

Safety-critical SW development for integrated modular avionics

by Paul Parkinson and Larry Kinnan, Wind River

The shift of the avionics industry toward IMA presents significant challenges for standards organisations, OEMs, and commercial vendors alike. However, it is clear that standards-based architectures, such as ARINC 653, will provide greater flexibility and portability, and enable existing federated applications to be re-used in an IMA environment.

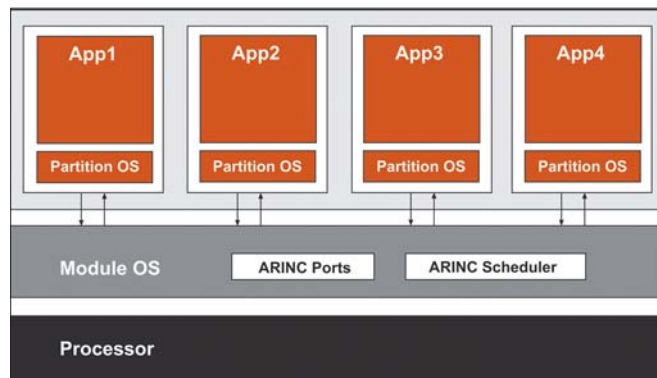


Figure 1. Example ARINC 653 architecture implementation

■ Many avionics systems have been successfully developed using custom hardware and software. However, in recent years, the full life-cycle costs of customized systems have forced programme offices and OEMs to consider the use of COTS-based systems. At the same time, there has been a noticeable migration away from federated architectures, where individual subsystems perform a dedicated function, towards generic computing platforms which can be used in multiple types of applications and, in some cases, run multiple applications concurrently. This approach, known as integrated modular avionics (IMA), results in fewer subsystems, reduced weight, and less platform redundancy.

A number of civil and military research programmes have sought to define IMA architectures, and while they have differed in their approaches, they have shared the same high-level objectives. 1) Common processing subsystems. These should allow multiple applications to share and re-use the same computing resources. This facilitates a reduction in the number of deployed subsystems which are not fully utilized and provides a more efficient use of system resources, leaving space for future expansion. 2) Software abstraction. This should isolate the application not only from the underlying bus architecture but also from the underlying hardware architecture. This enhances

portability of applications between different platforms and also enables the introduction of new hardware to replace obsolete architectures. 3) Cost of change. An IMA architecture should reduce the cost of change, since it facilitates re-use and lowers re-test costs because it simplifies the impact analysis by decoupling the constituent pieces of the platform of multiple applications executing on the same processor.

IMA also facilitates support for applications which have ever-increasing levels of functionality, including the interactions between complex applications such as head-up displays, map display systems, and weather radar displays. Although a number of IMA architectures and standards have emerged, the ACR specification and ARINC specification 653 appear to have the widest adoption in the avionics community.

The ACR specification ARINC 653 application development defines two important concepts which are widely used in IMA – spatial partitioning and temporal partitioning. Spatial partitioning defines the isolation requirements for multiple applications running concurrently on the same computing platform, often referred to as a module. In this model, applications running in an IMA partition must not be able to deprive each other of shared application resources or those provided by the RTOS kernel. This is most often achieved through the use of

different virtual memory contexts. These contexts are referred to as partitions in ARINC 653, and contain an application with its own heap for dynamic memory allocation, and a stack for the application's processes (the ARINC 653 term for a context of execution). These requirements have an impact on the design and implementation of the RTOS kernel and language runtime system, in particular, the use of the processor's memory management unit (MMU) to prevent accidental or malicious accesses to program code by errant applications.

However, in an IMA environment, memory protection alone would not prevent an errant application running in a partition from consuming system resources, which might have a detrimental effect on an application running in another partition. This can have serious consequences where multiple applications of differing levels of criticality are running on the same processor. This problem cannot be resolved through the use of a full process model alone; instead it requires the development of an RTOS which specifically addresses the needs of IMA. In order to implement the ARINC 653 model, a number of architectural approaches can be taken, an example of which is shown in figure 1. This architecture comprises the following elements. 1) The module OS interacts directly with the computing platform (core module). It provides global resource manage-

