Codethink

www.codethink.co.uk

Software Readiness and Long-Term Maintainability for RISC-V in Safety-Critical Automotive Systems

How Lessons from Five Years of Upstream RISC-V Engineering Support Safe, Long Life Cycle Automotive Deployment

Co-authored by Roan Richmond, Lawrence Hunter and Ben Dooks

Contents

1. RISC-V Adoption in the Automotive Industry is Accelerating

The evidence of the past year suggests RISC-V adoption is increasing in automotive and there may be a significant shift in how hardware architecture is sourced and governed on the horizon. In March 2025, Infineon Technologies, a global market leader for automotive microcontrollers, announced it would launch a new automotive microcontroller family based on RISC-V, becoming the first semiconductor supplier to do so.[1] Original equipment manufacturers (OEMs) are equally direct. Hartmut Schittko, Head of Semiconductors at Volkswagen's software group CARIAD, speaking at RISC-V Summit Europe 2025 in Paris, said: "RISC-V is a great opportunity for us as a car OEM. We grow closer to the community that does chips and software, because these really need to work flawlessly together, especially when it comes to high performance compute (HPC) tasks."[2] On top of that, NVIDIA shipped one billion RISC-V cores in 2024, deployed across function-level controllers, chip and system-level control, and data processing applications in its GPU product portfolio.[3]

For RISC-V to work in automotive, RISC-V hardware must first be certifiable to automotive safety standards. Some progress has been made in this area. SiFive certified multiple automotive processor variants in 2024, including the 64-bit S7-AD.[4] Following that, in 2025, Andes Technology's D45-SE and Codasip's L739 both achieved ISO 26262 ASIL-D certification.[5,6]

We can assume from this significant growth that an evaluation of RISC-V in automotive is already underway. However, for automotive engineering teams, hardware transitions create a porting problem. RISC-V is an instruction set architecture (ISA) which defines how a processor executes code but not the software that runs on it. As is the case with all architecture, software stacks which run on it must be safe, maintainable, and compliant across a vehicle's operational life. The decisions made at the software layer will strongly influence whether RISC-V accelerates in automotive or stalls.

The ISA's royalty-free, openly governed model changes the economics of automotive compute and with it the competitive dynamics of silicon supply. Comparable hardware costs less without royalty payments to ARM or Intel, a material saving at automotive production volumes. However, those commercial advantages are only realised if the software built on top of the hardware can be sustained across the full lifecycle automotive demands. This requires Long-Term Maintainability (LTM): keeping the software stack as close to upstream mainline as possible throughout a product's operational life, so that security patches, kernel updates, and hardware revisions land naturally rather than requiring costly backporting efforts. RISC-V economics combined with LTM practice makes safe, long life cycle automotive deployment tractable.

Codethink has been working on **RISC-V since 2021**. Due to the possible benefits of RISC-V's use in automotive and its alignment with open source principles, we have been porting, testing, and contributing software to RISC-V by finding gaps in the ecosystem and addressing them. This whitepaper consolidates our work and demonstrates what we found.

2. RISC-V and the Economics of Automotive Compute

RISC-V is owned by no single company. Its specifications are openly governed by RISC-V International and can be implemented by any organisation without paying royalties or licensing fees. Hardware vendors building on RISC-V compete on equal terms, and automotive OEMs and Tier 1 suppliers benefit directly from that competition. Vendors are incentivised to differentiate through better silicon, better tooling, and better software support, rather than through proprietary lock-in.

Removing royalties and open-sourcing the ISA governance delivers three specific advantages for automotive programmes:

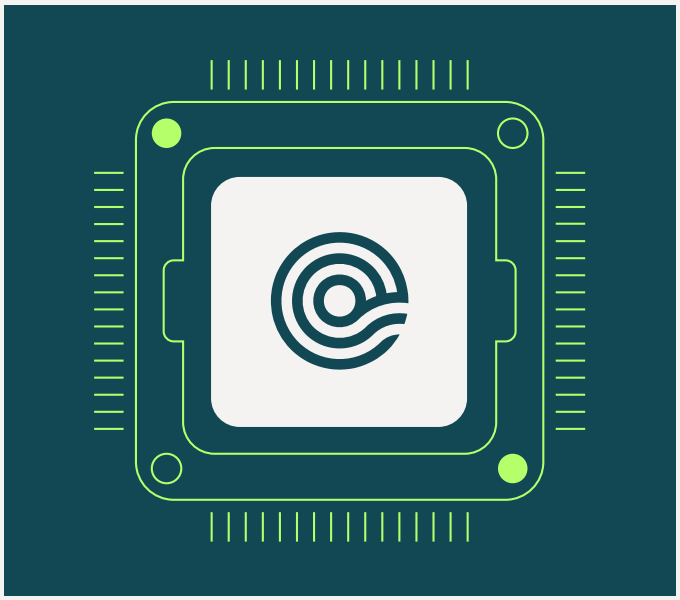
1) There are no per-unit royalties and no per-design licensing fees providing a material saving across automotive production volumes.

