



Safety Islands: Enabling Safe Software-Defined Vehicles

Issue 1.0 - September 16, 2026

Contents

Contents.....	2
List of figures.....	3
List of notation.....	3
 CHAPTER 1 Introduction.....	 4
 CHAPTER 2 The Evolution of SDV Architectures.....	 5-6
2.1 Zonal Controllers.....	6
2.2 Central Compute Platforms.....	6
 CHAPTER 3 The Need for Safety Islands.....	 7-10
3.1 Safety Islands	7
3.2 How Safety Islands Work.....	8
3.3 Safety Island Implementation.....	8
3.4 Why Has Cortex-R52 Become the Preferred Safety-Island Processor?.....	8-9
3.5 Cybersecurity & Safety.....	9
3.6 Software Requirements for a Safety Island.....	10
 CHAPTER 4 Introducing SAFERTOS®.....	 11
4.1 Supporting ISO 26262 Compliance.....	12
 CHAPTER 5 What this means for OEMs and Tier 1 Suppliers.....	 13
 CHAPTER 6 Beyond Automotive	 14
6.1 Humanoid Robotics	14
6.2 Industrial Autonomy & Agricultural Automation.....	14
6.3 Medical Systems.....	14
 CHAPTER 7 Conclusion.....	 15
Contact Information.....	16

List of Figures

Figure 1 Traditional distributed ECU architecture5

Figure 2 Centralized SDV architecture5

Figure 3 Safety Island Operation within a Centralized SDV Platform9

List of Notation

ADAS	Advanced Driver Assistance Systems	GPU	Graphics Processing Unit
		OEM	Original Equipment Manufacturer
ASIL	Automotive Safety Integrity Level	RTOS	Real Time Operating System
ECU	Electronic Control Unit	SDV	Software-Defined Vehicle
CPU	Central Processing Unit	WHIS	WITTENSTEIN high integrity systems

References

1. Meet the new R1S and R1T by Rivian - Rivian Stories | Electric Vehicle Adventures

2. ISO 26262: Road Vehicles — Functional Safety.

3. ISO/SAE 21434: Road Vehicles — Cybersecurity Engineering.

4. Koon, J. For SDVs, Software Is The Biggest Challenge. Semiconductor Engineering, November 2023.

5. Issues Facing Automotive Software Developers, WITTENSTEIN high integrity systems. September 2017.

CHAPTER 1 Introduction

The transition to Software-Defined Vehicles (SDVs) is fundamentally transforming automotive electronic architectures. Building on themes explored in our previous white paper, Issues Facing Automotive Software Developers [5], the industry is moving away from distributed systems built around numerous dedicated Electronic Control Units (ECUs) and toward centralized compute platforms and zonal architectures capable of supporting increasingly sophisticated software-defined functionality.

While this shift enables capabilities such as over-the-air updates, Advanced Driver Assistance Systems (ADAS), and autonomous driving, it also introduces new functional safety challenges. As safety-critical and non-safety applications are consolidated onto shared computing platforms, ensuring freedom from interference between them becomes essential.

As compute becomes centralized, safety must become independent.

As centralized computing becomes mainstream, independent safety supervision is increasingly a prerequisite for SDV architectures rather than an optional design choice. Beyond functional safety, independent safety supervision also delivers important business and engineering benefits. By separating safety-critical supervision from application software, OEMs can reduce certification scope, support parallel development teams, simplify platform evolution, and accelerate deployment of new, software-defined features throughout the vehicle lifecycle.

In many respects, the long-term success of the Software-Defined Vehicle depends on the industry's ability to separate innovation from safety, allowing both to evolve without compromising the other.

Safety islands have emerged as the architectural mechanism that makes this possible and are increasingly shaping the way next-generation vehicle platforms are designed. By providing an independent domain for safety supervision, fault management, and safe-state control, they enable manufacturers to pursue aggressive software innovation without compromising functional safety.

