WIND™

Safety-Critical Software Development for Integrated Modular Avionics

Paul Parkinson, Principal Systems Architect, Aerospace & Defense, Wind River

WHEN IT MATTERS, IT RUNS ON WIND RIVER

EXECUTIVE SUMMARY

This technical paper presents recent trends in the development of safety-critical avionics systems. It discusses the emergence of Integrated Modular Avionics (IMA) architectures and standards, the resulting impact on the development of an ARINC 653–compliant commercial off-the-shelf (COTS) real-time operating system (RTOS), and support for multi-core processor architectures.

TABLE OF CONTENTS

WIND

INTRODUCTION

Many avionics systems have been successfully developed using custom hardware and software. However, over the last decade, the full lifecycle costs of customized systems have forced original equipment manufacturers (OEMs) to consider the use of COTS-based systems. At the same time, there has been a noticeable migration away from federated architectures, where each individual subsystem performs a dedicated function, toward generic computing platforms that can be used in multiple types of applications and, in some cases, can run multiple applications concurrently. This approach, known as integrated modular avionics, or IMA, results in fewer subsystems that take up less space and have reduced weight and power consumption (often referred to as SWaP). A number of civil and military research programs have sought to define IMA architectures, and while they differ in their approaches, they share the same high-level objectives:

- **Common processing subsystems:** An IMA architecture should allow multiple applications to share and reuse the same computing resources so that fewer subsystems need to be deployed, resulting in more efficient use of system resources and leaving space for future expansion.
- **Software abstraction:** An IMA architecture should isolate the application not only from the underlying bus architecture but also from the underlying hardware architecture. This practice enhances portability of applications between different platforms and also enables the introduction of new hardware to replace obsolete architectures.
- **Maximize reuse:** An IMA architecture should allow for reuse of legacy code. This practice reduces development time while affording the developer a method of redeploying existing applications without extensive modifications.
- **Reduced cost of change:** An IMA architecture should reduce the cost of change, both because IMA facilitates reuse and because, by decoupling the constituent pieces of the platform that execute on the same processor, it simplifies the impact analysis, lowering retest costs.

IMA also facilitates support for applications that have ever-increasing levels of functionality, including the interactions between complex applications (such as head-up displays, map display systems, and weather radar displays). IMA can also now exploit the technology of multi-core processors in order to facilitate such diverse applications as I/O offload; executing multiple, disparate operating environments (such as Linux or other non-COTS operating systems) through virtualization; and addressing the increasingly demanding needs of processing power and speed required by the previously listed applications.

Although a number of IMA architectures and standards have emerged, the ACR Specification[1] and ARINC Specification 653[2] appear to have the widest adoption in the avionics community. The ACR Specification addresses architectural considerations, whereas ARINC Specification 653 defines at a high level an instance of a software implementation for an IMA architecture. The widespread adoption and support of ARINC 653 is also evident in the Future Airborne Capability Environment (FACE™) for U.S. military avionics programs. These and other IMA standards place new demands on the software architecture, especially the RTOS implementation provided by the COTS supplier. Wind River® has specifically addressed these needs by developing Wind River VxWorks® 653 Platform to support ARINC 653 and POSIX®. This support for open architectures has contributed to VxWorks 653 Platform being selected for and deployed in many avionics systems and safety-critical applications, including the C-130 Avionics Modernization Program, 767 Tanker, the Boeing 787 Dreamliner Common Core System (CCS), and the Airbus A330 MRTT. VxWorks 653 Platform has also subsequently undergone independent conformance verification, testing against the FACE Safety Base Profile, and is the first COTS RTOS to achieve conformance certification.[3]

The following sections consider the technical requirements for an integrated device software platform to support IMA applications and show how VxWorks 653 Platform (see Figure 1) fulfills these requirements—in particular within the context of ARINC 653 application development.