2) Organisations can participate in RISC-V International's technical working groups and actively shape the architecture's development. They are not dependent on a single vendor's roadmap.

3) Automotive suppliers can source RISC-V chips from multiple vendors including Infineon, SiFive, and Codasip. Because the ISA is not proprietary, teams avoid dependence on a single company's IP for generic RISC-V implementations, though individual boards and advanced chips remain vendor-specific.

Together, these advantages provide silicon sovereignty: the ability to make independent decisions about silicon supply, architecture direction, and licensing terms without being locked into a proprietary ecosystem.

The ratification of the RVA23 Profile by RISC-V International in 2024 gave software developers a common target across RISC-V implementations. RISC-V profiles are essential to software portability across many hardware implementations and help to avoid vendor lock-in, aligning implementations of RISC-V 64-bit application processors that will run rich operating systems stacks from standard binary OS distributions.[7] RVA23 standardises which ISA features are mandatory or optional across 64-bit RISC-V implementations, ensuring that software written for one compliant chip can execute on another without recompilation.[8]



RVA23 establishes a common baseline for application processors, reducing the bespoke software work required for each hardware target and freeing development teams to build on shared community work. For microcontroller deployments, which represent a significant portion of automotive RISC-V adoption including Infineon's AURIX commitment, an equivalent profile called RVM is currently under development within RISC-V International. Its ratification will extend the same standardisation benefits to the microcontroller domain.

That portability applies at the application layer rather than the board support layer. Moving from one vendor's RISC-V board to another still requires supporting that board's specific hardware configuration, including drivers, memory mapping, timers, and pinouts. The open source nature of the ISA allows organisations to steer the architecture and not pay royalties but vendors are not required to disclose how their chip is built.

The economics case for RISC-V in automotive is therefore real but conditional. As Roland Jancke, design methodology head of Fraunhofer IIS' Engineering of Adaptive Systems Division, commented: "Automotive manufacturers are increasingly looking into RISC-V due to the potential cost reduction, because in automotive you look for every cent to lower the price. However, you need to have an ecosystem. It is not enough to have the tools that are able to develop your processor, but also the software on top of that. RISC-V is gaining ground, but it still has a way to go until we have RISC-V processors as the main horse in the automotive area."[4]

The cost reduction and supply chain independence RISC-V offers are only realisable if the software stack is built to work with the open ecosystem. A codebase that diverges from upstream trades one form of lock-in for another, and the maintenance cost of that divergence compounds over a vehicle's operational life. LTM, the practice of keeping software as close to upstream mainline as possible, is what prevents that divergence.

3. The Software Challenge Automotive Demands of RISC-V

Vehicles must remain safely operational for 15 to 20 years, a commitment that has no equivalent in consumer electronics.[9] UNECE R155 requires OEMs to manage cybersecurity risks across the vehicle's full lifecycle, which means security vulnerabilities identified after launch must be addressable through software updates across all underlying hardware architectures, including RISC-V.[10] Meeting that obligation requires the entire software stack to remain maintainable, patchable, and updatable across the vehicle's operational life.

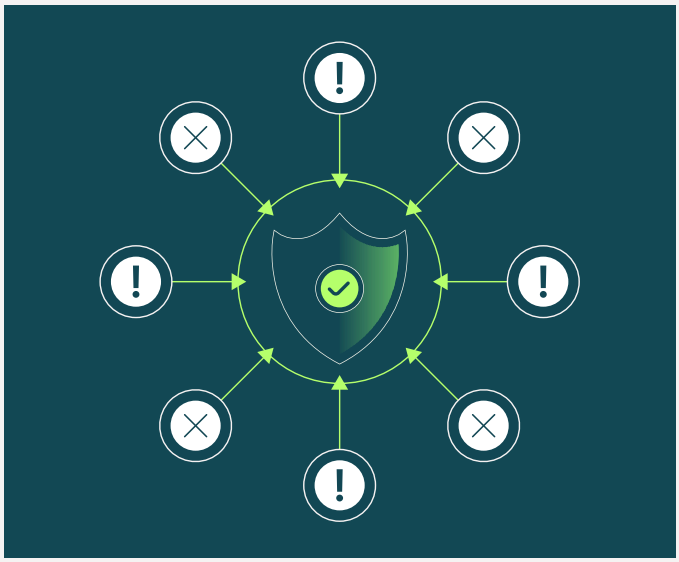
Established architectures, like Arm and x86, have a head start on supporting the fulfilment of compliance obligations, with decades of successful iterations, giving manufacturers access to mature toolchains, strong security frameworks, extensive software libraries, and legacy support that enable development and deployment.[11] For RISC-V, that same maturity does not yet exist across toolchains, profilers, certified compilers, software packages, and long-term support contracts.[12]

RISC-V engineering teams must build their own verification evidence for the toolchain, the software stack, and every update deployed across the vehicle's operational life. Long-Term Maintainability makes this tractable by keeping the codebase close enough to upstream mainline that security fixes can be applied and verified without specialist rewrite effort. Additionally, automated integration testing ensures each update can be proven safe on real hardware before it reaches a deployed vehicle.

