Harmonic Firmware Initiative

Bringing the Personal Computer Experience
to the RISC-V World

BY

Yuri Zaporozhets

QSOE SYSTEMS

Ravensburg · 2026

1 The RISC-V inflection point

RISC-V has crossed the line from curiosity to platform. An open instruction set has drawn in an ecosystem of silicon that grows month over month, and the roster of available boards now reads like a young industry finding its footing — SiFive, StarFive, MilkV, T-Head, ESWIN, SpacemiT, and more arriving. The cores are real, the toolchains are mature, and the software above them is catching up quickly.

What has *not* kept pace with the silicon is the experience of *using* these machines. The maturity gap is no longer in the processor; it is in everything a person encounters between pressing the power button and reaching a usable system. At first contact, nearly every RISC-V board is a laboratory instrument: a UART at 115 200 baud, a scroll of initialization messages, and an unspoken assumption that the operator owns a serial cable and knows what to do with it. That is precisely the right shape for a development kit. It is precisely the wrong shape for a product.

The next phase of RISC-V adoption will not be won on benchmarks. It will be won, or lost, on the ordinary expectation that a computer behaves like a computer when you switch it on.

2 A development board is not a product

The distance between a development board and a product is easy to feel and worth stating precisely. Three different people form their first impression of a RISC-V board, and today all three are met by the same thing.

The reviewer — at Phoronix, or any outlet whose photographs shape how a platform is perceived — powers the board on and is met by a blank screen and a log. The developer, unboxing a first RISC-V machine, reaches for a serial adapter before reaching for anything to *do*. The procurement evaluator, weighing platforms for a product of their own, sees no evidence that this is a finished thing rather than a component. First impressions are cheap to make and expensive to undo, and the RISC-V world is, at this moment, making the wrong one by default.

Beneath the first impression lies a deeper, structural problem: *fragmentation*. Each vendor ships its own fork of U-Boot, its own display bring-up, its own boot flow, and its own accumulated quirks. There is no shared notion of what a RISC-V computer *does* when you switch it on. The cost of this absence is paid everywhere at once — every vendor solving the same problem, in isolation, incompletely — and the solutions do not compose. A buyer moving between two RISC-V boards encounters two unrelated firmware worlds with nothing in common but the architecture underneath.

This is the gap the rest of this paper addresses. It is not a missing feature. It is a missing *layer*.

3 Why the existing answers fall short

The gap is not for want of firmware. It is that each of the available answers is built for a different job, and none provides the one the ecosystem is missing.

Vanilla U-Boot. The near-universal RISC-V bootloader is powerful, scriptable, and ubiquitous, and it boots systems admirably. But it is a developer's tool. Its native surface is the serial console; its idea of interaction is a command prompt; its idea of a machine is a log. It was never meant to be the face a product shows its user.

UEFI and EDK2. The standardized answer inherited from the PC and server world is comprehensive and specification-driven. It is also heavy to port, shaped throughout by decades of x86

assumptions, and — most importantly — specification-compliance is not the same thing as a product experience. A fully UEFI-compliant board can still greet its user with nothing memorable at all.

coreboot and LinuxBoot. These excel at fast, open, minimal initialization for hyperscale and enthusiast contexts. Their explicit goal is to get out of the way as quickly as possible — to serve the operator who never wanted a firmware screen in the first place. That is a virtue for their audience and the opposite of what a board needs in order to present an identity.

Vendor BSPs. Each vendor's board-support package solves pieces of this — but once, in isolation, and non-portably. The work does not accrue to the ecosystem; it is re-spent on every new board and every new SoC, and none of it carries across a vendor boundary.

The pattern is consistent: each option is good at its intended job, and none provides a *portable product-experience layer* — one that is lightweight, built on the U-Boot vendors already ship, consistent across vendors, and designed to make a board feel finished. That layer is missing, and its absence is structural rather than incidental.

4 What the IBM PC got right

The IBM PC's most durable architectural gift was not a chip. It was an *interface*.

