ENGINEERING & USER MANUAL

XType

Distraction-free Markdown writing firmware for the Xteink X4 (and experimental X4 Pro), built from scratch for fast e-paper typing, USB browser flashing, and seamless dual-booting alongside CrossPoint Reader.

HARDWARE TARGETS
Xteink X4 (ESP32-C3, verified)
Xteink X4 Pro (ESP32-S3, experimental)

DUAL-BOOT FIRMWARE
XType (ota_0) ↔ CrossPoint (ota_1)
Web Flasher via Web Serial

HOST VERIFICATION
11,138,504 ASan/UBSan checks
Web manifest & MD5 verification

INSIDE THIS GUIDE

Part 1 — How It Was Built

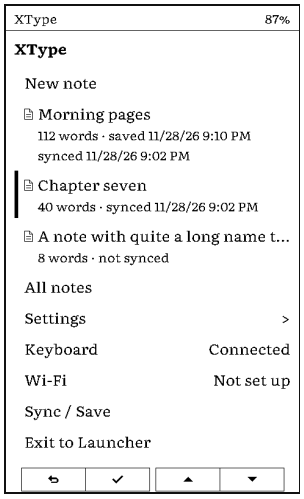
- Hardware constraints & X4 Pro port
- Portable C++ core vs. firmware HAL
- Fast SSD1677 typing waveforms
- Non-blocking serial & BLE key guard
- Safe SD saves & otadata dual boot

Part 2 — Using XType

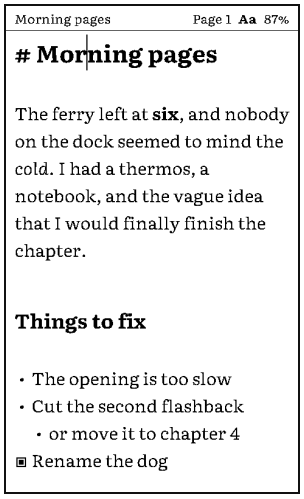
- Flashing over USB in the browser
- Pairing a Bluetooth LE keyboard
- Plain & live rendered Markdown
- Half-page scrolling & shortcuts
- Fonts, refresh speeds & Drive sync

Part 3 — Moving to CrossPoint

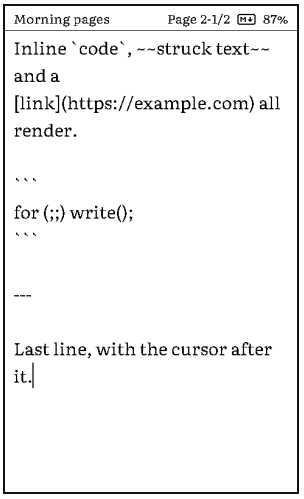
- 16 MB two-slot flash layout
- Switching back and forth on-device
- Persistence across sleep & power-off
- Shared NVS, UI themes & updates
- USB recovery back to XType



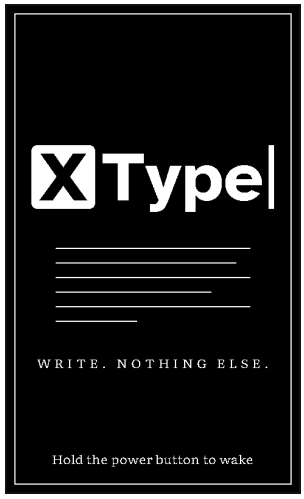
Main menu & recents



Live rendered (Aa)



Plain Markdown & 1/2 page



Deep-sleep splash

PART 1

How XType Was Built

Writing smoothly on an e-paper reader presents two unusual bottlenecks: normal e-paper updates take ~430 ms per frame, and cheap ESP32 boards without PSRAM leave only ~25–30 KB of usable contiguous heap once BLE and TLS have run. XType was architected around both constraints from day one.