Long-Term Maintainability Supports Software Safety Across the Vehicle Lifecycle

The dominant approach to managing software across the vehicle lifecycle, which is freezing a software version at launch and backporting critical fixes onto a static codebase, does not meet the standards for safety across the timescale required in automotive and required by UNECE R155. As Codethink's **Sam Thursfield has noted**, few people are maintaining decade-old versions of Linux, and these specialists can only do so much in terms of backporting fixes for today's complex security issues onto older kernels.[13] In vehicles, the maintenance cost of that divergence is both a commercial liability and safety concern. For RISC-V, where upstream contributions are still being established across the kernel and toolchain, a codebase that diverges from mainline loses the community maintenance benefit precisely when it is most needed and must absorb the full backport burden itself.[14]

LTM addresses the divergence problem directly by designing the software development process so that the codebase never diverges far enough from upstream to make security fixes infeasible. Rather than relying on specialists to bridge an ever-widening gap between in-house code and the upstream project, LTM keeps the product within reach of the community's ongoing work so that the fixes can be applied proactively, not reactively to an archaic codebase that's hard to change without entailing large costs.



Integration Testing Supports Compliance Across Software Updates

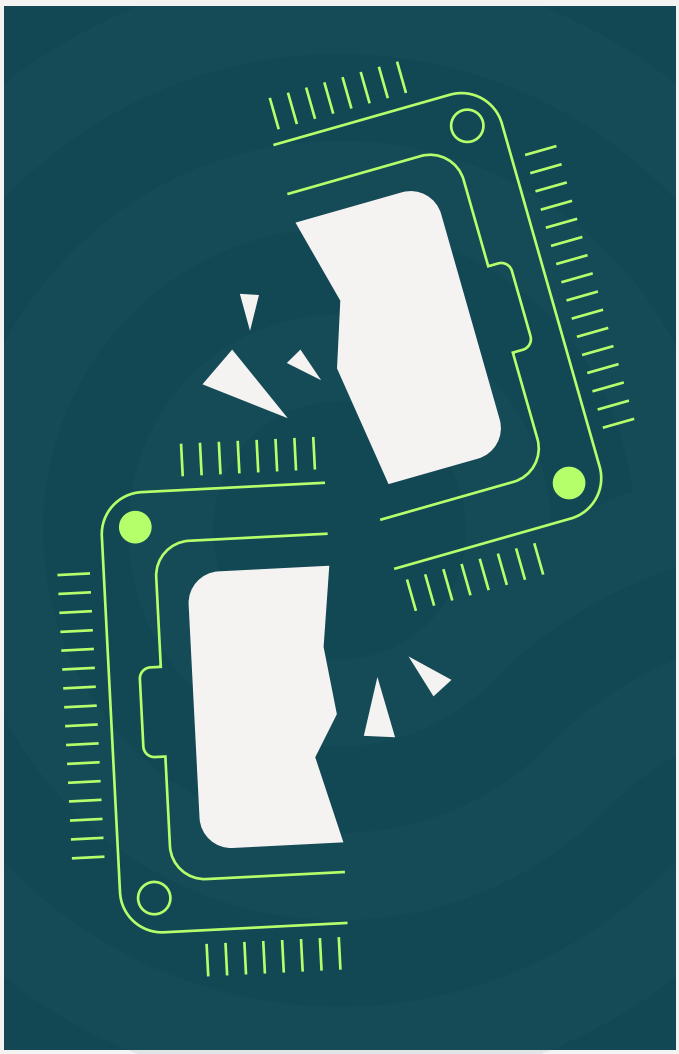
UNECE R155/156 compliance requires updates to be feasible and demonstrably safe. In order to demonstrate this, testing at the full system level is a requirement. Unit tests validate components in isolation, whereas integration tests can reveal whether component interactions behave correctly under real conditions. An engineering team that relies on unit tests alone is validating the parts without validating the whole.

NASA's Mars Climate Orbiter illustrates the consequence of that gap precisely. Ground software supplied by Lockheed Martin produced thruster impulse values in US customary units, while NASA's navigation software expected metric units. The mismatch went undetected throughout the mission. The Orbiter encountered Mars on a trajectory that brought it too close to the planet and was destroyed in the atmosphere. NASA later acknowledged that proper end-to-end testing could have caught the mistake before the spacecraft was lost.[15]

On proprietary architectures, testing infrastructure is tied to vendor-licensed debug tooling, which means its long-term availability depends on commercial relationships that may not persist across a 15 to 20 year vehicle lifecycle. RISC-V's open debug specification means the testing infrastructure engineering team's build is not subject to those dependencies. Section 4.2 documents how Codethink built an analogous kind of independent testing infrastructure on physical RISC-V hardware.