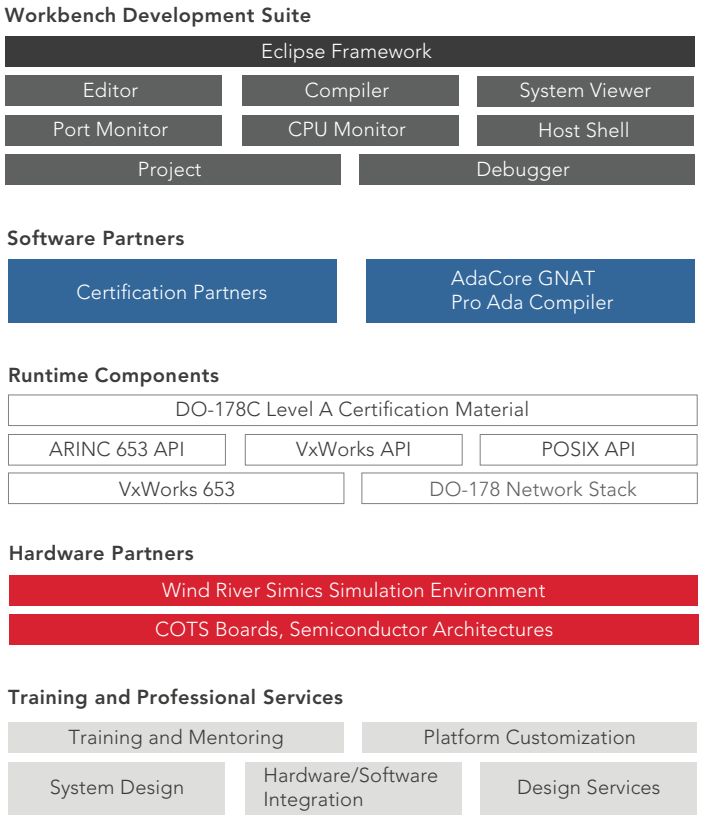

Figure 1. Wind River VxWorks 653 Platform

APPLICATION DEVELOPMENT WITH VXWORKS 653 PLATFORM

The ACR Specification defines two important concepts widely used in IMA: *spatial partitioning* and *temporal partitioning*.

Spatial Partitioning

Spatial partitioning defines the isolation requirements for multiple applications running concurrently on the same computing platform, also known 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 state is most often achieved through the use of different virtual memory contexts enforced by the processor's memory management unit (MMU).

These contexts are referred to as partitions in ARINC 653. Each partition contains 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 affect the design and implementation of the RTOS kernel and language runtime system. For example, VxWorks 5.5 uses a shared virtual address space for applications and provides basic support through the MMU to prevent accidental or malicious access to program code by errant applications, without incurring the performance overhead of a full process model. VxWorks 6, VxWorks 7, and VxWorks 653 provide environments that use the MMU to enforce separate contexts.

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 issue 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 that specifically addresses the needs of IMA. The VxWorks 653 operating system was designed specifically for this purpose and supports the ARINC 653 model in the implementation of the kernel architecture (see Figure 2).

- The *module OS* interacts directly with the computing platform (core module), providing global resource management, scheduling, and health monitoring for each of the partitions. It also uses a board support package (BSP), the hardware-specific configuration required to run on different processors and hardware configurations. Multi-core processors are also supported by the module OS and its BSP. Executing on a multi-core processor allows some additional capabilities to be exploited, such as running in an asymmetric multiprocessing (AMP) configuration and being able to assign multiple cores to application partitions for use by the application, accomplished through the use of APIs provided in ARINC 653 Part 1, Supplement 4. This ability can also allow individual partitions to be executed on their own core for additional spatial separation.
- The *partition OS* is implemented using the VxWorks microkernel and provides scheduling and resource management within a partition. Communication with the module OS occurs through a private message-passing interface to ensure robustness. The partition OS also provides the ARINC 653 application/executive (APEX) interfaces for use by applications.

This architecture, which represents the virtual machine approach as described in "Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance,"[4] provides a means of fulfilling the requirements of ARINC 653 while providing a flexible, extensible framework not easily achieved with a monolithic kernel implementation or UNIX-like implementations. 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 (virtual machines). Each partition has its own stack and local heap, which cannot be usurped by applications running in other partitions. The partitions also prevent interference from errant memory accesses by applications running in other partitions.

Figure 2 shows the conceptual implementation of the VxWorks 653 architecture. The RTOS features the ability to have a shared partition OS library in order to simplify configuration, testing, and certification; it can also have separate, distinct configurations of the partition OS for one or more partitions (referred to as MPOS, or multiple partition operating system), which allows for significant flexibility for developers of applications to exploit various features of the partition OS that are unique to each application.