Many modern automotive safety-island implementations are built around dedicated real-time processors such as Arm Cortex-R52, which provide the deterministic execution, isolation, and fault-management capabilities required for independent supervision. As vehicle architectures continue to evolve, safety islands are becoming a foundational element of modern automotive system design.

This paper examines the role of safety islands in Software-Defined Vehicles and explains why safety-certified RTOS technologies such as SAFERTOS® are increasingly being used to implement these critical supervisory functions.

Safety Island Adoption in Modern SoCs

Safety islands have become a common architectural feature across leading automotive semiconductor platforms. Examples include:

- Texas Instruments Jacinto: Dedicated safety MCU domains for safety monitoring and fault management.
- Qualcomm Snapdragon Ride: Cortex-R52-based safety processors supporting ASIL-D supervision.
- NVIDIA DRIVE Orin & Thor: Independent safety processing subsystem for autonomous driving workloads.
- NXP S32G and S32Z/E: Integrated safety cores supporting zonal and vehicle-control architectures

These examples highlight how independent safety supervision has become a common architectural pattern across modern automotive compute platforms, regardless of semiconductor vendor.

CHAPTER 2 The Evolution of SDV Architectures

The automotive industry is transitioning from traditional ECU-based architectures to Software-Defined Vehicles (SDVs), where software increasingly determines vehicle functionality, performance, and user experience. This shift is driven by the need for greater flexibility, faster feature deployment, and support for advanced technologies such as autonomous driving and connected services.

Historically, vehicles relied on dozens of dedicated Electronic Control Units (ECUs), each responsible for a specific function. As vehicle functionality expanded, this approach led to increased hardware complexity, software integration challenges, and higher costs.

To overcome these limitations and reduce costs, manufacturers are adopting centralized computing platforms and zonal controllers. High-performance centralized compute platforms now consolidate workloads that were previously distributed across many ECUs, while zonal controllers manage local sensors and actuators. This reduces wiring, simplifies system design, and enables more efficient software deployment.

The benefits of zonal architectures can be substantial. For example, Rivian has reported [1]:

“Shifting to a zonal system enabled us to remove over 1.6 miles (2.6km) of wiring from each vehicle, shedding 44 lbs (20kg) of weight. This reduction in wiring, along with the elimination of 10 ECUs, greatly reduces production costs. Fewer components needing to be installed during assembly also improves manufacturing efficiency, allowing us to build vehicles more quickly and with better quality. With less components onboard, vehicles are easier to service and require us to stock fewer parts, improving serviceability.”