Current Limiting Factors to RISC-V Automotive Adoption

Engineering teams evaluating RISC-V for automotive production today will encounter three software gaps where established architectures are further along, having had decades of production use to develop and refine solutions that are still maturing for RISC-V.



1. Driver, Middleware, and Vertical-Specific Library Coverage

Compiler support for RISC-V has improved significantly since 2024, with native builds now available for most developer tools.[16] However, tuning, libraries, and vertical-specific gaps remain.[17] Driver coverage for peripheral hardware is incomplete in places, and middleware components that automotive programmes depend on have sometimes required additional porting work before a production-grade software stack can be assembled.[18] Many projects still do not produce RISC-V releases as standard, and some do not yet support RISC-V at all.[19]

2. Secure Boot Tooling

Production-grade secure boot implementations are well understood and widely available on established architectures. ARM provides TrustZone and Intel provides a Trusted Execution Engine which are standardised, on-chip trusted execution environments with decades of production use behind them. QEMU and the RISC-V server specification support TPMs as external devices, enabling secure boot and attestation, but there's no comparable on-chip security infrastructure to that ARM and Intel offer for RISC-V. Integrating third-party IP for specialised functionalities may require complex licensing agreements, making compatibility challenging.[20]

3. Hardware Immaturity Creates Software Engineering Overhead

RISC-V silicon is still catching up to ARM and x86 benchmarks in many application domains, and that gap has direct consequences for software development workflows. Industry experts project performance parity between high-end ARM and RISC-V CPU cores by the end of 2026, signalling that the gap is rapidly narrowing rather than widening.[10] Where teams build natively on RISC-V hardware rather than cross-compiling from a faster host, current silicon performance adds overhead.

These limitations are not constitutional but rather expected of a new and developing architecture. The RISC-V ecosystem is moving fast, with CPU performance increasing approximately 8x over the past few years.[21] The engineering teams that invest in software support now will be better positioned than those that wait for the hardware to mature before starting.

4. Five Years of Software Engineering on RISC-V: What Codethink Found

Codethink has contributed to the RISC-V software ecosystem since 2021. The work documented here reflects engineers engaging with a new architecture by finding gaps and addressing them.

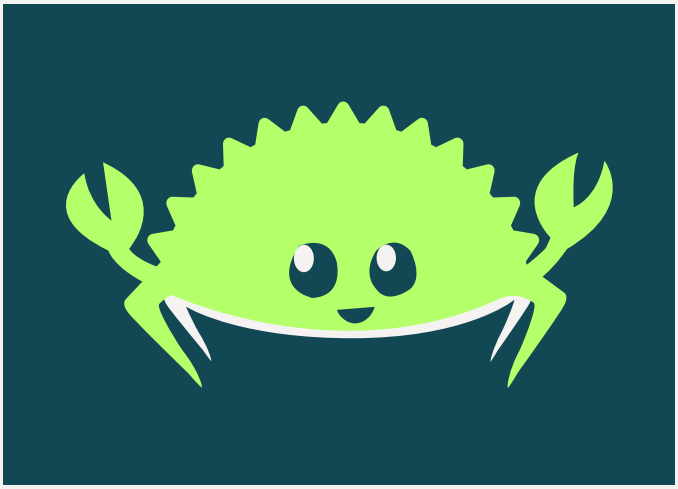
Each published piece is available to any team building on RISC-V today. Taken together, they demonstrate that the software required for safe, maintainable RISC-V automotive deployment is not theoretical. It has been built, tested, and contributed upstream.

4.1 Developing a Cryptographically Secure Bootloader for RISC-V in Rust

Lawrence Hunter, Codethink, August 2024

https://www.codethink.co.uk/articles/2024/secure_bootloader/

Memory safety is a persistent issue in system software due to their privileged operation, especially bootloaders, as they are the first modifiable link in the secure boot chain. Exploiting vulnerabilities arising from a lack of memory safety leads to data leaks, denial-of-service, and most dangerously, arbitrary code execution.



C, C++, and Assembly have long been the only viable languages for system software such as bootloaders, because their performance characteristics meet the constraints of low-level applications. Languages that promote memory safety through runtime checks incur computational costs that make them incompatible. The trade-off has therefore been performance or safety, such as accepting memory vulnerabilities in exchange for the performance that system software requires. Rust resolves this trade-off by performing static analysis at compile time rather than incurring runtime performance penalties, making it a viable memory-safe alternative for system software.

Codethink sponsored a University of Manchester final-year project to address this directly on RISC-V. The result was SentinelBoot: a cryptographically secure RISC-V bootloader written in Rust. SentinelBoot is aimed at enhancing boot flow safety of RISC-V through memory-safe principles, predominantly leveraging the Rust programming language with its ownership, borrowing, and lifetime constraints. Additionally, SentinelBoot employs public-key cryptography to verify the integrity of a booted kernel by the use of the RISC-V Vector Cryptography extension. Codethink contributed support for this extension to QEMU, sponsored by SiFive, as one of the required steps toward ratification.