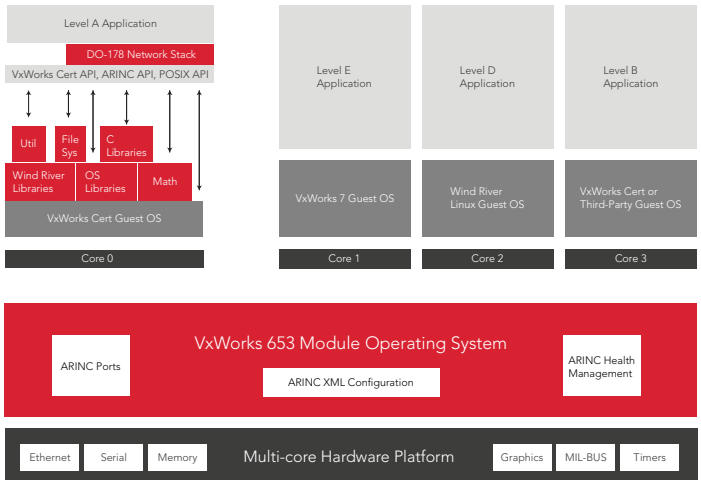

Figure 2. VxWorks 653 RTOS architecture

Temporal Partitioning

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 addresses the problem by defining an implementation that uses partition-based scheduling. A partition is scheduled for a time slot of defined width, and other partitions may be allocated time slots of similar or differing durations. Within a time slot, a partition may use its own scheduling policy, but at the end of the time slot, the ARINC scheduler forces a context switch to the next partition in the schedule. This model is sufficiently flexible to enable existing federated applications or new IMA applications developed in isolation to be hosted on a core module. But partition scheduling and verifying that boundaries and schedules are not violated, and taking appropriate corrective action, inevitably create additional complexity.

In VxWorks 653, the module OS performs ARINC 653 scheduling of the individual partitions. Within each time slot, the partition OS uses the VxWorks scheduler to perform preemptive, priority-based scheduling. This means that all process-level scheduling occurs within the partition space, enabling greater scalability and stability (minimal jitter) in the system even at high system clock rates (e.g., greater than 1 kHz frequency; i.e., 1 millisecond period).

The implementation of VxWorks 653 Platform, Multi-core Edition provides an ARINC 653 Part 1, Supplement 3 environment for applications.[5] It also supports optional mode-based scheduling, where up to 16 schedules can be predefined and used for different modes of flight or for staged initialization. The health monitoring system (HMS) validates the new schedule before being adopted (this feature is discussed later in the section "Health Monitoring System and Restarts").

ARINC 653 APPLICATION DEVELOPMENT

The ARINC 653 APEX, sometimes referred to as the ARINC 653 API, provides a general-purpose interface between the operating system and the application software. The ARINC 653 API also provides an abstraction layer that hides the implementation details of a particular ARINC 653–compliant RTOS from the application and the underlying architecture of the core module. This feature facilitates porting of the application to other ARINC 653 platforms, an important consideration for safety-critical IMA systems.

The ARINC 653 APEX also provides a model of static system configuration and initialization. Here, the number of ARINC

processes is known ahead of time, and they are created in the partition through startup code using the CREATE_PROCESS( ) API. All other partition objects are created from the partition heap, and once this has been done, the partition is activated through a call to SET_PARTITION_MODE(NORMAL). At this point, the partition OS scheduler is activated and schedules the processes within the partition. Once the application has started, no further objects or processes may be created dynamically. This ensures a controlled, deterministic startup sequence for safety-critical applications and deterministic, fixed usage of resources.

ARINC 653 also provides excellent constructs for both intra-partition and inter-partition communication. ARINC 653 blackboards and buffers assist with intra-partition communication. Blackboards provide a convenient write-once/read-many mechanism; buffers provide the ability to send and receive messages that are always stored in first-in, first-out (FIFO) order but permit the receiving process to receive them in either FIFO or priority order. Additionally, semaphores and events can be used for synchronization.

ARINC 653 ports facilitate inter-partition communication. The same naming scheme can be used for ports resident on the same processor or on another core module in the same IMA cabinet or in another IMA cabinet. This prevents applications from making architecture-dependent and configuration-dependent assumptions, aids portability, and eases reconfiguration by the systems integrator. The ARINC 653 ports facility provided by VxWorks 653 also allows the definition and use of a pseudo port, whereby an ARINC sampling or queuing port is connected to a module OS device driver to achieve inter-module communications while presenting a standard ARINC API to the application.