The BIOS defined a stable contract between the machine's firmware and everything above it, and the option-ROM mechanism extended that contract to hardware IBM had never seen. An expansion card carried a small ROM marked with a known signature; during power-on the BIOS scanned for it and executed it, letting the card initialize itself and claim standard services — the video card's ROM taking ownership of the display being the canonical example. The consequence was profound: a user's power-on experience stayed consistent across an explosion of hardware from vendors who never coordinated, because they all targeted the same contract.

RISC-V today has the silicon diversity that the PC ecosystem had, and none of the firmware contract. Each board is its own island.

The lesson is emphatically *not* to reimplement the PC BIOS — its real-mode heritage and x86 assumptions are exactly what RISC-V is fortunate to be free of. The lesson is to reclaim the *idea*: a published contract that decouples experience from hardware, so that adding a new controller does not mean rebuilding the world above it.

5 HFI: the Harmonic Firmware Initiative

HFI is a firmware experience layer built as an *extension* of U-Boot — not a replacement for it. This is deliberate. U-Boot is already the common denominator of the RISC-V board world; HFI leverages that ubiquity rather than fighting it, reading platform inventory — memory, board identity, PCI devices, storage — through U-Boot's own subsystems and presenting it as a coherent machine.

On a real display, and not merely a serial line, HFI delivers what a person associates with a finished computer: a power-on identity and POST screen; boot-device and boot-order menus; a configuration editor in the idiom of a classic system BIOS; and a system console beneath it all for the operator who wants the U-Boot prompt. A kit becomes a machine.

The mechanism that makes this *harmonizing* rather than board-specific is the **unified VideoBIOS interface**. Each board's display controller is described and driven through one common format that HFI consumes — the option-ROM idea, reborn. Where the PC stored that code in a

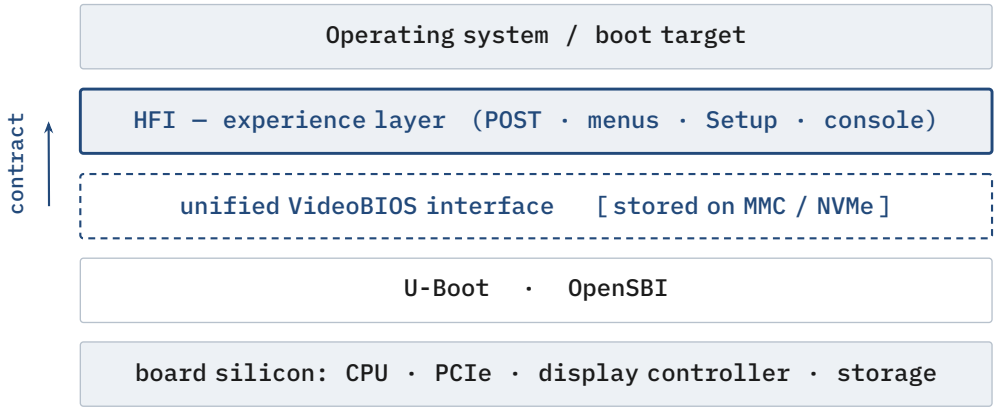

Figure 1. HFI sits above U-Boot and below the OS. A published interface --- not a board-specific driver --- carries the experience across vendors.

ROM on the card, HFI stores it on the storage the board already carries: eMMC, MMC, or NVMe. Porting the interface once for a given controller makes the entire experience above it available, unchanged, and identical to every other board that speaks it — and it carries forward to the vendor's next SoC without re-doing the work. The interface is a portable ABI, not a property of any one GPU.

Because HFI links against U-Boot, it inherits U-Boot's GPL-2.0-or-later licensing. This is not a constraint to be managed. It is the point: the contract is open *by construction*, which is exactly what a harmonizing layer must be.

6 Reference implementation: the hardest case first

The reference implementation runs today on a SiFive HiFive Unmatched, and it drives its display through a discrete NVIDIA GK-208 (GT 710) seated in the board's PCIe slot — brought up natively from inside U-Boot, with no legacy VGA hardware, no video BIOS ROM to execute, and no x86 real-mode assumptions to lean on.