Unlike C and C++, where memory vulnerabilities are discovered at runtime and often in production, Rust's ownership model eliminates whole classes of them at compile time. The result is a demonstration that a memory-safe, cryptographically verified bootloader for RISC-V can be implemented in a single Rust binary, checking kernel integrity against a hardcoded SHA without depending on vendor-specific hardware security modules, demonstrating the viability of the approach.

4.2 Automated Kernel Validation to Produce Repeatable, Hardware-Level Test Evidence that Compliance Processes Accept

Neill Whillans, Codethink, September 2023

https://www.codethink.co.uk/articles/2023/riscv-kernel-testing/

UNECE R155/156 compliance requires software updates to be demonstrably safe across a vehicle's full operational life. Demonstrating that safety requires testing at the full system level, continuously, on the actual target hardware. Staying close to upstream mainline is the principle that underpins LTM. However, it only delivers compliance benefits if each update can be validated before it reaches a deployed vehicle. Automated hardware-level testing is what makes that validation possible.

Codethink are strong proponents of using robust testing pipelines to aid in the goal of LTM of complex software platforms, such as the Linux kernel, advocating to align as closely to mainline as possible. Codethink built an automated kernel testing pipeline for RISC-V using LAVA and OpenQA, running on physical SiFive Unmatched boards; the same tooling used for x86 and ARM kernel testing. The pipeline fetches Linux kernel sources, builds a kernel image for the RISC-V board, triggers a LAVA job that boots the SiFive Unmatched board, logs in automatically, and runs a set of OpenQA-controlled tests on the OS. Results are provided upstream to Linux kernel stable branch maintainers contributing to the RISC-V ecosystem.

Continuous hardware-level testing on real RISC-V boards produces the repeatable, documented test evidence that long-lifecycle compliance demands. For a vehicle expected to remain on the road for 15 to 20 years, this is the infrastructure that makes fearless kernel upgrading tractable.

4.3 Adding RISC-V Vector Cryptography Extension Support to QEMU

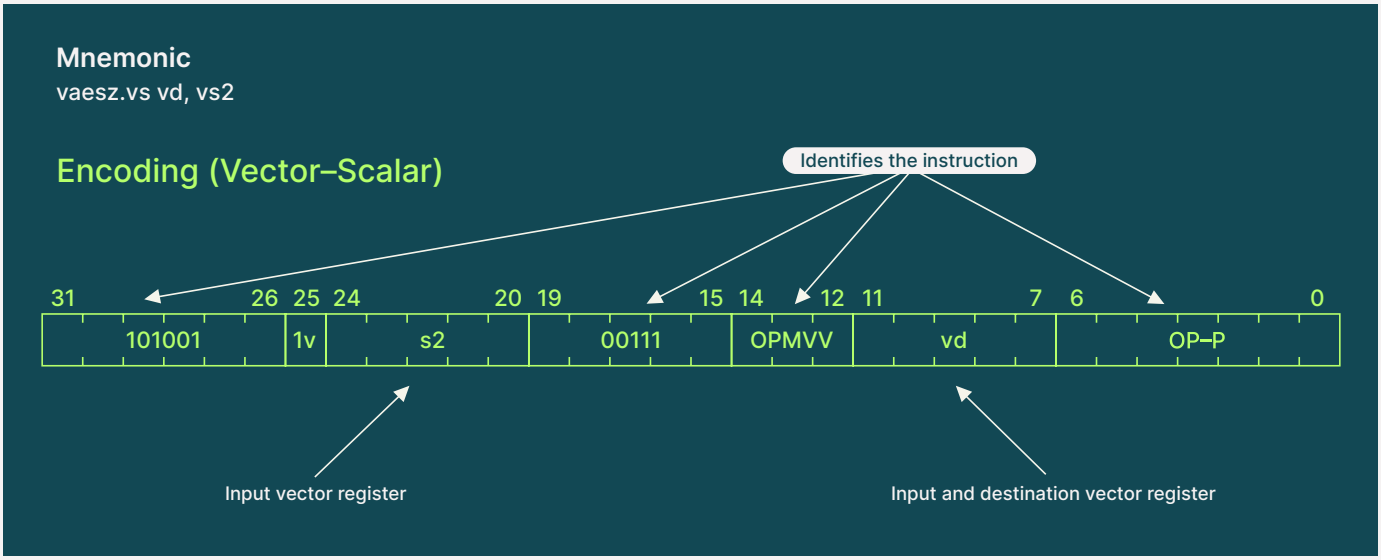
Lawrence Hunter, Nazar Kazakov, Kiran Ostrolenk and Dickon Hood, Codethink, June 2023

https://www.codethink.co.uk/articles/2023/vcrypto_qemu/

QEMU is the standard emulation environment for RISC-V development now. But when Codethink began working on hardware-accelerated cryptography for RISC-V, the RISC-V Vector Cryptography extension had no prior implementation in QEMU and it wasn't even ratified yet. That meant teams building cryptographically secured software could not accelerate implementations before physical hardware was available.