HETEROGENEOUS APPLICATION SUPPORT

Although many IMA applications are developed from scratch, there is a wealth of existing applications on federated systems. These applications may be developed in different programming languages and use different scheduling models but may still need to communicate with each other in an IMA environment.

Wind River has addressed this need by providing support for heterogeneous applications running within separate ARINC partitions. This enables an Ada application using the Ravenscar restricted-tasking profile to run on top of the VxWorks 653 partition OS alongside other C applications in separate partitions (this is discussed at length by Parkinson and Gasperoni[6]). Similarly, a FACE-conformant and POSIX-conformant[7] application can run in separate partitions, and communication between these two heterogeneous applications can be implemented using ARINC 653 ports (see Figure 3).

With the availability of multi-core processors with support for hardware virtualization, an additional avenue of development opens up for IMA platforms, which allows the possibility of running multiple operating environments (referred to as guest operating systems or GOS). These GOS allow a number of options with which it previously would have been considered difficult or impossible to achieve sufficiently safe operation. These configurations can include use of low safety certification level environments, such as commercial or roll-your-own Linux environments, to address systems such as in-flight entertainment, galley controls, and other similar aircraft systems. The other potential use of multi-core and virtualization is to host a legacy OS environment (COTS or in-house developed) to allow application reuse, as well as to serve as an asset bridge to migrate platforms forward while reducing the risks and costs introduced by having to rewrite large portions of those environments and applications all at one time. VxWorks 653 Platform, Multi-core Edition opens the door to these and other possibilities for developers through the use of multi-core processors and their capabilities.

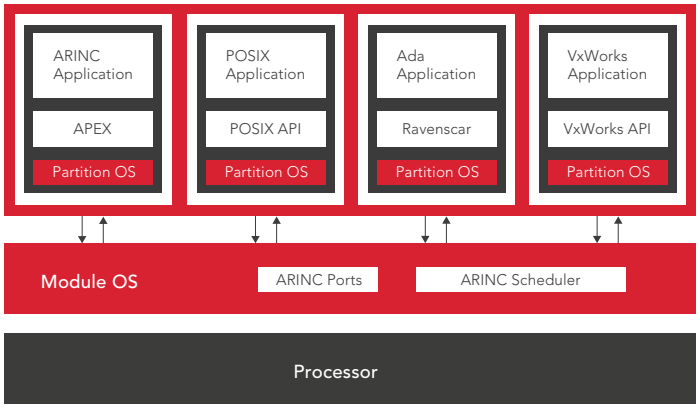

Figure 3. VxWorks 653 heterogeneous application support

SYSTEM CONFIGURATION

The ARINC 653 architecture guarantees resource availability through the use of system and partition configuration records (also known as *system blueprints* in the IMA community). Their purpose is to enable the configuration of IMA applications developed by one or more OEMs onto a shared IMA platform that is configured by the systems integrator. The partition configuration records define the characteristics of each OEM application in terms of memory requirements, processor requirements, and ARINC port utilization. The system configuration record defines the capabilities and capacities of the IMA platform and references and validates individual partition configuration records. This scheme enables the systems integrator to ensure that the demands of the applications are consistent with the performance of the platform and that individual applications do not exceed their allocated resources.

The current revision of ARINC Specification 653 provides only a high-level definition of the structure and content of the configuration records and leaves the implementation up to the RTOS implementer, although an example XML-based configuration is presented. VxWorks 653 uses the following process:

Step 1: At system initialization, the boot code loads the module OS and system and partition configuration records.

Step 2: The module OS initializes itself, starting its own subsystems.

Step 3: The module OS loads the application partitions and their applications.

This process decouples the configuration of the module OS binary image and the system and partition configuration records binary image from the partition applications. In this way, individual applications and subsystems can be developed separately, then integrated easily on the target file system. Individual partition applications can also be upgraded in a straightforward manner without requiring changes to the module OS configuration. This results in significantly less recertification effort and greater flexibility for OEMs and systems integrators.

