# Enabling the Migration to Future Aerospace and Defense Systems

*Paul Parkinson*
*Director, Field Engineering, Aerospace & Defense, EMEA*

WHEN IT MATTERS, IT RUNS ON WIND RIVER

## EXECUTIVE SUMMARY

Embedded software used in aerospace and defense (A&D) systems is currently undergoing a dramatic evolution. Historically, these A&D systems had varying levels of network connectivity, performed fixed functions, and may have undergone manual software updates in the field as part of long in-service lifetimes. The advent of ubiquitous network connectivity through civilian and military communications infrastructure has accelerated innovation of these systems, known as edge devices, at the edge of the network. This has driven demand for enhanced application capabilities that perform traditional automation and control functions while adding greater intelligence and the ability to support more dynamic behavior.

There is now a growing demand across the aerospace and defense sector for devices to be able to support the greater intelligence required to transition from automated to autonomous systems. This is driving technical requirements for open standards–based software-defined architectures to enable consolidation of multiple applications, including those at different levels of criticality, onto common computing platforms. This approach enables application migration, portability, and interoperability in order to avoid being locked into proprietary architectures.

Wind River® Helix™ Virtualization Platform addresses these common requirements by providing a flexible virtualization platform that supports open standards–based software-defined architectures. Based on proven software technologies, it enables systems to meet civil and military software safety certification requirements.

## TABLE OF CONTENTS

## EVOLUTION OF EMBEDDED SYSTEMS FROM AUTOMATION TOWARD AUTONOMY

Over the last decade, embedded systems have undergone a continual evolution in terms of performance, connectivity, and capability. Historically, at one end of the capability continuum, some embedded devices have performed fixed functions and had long in-service deployment lifetimes. They may rarely have undergone system upgrades that added new functionality or deployed security updates to address the latest security vulnerabilities. Systems without any network connectivity may have needed to be manually updated in the field. This could be very time-consuming, error-prone, and expensive, depending on the number of devices deployed and the level of difficulty of access to those actual physical devices.

Processor performance has increased dramatically for many years, per Moore's Law, and the cost of network connectivity has continued to fall, resulting in a dramatic increase in edge device connectivity. This has accelerated innovation, enabling new application functionality to be deployed more rapidly to edge devices using secure communications sessions over network infrastructure. This foundation is now enabling the development and deployment of the next generation of intelligent devices and the transition from automation to autonomy.

In the commercial and defense aerospace sector, there has been a significant trend in avionics design, with a marked shift away from federated avionics architectures using fixed-function line replaceable units (LRU) to integrated modular avionics (IMA) architectures employing common computing platforms hosting multiple applications at different levels of safety criticality. This transition has been driven by a common requirement to significantly reduce the overall size, weight, and power (SWaP) requirements of avionics systems, particularly as the number of federated LRUs have increased in order to support new avionics functions. The adoption of IMA has resulted in significant reduction in weight, enabling aircraft to fly with reduced fuel load, or with more passengers or payload, respectively. The use of standards-based IMA software architectures to enable interoperability and integration has also had a significant positive impact on both commercial and military programs, reducing the risk of design lock-in, driving innovation, and reducing whole-life costs.

The military land vehicle and maritime sectors face similar challenges in relation to SWaP—the requirements for more functionality and enhanced capabilities being in tension with the fixed limited space on land vehicles and on ships and submarines.

Finally, the deployment of autonomous driving systems will require significant increases in on-board computing power, including general-purpose graphics processing units (GPGPUs), in order to support artificial intelligence (AI) and machine learning (ML) applications. In the case of military land vehicles, these systems, along with their associated cabling, have the potential to increase the weight of the vehicles, which would have a negative effect on vehicle performance, fuel economy, range, and cost (just as in the aerospace market). As a result, land vehicle systems programs are now using common embedded computing platforms to consolidate applications and reduce SWaP.

