

Fader Control — User Manual

VIOLET AUDIO · DMIX128 CONTROL SURFACE · VERSION 2.0

Introduction

Connecting to the dMix

The Mixer Surface

Channels & Buses

Hardware Control Surfaces

MIDI Mapping & Learn

View Groups & Mute Groups

Keyboard Commands & GUI Control

Snapshots, Shows & Scenes

Touchscreen, Tabs & User Pages

Touch & Turn Encoder

Wireless Monitoring

The Companion Bridge (`dmix_bridge.py`)

Display & Settings

Debug & Logging

Troubleshooting

Appendix — Protocol Reference

Introduction

Fader Control is a complete control surface for the **dMix128** digital mixing console. It turns a computer — and, optionally, one or more MIDI control surfaces — into a full tactile mixing desk, talking to the engine over its native protocol with no proxy hardware in between.

Run it as a signed native app on **macOS** or **Windows**, or in a modern browser. It drives up to **four Mackie Control (MCU) surfaces** (32 motorised faders) plus HUI surfaces, gives you on-screen faders and a touchscreen layout, mirrors everything back to the hardware (motors, LEDs, scribble-strip LCDs), and even monitors your **wireless microphones and IEMs**.

This manual covers every area of the app: connecting, mixing, hardware surfaces, MIDI mapping, view/mute groups, GUI control, snapshots, the touchscreen pages, wireless monitoring, the companion bridge, settings, and troubleshooting.

Two ways to run it. As a **native desktop app**, Fader Control does its own MIDI, keystrokes and mixer handshake with no helpers required. In a **browser**, some features (OS keystrokes, wireless monitoring, and the mixer handshake on a non-mixer-hosted page) use the small **companion bridge**, `dmix_bridge.py`.

Connecting to the dMix

Making a connection

1. Click **CONNECT** in the top bar.
2. In the **Connect to dMix128** dialog, pick a preset server or type the mixer's address in the custom field and press Enter. Presets include **dMix Local (mDNS)** `dmix.local:80`, **Localhost** `127.0.0.1:80`, and a cloud dev server.
3. The address is remembered for next launch.

The status dot and label in the top bar show the state — grey while connecting, **green** with the mixer's address when connected, red on error — alongside a parameter count, the current show/snapshot name, and the MIDI and strip-count pills.

Fader Control speaks **Socket.IO (Engine.IO v3)** over a WebSocket. On connect it performs the handshake, then sends an **INIT** to pull a full snapshot of the desk's state (every channel, bus, name, colour and value) and brings the surface live.

dMix V2 — the allocation handshake

Newer **dMix V2 firmware** requires each browser-style client to identify itself with the mixer's current **allocation identity** before it can make changes. Fader Control handles this automatically: before sending INIT it reads the identity from the mixer's `GET /api/setup` and includes it as `allocIdentity`. **Control writes stay paused** until the mixer confirms the identity in its INIT reply. (Older V1 mixers, which have no `/api/setup`, simply fall back to the classic INIT.)

Fader Control fetches `/api/setup` by whichever path is available, in order:

1. **Native (app)** — the desktop app reads it in its Rust backend, which is not subject to browser cross-origin (CORS) rules. *Nothing else needed.*
2. **Direct browser fetch** — works when the page is **served from the mixer** (same origin) or the mixer sends CORS headers.
3. **Bridge relay** — if a browser page is blocked by CORS, it asks the companion `dmix_bridge.py` (on `ws://127.0.0.1:8765`) to fetch `/api/setup` for it.

If none of these can supply the identity, the log tells you to run the bridge, serve the page from the mixer, or enable CORS — and the desk will reject writes with `stale_allocation` until

an identity is provided.

Staying connected

If the mixer hands control to another client, Fader Control receives `ALLOC_INVALIDATED` (or `ALLOC_BUSY` during a transition). It **pauses writes, re-reads the identity and re-initialises automatically** with a backing-off retry — it will not hammer the desk. Use **RE-INIT** (in the log panel) to force a fresh handshake at any time.

