START HERE

01

Recover in small, visible steps

A board that stops answering is not automatically broken. In about **30 minutes**, work from the first missing observation, preserve readable files, and finish with a small program that starts after reset.

Put these three items in front of you



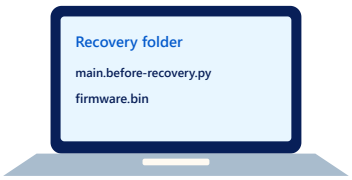
ESP32-S3-N16R8 board

The device you are recovering. Leave external circuits and power disconnected.



USB data cable

Use a cable already known to carry data, not only power.



Computer + private folder

Holds tools, the matching image and copies of your board files.

> **Orient the board:** components toward you, antenna up, sockets down. Native **USB is left**; COM is right. On the front right, **upper RST** resets and **lower BOOT** selects download at reset.

Before you start

- Use ML-C04 if the computer cannot detect the USB connection.
- Have desktop Python 3.10 or newer and internet for the tool and firmware downloads.
- Decide which files or settings you need to preserve before any erase.

> **Use one USB cable and a bare board.** Unplug USB before removing attached hardware. This recovery exercise does not need sensors, an external supply or a battery.

No Python expertise is assumed. The lesson introduces the prompt and every command before you use it.

CHOOSE A RECOVERY PATH

02

Find the first failing layer

A serial **port** is the computer's connection to the board. The **REPL** is MicroPython's interactive prompt, written >>>. They are different checkpoints.

1 · No USB port

Check cable, native USB and the computer.

2 · Port cannot open

Close the other serial owner; check access.

3 · Python replies

Path A: save files and repair the application.

4 · No usable runtime

Path B: known firmware, only after the erase decision.

A missing prompt alone does not prove the board is broken.

Start at the first missing observation, then repeat that check after one change.

Path A · files still accessible

If Python answers or mpremote can list files, save what you need and isolate the failing application first.

Path B · recover firmware

If the runtime is unusable, use the chip's download loader to install a known image after deciding about data loss.

Predict: the computer shows no new port when you plug in. Is an erase the next step?
Check: no. First check the native USB socket, a known data cable and the computer connection.

COMPUTER TERMINAL

Prepare the tools and a fresh folder

03

Firmware is the installed runtime. Your application is made of saved Python files. A bad file can fail while the runtime still works.

Application files	MicroPython runtime
main.py / boot.py	Installed firmware
Your saved program	Runs Python instructions
Repair or rename one file	Reinstall only when needed

A bad program can fail while the runtime still works.

Use mpremote for Python/file access and esptool for firmware recovery.

Create a recovery folder on your computer

Open a terminal there. With desktop Python installed, create an isolated tool environment:

```
python -m venv .venv
.venv\Scripts\python.exe -m pip install "esptool>=5,<6" mpremote
```

The second line is for Windows PowerShell. Below, replace `PY` with `.venv\Scripts\python.exe` on Windows, or `.venv/bin/python` on macOS/Linux. Replace `PORT` with your board's current port name.

```
PY -m mpremote connect list
```