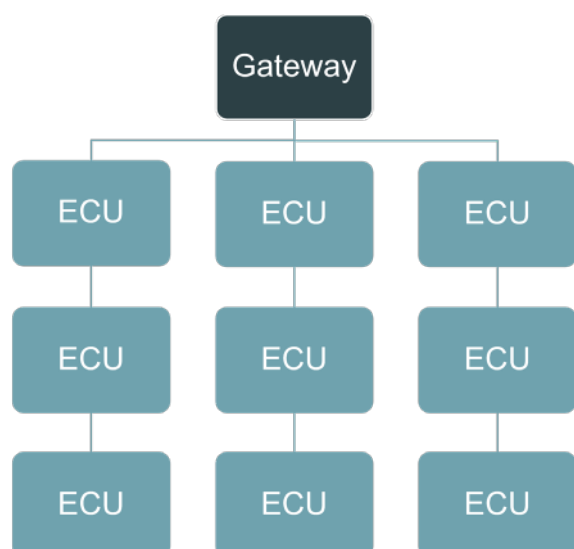


Figure 1. Traditional distributed ECU architecture

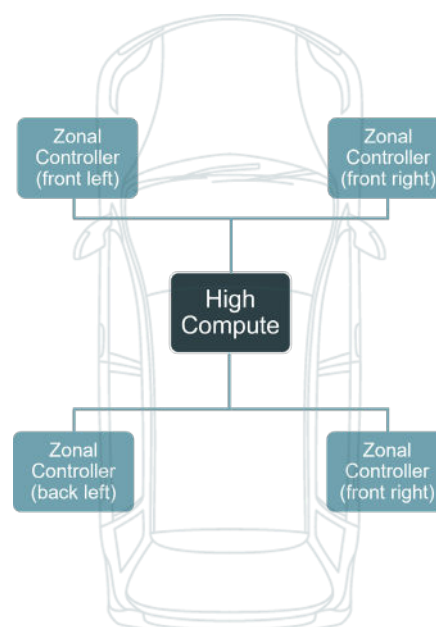


Figure 2. Centralized SDV architecture

2.1 Zonal Controllers

A typical SDV architecture consists of four to eight zonal controllers distributed throughout the vehicle and connected to one or more central computing platforms. Each zonal controller manages the sensors, actuators, power distribution, and communication networks within a defined physical region of the vehicle, such as the front-left, front-right, rear-left, rear-right, or cabin zone. It acts as the interface between local vehicle electronics and the centralized compute platforms.

Modern zonal controllers are typically built around heterogeneous SoCs that integrate multiple processing domains within a single device. These commonly include high-performance application processors, often based on Arm Cortex-A cores, for communication services, diagnostics, and middleware, together with real-time Cortex-R or Cortex-M cores responsible for deterministic control and I/O processing. Many devices also incorporate dedicated safety cores, frequently operating in lockstep configurations to support functional safety requirements, as well as Hardware Security Modules (HSMs) that provide secure boot, key management, and cryptographic services.

In addition to these processing elements, zonal controller devices integrate communication interfaces such as Automotive Ethernet, CAN/CAN-FD, and LIN, enabling the consolidation and management of local vehicle networks. By combining communication, control, safety, security, and networking functions within a single platform, zonal controllers provide the foundation for the scalable and centralized architectures used in modern Software-Defined Vehicles.

Representative zonal controller platforms include the NXP S32G and S32Z/E families, Texas Instruments Jacinto devices, Renesas RH850 processors, and Infineon AURIX microcontrollers.

2.2 Central Compute Platforms

Central compute platforms provide the processing capability required for software-intensive vehicle functions. Unlike zonal controllers, which focus on local I/O management and network aggregation, central compute platforms execute vehicle-wide applications such as the digital cockpit, infotainment, Advanced Driver Assistance Systems (ADAS), automated driving functions, connectivity services, and over-the-air (OTA) updates.

These systems are typically based on high-performance SoCs containing multi-core Arm CPUs, GPUs, AI accelerators, image processing units, hardware virtualisation support, and dedicated safety and security subsystems. Examples include NVIDIA DRIVE Orin and Thor, Qualcomm Snapdragon Ride, Renesas R-Car, and Mobileye EyeQ platforms.

The industry's move toward centralized computing therefore creates a new challenge: how can the benefits of workload consolidation be realized without compromising the independence required for functional safety?

CHAPTER 3 The Need for Safety Islands

The transition to Software-Defined Vehicle (SDV) architectures fundamentally depends on the ability to safely consolidate vehicle functions onto centralized compute platforms. As vehicle software grows to hundreds of millions of lines of code [4], managing interactions between applications with different safety requirements has become a major architectural challenge.

A decade ago, many safety-related functions could be isolated simply by assigning them to dedicated ECUs. That approach becomes increasingly difficult in centralized SDV architectures, where multiple workloads coexist on the same SoC. As workloads that were once distributed across dozens of dedicated ECUs are brought together on a small number of high-performance devices, applications with very different safety requirements must coexist within the same system.

Safety-critical vehicle control functions may execute alongside non-critical infotainment, connectivity, and digital cockpit applications. Without effective separation, faults occurring within a non-safety application could undermine functions required for safe vehicle operation.