The Mixer Surface

Fader strips

The main view shows a row of channel strips — **16 by default**, selectable from 8/12/16/24/32 (or a custom 1–32) in the MIDI dialog. The desk has **128 channels**; the strips are a window onto them, moved with the **bank** controls (by a full page, or one channel at a time).

Each on-screen fader:

- **Drag** to set level; the dB value is shown live (with a −∞ floor).
- **Double-click** snaps to **0 dB (unity)**. The unity point is marked on the travel, and tick marks sit at the usual dB points.
- Uses a proper audio-taper law (unity sits high on the travel, as on a real console fader).

The multi-mode knob

Each strip has a knob whose function follows the current **mode** (PAN / GAIN / HPF / LPF):

Mode	Knob controls	Double-click	Press-and-hold
PAN	Pan L–C–R	Centre	(in sends-on-fader: toggle send PRE/POST)
GAIN	Preamp gain (−10... +60 dB)	0 dB	48 V phantom on/off
HPF	High-pass frequency	20 Hz	Cycle slope 6/12/24 dB/oct
LPF	Low-pass frequency	20 kHz	Cycle slope

Mute, solo, metering

- **MUTE** and **SOLO** per strip, mirrored to hardware LEDs.
- **VU metering** is decoded live from the desk — pre/post-fader level, plus **compressor gain reduction** and **gate** state — and animated on the strips and the master meter. Simulated meters show until real VU arrives.

- A per-strip **OLED/scribble** area shows the channel label, name, value, and pre/post + mute-group indicators.

The master strip

The master strip is **context-sensitive**: normally it's the **main L/R** fader and mute, but it becomes the master of whatever layer you're in — a sends-on-fader bus, a DCA/SUB spill, or a matrix.

Channels & Buses

Fader Control exposes the full dMix bus architecture through the **MASTERS** buttons and the **BUS** page:

- **SUB** — 12 subgroups
- **AUX** — 24 aux buses
- **DCA** — 16 DCAs/VCA's
- **MTX** — 8 matrix buses
- **FX** — 12 FX engines (sends and returns)

Sends-on-fader (SOF). Select an AUX (24) or FX (12) send and the channel faders become that bus's send levels; the master strip becomes the bus master. **Fader Flip** swaps faders and encoders in SOF.

DCA / SUB spill. Tap a DCA or subgroup on the BUS page to *spill* its member channels across the faders for direct access.

Matrix mixer. Tap a matrix to open its input mixer — a per-input fader, knob and mute for the 12 matrix inputs.

Names & colours come from the desk, using the dMix 16-colour palette, and appear on the strip colour bar and scribble.

Hardware Control Surfaces

Mackie Control (MCU)

Fader Control drives up to **four MCU surfaces** — an 8-fader unit each, for **32 motorised faders** total. Surface 1 covers channels 1–8 (its 9th fader is the master), Surface 2 covers 9–16, and so on.

What it handles, in both directions:

- **Motorised faders** — 14-bit pitch-bend; motors are driven to match the desk.
- **Buttons** (notes) — REC, SOLO, MUTE, SELECT, V-Pot push, assignment keys, BANK/CHANNEL, FLIP, function keys F1–F8, transport, cursor keys, and user mute/view-group keys.
- **V-Pots / encoders** — relative CC; adjust pan/gain/HPF/LPF per the active mode. **V-Pot push** cycles filter slope (tap) or toggles the filter (hold) in HP/LP mode, or toggles 48 V in GAIN mode.
- **Jog wheel** — banks through channels.
- **Fader touch** — motors won't fight your hand while you're touching a fader.
- **Feedback out** — MUTE/SOLO/REC/SELECT **LEDs**, the **V-Pot LED ring** (pan/gain/frequency), and the **scribble-strip LCD** (channel names and dB values via SysEx), plus **meter** data pushed to the surface's meter bridge.
- **Echo suppression** — surfaces that echo their own LED changes (iCON, Platform M, Persons, ...) are detected and a brief push-lock keeps those echoes from being misread as presses.

