use of ARINC 653 beyond IMA to include many other safety critical applications.

The current version of the ARINC Specification 653 defines the first phase of APEX that is focused on the requirements of critical systems. As such, this phase of APEX was designed to provide an interface to application software only; it does not define an interface to the Ada runtime system and it does not directly support Ada tasking. However, some of the goals of the APEX include satisfying the real-time requirements of Ada.

Given that Ada Issue 249 (AI-249) [2] defines a pragma-based approach for defining run-time profiles, this methodology may be used to define an ARINC 653 compliant profile that supports the goals of the first phase of APEX. Presently, AI-249 defines one profile, Ravenscar. This paper presents the definition of a new profile, ARINC_653_Processes, for use with an ARINC 653 compliant operating system.

There is a growing interest in the use of COTS Operating Systems that implement the ARINC 653 Application Executive (APEX) for the development of software systems that satisfy both space and time partitioning requirements. For example, BAE Systems Controls provides a fully compliant ARINC 653 Operating System (OS) with their CsLEOS™ product. In addition to the space and time partitioning provided by the ARINC 652 standard, BAE has demonstrated that CsLEOS™ is certifiable to DO-178B Level A [3] meeting the additional validation and verification requirements of safety-critical applications.

Within the APEX environment, applications are encapsulated in partitions that are managed by the APEX operating system. Each application is comprised of one or more APEX processes. The ARINC 653 specifies an application interface to provide access to the facilities within the APEX environment. ARINC 653 is defined to be language independent.

In the development of a system of applications that are operating on top of the APEX interface, it is possible to define applications in a variety of languages each of which represents threads of execution within the application as APEX processes. In this environment, one mode of operation for Ada applications is to replace Ada tasks and scheduler with the ARINC 653 processes and scheduling. In using the ARINC 653 processes and scheduler for Ada applications, the application cooperates with other applications in the ARINC 653

system through a common ARINC 653 interface and environment.

2 Integration with Ada

There are several approaches available to integrate an Ada application into an ARINC 653 compliant system. One option is to encapsulate the entire Ada application into a single ARINC 653 partition by integrating the Ada run-time system with the ARINC 653 operating system. Another alternative is to eliminate the need for the Ada run-time system and rely solely on the underlying ARINC 653 compliant operating system.

Using the first implementation strategy an Ada compilation system would generate an Ada application that is integrated with an existing Ada run-time system. This entire application would then be encapsulated into an ARINC 653 partition and incorporated into the larger application. To facilitate communication with other partitions in the system, the Ada application would need to have an interface with the ARINC 653 OS such that it could access the ARINC 653 inter-partition communciation procedures. Similarly, there would need to be some modifications to the Ada environment to facilitate the integration with the fault handling and interrupt handling mechanisms provided by ARINC 653. This approach is documented in [4] where the Ada application makes use of the Ravenscar profile to offer a predictable Ada tasking model within the APEX environment.

The approach considered in this paper is to take advantage of the new **pragma** `Profile` to define a set of restrictions that will eliminate all dependencies on the Ada run-time system that would interfer with the underlying ARINC 653 compliant operating system. In addition, applications in this environment would make use of ARINC processes and scheduling facilities.

3 A New Profile Definition

Taking advantage of the new **pragma** `Profile` capability that is to be incorporated into the upcoming revision of the Ada Language Standard [5], it is possible to define a set of Ada restrictions that enable an application to be built to use APEX Processes rather than Ada tasks. This profile definition enables the use of standard Ada language features to join with the standard API defined in the APEX standard. Thereby providing a solution that is common to two standards and is not vendor specific.

The recommended restrictions defining `ARINC_653_Processes` profile are:

```
Max_Tasks => 0,
No_Protected_Types,
No_Asynchronous_Control,
No_Synchronous_Control.
```

`Max_Tasks => 0` prevents any task creation and, if a program contains a task creation, it is illegal. If an implementation chooses to detect a violation of this restriction, `Storage_Error` should be raised; otherwise, the behavior is implementation defined.

`No_Protected_Types` indicates that there are no declarations of protected types or protected objects.

`No_Asynchronous_Control` specifies that there are no semantic dependences on the package `Ada.Asynchronous_Task_Control`.

`No_Synchronous_Control` is a new restriction that specifies that there are no dependencies on the package `Ada.Synchronous_Task_Control`.

This set of restrictions enables the development of runtime systems that does not include any part of an Ada tasking kernel. Several additional restrictions were considered to reflect more of the behavior of ARINC 653 including:

```
No_Allocators,
No_Exceptions,
No_Delay.
```

After consideration from the members of the IRTAW12, these restrictions were viewed as being implementer dependent and not necessary for the definition of the `ARINC_653_Processes` profile.

Thus, an Ada application that is developed to execute using the APEX API would be compiled using:

```
pragma
    Profile(ARINC_653_Processes)
```

This would indicate to the compilation system to check at compile-time for any violations of the restrictions listed above. It would also tell the compilation system to use a runtime system that does not include any support for Ada tasking or scheduling. In this manner, the APEX processes and scheduler could be used as provided by the underlying APEX compliant operating system.

4 Conclusions

The proposal provided here offers an vendor independent approach that enables the user to control all process scheduling and communications as well as fault handling for all applications within the APEX domain using the common process scheduler provided by the underlying operating system. With the use of **pragma** `Profile` the Ada applications will be checked at compile time to ensure compliance with the execution time profile defined as `ARINC_653_Processes`. Thus, these applications benefit from the early detection of errors in the use of the ARINC 653 profile. Applications that are comprised of module written in Ada and modules written in other programming languages such as C make use of the common APEX interface thereby sharing the common scheduling and communications facilities defined in the APEX. The overall system benefits from the use of a common set of

processes and a common scheduler throughout all applications.

References

[1] *Avionics Application Software Standard Interface*, ARINC Specification 653, January 1997.

[2] Ada Issue 249, *Ravenscar Profile for High-Integrity Systems*, December 2002.

[3] *Software Considerations in Airborne Systems and Equipment Certification*, RTCA Document DO-178B.

[4] B. Dobbing, *Building Partitioned Architectures Based on the Ravenscar Profile*, SIGAda 2000, Ada Letters, Vol. XX, No. 4, December 2000.

[5] *Consolidated Ada Reference Manual*, International Standard ISO/IEC 8652/1995 (E) with Technical Corrigendum 1, Lecture Notes in Computer Science 2219, Springer-Verlag, 2000.