## THE NEED FOR A CONVERGED EMBEDDED VIRTUALIZATION PLATFORM

Regardless of the new capabilities aerospace and defense suppliers are trying to fit into their systems, there will always be a common need for a safe, secure, and reliable software platform that follows the same high-level requirements:

- **Mixed-criticality:** A common requirement for a consolidation platform is to host both critical real-time control applications and general-purpose applications. It must isolate safety components and reduce their dependencies on the rest of the platform in order to ensure that the safety-certification costs of the overall platform remain affordable.
- **Open standards:** There is an increasing requirement for each sector to adopt open standards to enable application migration and portability as well as competition, and to prevent design lock-in. The commercial aerospace sector has embraced the ARINC 653 standard for open avionics architectures to meet these requirements. The U.S. government has adopted a similar philosophy through Modular Open Systems Approach (MOSA). The Future Airborne Capability Environment (FACE™) Technical Standard has also been developed to provide an open avionics architecture for military avionics that uses commercial open standards (ARINC 653, POSIX®). This enables a greater adoption of commercial off-the-shelf (COTS) hardware and software and provides a flexible and scalable framework.

- **Reuse and scalability:** This involves the ability to reuse intellectual property and previously developed applications (and in some use cases, previously certified applications) on new embedded computing platforms. This includes utilizing the performance and scalability provided by multi-core processor architectures with hardware virtualization support, while managing and isolating the application developer from the complexity of the underlying hardware architecture, and also providing the ability to enable the deployment of new applications on the platform to deliver additional capabilities.
- **Security:** There is an increasing need to ensure security throughout the lifecycle of a device. This includes design, production, and commissioning, and also secure operation through deployment and decommissioning at the end of the life cycle. During deployment, the device needs to operate securely in different phases: It needs to perform secure initialization to verify the integrity of the deployed software, to ensure that it has not become corrupted or tampered with; it needs to provide secure communications during operations and resilience against cyberattacks; and it needs to securely store sensitive data in process and in data at rest, including when powered down.
- **Support for modern development methods:** Aerospace and defense suppliers are now under pressure to deliver new, high-quality software capabilities within shorter time scales. Many companies are now adopting agile development processes to accelerate the pace of change and are employing continuous integration (CI) and continuous delivery (CD) approaches through automation to dramatically reduce their integration and delivery time scale, as part of a DevOps culture to increase responsiveness to customer and program needs.

## AEROSPACE SYSTEMS LANDSCAPE

During the last decade, there have also been advances in processor technologies, which have had a significant impact on deployed architectures; in particular, the evolution and maturity of processor hardware virtualization support has enabled enterprise and cloud computing platforms to host virtualized applications efficiently and at scale. Hardware virtualization support has also been successfully deployed in embedded systems, particularly in safety-critical avionics systems using VxWorks® 653 Multi-Core Edition, where multiple virtualized applications can be hosted on a common avionics computing platform. This convergence of hardware virtualization, multi-core processor architecture, and open standards–based software-defined architecture provides the opportunity for a common embedded virtualization platform for aerospace and defense systems.

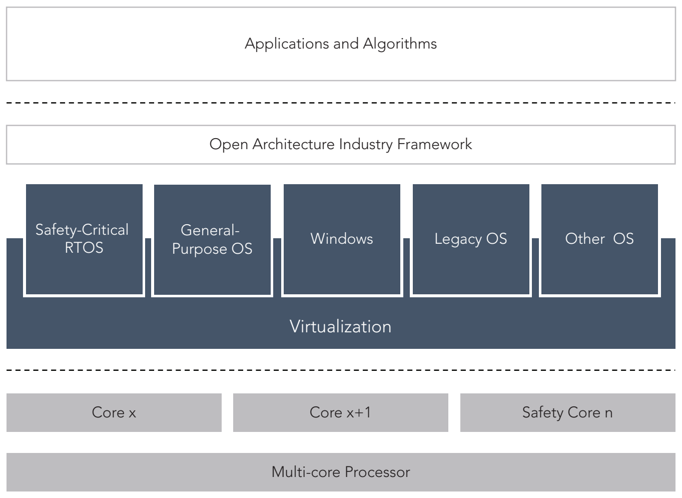


*Figure 1. Software-defined architecture for common computing platforms*

Wind River has addressed this market need by leveraging decades of experience and continuing industry leadership in safety-critical, security, and embedded hypervisor software technologies to develop Helix Platform.

### Hypervisor and Virtual Machines

Helix Platform uses a Type 1 hypervisor (also known as a bare metal hypervisor), as shown in Figure 2, which runs directly on the processor, providing near-native real-time performance using direct interrupts and providing the scalability, determinism, and small footprint suitable for safety certification. This approach contrasts with Type 2 hypervisors, which focus on abstraction from the underlying physical environment and provide best-effort performance that is not suitable for hard real-time applications. Additionally, the footprint is too large to undergo safety certification. Alternatively, the Type 1 hypervisor approach provides efficient scalability on multi-core processor architectures, since it can scale to a higher number of cores compared to traditional monolithic and micro-kernel architectures.