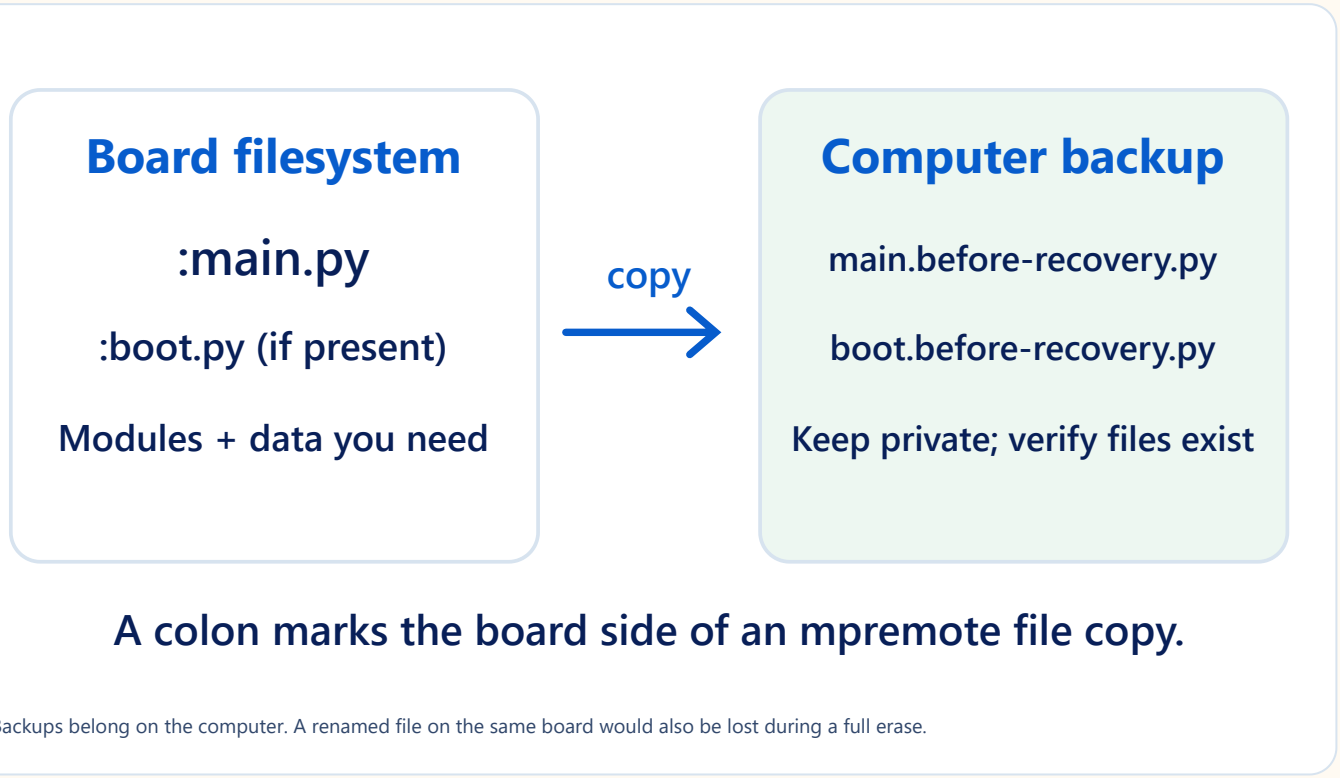
> **Find PORT:** list once unplugged, connect native USB, then list again. `COM7`, `/dev/ttyACM0` and `/dev/cu.usbmodem…` are only example names. Close other serial tools before opening the port.

PATH A · PRESERVE FIRST

Save readable files before changes

04

If MicroPython can answer, recover the files before replacing a program or erasing flash. Keep the computer recovery folder private; configuration files can contain passwords.



List, then copy the files you need

```
PY -m mpremote connect PORT fs ls
PY -m mpremote connect PORT fs cp :main.py main.before-recovery.py
PY -m mpremote connect PORT fs cp :boot.py boot.before-recovery.py
```

Copy `main.py` and `boot.py` only when listed. If a file does not exist, that copy command reports an error; inspect the list rather than inventing a replacement. Repeat the copy operation for your required modules and data files.

Verify: confirm the copied files exist in the computer folder and are readable. Use a fresh folder or unused local names so earlier backups are not overwritten.

A two-file copy is not a full project backup. Preserve the libraries, data and settings your application actually uses. If files cannot be read and you need them, stop before erasing and seek help with data recovery.

mpremote file operations may interrupt and soft-reset Python. Its raw-REPL reset skips `main.py`; a problematic `boot.py` can still prevent access.

PATH A · KEEP THE ORIGINAL

05

Stop one failing startup file

Open the board console. Press Enter; use **Ctrl-C** to interrupt a running Python program. If the program catches that interrupt, it may not return a prompt.

```
PY -m mpremote connect PORT repl
```

Before	After renaming
main.py	**main.disabled.py**
Runs at normal startup	Kept for later inspection
Causes the known failure	Not the startup filename

Choose an unused name. Preserve the file; do not delete it.

A different filename keeps the content for inspection without running it automatically as main.py.

At >>>, type these Python statements after backing up the file. Do not type the prompt symbols:

```
import os
print(os.listdir())
os.rename("main.py", "main.disabled.py")
```

Check both names first. `main.py` must exist, and `main.disabled.py` must be unused. Choose another unused name if necessary. Do not overwrite an earlier recovery copy.

Then: press Ctrl-D at the normal prompt to restart Python. If the failure names `boot.py`, preserve that file too and have its role checked before renaming it. Do not modify the runtime's internal `_boot.py`. If the prompt is usable, continue with the known small program on page 9.

Ctrl-] exits the mpremote console back to the computer terminal.

PATH B · BEFORE AN ERASE

06

Choose the known recovery image

For this N16R8 module, use the generic **ESP32-S3 Octal-SPIRAM** variant. N16R8 identifies 16 MB flash and 8 MB Octal PSRAM; do not choose a firmware file only because its name contains “ESP32.”

Chip	Memory	Format	Address
ESP32-S3	SPIRAM_OCT	Merged .bin	0x0

N16R8: 16 MB flash + 8 MB Octal PSRAM

Worked stable image: v1.29.0

The image format and its documented write address must match.