Maintaining functional safety therefore requires an independent mechanism capable of supervising the centralized compute platform, detecting faults, and ensuring appropriate safe-state behaviour when failures occur.

As compute becomes centralized, safety must become independent. The industry's response to this challenge is the emergence of safety-island architectures, which provide the separation required to support both software innovation and functional safety.

This separation delivers benefits beyond safety alone. By isolating safety-critical supervision from application software, vehicle manufacturers can reduce development dependencies, simplify certification activities, accelerate feature deployment, and create greater flexibility for future platform upgrades. It also enables vehicle platforms to be scaled across multiple product generations without requiring fundamental changes to the underlying safety architecture.

As Software-Defined Vehicle architectures continue to evolve, this architectural independence is becoming an increasingly important enabler of both safety and innovation. Safety islands are therefore emerging not simply as a functional-safety mechanism, but as a foundational architectural element that enables the long-term scalability, maintainability, and evolution of modern vehicle software platforms.

3.1 Safety Islands

Safety islands provide an independent supervisory domain within a vehicle SoC. Their primary purpose is to monitor system health, manage safety-related diagnostics, and maintain control of safe-state transitions independently of the primary application environment. In many respects, safety islands represent a practical implementation of a Safety Before Silicon approach, where safety architecture is considered alongside compute architecture from the earliest stages of platform design rather than added later as a compliance exercise.

By separating supervision functions from the main compute domain, safety islands help satisfy key functional safety objectives including fault containment, independence, and freedom from interference.

3.2 How Safety Islands Work

A safety island operates as an independent supervisory domain within a larger automotive SoC. While the primary compute platform executes applications such as infotainment, connectivity, ADAS, and autonomous driving software, the safety island continuously monitors system health and safety-critical functions.

The safety island collects status information from across the system and evaluates it against predefined safety criteria. When a fault is detected, the safety island assesses its severity and determines the appropriate response. Depending on the nature of the failure, this may involve restarting software, disabling affected functions, transitioning the vehicle into a degraded operating mode, or initiating a safe-state response to maintain safe vehicle operation.

By remaining physically and logically independent from the application domain it supervises, the safety island provides the fault containment, monitoring, and control mechanisms required to support centralized Software-Defined Vehicle architectures.

3.3 Safety Island Implementation

Typically integrated within a zonal controller or central compute SoC, a safety island comprises dedicated processor cores, memory, peripherals, and hardware safety mechanisms that operate independently of the primary application domain. This separation allows safety supervision functions to remain operational even when faults occur within the main compute environment.

The exact partitioning of functionality between the application domain and safety island is an important architectural decision that varies between vehicle platforms. Safety goals, system complexity, performance requirements, and the capabilities of the underlying SoC all influence which responsibilities are entrusted to the safety island and which remain within the application domain.

3.4 Why Has Cortex-R52 Become the Preferred Safety-Island Processor?

In many modern automotive platforms, the safety island is implemented using Arm Cortex-R52 processors. Designed specifically for safety-critical and mixed-criticality systems, Cortex-R52 combines deterministic real-time execution with functional safety features that make it particularly well suited to supervisory applications. Unlike high-performance application processors that prioritise computing throughput, Cortex-R52 is optimised to deliver predictable execution and bounded response times, enabling dependable monitoring and control functions.

Key capabilities include support for split-lock operation, allowing CPU cores to operate independently or in lockstep for fault detection, low-latency interrupt handling, hardware-assisted virtualization, and robust memory-protection mechanisms. Together, these features support the isolation, fault containment, and deterministic behaviour required by safety-critical systems.

In many automotive SoCs, Cortex-R52 cores form the trusted supervisory subsystem responsible for functions such as watchdog management, diagnostic monitoring, safety communications, fault collection, and safe-state control. This enables high-performance application processors running software stacks such as Linux, QNX, or AUTOSAR Adaptive to focus on workloads including ADAS, digital cockpit, AI inference, connectivity, and over-the-air updates, while safety-critical supervision remains isolated within a dedicated execution environment.