Surfaces **auto-reconnect** on launch, and a “**2nd controller display ID**” option switches the LCD SysEx product id (Mackie Control vs Extender/XT) to fix blank or stale text on a second unit.

HUI surfaces

Fader Control also supports **HUI** surfaces (e.g. Waves eMotion LV1 control) via a translation layer over the MCU engine — enable it in the **HUI** dialog, pick In/Out ports and the first-fader channel, and it handshakes, decodes faders, V-pots, jog and zone buttons, and pushes faders, scribble, LEDs and meters back.

MIDI Mapping & Learn

Setting up a surface — MCU Setup dialog

Click **MIDI** to open **MIDI Controller — MCU Setup**: pick the IN/OUT device for each of Surfaces 1–4, rescan for hotplugged gear, connect/disconnect, choose the **fader strip count**, set the 2nd-controller display id, and optionally **log all incoming MIDI** for mapping. An MCU protocol quick-reference is built in.

Mapping buttons to actions — MIDI Button Mapping

Click **MAP** to open the mapping table. Each row is a **trigger** → **action**.

- **LEARN** — click it, then press a controller button; Fader Control captures that control's **full signature** and adds a row.
- **Full-signature matching** — a mapping is keyed by **type** (Note or CC) · **MIDI channel** (1–16) · **number** (note/CC #) · **value** (velocity/CC value). This means the *same note on different channels*, a *CC vs a Note*, and *buttons that use a specific velocity* all map **distinctly** — no collisions.
- **Presets** — keep several named maps (NEW / RENAME / DEL), **EXPORT** and **IMPORT** them as JSON, and **RESET** to the X-Touch/MCU default.

What you can map (action registry)

- **Knob modes** — PAN / GAIN / HPF / LPF
 - **Banking** — Bank Left/Right (by fader count), Channel Left/Right (by 1)
 - **Clears & utility** — Solo Clear, Mute Clear, Tap Tempo
 - **Master layers** — SUB / AUX / DCA / MTX
 - **Mute Groups** — MG 1–6 (protocol)
 - **View Groups** — Custom VG 1–6, plus View Group Cycle
 - **Protocol channel select** — Prev / Next channel, Mute FX
 - **Protocol page nav** — MIXER / GAIN / INPUT / AUX SEND
 - **Protocol bank/type GOTO** — Ch 1-16 ... 113-128, AUX, SUB, DCA, MTX, FX
 - **Fader Flip, Snapshot Prev/Next, Touch & Turn Param** ◀ ▶
 - **Keyboard commands** — every GUI shortcut (see *Keyboard Commands*) is mappable
-

View Groups & Mute Groups

- **Native View Groups (VG 1–6, plus V7–14)** filter the surface to the channels that belong to that view group on the desk.
 - **Custom View Groups (6, user-programmable)** let you build your own sets from **any** strips — inputs, aux, FX, subs, DCAs or matrices. Double-tap a VG to open the editor: add/remove strips, **drag to reorder**, rename, clear, and **export/import** as JSON.
 - **Mute Groups (MG 1–6)** toggle the desk's mute-group mask; the buttons and the MCU LEDs light to match, and channels muted by a group are dimmed on the surface.
-

Keyboard Commands & GUI Control

Fader Control can fire the dMix **web GUI's keyboard shortcuts** to drive the on-screen mixer — handy for jumping to edit pages the control protocol doesn't cover.

Shortcut set (`KEY_CMDS`) includes the number-row page modes (Home, Gain, AUX, FX, MTX, Routing, Settings...), channel-edit pages (Source, Input/HPF-LPF, Gate, Comp, EQ, GEQ, Multiband, De-esser, Insert, Aux sends, FX sends), numpad mute groups, and navigation (channel mode, FX returns/sends, SUB/AUX/DCA master, previous page, prev/next channel).

A keystroke is delivered three ways, whichever applies:

1. **Native (app)** — a **real OS keystroke** posted by the Rust backend; it reaches the focused dMix GUI even cross-origin, with no bridge. (*macOS: grant Accessibility on first use.*)
2. **Browser** → **bridge** — sent to `dmix_bridge.py`, which types it at the OS level into a separate GUI window.
3. **Synthetic events** — dispatched into a *same-origin* embedded or popped-out GUI.