The hypervisor uses the processor's dedicated hypervisor privilege level and full hardware virtualization support, enabling 32-bit and 64-bit guest operating systems and associated applications to run at separate privilege levels inside virtualized environments (commonly known as virtual machines). This includes VxWorks, Wind River Linux, Microsoft® Windows® (64 bit), Android, bare metal, and third-party operating systems, including other Linux flavors.

The hypervisor uses the processor's memory management unit (MMU) to enforce isolation of individual virtual machines. This prevents a guest operating system (OS) and its associated application from making any unauthorized programmed I/O accesses to another virtual machine or privileged system resource, and any attempted unauthorized accesses are reported to the hypervisor.

Many modern processors also provide direct memory access (DMA) engines to perform efficient transfers of blocks of data between source and destination memory locations, for example between system memory and external I/O devices such as network interfaces. The hypervisor also uses another processor hardware resource, the IOMMU, to ensure that the guest OS and applications only perform DMA transfers between authorized source and destination addresses. This means that Helix Platform is able to host a virtual machine containing a guest OS that uses a third-party device driver of unknown pedigree, in the knowledge that an unauthorized DMA access will be detected and prevented by the IOMMU. This isolation capability provides benefits in terms of safety and security, and it supports the consolidation of multiple applications onto a common embedded processing platform.

These hardware virtualization and isolation capabilities enable Helix Platform to host real-time and safety-critical VxWorks applications alongside embedded Linux applications and even a general-purpose OS, such as Microsoft Windows and other third-party and legacy operating systems. This enables previously developed software to be rehosted, preserving investment in existing intellectual property and serving as an asset bridge between existing deployed systems and next-generation systems, thereby reducing technical risk when upgrading in-service systems.

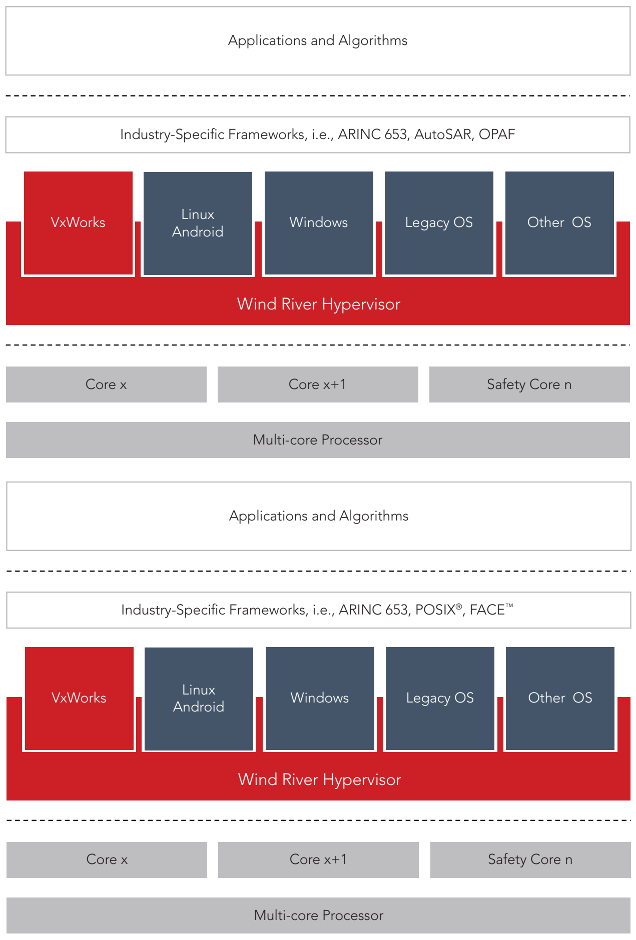


*Figure 2. Helix Platform architecture*

### Inter-partition Communication

A consolidation platform also needs to support communication between individual virtual machines, and Helix Platform provides a number of different inter-partition communication methods:

- Shared memory and VNIC (Virtualized Network Interface Controllers) enable communication between VxWorks, Linux, and Windows guest operating systems as well as other third-party guest operating systems.
- Safe IPC is a controlled shared memory implementation to enable communication between multiple guest operating systems, potentially running at different levels of criticality.