Our project, sponsored by SiFive, was to add full support into QEMU for RISC-V's vector cryptography, vcrypto, extension set. This extension set provides instructions for implementing various cryptographic programs, including AES, SHA-2, and the ShangMi suites. The vector cryptography extension uses vector registers to accelerate the base cryptographic operations already defined in the RISC-V ISA. Where a scalar cryptographic operation works on one fixed-width chunk of data at a time, the vector extension uses many registers in parallel. To give one example, it is able to operate on 8x64 bits of data in a single operation rather than processing each 64-bit chunk sequentially. The result is a significant increase in cryptographic throughput without changing the underlying cryptographic operations themselves.

This work unblocked the entire development path for hardware-accelerated cryptography on RISC-V. It was contributed upstream to QEMU and republished by RISC-V International on their own site.[23]



4.4 Bringing up a New Distro for the CVA6 RISC-V FPGA Processor

Nazar Kazakov, Ben Dooks, Lawrence Hunter and Joseph Baker, Codethink, October 2025

https://www.codethink.co.uk/articles/running-freedesktop-sdk-on-cva6/

Getting from bare RISC-V silicon to a production-grade, reproducible, maintainable Linux environment requires experience across firmware, kernel BSP, OS, and application stack. Codethink has done this across multiple RISC-V platforms: SiFive Unmatched, Microchip PolarFire Icicle Kit and the open source CVA6 FPGA processor.

The goal for the CVA6 processor was to bring it up with Linux and replace the existing OS with one based on Freedesktop SDK, updating the kernel version along the way. CVA6 is being developed by the OpenHW Group, part of the Eclipse Foundation, of which Codethink is a member. Codethink's contributions to the Freedesktop SDK brought RISC-V support into the mainline release as a concrete upstream contribution that persists and benefits any team building on RISC-V, rather than remaining in a private fork.

This work is most directly relevant to teams building their own silicon rather than integrating an off-the-shelf RISC-V platform. However, it demonstrates the principle that a reproducible build and bring-up process, with contributions to mainline, is what makes safe updates at scale possible across a 15 to 20 year deployment.

4.5 Porting an Automotive Operating System to RISC-V

Roan Richmond, Codethink, 2026

https://www.codethink.co.uk/articles/2026/CTRL-RISC-V.html

CTRL is Codethink's safety-oriented, trustable, and reproducible Linux operating system, used in automotive and industrial projects. It is built on the foundation of Freedesktop SDK, which already supports RISC-V, making it well placed to answer a practical question: how hard is it to port a Linux-based OS to a RISC-V target?

Porting CTRL to RISC-V was structured as a phased adoption, broken into three stages:

1. **Supporting RISC-V target compilation in the build environment**
2. **Building CTRL for RISC-V and running on a virtual target**
3. **Adding support for an actual RISC-V hardware target**

The first stage involved updating the build container to support RISC-V targets, ensuring all tools supported the RISC-V ISA and adding the RISC-V container build as an optional step in the CI pipeline. The main challenges centred on components that did not produce releases for RISC-V, including OpenTofu and Jsonnet, which could be excluded from the initial implementation as they were not required for later stages.

The second stage added support for virtual targets using QEMU. Since CTRL already supports both Aarch64 and x86 QEMU virtual targets, supporting qemu-system-riscv64 was a logical extension. The main challenges were related to intricacies in the UEFI boot stage, specifically that UEFI expects the next-stage binary, grub2, to be named bootriscv64, which differs from the x86 and Aarch64 naming patterns.

The third stage ported CTRL to physical RISC-V hardware, specifically the Banana Pi BPI-F3, chosen for its availability and broad support across the RISC-V ecosystem, containing a SpacemiT K1 octa-core RISC-V processor. The process covered building the bootloader components, OpenSBI and U-Boot, building the BPI-F3 Linux kernel, and constructing a flashable image for an SD card. The sources for the BPI-F3 versions of OpenSBI, U-Boot, and the Linux kernel are maintained in forked repositories rather than being upstreamed to official open source repositories, a common pattern among RISC-V hardware that is detrimental to the industry.

A single Codethink engineer completed all three stages in under a week indicating that the move to RISC-V has never been easier. BuildStream's in-built caching of components supported a significantly quicker rebuilding process and reduced kernel rebuild times to under two hours, helping to overcome the hardware performance gap.

5. RISC-V is Ready for Automotive

Significant milestones in 2025, such as the hardware commitment from Infineon, the public endorsements from OEMs, and the maturing certification landscape signal the architecture is up for production consideration in automotive.