These processors are frequently paired with safety-certified RTOS technology to provide the deterministic software environment required for monitoring, fault management, and safe-state control. While the processor provides the hardware foundation for safety supervision, the software environment running on it is equally important to achieving the levels of predictability, isolation, and certifiability required by modern safety-critical systems.

This architectural separation supports the ISO 26262 [2] principle of freedom from interference by helping ensure that failures within the application domain cannot compromise safety-critical monitoring and control functions.

The appearance of safety islands across multiple automotive SoC vendors suggests the industry is converging on a common architectural approach to functional safety. While implementations differ, the underlying principle remains consistent: safety supervision is increasingly separated from the primary application domain and entrusted to an independent safety-processing subsystem.

A practical example can be seen in Qualcomm Snapdragon Ride, which employs dedicated Cortex-R52 safety processors to supervise the primary compute domain and support ASIL-D safety objectives. Similar architectures can be found in platforms such as NVIDIA DRIVE Orin & Thor, NXP S32G, and Texas Instruments Jacinto devices, reflecting a broader industry shift toward independent safety supervision within centralized vehicle architectures.

3.5 Cybersecurity & Safety

As vehicle platforms become increasingly connected, functional safety and cybersecurity are also becoming more closely intertwined. A cybersecurity incident affecting a central compute platform can have direct implications for vehicle safety, requiring systems to tolerate both accidental failures and malicious attacks. While ISO 26262 [2] addresses functional safety and ISO/SAE 21434 [3] addresses cybersecurity, safety islands provide a common architectural mechanism that helps support both objectives by containing failures and maintaining the availability of critical safety functions.

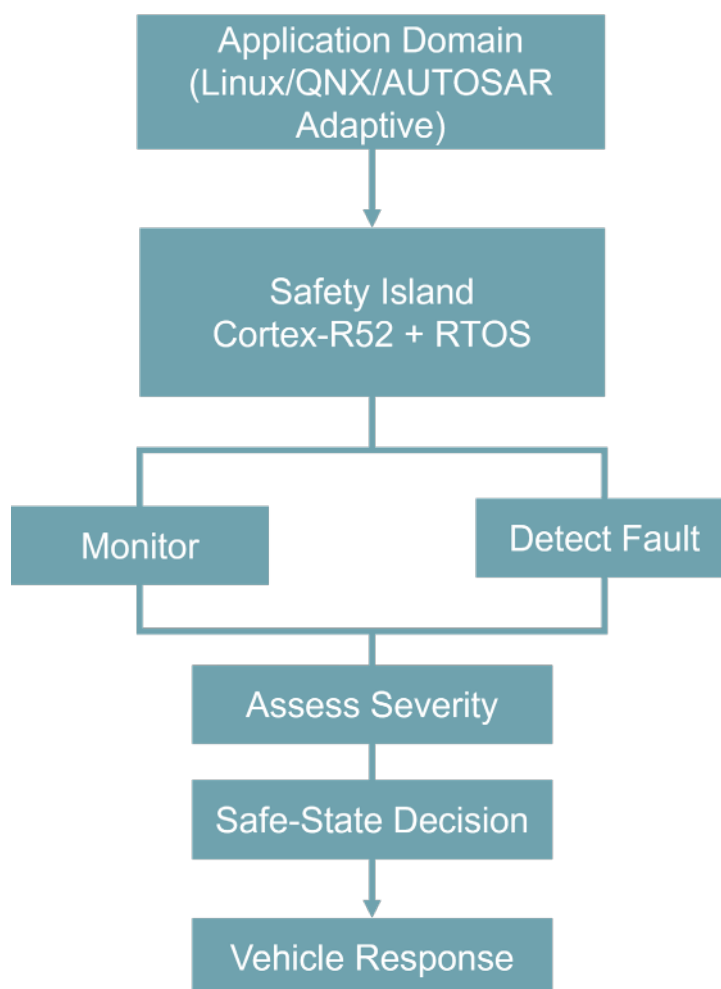


Figure 3. *Safety Island Operation within a Centralized SDV Platform*

3.6 Software Requirements for a Safety Island

The effectiveness of a safety island depends not only on hardware isolation but also on the software executing within the safety domain. A safety-island operating environment typically requires:

- Deterministic scheduling and bounded response times for predictable safety supervision.
- Strong memory protection and task isolation to support freedom from interference.
- Minimal software complexity and a small trusted computing base to simplify verification and reduce risk.
- Robust fault-detection and error-handling mechanisms to maintain safe operation during failures.
- Certification artefacts and compliance evidence to support ISO 26262 [2] functional-safety activities and ISO/SAE 21434 [3] cybersecurity assurance activities.

These requirements often make general-purpose operating systems unsuitable for safety-island deployments. General-purpose operating systems are typically optimised for functionality, flexibility, and broad software ecosystems. Safety islands have different priorities. They require predictable behaviour, limited complexity, and demonstrable compliance with functional-safety objectives. For this reason, many safety-island implementations rely on specialised safety-certified RTOS technology such as SAFERTOS®.



CHAPTER 4 Introducing SAFERTOS®

SAFERTOS® is specifically designed for high-integrity embedded applications and provides the deterministic behaviour, certification artefacts, integration guidance, and Safety Manual needed to support safety-island software design. It is derived from the widely adopted FreeRTOS functional model but has been specifically developed for use in safety-critical systems. The familiarity of the FreeRTOS programming model can help reduce adoption effort. In addition, a straightforward migration path from FreeRTOS to **SAFERTOS®** allows developers to prototype using FreeRTOS before migrating to **SAFERTOS®** at the start of formal safety-related development.

Unlike the primary application domain, a safety island is not designed to host feature-rich software stacks, middleware frameworks, or user-facing applications. Its purpose is to provide trusted supervision, fault management, and safe-state control. As a result, simplicity, determinism, and certifiability often become more important than software ecosystem breadth. The operating system must be predictable, verifiable, and capable of supporting safety objectives without introducing unnecessary complexity.

Safety islands place unique demands on software infrastructure. They must remain operational when faults occur elsewhere in the system, respond within defined timing constraints, and provide a trusted environment for safety-related decision making. **SAFERTOS®** aligns closely with these requirements through its deterministic scheduling model, small trusted computing base, and comprehensive safety documentation.

Within modern Software-Defined Vehicle architectures, **SAFERTOS®** is commonly deployed on safety islands to host supervisory and control functions. Typical safety-island applications deployed on **SAFERTOS®** include watchdog supervision, health monitoring, communication monitoring, fault management, power-state supervision, and safe-state control. **SAFERTOS®** is particularly well suited to this environment because it combines deterministic real-time behaviour with a small and highly trusted software footprint. Its predictable scheduling helps ensure that critical monitoring and control functions execute within their required timing constraints.

Its compact architecture minimises memory consumption and reduces the size of the trusted computing base, simplifying verification and safety analysis. This is especially important within safety islands, where resources are often more constrained than on the primary compute platform.

SAFERTOS® is supplied with certification artefacts, integration guidance, traceability documentation, verification evidence, and a Safety Manual that can significantly reduce the effort required for ISO 26262 [2] safety qualification activities.

By combining deterministic real-time performance, a small trusted computing base, and proven safety credentials, **SAFERTOS®** provides an effective software foundation for implementing the supervision and fault-management functions required by modern Software-Defined Vehicle architectures.

Example: ADAS Processor Failure

A centralized compute platform running Linux hosts an ADAS perception stack. If the perception software crashes or stops responding, the safety island detects the missing heartbeat, determines that the ADAS functionality can no longer be trusted, disables the affected feature, notifies the driver, and places the vehicle into a predefined safe state. Because the safety island operates independently of the failed software stack, safety supervision remains available throughout the event.

