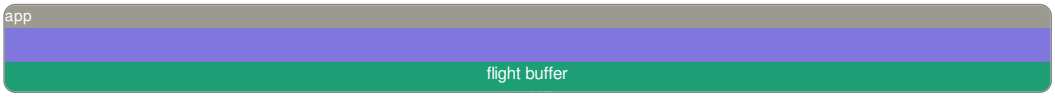
Data storage

Jupiter carries 32 MB of flash on a quad-SPI bus clocked at 80 MHz, giving a peak transfer rate of 40 MB/s, and divides it into regions that each do one job in the way that job is best done: a raw high-speed buffer for the flight itself, a conventional FAT16 volume for finished flight logs, a small filesystem for the device's own housekeeping, and a rolling black box of the console. The mental model that makes sense of all of it: the website is the archive, with no storage limits on uploaded flights, and the device is the working store plus a cache of your most recent flights.



LittleFS 1 MB, housekeeping console black box 1 MB firmware + system regions

The flight buffer

During a flight, nothing as slow as a filesystem is allowed near the data, and the flight core never touches flash at all. Samples stream from the 600 Hz flight loop into buffers in RAM, and Jupiter's second processor core does the collecting: it continuously drains those buffers and commits the data into a dedicated 7.8 MB raw flash region, no directory structures, no allocation tables, just an append-only stream written in efficient blocks. One rule bounds the risk: a partial block is committed to flash at least every 500 milliseconds, so however the flight ends, at most half a second of data can ever be in transit. The buffer holds nearly nine minutes of recording at the full 600 Hz rate, and the sample-ratio setting multiplies that span for long flights. This split is also what makes a flight crash-proof: if power were lost mid-flight, everything already committed survives, and at the next boot Jupiter finds the held flight and completes it into a proper log, so the data you recorded is the data you keep, even on the worst day.

That recovery is worth stating in the strongest terms. If something catastrophic happens mid-flight, a power cut, a hard landing, a failure of the airframe itself, the flight log up to the moment of failure is recovered automatically at the next power-on, whenever and wherever that is. And because firmware and data alike live on the one flash chip, even a physically destroyed board is not necessarily the end of the story: it is technically possible for a professional to transplant the flash chip into a replacement Jupiter, and on its next power-on the recovery runs exactly as it would have on the original, completing the held flight into a proper log. The board is replaceable; the record of what happened often matters more.

The FAT16 store

When a flight ends, it is compressed, signed and written as a .aclz file into a 20 MB FAT16 volume, the same filesystem every computer on earth reads without drivers. Because the ACLZ format compresses flight data to roughly 100 KB per minute at the full recording rate, the store's capacity is best thought of in minutes: with around 2 MB of the volume in practice serving the console logs, the remaining space holds an estimated 180 minutes of total flight time at full rate, and around 500 minutes under the default hybrid sample-ratio. In flight-count terms that is typically fifty or more flights on the device at once, before space is even a question. The console black box lands here too: each session's log is rolled out as a text file at boot, so the device's own diagnostic history rides alongside your flights.

Housekeeping: LittleFS

A separate 1 MB filesystem holds the device's working state, the GNSS orbit and weather caches, the latest test-fire trace, upload bookkeeping and similar, kept deliberately apart from your flight data so that housekeeping churn and flight storage never share a volume.

Making room

When a new flight needs space and the store is full, the oldest flight on the device is removed first, automatically. This is safe precisely because of the archive model: flights upload to AltimeterCloud, where there are no storage limits, so the copy on the device is your recent-flights cache, not the only copy. The website keeps everything, forever; the device keeps what's newest.

Checking the volume

FAT16 keeps two copies of its allocation table, and Jupiter uses that. At every boot, before the volume is even mounted, a raw health check runs against the storage: the two table copies are compared, and if they disagree, the damaged one is repaired from the intact one, catching the classic corruption pattern before any software has cached a stale view. The volume's structural records are checked for consistency and patched if needed. And one rule is absolute: a volume that contains data but will not mount is never automatically reformatted. It is preserved exactly as found and still exposed over USB, so a PC recovery tool gets its chance at your data; formatting only ever happens on genuinely blank flash at first use.

USB access

Plug in a USB cable and Jupiter appears as an ordinary USB drive containing your flight logs and console history, no software, no vendor tool, any operating system. The drive is deliberately read-only to the computer: you can copy anything off, but the host can never write, which means no operating system, indexer or antivirus can ever corrupt the volume or your flights. Copy an .aclz off directly if you want to work with it yourself, the format and a reference converter are documented on the ACLZ format page, or simply let the flights upload themselves and never think about any of this at all.