### Static and Dynamic Configuration

Helix Platform builds on the capabilities of safe and secure Wind River platforms that have a world-class certification track record under multiple certification standards across different industry sectors. It has evolved the proven code base to support a broader range of use cases, including both static and dynamic behavior, as shown in Figure 3.

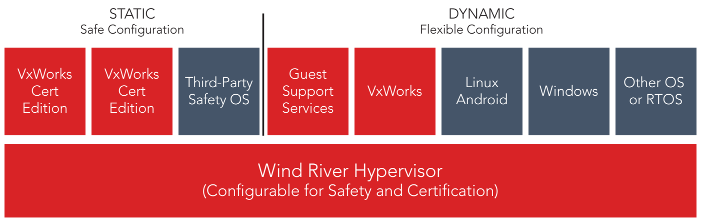


*Figure 3. Static and dynamic configuration*

Static configuration is used for systems that contain a safety-critical application, enabling system resources to be allocated at system configuration and build time, then used in a predictable and deterministic manner at runtime. The robust partitioning described earlier means that this configuration can support multiple safety-critical applications running in different partitions, or mixed-criticality systems comprising multiple applications at different integrity levels. The Independent Build Link and Load (IBLL) approach, pioneered by Wind River in VxWorks 653, enables applications to be configured, built, linked, and loaded independently. This enables individual applications to be updated independently throughout the platform lifecycle without impacting other components, enabling incremental certification and dramatically reducing whole-life costs for the platform.

Dynamic configuration is used for systems that require greater flexibility in terms of configuration of applications and system resources, and that may need to dynamically change system configuration and runtime operation to handle an increase in service requirements or in response to the external environment or mode of operation. This approach enables a wide range of guest operating systems and applications to be hosted, including those that have less stringent requirements for determinism than a safety-critical guest OS that uses a static configuration.

### Scheduling

Helix Platform also supports two distinct types of scheduling in order to support a broad range of application use cases:

- **Priority preemptive scheduling:** This scheduling type uses threads running in the hypervisor, with one hyper-thread per guest OS per core. This approach enables an individual guest OS to be scheduled on a priority-based preemptive basis (as shown in Figure 4) without affecting the scheduling within an individual guest OS, which uses its own local scheduling policy. This scheduling model enables applications with dynamic behavior and event-driven systems to be hosted on Helix Platform.

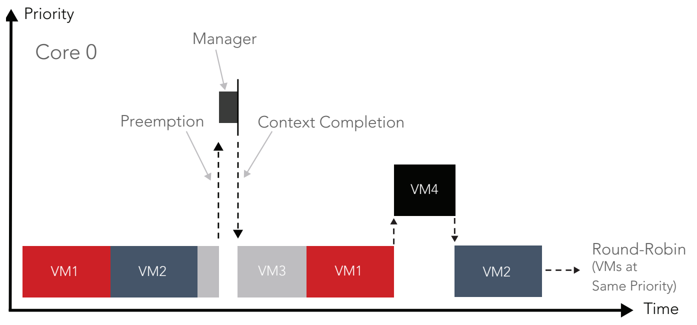


*Figure 4. Priority preemptive scheduling*

- **Frame scheduling:** This form of scheduling uses a system schedule of multiple individual time slots of fixed durations that define how long individual guest operating systems and their associated applications can execute. The frame scheduler can be used to ensure that a general-purpose application does not overrun its allocation time period and adversely impact the execution of a safety-critical application hosted on the same platform (as shown in Figure 5). Helix Platform also allows the schedules for individual cores to be synchronized. The frame scheduler enables multiple schedules to be defined at system configuration and build time, and it allows the transition from one schedule to another through the use of a dedicated hypervisor call, known as a hypercall, by a trusted guest OS partition. This support for multiple schedules enables use of different operational modes, e.g., initialization mode, operational mode, and maintenance and/or diagnostic mode. The frame scheduling model is widely used in safety-critical avionics systems (using an ARINC 653 scheduler), but this approach is also equally applicable to safety-critical systems in other market sectors needing to host general-purpose and safety-critical applications on the same computing platform.