4.1 Supporting ISO 26262 Compliance

The challenge is not simply achieving functional safety, but achieving it without creating excessive certification overhead. As Software-Defined Vehicle platforms continue to grow in complexity, the effort required to verify, qualify, and maintain safety-related software can become a significant development constraint.

Functional safety is a fundamental requirement for modern Software-Defined Vehicle architectures, particularly as vehicle functions are consolidated onto centralized compute platforms. ISO 26262 [2] requires that safety-related systems are capable of detecting faults, controlling failures, and maintaining safe operation throughout the vehicle lifecycle.

Safety islands play an important role in meeting these requirements by providing an independent execution environment for safety-critical software. Their separation from the primary application processors supports key ISO 26262 [2] concepts such as freedom from interference, fault containment, independence, and diagnostic coverage. This allows safety-critical functions to remain operational even when faults occur within non-safety applications running on the same SoC.

SAFERTOS® complements this architecture by providing the deterministic behaviour required for safety-critical monitoring and control functions. Developed using a rigorous safety lifecycle, SAFERTOS® is supported by comprehensive certification artefacts, traceability documentation, and verification evidence that simplify safety qualification activities and reduce development risk.

Together, safety islands and SAFERTOS® enable OEMs and Tier-1 suppliers to implement centralized SDV architectures while maintaining compliance with functional safety requirements up to ASIL-D.

CHAPTER 5 What this means for OEMs and Tier 1 Suppliers

The combination of safety islands and SAFERTOS® enables OEMs and Tier-1 suppliers to realize the benefits of Software-Defined Vehicle architectures while reducing development risk and long-term lifecycle costs.

Key benefits include:

Faster SDV Deployment

By separating safety supervision from application software, teams can develop and validate these domains in parallel. This reduces dependencies and helps accelerate feature deployment and OTA update programs.

Reduced Certification Risk

By isolating safety functions within a dedicated execution environment, certification activities can often be limited to a smaller portion of the software stack, reducing verification effort and project risk.

Improved Platform Scalability

A reusable safety-island architecture allows OEMs to add new applications and services without redesigning the underlying safety framework for each vehicle generation

Table 2. *Benefits of Safety Islands for OEMs and Tier-1 Suppliers*

Challenge	Safety Island Benefit
Complex software integration	Clear separation of safety and application domains
ISO 26262 compliance effort	Reduced certification scope
SDV feature deployment	Parallel software development
Platform evolution	Reusable safety architecture
OTA updates	Reduced impact on safety-certified components
Long-term maintenance	Lower validation and regression-testing effort

Taken together, these benefits help OEMs and Tier-1 suppliers manage the growing complexity of Software-Defined Vehicle architectures while maintaining functional safety, reducing development risk, and improving long-term platform scalability. Beyond safety, safety islands provide architectural freedom, enabling OEMs to deploy different operating systems, middleware stacks, and software environments for safety-critical and application workloads while maintaining appropriate separation between domains. As vehicle software platforms continue to evolve, this flexibility becomes increasingly important for supporting continuous innovation without requiring fundamental changes to the safety architecture.

CHAPTER 6 Beyond Automotive

The architectural principles behind safety islands are gaining relevance beyond the automotive sector. As systems become more software-defined, AI-driven, and reliant on centralized computing, independent safety oversight is becoming essential. The separation of compute and safety may be as important to the future of autonomy as centralized computing itself, providing the foundation for safe AI-enabled systems across industries.

6.1 Humanoid Robotics

As the architectural concepts pioneered within Software-Defined Vehicles continue to mature, many of the same technologies are now being adopted by the rapidly expanding humanoid robot and collaborative robot (cobot) markets.