VxWorks 653 has extended the example XML-based configuration to provide application developers, platform providers, and systems integrators with a complete and qualified toolset, coupled with data files to configure and initialize an IMA platform. This process, referred to as *independent build link and load* (IBLL), reduces the cost of change while providing a fully configurable runtime environment. The process also fully realizes the goals as stated in RTCA DO-297/EUROCAE ED-124, "Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations."[8]

System and partition configurations can be changed without rebuilding the entire application or platform, which significantly reduces the impact-analysis burden for the systems integrator when upgrading and modifying an existing system.

Since the tools used to generate the configuration records in VxWorks 653 generate binary data directly from the XML configuration data, the tool is much simpler to use and therefore more easily qualified than other implementations. Those often rely on more general-purpose mechanisms, such as a C compiler, to generate the binary configuration data for use by the system.

HEALTH MONITORING SYSTEM AND RESTARTS

ARINC 653 defines the concept of a health monitor (HM) within an IMA system. The HM is responsible for "monitoring hardware, application, and operating system faults and failures," and it is the role of the HM to help "isolate faults and to prevent failures from propagating." Though the concept may appear straightforward, it is actually complex, requiring a sophisticated system-wide health monitor to track errors and perform reconfiguration and recovery. The response to an individual fault depends on the nature of the fault, its severity, and the error management policy defined by the systems integrator.

The VxWorks 653 HMS is a sophisticated framework that acts as an intrinsic part of the VxWorks 653 architecture. It fulfills all of the requirements of ARINC 653, and provides extensions relevant to systems integrators intending to use dynamic reconfiguration (in particular, mode-based scheduling). The design and implementation of the VxWorks 653 HMS is sufficiently rich that only an overview can be provided here. The HMS architecture consists of a system-wide HM server and HM agents (called *process-level handlers* in ARINC 653) residing in individual partitions, and the architecture also includes support for the module OS. The HMS processes events occurring in the system that need attention; these are known as *faults*, though they may represent either a negative or a positive event, such as a hardware exception or crossed threshold

(a fault is represented in software by an *alarm*). The framework also supports *messages*, another type of event used for logging or other behavior configured by the systems integrator. Note that HMS use is not bound only to the ARINC support in VxWorks 653, giving added flexibility to the application developer.

The framework provides the ability to perform health monitoring at three levels—process HM, partition HM, and core module HM—through three types of services: alarm detection, alarm logging, and alarm response. The partition HM and module HM are table-driven and provide a mapping between a code and an appropriate handler. To aid portability, the framework uses ARINC 653 definitions for error codes, including events such as missed deadline, numeric error, illegal request, and power fail. XML-based configuration data is used to configure the framework and create the table-driven mapping for runtime usage by the system. Alarm response depends on the error level: module-level responses include reset or shutdown; partition-level responses include the restart of a partition.

At partition creation, a cold start is used to allocate and initialize partition objects, whereas a warm start, which reinitializes but does not allocate objects, is used at partition re-initialization or restart. For process errors, the responses are application driven; the action taken is dependent on the error type and its context. To facilitate warm restarts, VxWorks 653 supports the use of persistent data types that provide for preservation of critical data during the warm start operation. This simplifies the operation and provides a mechanism to increase the speed of startup in such situations.

TOOLS FOR SAFETY-CRITICAL SYSTEMS DEVELOPMENT

Although RTOS runtime functionality is a major consideration, a discussion of IMA application development would be incomplete without reference to simulation, development, and debugging tools. The quality of simulation development and debugging tools can have a dramatic effect on development time scales. Wind River Simics®, a software simulation environment that enables unmodified target binaries to be run on a virtual platform, enables very early prototyping before physical hardware is designed and built.

Tools designed for federated application development may not be suited to IMA development, as they need to support IMA models and scheduling modes. VxWorks 653 Platform provides an integrated development environment with the Eclipse-based Wind River Workbench development suite. This state-of-the-art environment includes project configuration, code browsing and build, target debugging, and the Wind River System Viewer analyzer.

Figure 4 shows Workbench debugging an application running in an ARINC partition. In addition to capabilities provided by Wind River, Eclipse plugins for open source and partner tools can further extend and customize the environment. The dynamic visualization capabilities are a real benefit to an application developer because they provide graphic feedback on the behavior of ARINC 653, POSIX,[7] and VxWorks applications; interactions between partitions; and the operations of the HMS. Workbench can be used to browse, navigate, and comprehend Ada, ARINC 653, POSIX, and VxWorks applications. Wind River System Viewer is partition aware and can display ARINC 653 processes, POSIX threads, and VxWorks tasks running on different cores. It can be invaluable in displaying the internal behavior of an application, as shown by the inter-process communication between ARINC 653 processes through ARINC queuing ports in Figure 5.