Embedded GUI. The  **GUI** button opens the dMix web GUI in a pane (or a pop-out), with its own address field and LOAD. (A cross-origin embedded GUI can't receive synthetic keystrokes — use the native app or the bridge for that.)

Follow-Me. With **FOLLOW ME** on, the surface drives the dMix GUI's channel and page focus: selecting a channel sends `SELECT_CHANNEL` / `SCROLL_TO_CHANNEL` and a sync broadcast; changing mode/page switches the GUI page or fires its shortcut; changing master layer scrolls the GUI and keeps its visible bank in step. Fader Control also **follows** remote selection changes when sharing a **Sync ID** group.

Snapshots, Shows & Scenes

- **Snapshot –/+** (HOME) step through the desk's snapshot list (`SNAP_LOAD`).
- **NEW SNP / SAV SNP** (FXFN) create or save a snapshot.
- **LOAD SH / SAVE SH** (FXFN) load or save a whole show.

The current show/snapshot is shown in the header, and the snapshot list is kept in sync from the desk.

Touchscreen, Tabs & User Pages

The centre touch panel has tabs: **HOME**, **AUX**, **FXFN**, **BUS**, **VG**, and **USR 1–4**.

- **HOME** — the channel bank grid, View Groups, Mute Groups, Masters, Mode selector, and a utility row (Solo Clear, Mute Clear, Tap, Snap –/+).
- **AUX** — 24 aux sends-on-fader buttons + bank nav.
- **FXFN** — 12 FX sends, a FUNCTIONS row (Flip, Bank ◀/▶, Mute FX, Load/Save Show, New/Save Snapshot), extended View Groups 7–14, and **F-keys F1–F8**.
- **BUS** — DCAs (tap to spill), subgroups (tap to spill), matrices (tap for inputs).
- **VG** — custom view-group list + editor.
- **USR 1–4** — your own pages. In **ASSIGN mode** you can pick **any** button from any page and drop it into a slot (tap to place, long-press to pick for a swap). Build the exact layout your show needs.

Assignable F-keys run a stored action on short-press and are **re-assigned** by long-pressing (1.5 s). **Tap Tempo** averages your taps into the desk's BPM.

Touch & Turn Encoder

A single **Touch & Turn** encoder follows the GUI-selected parameter. The on-screen ring shows the parameter name, in its colour, and its value, and can be dragged (speed-sensitive). **Param stepping** moves through logical groups — EQ band gain↔freq↔Q, HPF/LPF freq↔slope, compressor and gate parameters — mappable to **Param** ◀ / ▶.

A **physical** Touch & Turn encoder can be learned (**LEARN ENC**), with optional **ring feedback** (LED ring shows the value) and **colour feedback** (an RGB pad lit to the parameter's colour).

Wireless Monitoring

Fader Control can monitor your **wireless microphone and IEM** receivers and overlay their status right on the mixing strips — via the companion bridge.

- Open **WIRELESS**, enable it, and add receivers by model + IP. A **Simulate** mode previews the layout with no hardware.
 - **Supported families** include **Shure** (ULXD, QLXD, SLXD, Axient AD, AXT, PSM1000 IEM), **Sennheiser** (EW-DX, EW-D, Digital 6000, SR IEM G4), **Lectrosonics DSQD**, **Audio-Technica 3000/5000**, and generic custom RX/IEM. (*Shure is fully implemented today; others degrade gracefully to “no link” until their adapter ships.*)
 - **Telemetry** shown per channel: **battery** (bars / %, runtime), **RF** level and quality, **audio** level, frequency, TX power, and **interference warnings**. Mics map to input channels; IEMs map to aux/matrix/group/direct outputs.
 - Each strip’s scribble cell gets a live badge: battery, RF bars, an IEM tag, a warning “!”, or “NO LINK”.
-

The Companion Bridge (`dmix_bridge.py`)

The bridge is a small, UI-less Python program that gives a **browser-based** Fader Control the few things a web page can't do itself. It listens on `ws://127.0.0.1:8765` and handles three jobs over that one connection:

1. **Keystrokes** — types dMix GUI shortcuts at the OS level (needs `pynput`).
2. **Wireless monitoring** — talks to the receivers and relays telemetry.
3. **V2 setup relay** — fetches the mixer's `/api/setup` for a CORS-blocked page.