This case was chosen because it is the hardest one available: a scavenged consumer GPU on a foreign architecture, initialized from first principles. The argument it settles is *portability*. If the unified interface can present a coherent BIOS on an unmodified Kepler-class card over PCIe on RISC-V, then a board whose display controller is integrated into its own SoC — the StarFive VisionFive's JH7110, for instance — is a materially *easier* target, not a harder one.

The same implementation renders a full two-panel configuration editor. This matters because it demonstrates that HFI is not a splash screen bolted onto a boot log, but a functioning system BIOS: identity, menus, settings bound to the board's real-time clock and environment, and a deliberate escape hatch to the console underneath. It is photographed on a period display precisely because the point is that it behaves like a computer of the era whose expectations it revives.

7 What changes if this wins

Suppose the major RISC-V boards came to share this layer. What changes, and for whom?

For the board vendor, differentiation a datasheet cannot express: the board that looks finished the moment it is switched on, the one a reviewer photographs and a buyer remembers. Product-experience firmware without standing up and maintaining a firmware-UX team in-house. And forward-carry — the interface, ported once, serves the next SoC as well as this one.

Figure 2. HFI's power-on identity screen: board inventory, storage and USB enumeration, a system console, boot menus---rendered by a discrete GPU on a RISC-V board, photographed on a period CRT.

For the ecosystem, a shared firmware contract where today there are only forks. Effort that accrues to everyone rather than being re-spent per board. Interoperability as a property of the platform rather than a coincidence.

For the developer and the end user, a consistent, familiar, trustworthy power-on experience across vendors — the quiet confidence that the machine is a machine.

QSOE Systems' role in this is that of a neutral steward: maintaining the interface, providing a high-quality reference implementation, and porting it to new controllers. The same firmware foundation is, in time, the natural place where later concerns attach — verified boot, and the partitioning of a general-purpose operating system alongside a real-time one on a single machine — but those are subjects for their own papers.

8 An invitation

HFI is offered as an *initiative*, not as a product to be licensed. The unified VideoBIOS interface is published and open; the reference implementation is open by construction, under the same license as the U-Boot it extends. Vendors are invited to adopt it, to co-develop the interface for their controllers, and to be — for their platform — first.

Harmonization is not a standard imposed from above. It is a contract that nobody owns and everyone can target. QSOE Systems stewards that contract today and stands behind the reference implementation — porting, integration, and support — with the expectation that, as the interface matures, its stewardship can broaden to the wider RISC-V community and its standards bodies.

The RISC-V world has the silicon. What it is missing is the layer that makes a board a computer. We invite you to help us build it.

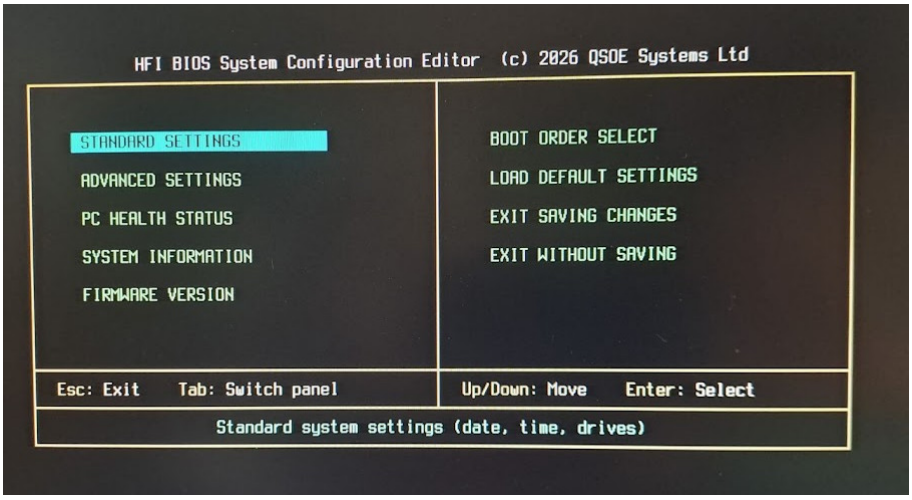

Figure 3. The configuration editor: standard and advanced settings, integrated peripherals, boot order, persisted across power cycles. A real BIOS, not a boot banner.