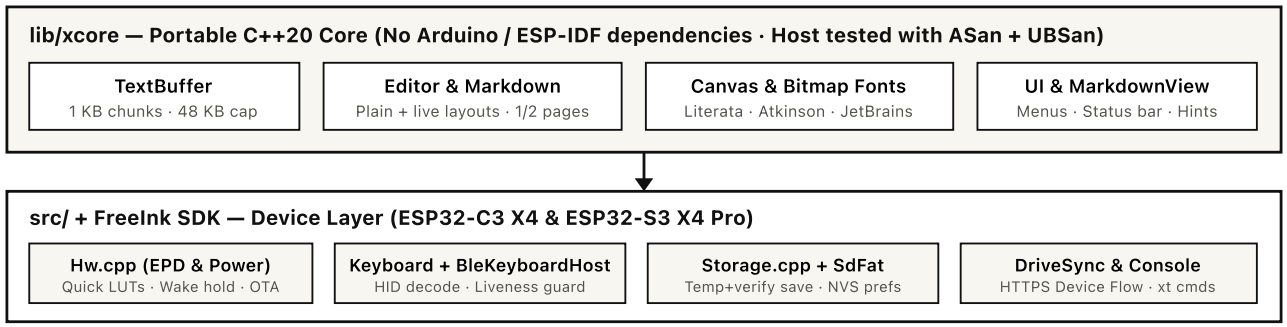


Figure 1 — Clean separation between the host-tested C++ core (`lib/xcore`) and ESP32 hardware drivers (`src/`).

1. Portable Core & Host Test Suite

Every editing rule, Markdown parser edge case, word-wrap calculation, glyph rasterizer, and half-page scroll step lives inside `lib/xcore` with zero Arduino or FreeRTOS headers. This enables running over 11.1 million automated checks under Clang's AddressSanitizer and UndefinedBehaviorSanitizer in seconds via `tests/host/run.sh`, plus headless PNG screen rendering via `tests/host/render.sh`.

2. Custom Quick Waveforms & Lag Elimination

- **Direct SSD1677 typing waveforms (`src/Hw.cpp`)**: Instead of a 430 ms whole-screen SDK partial pass plus ~85 ms rail power-cycling, XType hashes framebuffer rows and byte columns to locate the exact dirty rectangle, uploads only that window to both controller memories (BW and RED RAM), and runs a short 20 ms/frame custom LUT (`Type`).
- **Greying & contrast management**: Voltage rails stay warm for 2 seconds while you keep typing so keystrokes don't pay the 85 ms rail startup cost. Every 16th quick update runs `TypeHold` to nudge unchanged pixels back to their own color without extra delay, and a 2-frame `Settle` pass fires 350 ms after typing pauses to crisp up strokes. On X4 Pro units with UC8179/UC8279 controllers, XType automatically falls back to the SDK's window/fast refresh.
- **Zero-wait USB serial logging**: Arduino's USB-CDC `Serial` blocks up to 2 seconds per log line when plugged into a host that isn't reading the port. Calling `Serial.setTxTimeoutMs(0)` at boot drops uncollected debug lines immediately, restoring interactive typing latency even while charging from a laptop.

3. Memory Discipline, BleKeyboardHost Guard & Safe Saves

- **1 KB block-based `TextBuffer`**: Notes up to 48 KB (~8,000 words) are stored as small fixed chunks rather than a single contiguous `std::string`, preventing heap exhaustion after Wi-Fi TLS permanently

keeps ~22 KB of internal DRAM fragmented.

- **Stuck-key auto-repeat protection**: `lib/BleKeyboardHost` adds an asynchronous `requestInputReport()` GATT probe fired every 150 ms while a key is held. If the keyboard fails to answer within 450 ms (for instance after a dropped BLE link or lost release notification), key repeat halts automatically instead of printing endless characters.
- **Verified SD saves**: Every autosave or manual save writes to a temporary file, flushes, reads back and verifies contents before replacing the note, and refuses sleep or firmware switching if a save fails. On the X4's shared SPI bus, SdFat is built with CrossPoint's three cache-safety patches.

4. Xteink X4 Pro Support & On-Screen Keyboard (Experimental)

Adding the `[env:x4pro]` ESP32-S3 target in `platformio.ini` reuses the same `lib/xcore` UI and derives display pins, 1-bit native SDMMC storage (`GPIO41/42/40` + `GPIO5`), `BoardConfig::holdPowerRails()` (asserting `GPIO1` for GT911 touch), CW2017 fuel gauge, and `ESP_SLEEP_WAKEUP_EXT1` wake handling from `freeink-sdk`. An orientation-aware capacitive on-screen keyboard (`TouchKeyboard`) with letters, numbers, and full ASCII symbols works across every note and text prompt and can be dismissed or reopened at any time. Because XType hasn't yet been run on physical X4 Pro hardware, X4 Pro builds are labeled *Experimental* in the Web Flasher alongside a 16 MB flash backup tool.