Download the worked stable image

1. Open MicroPython’s ESP32-S3 download page.
2. Use **Firmware (Support for Octal-SPIRAM)**, then the stable release’s merged `.bin`.
3. Keep the original filename in your notes. Make a copy called `firmware.bin` in your recovery folder.

> **Worked file:** `ESP32_GENERIC_S3-SPIRAM_OCT-20260824-v1.29.0.bin`
> Use this merged BIN at **0x0**, not an application-only `.app-bin`, `.elf` or `.uf2`.

> **Full erase removes all flash contents:** the runtime, Python files and stored settings. Confirm your computer backups first. If you accept losing unreadable files, make that decision before the next page. An erase cannot recover a lost file.

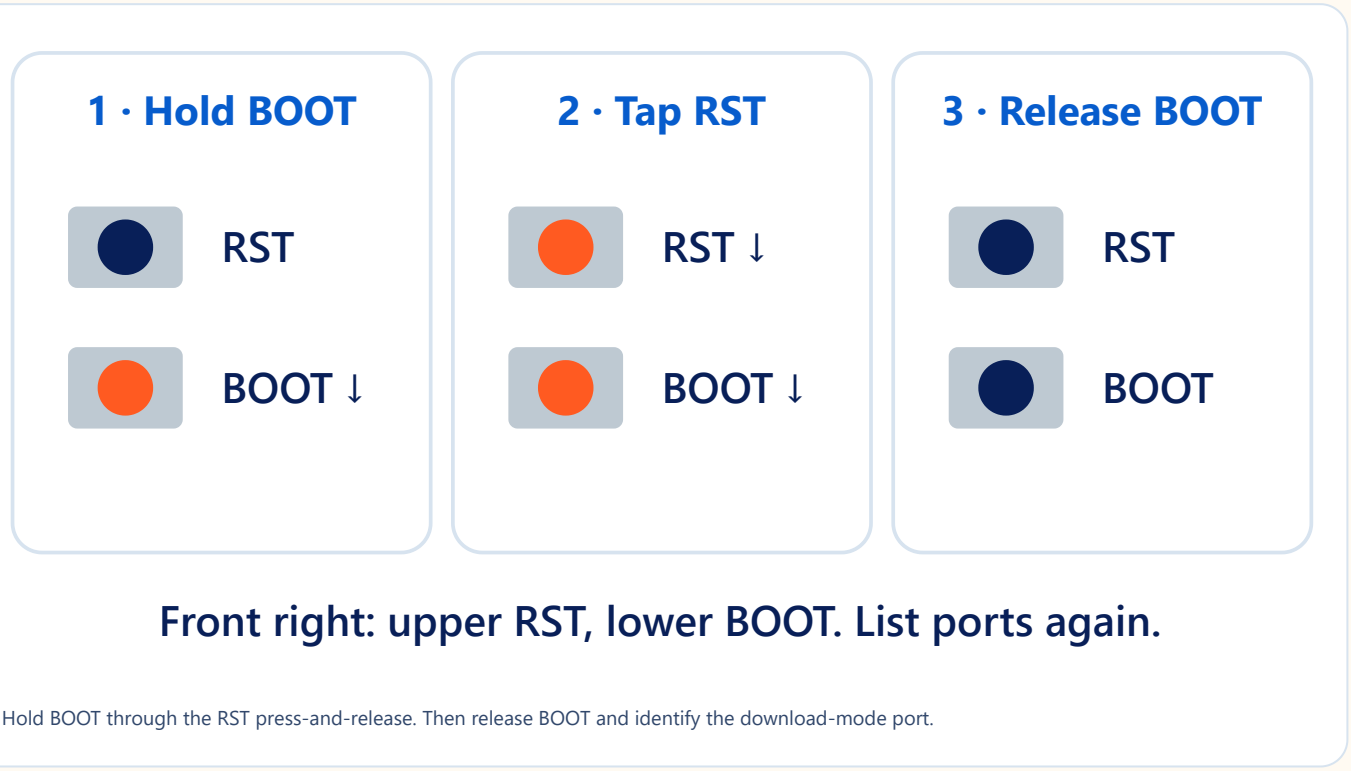
Do not use force options to bypass a secured/encrypted device’s protections. Get its owner’s recovery instructions instead.

PATH B · EXACT SEQUENCE

Enter download mode, then install

07

Keep the known data cable connected to native USB. With the board's front toward you, use the lower BOOT and upper RST buttons:



Hold BOOT through the RST press-and-release. Then release BOOT and identify the download-mode port.

List ports again and replace `PORT` with the current download-mode name. In the computer terminal, inspect the target first:

```
PY -m esptool --chip esp32s3 --port PORT flash-id
```

Proceed only for ESP32-S3 with the expected 16 MB flash. Stop at a chip/size mismatch. Flash ID does not identify the PSRAM type.