The software ecosystem has kept pace. SentinelBoot demonstrates a memory-safe, cryptographically verified approach to RISC-V boot integrity in a single Rust binary. Automated kernel testing on physical SiFive hardware produces repeatable, hardware-level test evidence suitable to long-lifecycle compliance. Additionally, QEMU support for the RISC-V Vector Cryptography extension means accelerated cryptographic development and testing can begin before silicon arrives. Furthermore, Linux bring-up across SiFive, PolarFire and CVA6 platforms, alongside CTRL OS ported to physical RISC-V hardware and contributions back to the Freedesktop SDK mainline, means a reproducible, maintainable application stack is available to any team building on RISC-V today.

What makes this sustainable across a vehicle's operational life is the principle of upstream first underlying it. Every fix, driver, and kernel contribution that lands in mainline rather than a private fork reduces future maintenance burden rather than deferring it. A software stack built this way absorbs the kernel updates, security patches, and hardware revisions of the next decade as a matter of routine, without requiring the kind of specialist backport effort that has made long-lived automotive software so costly to maintain under the traditional model. That is how Codethink has approached its RISC-V work since 2021 and it is what we bring to automotive programmes where the software commitment outlasts the silicon it runs on.

6. Codethink's Approach to Open Source Software in Safety-Critical Systems

In 2016, Codethink started out on a journey to discover how open source software can be safely used to build safety-critical systems where people could be harmed if software fails.

Our research and development has produced three concrete, verifiable outcomes directly relevant to engineering teams evaluating RISC-V for automotive deployment.

ISO 26262 ASIL D Tool Certification

Working with Exida, Codethink achieved ISO 26262 ASIL D tool certification for its Deterministic Construction Service (DCS), a framework for enforcing reproducibility, repeatability, and traceability of builds using entirely open source tooling. This matters for RISC-V deployment because it demonstrates that an open source, reproducible build pipeline can be independently assessed to the highest automotive safety integrity level. Additionally, the DCS reference implementation is built on Freedesktop SDK, which supports RISC-V.

The Trustable Software Framework

Codethink created the Trustable Software Framework (TSF) as an evidence-based approach to evaluating risk in modern software projects. It is now governed and hosted by the Eclipse Foundation, the world's largest open source software foundation, reflecting the same upstream-first philosophy that underpins Codethink's RISC-V work.

The TSF provides a methodology for engineering trust into the way software is designed, built, and delivered, which is applicable to open source software stacks including those built on RISC-V. The TSF has received a positive functional safety assessment against IEC 61508 from Exida and domain expert assurance of compatibility with ISO 26262.

ELISA and AGL Membership

Codethink is an active member of ELISA (Enabling Linux in Safety Applications) and AGL (Automotive Grade Linux), contributing directly to the industry's effort in qualifying Linux-based operating systems for safety-critical deployment. Ongoing participation means Codethink's understanding of what safety qualification requires of a software stack is current, community-tested, and directly applicable to RISC-V deployments where the software is open source by design.

7. Conclusion

Software is never a temporary feature, especially in the fast-moving world of embedded systems. In the automotive industry, software is now at the core of the product's function, yet time and time again, companies treat it like a short-term asset.

To succeed, OEMs need automated and effective integration testing, a rock-solid over-the-air (OTA) update mechanism, and always up-to-date software, thanks to a long-term maintainability strategy.

RISC-V makes this more achievable than it has been under proprietary architectures. Its royalty-free, openly governed model reduces the cost of automotive compute, and removes per-design licensing fees, freeing investment for the software layer. When combined with an upstream-first approach to software engineering, RISC-V gives automotive teams a platform that can be updated, patched, and maintained without accumulating backport debt, which has made long-lifecycle software unreasonably expensive under the traditional model.

About Codethink

Codethink is a world-class provider of software engineering and strategic technical consultancy services. We work with clients in safety-critical sectors where software failure has real-world consequences and the lifetime of a product is measured in decades, not years. Our work spans the full embedded software stack: Linux kernel and BSP services, build engineering, embedded systems, long-term maintainability, and delivering trustable software.

If your team is beginning a RISC-V evaluation now, we can help you solve the problems we've already encountered and work with you to find future solutions with real longevity.

To discuss how Codethink can support your RISC-V software strategy, contact us at connect@codethink.co.uk or visit www.codethink.co.uk

8. Endnotes

[1] "Infineon Introduces RISC-V to the Automotive Industry, Becoming the First to Unveil an Automotive RISC-V MCU Family," Embedded.com, accessed April 22, 2026,

https://www.embedded.com/infineon-introduces-risc-v-to-the-automotive-industry-becoming-the-first-to-unveil-an-automotive-risc-v-mcu-family/