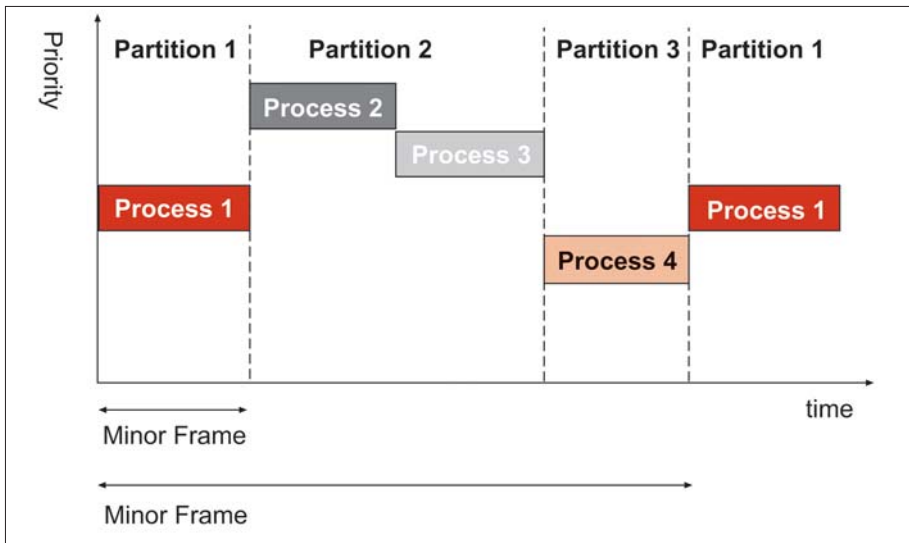


Figure 2. ARINC 653 temporal partitioning

ment, scheduling, and health monitoring for each of the partitions. It also provides the hardware-specific implementation required to run on a specific processor and hardware configuration. This is also referred to as the board support package. 2) The partition OS provides scheduling and resource management within a partition. Communication with the module OS is through a private message-passing interface to ensure robustness. The Partition OS also provides the ARINC 653 APEX interfaces for use by applications.

This architecture provides a means of fulfilling the requirements of ARINC 653, while providing a flexible, extensible framework not easily achieved with a monolithic kernel implementation. Within the framework, individual partitions are implemented using memory-protected containers into which processes, objects, and resources can be placed, with partitioning enforced by the MMU. Each partition has its own stack and local heap, which cannot be usurped by applications running in other partitions. Partitions also prevent interference from errant memory accesses by applications running in other partitions.

Temporal partitioning defines the isolation requirements for multiple applications running concurrently on the same computing platform. This ensures that one application may not utilize the processor for longer than intended to the detriment of the other applications. ARINC 653 defines an implementation using partition-based scheduling where a partition is scheduled for a timeslot of a defined width, and other partitions may be allocated timeslots of similar or differing durations. At the end of the timeslot, the ARINC scheduler forces a context switch to the next partition in the schedule. Within a timeslot, a partition may use its own scheduling policy. This model is sufficiently flexible to

enable existing federated applications, or new IMA applications developed in isolation, to be hosted on a core module. Inevitably, the scheduling of the partitions and the verification that boundaries and schedules are not violated, or that, when this does occur, that appropriate corrective action can be taken, introduces additional complexity.

In this example, ARINC 653 architecture implementation, the module OS performs ARINC 653 scheduling of the individual partitions, and within each timeslot, the Partition OS performs pre-emptive priority-based scheduling (see figure 2). There is also a requirement to ensure that applications do not interfere with each other through use of kernel processing on behalf of the application. In ARINC 653, this is prevented through the use of a sophisticated model. Here, a partition layer is responsible for servicing APEX calls, and only those which require interaction with the core layer are passed over. The time spent servicing the kernel calls is scheduled as part of the partition, which prevents the application from stealing time from other partitions.

Although the certification of IMA systems is a relatively new endeavour, many aspects build on the methods used in the certification of existing federated systems to certification standards. The concept of re-usable software components with DO-178B certification evidence in a safety critical application is well documented. ARINC 653 implementations may now enable this concept to be extended further through the use of temporal and spatial partitioning. For example, federated application which has been previously certified to DO-178B level C could be used in a separate partition on the same IMA platform as a new DO-178B level A application, without the level C application needing to be re-certified to Level A. ■