After the backup/data-loss decision

```
PY -m esptool --chip esp32s3 --port PORT erase-flash
PY -m esptool --chip esp32s3 --port PORT write-flash 0x0 firmware.bin
```

Wait: erase completes, writing completes and verification succeeds without errors. Do not unplug during writing. If a command cannot reconnect, repeat BOOT → RST → release BOOT and rediscover PORT.

Start MicroPython: release BOOT and tap RST alone. The runtime USB port may differ from the download port. List again before opening the console.

VERIFY THE RUNTIME

08

Prove that Python is running

A successful flash write is one checkpoint. The next is a reply from the installed runtime. Reconnect using the current **runtime** port:

```
PY -m mpremote connect PORT repl
```

Example board console

```
>>> import sys
>>> print(sys.implementation.name)
micropython
>>> 2 + 3
5
>>> print("ML-C10 ready")
ML-C10 ready
```

Example input and expected replies. Type the white statements after the prompt; do not type the prompt symbols.

Read three useful observations

- `sys.implementation.name` replies `micropython`.
- `2 + 3` replies `5`.
- Your harmless greeting appears, followed by another prompt.

Record the version too: at the prompt, type `print(sys.version)`. Keep the firmware filename and current port in your recovery notes; do not include passwords or raw private configuration.

No prompt? Release BOOT, tap RST, list ports again, press Enter and try Ctrl-C. Recheck the selected S3/Octal image and 0x0 address if startup keeps failing. Repeated erases will not fix a power-only cable.

VERIFY THE NEXT RESTART

09

Install a tiny known-good startup file

Use a small application that prints once and exits. It uses no pins, network or external hardware. Save ready_main.py from the practice folder into your recovery folder:

```
print("ML-C10 ready")
```

Normal restart	Startup file	Visible result
Runtime starts	main.py executes	ML-C10 ready

Copying a file is not the final check. Restart and observe.

The test checks a saved startup file, not only a line typed into the current session.

Exit the console with Ctrl-]. Preserve any existing `main.py` before this copy. In the computer terminal:

```
PY -m mpremote connect PORT fs cp ready_main.py :main.py
PY -m mpremote connect PORT repl
```

Success: at a normal prompt, press Ctrl-D and see `ML-C10 ready`. Tap RST with BOOT released for a hardware-restart check; reconnect if the port changes. If the early message was missed, Ctrl-D at the prompt shows it again.

Keep this checkpoint: save the known firmware filename and this small program on the computer. Do not restore the failing application yet.

OPTIONAL FILE EXERCISE

10

Rehearse recovery without a board

Practice the file decisions on the computer before doing them on a device. The included script creates a fresh folder; it does not open a serial port or flash anything.

Run the included rehearsal

Open the practice folder's rehearse.py with desktop Python from that folder:

```
python rehearse.py
```

simulated_board/

Contains a deliberately failing `main.py` and an unrelated `keep.txt`. This is an ordinary computer folder, not a board emulator.

computer_backup/

Holds a byte-for-byte copy of the original file before it is renamed.

1. The failing script reports its deliberate `RuntimeError`.
2. The original is copied to the backup and renamed to `main.disabled.py`.
3. A known small `main.py` runs and prints `ML-C10 ready`.
4. The backup, disabled original and unrelated file remain intact.

Expected: four PASS lines and a final folder location. Inspect that folder to point to all three versions. The rehearsal refuses an occupied quarantine name or a missing/mismatched backup.

Explain the boundary: these file checks do not prove the board can enumerate, flash or run MicroPython. Complete the board reply and restart observations for those questions.

FINISH AND TROUBLESHOOT

11

Restore one piece at a time

Keep the recovered baseline small. A broken application can recreate the same symptom as soon as you copy it back.

1 · Keep a clean baseline

Save the firmware name and a working small program.

2 · Add one dependency

Restore one known module or setting, then retest.

3 · Add the application last

Repair the failing file before returning it to main.py.

Restore deliberately, checking the small known program after each change.

Loader connects, no Python

Release BOOT, tap RST, find the runtime port. Recheck image variant, file type and address before another deliberate flash.

Failure returns with one file

Stop, save the new traceback privately and isolate that file. Repair the application instead of erasing a working runtime again.

Port absent or busy

Return to ML-C04: native USB, known data cable, one serial owner and account access. Do not solve a link issue with an erase.

Write fails midway

Keep a direct USB connection and known data cable. This lesson uses the default transfer speed; remove optional fast-baud settings if you added them.

Your completion check

- □ I identified the first failing layer and chose Path A or B.
- □ I preserved the files I need or understood the erase's loss.
- □ The runtime identifies itself and answers the small tests.
- □ The saved small `main.py` prints after restart.

Keep the recovery checklist with the known firmware and a working copy of your program.