PART 2

How to Use XType

1. Installing via the Web Flasher (Recommended)

1. Open the XType Web Flasher (`https://xtype.aboutrussell.com/`) in desktop **Chrome**, **Edge**, **Brave**, or **Opera**.
2. Select your device: **Xteink X4** (ESP32-C3) or **Xteink X4 Pro** (ESP32-S3, experimental — run **Download 16 MB Backup** under *More tools* first).
3. Pick either **Dual Boot: XType + CrossPoint** (recommended) or **XType Only**.
4. Connect the device over USB-C, wake it (hold power ~1 s if asleep), click **Connect via USB**, verify the detected chip & layout, and click **Flash**. Wait for all regions to verify MD5 checksums and restart.

2. Pairing a Keyboard & Device Buttons

Use the physical buttons to navigate menus before your keyboard is paired (the device only supports **Bluetooth LE** keyboards with US QWERTY layout):

- **Pair once**: Select **Keyboard → Pair a new keyboard**, put your keyboard into pairing mode, and select it from the list. Afterwards it reconnects automatically on boot and wake.
- **On Xteink X4**: Icon button boxes (`↩` , `✓` , `▲` , `▼`) follow the physical button edge in all four orientations, with matching front and side rocker navigation.
- **On Xteink X4 Pro**: Physical button indicator boxes are hidden; tap any menu item directly to select it (and tap top-left `<` or capacitive **Home** for Back, or top-right `< 1/2 >` for menu pages). In notes, tap to toggle Plain/Rendered, long-press for **Note Settings**, and open/close the 4-row touch keyboard (full ASCII letters, `123` , `#+=`) from the top-right keyboard icon, its Hide key, or inside every text prompt in all four orientations.

3. Writing, Views & Half-Page Scrolling

- **Plain vs. Rendered Mode**: Press `Ctrl+P` (or the **OK** button) at any time inside a note to toggle between raw Markdown (status bar shows `[M↓]`) and live rendered Markdown (status bar shows `Aa`).
- **Half-page paging**: Page keys (`PgUp` / `PgDn` , side buttons, or Left/Right buttons) always move the viewport by half a screen so you never lose context. The top bar displays `Page 1` , `Page 1–1/2` , `Page 2` , etc.
- **Settings & refresh speeds**: Switch between **Literata** (Serif) and **Atkinson Hyperlegible** (Sans), pick sizes **XS–XL**, rotate through Portrait/Landscape orientations, and choose a Screen Speed preset from *Fastest* (lowest latency) to *Crisp*. Open **Markdown ReadMe** inside *Settings* for an in-app cheat sheet.
- **Sleep & Long-Press Wake**: Hold the power button (~0.7 s) to save and sleep (displaying the black XType splash screen), or tap it briefly on any screen for a full clean-up refresh. When asleep, hold the power button (~1 s) to wake — brief knocks in a bag go straight back to sleep untouched, and the button must be released after waking before a long press can sleep again.

SHORTCUT / CONTROL	ACTION IN A NOTE	MARKDOWN SYNTAX (LIVE RENDERED)	RESULT
`Esc` / Back button	Save & open main menu	`#`, `##`, `### Title`	Headings (H1–H3)
`Ctrl+P` / OK button	Toggle Plain (`[M↓]`) / Rendered (`Aa`)	`**bold**` / `*italic*`	Bold & italic text
`Ctrl+I` / Hold OK	Note settings (rename, Save / Sync, word count, timestamps)	`` `code` `` / `~~strike~~`	Monospace / strikethrough
`Ctrl+S`	Save immediately (also auto-saves)	`- item` / `1. item` / `- [x] task`	Bullets, numbered & checklists
`Ctrl+=` / `Ctrl+-`	Increase / decrease font size	`> quote` / ```` ```code``` ```` / `---`	Block quote, code block, rule
`Ctrl+L` / Tap Power	Clean-up full refresh (clears ghosting)	`Ctrl+Left/Right`, `Ctrl+Home/End`	Word jump & start/end of note

4. Wi-Fi & Google Drive Sync

Configure Wi-Fi under **Wi-Fi** in the main menu, then open **Sync / Save** (or **Save / Sync** inside a note's Note settings menu). First-time sign-in shows a Google verification URL, QR code, and short code (OAuth Device Flow) you enter on your phone or laptop. Notes are backed up one-way into the `/XType` folder on Drive; progress screens use clean-up refreshes, and XType automatically retries the clock handshake up to 3 times if Wi-Fi is connected. Each sync updates the device clock and records `saved` / `synced` timestamps in local time.