Modern robotic platforms are becoming more software-defined, AI-driven, and reliant on centralized high-performance compute platforms in much the same way as modern vehicles. This convergence is a key factor driving significant growth across the robotics industry and is creating new opportunities for safety-certified software technologies that have already been proven within automotive applications.

Unlike automotive platforms, robotic systems do not typically require AUTOSAR. However, developers increasingly require automotive-grade levels of reliability, determinism, and functional safety. As a result, many robotics architectures are adopting safety-island concepts similar to those used within centralized automotive compute platforms. Safety islands can provide independent supervision of actuator control, emergency stop functions, and fault management, ensuring safe operation even when faults occur within complex AI software stacks.

As robots become more autonomous and operate in closer proximity to humans, demand for certified safety software and independent safety-supervision architectures will continue to increase.

This growth in robotics further reinforces a broader industry trend: the architectural principles originally developed to support functional safety within Software-Defined Vehicles are now being applied across a growing range of intelligent, AI-enabled systems, where independent safety supervision is becoming a foundational requirement for safe autonomous operation.

6.2 Industrial Autonomy & Agricultural Automation

Autonomous industrial equipment, including warehouse robots, mobile platforms, and automated manufacturing systems, must continue operating safely in the presence of software or hardware failures. Safety islands can provide real-time monitoring, fault detection, and safe-state management independent of the primary control system.

Autonomous tractors, harvesting equipment, and precision farming platforms increasingly rely on centralized compute architectures for navigation, perception, and machine control. Independent safety domains can supervise machine operation and support safe-state transitions when faults are detected.

6.3 Medical Systems

Many medical devices combine sophisticated software, connectivity, and safety-critical control functions. Safety islands can provide an independent supervisory layer that monitors system behaviour, detects failures, and helps maintain safe operation when faults occur in the primary application environment.

CHAPTER 7 Conclusion

Software-Defined Vehicle architectures are reshaping automotive electronics by consolidating previously distributed functions onto centralized compute platforms. This shift enables faster feature deployment, over-the-air updates, reduced hardware complexity, and support for increasingly sophisticated ADAS and autonomous driving capabilities. However, it also creates a fundamental challenge: maintaining functional safety while enabling continuous software innovation.

Safety islands have emerged as the architectural mechanism that addresses this challenge. By providing an independent domain for supervision, diagnostics, fault management, and safe-state control, they enable vehicle manufacturers to separate safety-critical functions from rapidly evolving application software. This separation supports key ISO 26262 [2] principles while allowing centralized computing platforms to scale in capability and complexity.

As adoption accelerates across automotive platforms from vendors including Qualcomm, NVIDIA, NXP, and Texas Instruments, safety islands are increasingly becoming a standard architectural element of Software-Defined Vehicles. At the same time, the principles they embody are beginning to influence other software-defined and AI-enabled systems, including robotics, industrial automation, and autonomous machines.

SAFERTOS® provides the deterministic, safety-certified software foundation required to implement these supervisory functions. Its combination of predictable real-time performance, a small trusted computing base, and comprehensive safety evidence makes it well suited to the demands of safety-island deployments.

The industry's challenge is no longer whether systems can be centralized, but how they can be centralized without sacrificing safety, flexibility, or future innovation. Safety islands provide the architectural answer to that challenge, while SAFERTOS® provides the software foundation that helps make it practical.

Together they enable the separation of innovation from safety, allowing future vehicle platforms to grow in capability and complexity without compromising functional safety.

Contact Information

Your Feedback

We would be pleased to receive your comments and any suggestions for improvement to our software and documentation. Please email us at: support@highintegritysystems.com

Contact WITTENSTEIN high integrity systems

Address: WITTENSTEIN high integrity systems
3 Rivergate, 1st Floor
Temple Quay
Bristol
BS6 1EW
England

Americas +1 408 625 4712
ROTW +44 1275 395600

Email: support@HighIntegritySystems.com

Website www.HighIntegritySystems.com

All Trademarks acknowledged.