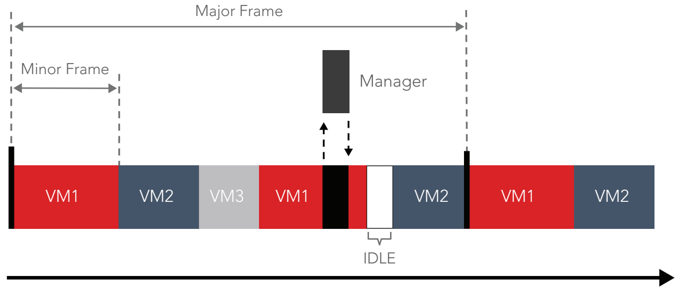


*Figure 5. Frame scheduling*

**Health Monitoring Framework**

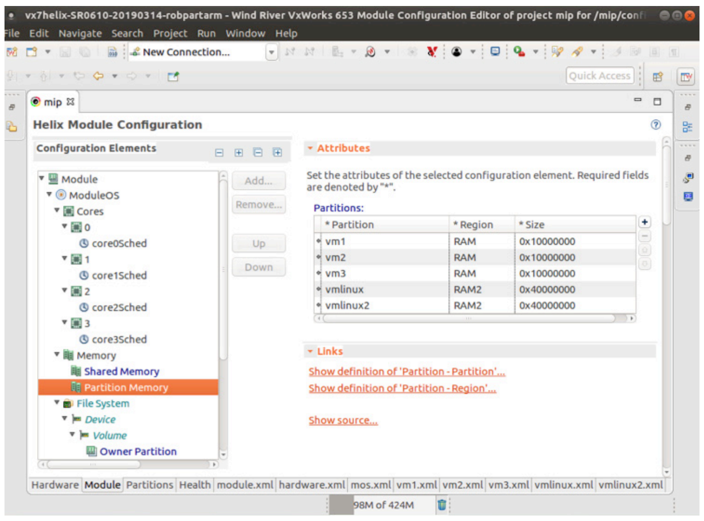
A system containing multiple heterogeneous virtual machines needs to be able to monitor, detect, and recover from hardware faults, guest OS faults within individual virtual machines, and application failures. This requires that a health monitoring framework is able to isolate faults and prevent failures from propagating. Although these requirements may appear to be straightforward conceptually, they are actually complex and require a sophisticated system-wide implementation in order to provide continued availability of the platform. Wind River has a proven track record in developing an ARINC 653 Health Monitoring (HM) framework for safety-critical IMA applications and has used this expertise in the development of the HM framework for Helix Platform.

The HM framework handles events that can be either an alarm, representing a fault that needs attention, or a message, providing a notification that can be processed or recorded. The HM framework uses a hierarchical table-driven implementation, which enables HM events to be handled at the application level, virtual machine level, or the embedded platform level (these are referred to as process level, partition level, and module level in the aerospace sector). The HM framework also allows HM events to be handled at the level at which they occur or passed to the next level; e.g., a fault within an individual virtual machine is contained, and the error handling could be routed from the virtual machine's guest OS to the hypervisor. This approach enables the system integrator to determine, at the time of system configuration, the course of action to take in the event of specific errors. The HM framework also provides support for cold and warm restarts at the virtual machine level and embedded platform level. This enables individual virtual machines to be restarted without interfering with other virtual machines.

## DEVELOPMENT TOOLS TO ENABLE FUTURE SYSTEMS

An embedded virtualization platform promises many benefits. In order to fulfill these promises, sophisticated development toolset support is needed to manage the complexity of system configuration and the building and development of applications for heterogeneous runtime environments.

Wind River has a proven track record in the development of integrated development environments, dynamic visualization tools, and simulation platforms for general-purpose and safety-critical applications. This expertise has resulted in the latest Wind River Workbench development environment for Helix Platform, which provides graphical support for system definition and configuration, including allocation of virtual machines to processor cores, and definition of schedules. Wind River Workbench performs continuous validation of system configuration in the background, providing feedback to the developer during system configuration. It also automates many of the build and configuration steps, reducing the burden on the system integrator and application developer.


vx7helix-SR0610-20190314-robpartarm - Wind River VxWorks 653 Module Configuration Editor of project mip for /mip/conf

File Edit Navigate Search Project Run Window Help
New Connection...
Quick Access
mip
Add...
Remove...
Up
Down
Hardware Module Partitions Health module.xml hardware.xml mos.xml vm1.xml vm2.xml vm3.xml vmlinux.xml vmlinux2.xml
98M of 424M

Helix Module Configuration

Configuration Elements

- Module
  - ModuleOS
    - Cores
      - 0
        - core0Sched
      - 1
        - core1Sched
      - 2
        - core2Sched
      - 3
        - core3Sched
    - Memory
      - Shared Memory
      - Partition Memory
    - File System
      - Device
        - Volume
          - Owner Partition

Attributes

Set the attributes of the selected configuration element. Required fields are denoted by "*".