It is important for the ARINC 653 application developer not only to visualize the behavior within an individual ARINC 653 partition but also to view application operation in the certifiable environment as well as inter-partition communication through ARINC ports and channels. This capability is achieved through the use of CPU time usage monitoring, memory usage monitoring, and port monitoring tools built into the RTOS and certified as part of the VxWorks 653 runtime system. These RTOS monitors are then coupled with RTCA DO-178B/EUROCAE ED-12B and RTCA DO-330/EUROCAE ED-215 qualified host tools for display and logging of the data in the test for credit environment. The monitors and tools provide unprecedented levels of insight into the operation of the system from development environment through final certified flight configuration.

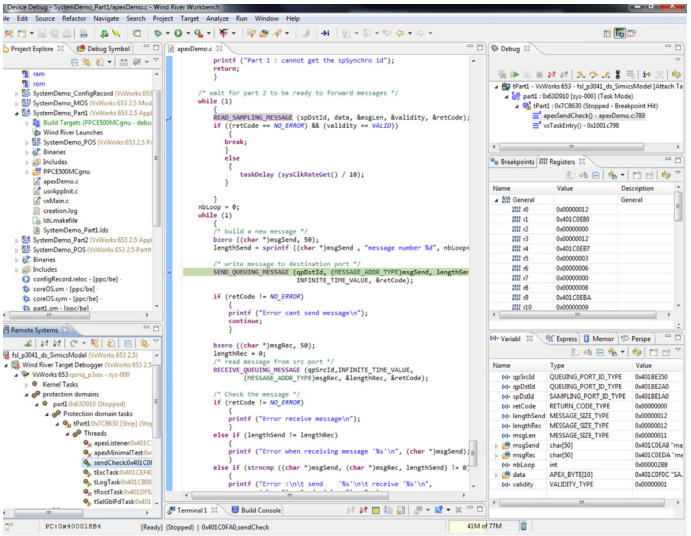

Figure 4. Wind River Workbench showing VxWorks 653 partition debugging

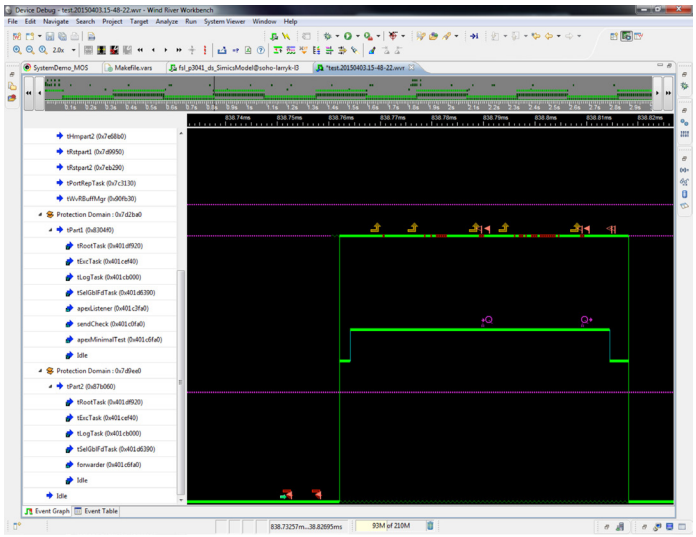

Figure 5. Wind River System Viewer showing ARINC 653 partition behavior

SECURITY CONSIDERATIONS FOR NETWORKED IMA SYSTEMS

Cyber security is now becoming a greater consideration in avionics systems, with guidance being provided by RTCA DO-326A/EUROCAE ED 202A.[9] Security at the aircraft level can be achieved through the use of firewalls that restrict interactions between different types of subsystems and separate flight systems from OEM systems and airline systems.[10] However, with the advent of IMA, there is a drive to increase—in a certifiable manner—the networking connectivity within these domains. This goal presents some interesting design challenges.

