https://farid.ps/articles/rtl9210_usb_to_nvme_bridge/en.html

Linux and the Realtek RTL9210 USB to NVMe bridge

> **Summary:**
> • **Symptoms:** Repeated USB resets, I/O errors, or drives disappearing under Linux.
> • **Affected:** Realtek RTL9210 (confirmed) and RTL9220 (possibly).
> • **Cause:** Firmware rollback to internal ROM (**f0.01**) after checksum failure.
> • **Impact:** Permanent instability, no Linux reflash tools available.
> • **Fix:** Only OEM Windows utilities can restore firmware - Realtek blocks open-source alternatives.

Preamble

In the year 2025 AD, it should be an entirely reasonable thing to boot a Raspberry Pi from an SSD connected over USB. Yet, thanks to the quirks of Realtek's firmware, that reasonable goal has become an adventure. After months of unexplained instability - random resets, vanishing drives, corrupted filesystems - the author exhausted every ordinary fix: new cables, powered hubs, kernel updates, USB quirks, and firmware tuning. The breakthrough came only when ChatGPT answered an odd late-night question: "Is it possible the USB-to-NVMe bridge reverted to an old firmware?"

Introduction

If your Realtek-based NVMe enclosure suddenly becomes unstable after weeks of flawless operation - repeated USB resets, I/O errors, or disappearing drives - you're not alone. The pattern has appeared across several brands, from no-name units to well-known OEMs such as Sabrent and Orico. The common denominator: **Realtek's RTL9210 and, possibly, RTL9220** USB-to-NVMe bridge chips.

At first, everything works. Then, seemingly without cause, the device starts disconnecting under load or during extended use, especially on Linux or Raspberry Pi systems. The true cause isn't the SSD or the power supply - it's the firmware controller itself quietly rolling back to its **ROM-embedded fallback code**, a version Realtek still ships internally as **f0.01**.

The Hidden Mechanism - Firmware Rollback by Design

Realtek's bridge chips store their operational firmware and configuration data in an external SPI flash. On power-up, the controller checks a simple checksum. If that checksum doesn't match, it refuses to load the external firmware and instead boots from its internal ROM.

That fallback firmware is ancient and defective. It lacks several USB stability fixes and link-state management improvements present in later revisions, leading to the classic sequence every Linux user recognizes:

> usb 3-2: reset high-speed USB device number 2 using xhci-hcd
> usb 3-2: device descriptor read/64, error -71
> EXT4-fs warning (device sda2): I/O error writing to inode ...

The checksum can become invalid when configuration data is rewritten - for example, when the bridge updates its power-management or UAS settings - and the device loses power mid-write. The next boot sees a corrupted checksum and falls back to the ROM firmware permanently.

At that point, your “high-performance NVMe enclosure” behaves exactly like the cheapest no-name shell, because internally it’s now running the same buggy base code burned into the silicon.

Verifying the Problem

You can confirm this state easily under Linux:

```
lsusb -v | grep -A2 Realtek
```

A healthy Realtek bridge reports a firmware revision (**bcdDevice**) above 1.00. A reverted one shows:

```
bcdDevice f0.01
```

That **f0.01** signature means the controller is booting from ROM - and no amount of unplugging, reformatting, or kernel tuning will fix it.

This rollback mechanism has been **confirmed on the RTL9210**. The **RTL9220** appears to share the same design architecture and firmware layout, so it may exhibit identical behavior, but this remains **probable rather than proven**.

Why You Can’t Repair It Yourself

In principle, the fix is trivial: re-flash the correct firmware to SPI. In practice, Realtek makes this impossible.

The company provides a closed-source Windows updater to OEMs and integrators. Linux users are offered nothing. Community developers reverse-engineered compatible flash utilities (**rtsupdater**, **rtl9210fw**, **rtsupdater-cli**) that allowed full firmware restoration from Linux systems - until Realtek issued **DMCA takedown notices** to suppress them.

There is no plausible intellectual-property rationale for blocking such utilities: they don’t expose microcode, only orchestrate the update sequence over USB. Realtek’s takedowns were not about protection. They were ideological.

The Cost of an Ideology

This isn't about open-source idealism. It's about a hardware vendor's **ideological hostility to open systems** breaking devices that are marketed as *Linux-compatible.*

Realtek's resistance to documentation and open tooling has persisted for two decades, spanning Wi-Fi, Ethernet, audio, and now storage controllers. That insularity might go unnoticed in a Windows-only world, but it becomes toxic when those same chips are integrated into multi-platform products like the **Sabrent EC-SNVE**, which openly displays the Linux logo on its packaging.

By forbidding Linux flashing utilities and blocking community maintenance, Realtek has effectively **criminalized self-repair**. The consequences ripple outward:

- Linux users see "supported" hardware devolve into instability.
- OEMs like Sabrent and Orico face unnecessary RMAs and warranty costs.
- Realtek's long-standing reputation for poor Linux compatibility is reinforced yet again.

In the end, it isn't open source that breaks Realtek's devices - it's **Realtek's hostility to open source** that breaks them.

A Rational Path Forward

The solution requires no ideological shift, only pragmatism. Realtek could:

1. Release a vendor-signed, command-line updater for Linux (no source disclosure needed).
2. Publish the checksum algorithm so integrators can safely revalidate flash images.
3. Adopt a DFU-style mode that accepts updates over USB Mass Storage, OS-agnostically.

Each of these would prevent warranty costs, protect OEM relationships, and restore confidence in Realtek's bridge chips among professional Linux users - from workstation builders to Raspberry Pi developers.

What You Can Do

If you suspect your enclosure has reverted to ROM firmware:

- Check with **lsusb -v | grep bcdDevice**.
- If it shows **f0.01**, report the issue to your OEM.
- Include the **dmesg** excerpt and point to this documented rollback mechanism.
- Ask your vendor to escalate the issue with Realtek, citing the need for a Linux-compatible updater.

Realtek's firmware policy doesn't just inconvenience enthusiasts; it creates tangible financial loss for its own ecosystem. The sooner that reality is acknowledged inside the company, the sooner both Linux users and OEM partners can stop wasting time on avoidable RMA cycles.

Manufacturer Responses

Both Realtek and Sabrent were invited to provide statements regarding the firmware rollback issue described above. Their responses - if received - will be appended here.

Appendix - Identifying Affected Devices

Controller	Vendor ID	Product ID	Notes	Status
RTL9210	0x0bda	0x9210	USB 3.1 Gen 2 10 Gb/s bridge	**Confirmed** rollback behavior
RTL9220	0x0bda	0x9220	USB 3.2 Gen 2×2 20 Gb/s bridge	**Possible**, similar architecture

Firmware fallback signature: **bcdDevice f0.01**
Known stable revisions: **1.23** – **1.31**