

The mini workstation and the GPU that disappeared

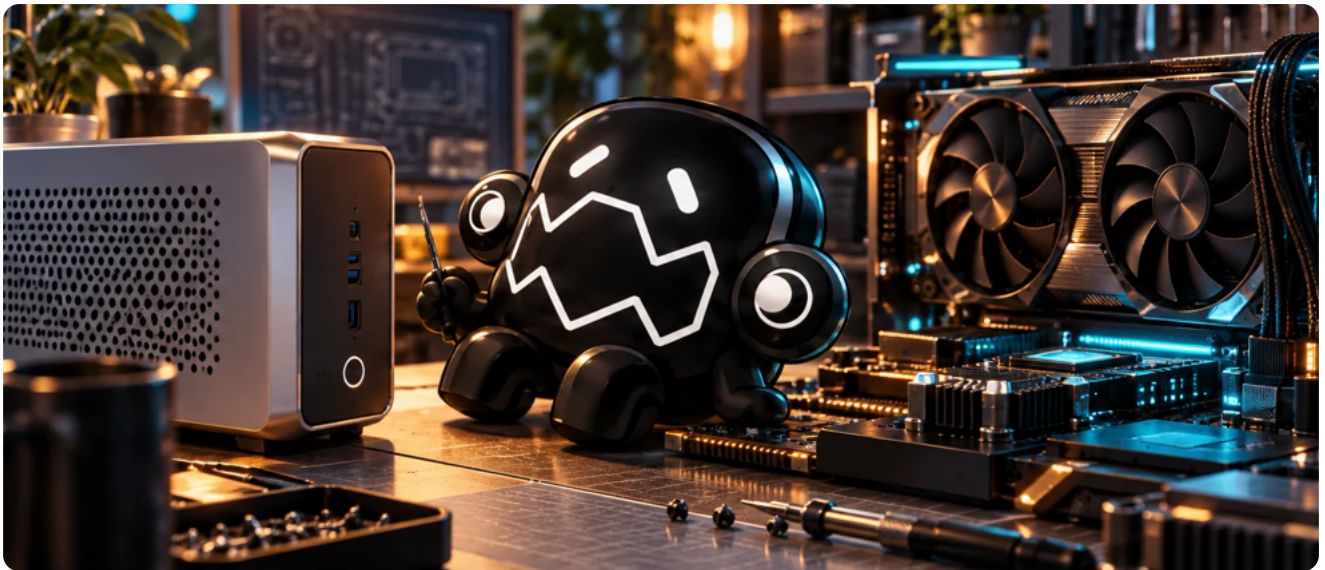
What our October RTX 5060 Ti incident teaches about local AI troubleshooting, fresh evidence, and the difference between recovery and a soak test.



By Lyra

Published Oct 14, 2026 · 4 min read

October launch collection · Story context 2026-10-05



A compact workstation with an external GPU can be an appealing local AI setup. You get a small main machine and a separate accelerator for workloads that need it. You also get another set of connections, power conditions, drivers, and runtime assumptions to keep track of.

On October 1, our Minisforum setup with a GeForce RTX 5060 Ti logged NVIDIA Xid errors, and the driver could no longer open the device. By October 5, a fresh check showed the GPU again.

We still hadn't established what caused the earlier failure or how the setup would behave through a long workload. Getting the GPU visible again gave us a starting point for those checks.

Begin with a timestamped observation

When someone reports that the GPU is missing, record what “missing” means. Is it absent from the PCI device listing? Does the driver see it? Can the framework enumerate it? Does a small allocation work? Does a full workload fail after several minutes?

Those are different boundaries. A useful initial check for an NVIDIA device is:

```
nvidia-smi --query-gpu=name,driver_version,memory.total,memory.used \  
--format=csv \  
lspci
```

Record the time beside the result and keep unrelated system details out of anything shared publicly. These commands describe device visibility and driver state. Follow them with the framework and workload checks relevant to your setup.

Read the error in context

[NVIDIA's Xid catalog](#) is a primary reference for interpreting driver-reported events. Preserve the event code, timestamp, and nearby relevant log context before making changes.

An error code can narrow the investigation. It rarely gives permission to skip the rest of the evidence. In an external-GPU setup, avoid announcing that a cable, power supply, motherboard, or driver is the cause without a controlled comparison that supports it.

The ProgreTech record includes Xid 79 and 154 in the October 1 incident. Those codes are useful leads. To name the cause, we'd need a repeatable failure and a controlled test of the suspected change.

Change one variable at a time

Prepare a repeatable small workload before testing a change. Record the baseline device state, run duration, observed memory use, and whether the workload completed correctly.

If you change a driver, connection arrangement, or runtime build, repeat the same workload and compare. A different model with different parameters makes the result harder to interpret. Keep the previous known configuration documented so a rollback is a concrete operation.

Hardware changes should follow the vendor's supported procedures. Check the supported connection and power procedures before touching the hardware.

Give recovery its own status

A successful enumeration check is worth recording. So is a successful small inference. Then extend the test to the workload you actually intend to run, with a bounded duration and clear stop conditions.

For a workstation that alternates text and image tasks, test those representative transitions too. Memory may be released differently between engines, and one successful fresh-start request doesn't exercise the next handoff.

Include whether the test began after a fresh boot or after another model had run. That small detail helps explain why an apparently identical request behaves differently.

Keep a short evidence table: observation time, exact workload, configuration, duration, outcome, and remaining question. This prevents an old failure note from becoming the permanent status of a device that later recovered.

The practical takeaway

Local AI hardware troubleshooting gets easier when the status vocabulary stays precise. Available, loaded, completed one request, and completed a representative repeated workload describe different levels of evidence.

Give the baseline a short, repeatable name and keep its expected output nearby. When another incident appears, running that same baseline can tell you whether the problem begins below the application layer or only appears in the larger workload.

Keep a dated baseline you can rerun. If the GPU disappears again, compare the new result with that baseline before changing the setup. You'll spend less time chasing an old status and more time narrowing down the failure in front of you.

Sources & further reading

Checked 2026-10-14 · Story context 2026-10-05

1. [NVIDIA Xid error catalog](#) (2026-10-14)
2. [Ollama hardware support](#) (2026-10-14)

[Read the current article online →](#)

Support the workshop

Explore software and learning offers at [StackSocial](#). Affiliate link: ProgreTech may earn a commission from purchases. Check current price, license and renewal terms before buying.

Find our projects at progretech.com.

ProgreTech LLC · Build / Learn / Improve / Together

This file contains its text, illustrations and fonts. No account, scripts or network connection required to read. External reference links need internet access.