The VxWorks 653 RTOS implements a number of capabilities using the security principle of least privilege to protect applications and data from malicious behavior. These include, but are not limited to, the following:

- Running application partitions in the processor's user mode, so that they are unable to execute privileged processor instructions
- Passing the partition OS (if it is unable to service an APEX call on behalf of an application) to the module OS, which performs validation of the address memory ranges, which should be within view of the partition boundary checks, module OS object access rights, and data structure integrity/consistency checking
- Providing a scalable system-call privilege mechanism, whereby one partition can have more authority than others (which provides a foundation to meet the requirements for high-assurance systems needing to undergo security evaluation)
- Implementing security restrictions in relation to the HMS, so that, for example, only a privileged partition acting as a mode manager for the system may request an ARINC 653 schedule change

TCP/IP and related networking protocols require considerable effort to certify, and systems designers have to achieve a balance between functionality and the suitability for certification. In particular, RTCA DO-178C Development Assurance Level (DAL) A certification of a full TCP/IP stack is rather onerous. Some have advocated the use of proprietary implementations utilizing a slave processor to implement the network stack for the master processor. However, this custom configuration uses additional hardware and restricts software portability. Since this defeats two of the goals of IMA outlined earlier, it can only be viewed as a retrograde step.

The Wind River approach is to provide the optional DO-178 Network Stack, which runs on the same processor as the RTOS and implements TCP/UDP/IPv4 capabilities and is available with an RTCA DO-178C/EUROCAE ED-12C DAL A certification evidence package. The DO-178 Network Stack provides a level of functionality most avionics customers require while still providing an extensible framework that provides the potential to integrate additional protocols as required.

The security capabilities in the VxWorks 653 Platform product can be augmented with additional cybersecurity capabilities to address specific customer application use case requirements. This can include the tailoring and integration of specific capabilities from Wind River Services' Information Assurance Foundation portfolio to implement specific confidentiality, integrity, and availability requirements related to RTCA DO-326A, DO-355, and DO-356, or domain-specific requirements. Wind River Services has previously integrated IA Foundation with VxWorks 653 Multi-Core Edition on a multi-core processor platform[11] and with other Wind River operating environments.

SAFETY CONSIDERATIONS FOR IMA SYSTEMS

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 existing certification standards.[12] For example, the concept of reusable software components with RTCA DO-178C / EUROCAE ED-12C certification evidence in a safety-critical application is well documented[13] and has been applied successfully to VxWorks certification in a federated application on an FAA program.[14]

The decoupling of the module OS and partitions in VxWorks 653 through the use of spatial partitioning enables this concept to be extended further. Now, a VxWorks application that has been previously certified to DAL C can be used in a separate partition on the same IMA platform as a new DAL A application, without the DAL C application needing to be recertified to Level A. This technique can also be applied to I/O drivers and networking stacks (such as TCP/IP). These are placed into a separate VxWorks 653 I/O partition that is isolated from both the module OS and application partitions. Communication with application partitions is achieved through the use of ARINC ports, and interaction with the module OS is restricted, preventing uncertified code from affecting the correct operation of the module OS or application partitions (see Figure 7).

At the time of writing, the FAA and EASA have not yet issued formal policy on safety certification on multi-core processor architectures. However, EASA released the MULCORS[15] report on multi-core research, and the FAA released the CAST-32A position paper[16] on multi-core processors that describes a number of objectives for use of multi-core processors in a safety-critical environment and in risk areas. Wind River took these objectives into consideration when developing the DO-178C DAL A certification package for VxWorks 653 Multi-core Edition in order to minimize certification risk on multi-core programs.

