LIBRARY · REFERENCE GUIDE

FLASHING FIRMWARE

How your compiled program gets into a microcontroller: the ESP32 enters its bootloader, esptool sends the binary over USB, and the next reset runs it.

ONE THOUSAND DRONES ENGINEERING TEAM · VERIFIED 2026-07

Flashing is how your compiled program gets into the microcontroller. The ESP32 enters its bootloader over USB, a tool sends the binary across, the tool writes it into flash, and the next reset runs it. That load-and-go loop is the same every time you hit upload, and understanding it turns most flashing problems from mysteries into checklists.

FROM SOURCE TO BINARY

Your code compiles to a binary image: the exact bytes the chip will execute. Flashing writes that image to a fixed address in the ESP32's flash memory, where it survives power-off. On the next reset, the tiny bootloader baked into the chip's ROM finds the image, loads it, and jumps into it. Change your code, recompile, reflash, and the loop repeats.

THE BOOTLOADER AND ESPTOOL

To accept a new image the chip must be in download mode, set by the boot strapping pin from the last lesson. Then `esptool`, Espressif's flashing tool, talks to the ROM bootloader over USB or a serial UART and streams the binary in. The ESP32-S3 has native USB built into the chip, so it can be flashed straight over a USB-C cable with no separate USB-to-serial part in the path at all.

- Espressif. esptool documentation (flashing over USB/UART, the ROM bootloader). `docs.espressif.com`

ADDRESSES, PARTITIONS, AND ERASING

Flash is divided into partitions: the second-stage bootloader, a partition table that maps the rest, your application, and often a data area for stored settings. Each lives at its own address, which is why a flash command usually lists several files with offsets. Erasing flash wipes all of it back to blank, which clears a bad image or stale stored data so you can start from a known-clean chip.

DEEP DIVE · WHY EVEN A BLANK CHIP STILL NEEDS THE BOOTLOADER FLASHED

The second-stage bootloader and the partition table live in flash right alongside your app, so a truly blank chip has all three written at once (that is what the multi-file flash command is doing). The very first stage, the one that receives the flash in the first place, is a small bootloader fixed in the chip's ROM at the factory and cannot be changed or erased. That ROM bootloader is your safety net: even if you erase everything and flash a broken image, you can always drop back into it and try again. (Espressif ESP-IDF)

- Espressif. ESP-IDF Programming Guide: Get Started (build, flash, and monitor). docs.espressif.com

COMPILE, SEND OVER USB TO THE BOOTLOADER, WRITE TO FLASH, RUN ON RESET.

> **YOU USUALLY DON'T NEED TO PRESS BUTTONS**
>
> On most ESP32 boards the flashing tool drives the reset and boot pins for you through the auto-reset circuit, so you just run the upload and it works. If a flash refuses to start, force download mode by hand: hold BOOT, tap and release reset, then release BOOT, and try the flash again.

CHECKPOINT

1. What must the ESP32 be in to accept a firmware flash?

a. Deep sleep

b. Its bootloader (download mode)

c. Full-speed run mode

ANSWER · B

The strapping pin selects download mode, where the ROM bootloader listens for the flashing tool.

2. What does 'erase flash' do?

a. Wipes the flash back to blank, clearing the image and stored data

b. Deletes the chip's ROM bootloader permanently

c. Turns off the CPU

ANSWER · A

It clears flash to a known-blank state; the ROM bootloader is fixed and untouched.

3. How can an ESP32-S3 be flashed without a separate USB-to-serial chip?

a. Only over Wi-Fi

b. It cannot be, one is always required

c. It has native USB, so a USB cable reaches the bootloader directly

ANSWER · C

The S3's built-in USB lets a plain USB cable talk to the ROM bootloader with no bridge chip.

- Prerequisite: boot and strapping pins
- Next: clocks and timers