[2] James De Vile, "7 Critical Components of the Car of Tomorrow," RISC-V International Blog, August 5, 2025,

https://riscv.org/blog/critical-components-car-tomorrow/

[3] Anisha Sharma, "How NVIDIA Shipped One Billion RISC-V Cores In 2024," RISC-V International, 25 February 2025,

https://riscv.org/blog/how-nvidia-shipped-one-billion-risc-v-cores-in-2024/

[4] SiFive, "SiFive Enhances RISC-V Automotive Safety Leadership with Critical Functional Safety Certification,"

https://www.sifive.com/blog/sifive-enhances-risc-v-automotive-safety-leadershi

[5] Andes Technology, "Andes Technology D45-SE Processor Achieves ISO 26262 ASIL-D Certification for Functional Safety," 23 January 2025,

https://www.andestech.com/en/2025/01/23/andes-technology-d45-se-processor-achieves-iso-26262-asil-d-certification-for-functional-safety/

[6] Codasip, "Codasip RISC-V Processors Certified up to ASIL-D," 1 October 2025,

https://codasip.com/press-release/2025/10/01/codasip-processors-certified-asil-d/

[7] Andrea Gallo, "RISC-V RVA23 — A Major Milestone," Electronic Design, 16 April 2025,

https://www.electronicdesign.com/technologies/embedded/digital-ics/processors/article/55283485/risc-v-international-more-about-the-risc-v-rva23-profile

[8]RISC-V International, "RISC-V International Announces Ratification of the RVA23 Profile Standard," 21 October 2024,

https://riscv.org/blog/risc-v-announces-ratification-of-the-rva23-profile-standard/

[9] Eve Stevens, "Software-Defined Vehicles and the 15-Year Challenge: Can Automotive Software Outlast the Hardware?," Movemnt,

https://movemnt.net/software-defined-vehicles-and-the-15-year-challenge-can-automotive-software-outlast-the-hardware/

[10] Alyssa Shames, "A Look at the UN R155 Regulation for Connected Vehicles," Finite State Blog, July 9, 2024,

https://finitestate.io/blog/buckle-up-for-security-a-look-at-the-un-r155-regulation-for-connected-vehicles

[11] "Fragmentation to Standardization: Evaluating RISC-V's Path Across Data Centers, Automotive, and Security," Embedded.com, 20 November 2025,

https://www.embedded.com/fragmentation-to-standardization-evaluating-risc-vs-path-across-data-centers-automotive-and-security/

[12] "RISC-V in 2026: AI, Automotive, and China Drive Adoption," Next Waves Insight, 26 April 2026,

https://nextwavesinsight.com/risc-v-2026-ai-accelerators-enterprise/

[13] Sam Thursfield, "Achieving Long-Term Maintainability with Open Source," Codethink, 18 April 2023,

https://www.codethink.co.uk/articles/2023/ltm-2023/

[14] "RISC-V Linux Kernel Upstreaming: Insights from the 2025 China RISC-V Ecosystem Conference," RISCstar, 21 May 2025,

https://riscstar.com/blog/risc-v-linux-kernel-upstreaming-china-conference/

[15] NASA, "Lost in Translation," NASA Safety Message, August 1, 2009,

https://sma.nasa.gov/news/safety-messages/safety-message-item/lost-in-translation

[16] RISC-V CPU Development in 2025: Is Progress Hindered by Ecosystem or Performance Issues?," PBX Science, 9 November 2025,

https://pbxscience.com/risc-v-cpu-development-in-2025-is-progress-hindered-by-ecosystem-or-performance-issues/

[17] RISC-V International, Annual Report 2025,

https://riscv.org/wp-content/uploads/2026/01/RISC-V-Annual-Report-2025.pdf

[18] Dmytro Humennyi, "Will RISC-V Adoption in Automotive Challenge Traditional Paradigms?" N-iX, July 17, 2024,

https://www.n-ix.com/risc-v-adoption-in-automotive/

[19] Richmond, Roan. "Porting an Automotive Operating System to RISC-V." Codethink. June 2, 2026.

https://www.codethink.co.uk/articles/2026/CTRL-RISC-V.html

[20] "Fragmentation to Standardization: Evaluating RISC-V's Path Across Data Centers, Automotive, and Security," Embedded.com, 20 November 2025,

https://www.embedded.com/fragmentation-to-standardization-evaluating-risc-vs-path-across-data-centers-automotive-and-security/

[21] Michael Larabel, "RISC-V CPU Performance Up 8x In Five Years: SiFive HiFive Unmatched To SpacemiT K3," Phoronix, 9 June 2026,

https://www.phoronix.com/review/risc-v-5-year-performance

[22] Hunter, Lawrence, Nazar Kazakov, Kiran Ostrolenk, and Dickon Hood. "Adding RISC-V Vector Cryptography Extension Support to QEMU." Codethink. June 2, 2023.

https://www.codethink.co.uk/articles/2023/vcrypto_qemu/

[23] RISC-V International, "Adding RISC-V Vector Cryptography Extension support to QEMU," ecosystem news, June 2023,

https://riscv.org/ecosystem-news/2023/06/adding-risc-v-vector-cryptography-extension-support-to-qemu/

www.codethink.co.uk
connect@codethink.co.uk