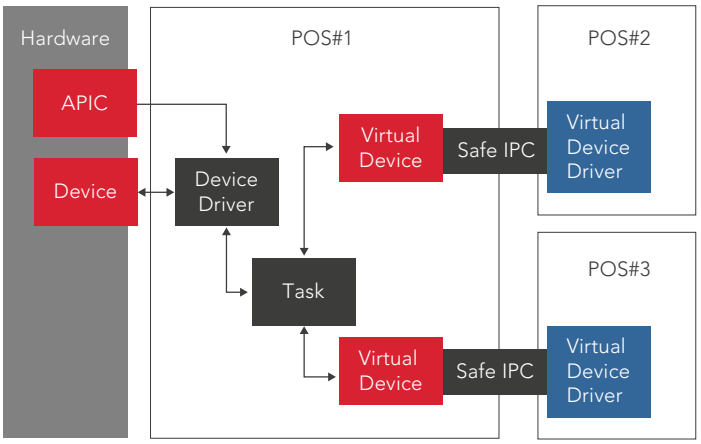

Figure 6. VxWorks 653 device driver model

SUMMARY

The avionics industry is in the midst of a major shift toward IMA, even though the continued evolution of IMA architectures and standards presents challenges for standards organizations, OEMs, and commercial vendors alike. VxWorks 653 Platform provides an integrated device software platform that brings together a standards-compliant COTS RTOS and all of the tools needed to successfully develop safety-critical IMA applications. The platform not only enhances developer productivity, it also makes sure that the complexity and effort involved in certification does not impinge on developers. In addition, the heterogeneous support for ARINC 653, Ada, POSIX, FACE, and VxWorks applications in an IMA environment facilitates maximum software reuse and porting of existing federated applications to VxWorks 653. With VxWorks 653 Platform, Multi-core Edition, this feature is now extended to legacy platforms and OS environments by employing multi-core processors and hardware virtualization to serve as an asset bridge for migration of both federated and IMA legacy platforms.

REFERENCES

1. RTCA DO-255/EUROCAE ED-96, "Requirements Specification for Avionics Computer Resource (ACR)." www.rtca.org and www.eurocae.net.
2. ARINC Specification 653, "Avionics Application Software Standard Interface." www.arinc.com.
3. "Major Organizations Achieve FACE™ Conformance as Program Gains Momentum," press release, The Open Group, 7 March 2017. http://www.opengroup.org/news/press/Major-Organizations-Achieve-FACE%E2%84%A2-Conformance-as%20Program-Gains-Momentum.
4. John Rushby, DOT/FAA/AR-99/58, "Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance," March 2000.
5. ARINC Specification 653 P1-3, "Avionics Application Software Standard Interface," November 15, 2010. www.arinc.com.
6. P. Parkinson, F. Gasperoni, "High-Integrity Systems Development for Integrated Modular Avionics Using VxWorks and GNAT," 7th International Conference on Reliable Software Technologies, Ada-Europe, 2002. http://link.springer.com/chapter/10.1007/3-540-48046-3_13.
7. POSIX Specification, ANSI/IEEE POSIX 1003.1-1995; ISO/IEC standard 9945–1:1996.
8. RTCA DO-297/EUROCAE ED-124, "Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations." www.rtca.org and www.eurocae.net.
9. RTCA DO-326A/EUROCAE ED-202A, "Airworthiness Security Process Specification." www.rtca.net.
10. Jean Paul Moreaux, EADS-Airbus, "Evolution of Future Aircraft Data Communications," NASA Workshop on Integrated CNS Technologies, May 2001.
11. A. Baker, P. Parkinson, "Cybersecurity Enhancements for an ARINC 653 Safety-Critical Avionics Platform," Proceedings of the 26th Safety-Critical Systems Symposium, February 2018. https://www.researchgate.net/publication/323019107_Cybersecurity_enhancements_for_an_ARINC_653_safety-critical_avionics_platform.
12. RTCA DO-178C/EUROCAE ED-12C, "Software Considerations in Airborne Systems and Equipment." www.rtca.org and www.eurocae.net.
13. FAA Draft Notice, N8110 RSC. www.faa.gov.
14. "Raytheon to Deliver Advanced Wide Area Navigation System to the FAA for Certification using Wind River's VxWorks Operating System," Wind River, February 5, 2002. www.windriver.com/news/press/pr.html?ID=393.
15. "MULCORS—Use of MULticore proCessORs in airborne Systems," Final Report, Research Project EASA. 2011/6, December 16, 2012. easa.europa.eu/system/files/dfu/CCC_12_006898-REV07%20-%20MULCORS%20Final%20Report.pdf.
16. "Multi-core Processors," Certification Authorities Software Team (CAST) Position Paper CAST-32A, FAA, November 2016. www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/media/cast-32A.pdf.

About the Author

Paul Parkinson is a principal systems architect for aerospace and defense with Wind River in the U.K., where he works with aerospace and defense customers. Paul is a chartered engineer (CEng) and a Fellow of the Institution of Engineering and Techology (FIET). Paul's professional interests include IMA, intelligence surveillance target acquisition and reconnaissance (ISTAR) systems, and information security (InfoSec).

WIND