Run it with:

```
pip install websockets pynput      # pynput only needed for keystrokes
python3 dmix_bridge.py             # macOS   (python on Windows)
```

On macOS, enable **Accessibility** for whatever launched it (Terminal / Python) so keystrokes can be injected.

The native app needs none of this — it does MIDI, keystrokes and the setup fetch itself. The bridge is for running Fader Control in a browser.

Display & Settings

- **DISPLAY** panel — toggle the scribble bridge, on-screen faders, and master cell; and tune button size, strip width, strip gap, the “gap every 8”, height, and label size to fit your screen or touchscreen.
 - **Scribble bridge** — a compact overview row mirroring every strip (name, ch, dB, colour, mute/solo, mini VU/comp/gate meters) for a second monitor; it also carries the wireless badges.
 - **Sync ID (0–6)** — the channel-select sync group shared between clients; Follow-Me uses it.
 - Almost everything is **remembered** between sessions (address, strip count, MIDI maps, custom VGs, F-keys, user pages, display options, wireless and Touch & Turn setup).
-

Debug & Logging

The **DEBUG** panel has three tabs:

- **Protocol/Log** — a colour-coded, timestamped log (info/send/recv/error/...), a **command box** to send arbitrary JSON or raw frames to the desk, and **RE-INIT / STOP VU** buttons.
- **Events** — decoded incoming events.
- **State** — a live dump of banks, channels, master and the key store, with REFRESH.

Turn on “**Log all incoming MIDI**” in the MIDI dialog to see raw MCU/HUI bytes while mapping.

Troubleshooting

“No allocation identity ... run dmix_bridge.py or serve from the mixer.” You’re on a browser page that can’t read the mixer’s `/api/setup` (cross-origin, no CORS). Fix it by: using the **native app** (no bridge needed); **running the bridge**; **serving the page from the mixer** (same origin); or having the mixer send an `Access-Control-Allow-Origin` header on `/api/setup`. Then **reconnect**.

The desk keeps replying `stale_allocation`. Same cause — the INIT went out with no identity. Supply one of the paths above and RE-INIT.

No MIDI messages received. Confirm the surface is selected under **MIDI** and the right IN device is chosen. In the **native app**, MIDI is handled by the built-in driver; in a browser, Web MIDI must be available (Chrome/Edge). Enable “Log all incoming MIDI” to see exactly what the controller sends.

Keyboard commands don’t reach the dMix GUI. In the native app, grant **Accessibility** (macOS: System Settings → Privacy & Security → Accessibility) and keep the GUI window/pane focused. In a browser, run the bridge, or host the app on the mixer so the embedded GUI is same-origin.

Buttons collide / don’t map. Re-learn them — the full-signature learn distinguishes note vs CC, MIDI channel, and velocity. If two physical buttons send identical bytes, reconfigure one in your controller’s own editor.

Appendix — Protocol Reference

Sent to the desk: INIT (with allocIdentity), SNAP_LOAD, SNAP_SAVE, SHOW_LOAD, SHOW_SAVE, SELECT_CHANNEL, SCROLL_TO_CHANNEL, SWITCH_PAGE, GOTO, BMSG{SYNC_ID}, raw {param:value} writes, and novurta (stop VU).

Received: INIT (full param dump), VU, RTA, SNAP_LIST, SELECT_CHANNEL, SYNC_ID, ALLOC_INVALIDATED / STALE_ALLOCATION / ALLOC_BUSY, and bulk parameter updates.

Fader Control — a Violet Audio product. © Violet Audio.