Partitions:

| * Partition | * Region | * Size |
|---|---|---|
| vm1 | RAM | 0x10000000 |
| vm2 | RAM | 0x10000000 |
| vm3 | RAM | 0x10000000 |
| vmlinux | RAM2 | 0x40000000 |
| vmlinux2 | RAM2 | 0x40000000 |

Links

Show definition of 'Partition - Partition'...
Show definition of 'Partition - Region'...
Show source...

*Figure 6. System configuration and application development using Wind River Workbench*

The patented Wind River System Viewer also provides graphical representation of the system, enabling the developer to view and understand the interaction between system events such as interrupts, guest OS primitives, VxWorks application tasks, POSIX threads, or ARINC 653 processes.

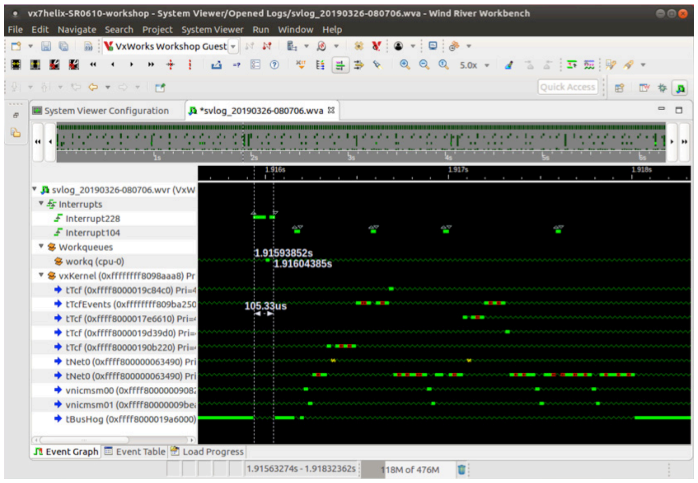

vx7helix-SR0610-workshop - System Viewer/Opened Logs/svlog_20190326-080706.wva - Wind River Workbench
File Edit Navigate Search Project System Viewer Run Window Help
VxWorks Workshop Guest
System Viewer Configuration
*svlog_20190326-080706.wva
svlog_20190326-080706.wvr (VxW
Interrupts
Interrupt228
Interrupt104
Workqueues
workq (cpu-0)
vxKernel (0xffffffff8098aaa8)
tTcf
tTcfEvents
tNet0
vnicmsm00
vnicmsm01
tBusHog
1.91593852s
1.91604385s
105.33us
Event Graph
Event Table
Load Progress
1.91563274s - 1.91832362s
118M of 476M

*Figure 7. Dynamic visualization using Wind River System Viewer*

Helix Platform also utilizes the modern Clang/LLVM compiler toolchain. This provides multiple benefits, including support for the latest C and C++ language standards and performance benefits, including fast compilation times, low memory usage, and faster execution from increased code optimization.

Applications can be developed using Helix Platform on an embedded target platform or using Wind River Simics® full-system simulator. Simics breaks the dependency on hardware availability and enables virtual prototyping to be undertaken earlier in the development lifecycle, reducing costly rework in later stages. Simics also enables many test cases to be run in parallel and at scale, supporting the development of applications for Helix Platform using continuous integration/continuous delivery (CI/CD).

## SUMMARY

Legacy aerospace systems are facing significant challenges in terms of functionality, maintainability, and obsolescence. Future systems also have demanding requirements in terms of software-defined architectures and the need to support open standards. Meanwhile, critical infrastructure systems increasingly need to meet stringent safety certification requirements.

Helix Platform addresses these challenges by providing an open virtualization platform that can act as an asset bridge, enabling the consolidation of legacy applications and previously developed intellectual property on a modern, scalable platform. It supports the development of new applications based on open standards using a diverse range of operating environments. It also provides flexibility through static and dynamic configuration options, enabling hosting and deployment of a broad range of use cases with mixed criticality, using modern development methods and processes.

For more information on Helix Platform, visit [www.windriver.com/announces/helix-platform](www.windriver.com/announces/helix-platform).