PART 3

Moving Between XType & CrossPoint

Instead of relying on an intermediate launcher that takes turns overwriting a single slot, XType Dual Boot uses a standard two-slot 16 MB ESP32 partition table (`partitions_dual.csv`) identical to CrossPoint's own layout. Both firmwares live permanently in flash at the same time.

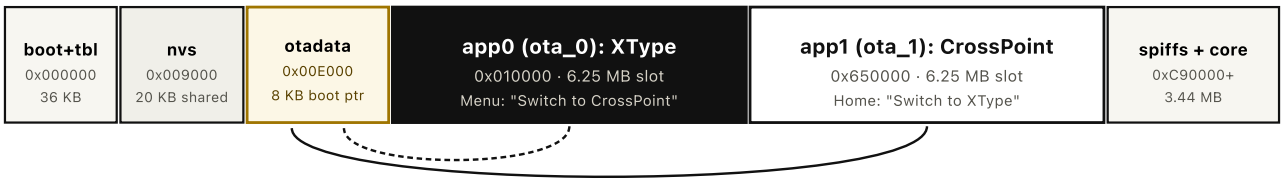


Figure 2 — Shared 16 MB dual-boot flash partition map (`partitions_dual.csv`). Blank `otadata` boots `app0` (XType).

1. Switching on the Device

- **From XType to CrossPoint**: Open the main menu (`Esc` or Back), scroll to the bottom, and select **Switch to CrossPoint**. XType saves your current note, checkpoints the clock, cleanly disconnects the Bluetooth keyboard, updates `otadata` to point at `app1`, verifies the write by reading it back, and reboots into CrossPoint.
- **From CrossPoint to XType**: Select **Switch to XType** at the bottom of CrossPoint's home menu (or tap the `XT` tab in *Cover Grid*, or select **Switch to XType** at the bottom of CrossPoint's top-level **Settings** menu). CrossPoint displays *"Switching to XType..."*, writes the next sequence entry into `otadata` for `app0`, and restarts back into XType.
- **Stays selected across sleep & power-off**: Because the bootloader checks `otadata` on every cold boot and deep-sleep wakeup, whichever app you switched to last remains active until you explicitly switch again.

2. Important Things to Know

Works in Every UI Theme
Whether you use *Classic*, *Lyra*, *Lyra 3 Covers*, *RoundedRaff*, or *Cover Grid* (whose bottom tab bar gets an `XT` icon), **Switch to XType** is always available both on the Home screen and at the bottom of CrossPoint's top-level **Settings** menu.

Shared NVS & Bluetooth Bonds
Both firmwares share the 20 KB `nvs` settings partition starting at `0x9000`. Your XType Wi-Fi credentials, Google OAuth token, and CrossPoint settings live side by side, and NimBLE Bluetooth bonding records (XType's keyboard and any CrossPoint BLE page-turner) share the same NVS bond list.

> **Why CrossPoint OTA & SD Firmware Updates Are Blocked**
> Stock CrossPoint's OTA and SD-card firmware updaters always write to *"the other OTA partition"* — which in this layout is `ota_0` where XType lives. To keep you from accidentally erasing XType, the patched CrossPoint build refuses both OTA and SD updates with *"Not in this build: it would overwrite XType"*. To update either firmware, use the Web Flasher or `scripts/flash_dual.sh` / `flash_app.sh`.

3. Updating or Recovering Over USB

- **Force boot back to XType**: Connect your X4/X4 Pro in the Web Flasher, open *More tools*, and click **Force boot to XType** — or run `scripts/flash_app.sh --start-xtype` from the CLI. This blanks `otadata` (`0xE000–0xFFFF`) without touching your notes, CrossPoint, or saved NVS settings.
- **Fast XType-only updates**: Choose **Update XType only** in the Web Flasher (or `scripts/flash_app.sh`) to update `ota_0` without rewriting CrossPoint or `nvs`.