Midicake CONNECT

User Manual

Your Midicake, on a bigger screen.

Manual · 2026
Tracks the current release at connect.midicake.com

Contents

Home

Getting started

The app

The panels

Reference

Midicake CONNECT - user manual

Your Midicake, on a bigger screen.

Patch management, live screen mirroring, MIDI visualization and rig planning - in the browser, no install required.

Manual - 2026. CONNECT updates continuously; this manual tracks the current release at connect.midicake.com (https://connect.midicake.com).

How to use this manual

CONNECT is designed to be explored - open it, plug in your Midicake, and most of it explains itself. This manual is for the parts that reward a closer read: what the two kinds of connection actually do, how the Patch Store keeps your library safe, and what to check when something doesn't behave.

Two reading paths, same as the UNA manual: a **quick path** (Chapters 1-2, then jump wherever) and a **reference path** (any chapter, any time). Chapter 5 - Connecting your device - is the one worth reading in full, because every panel builds on it.

If you're looking for the manual for the instrument itself, that's the Midicake UNA manual.

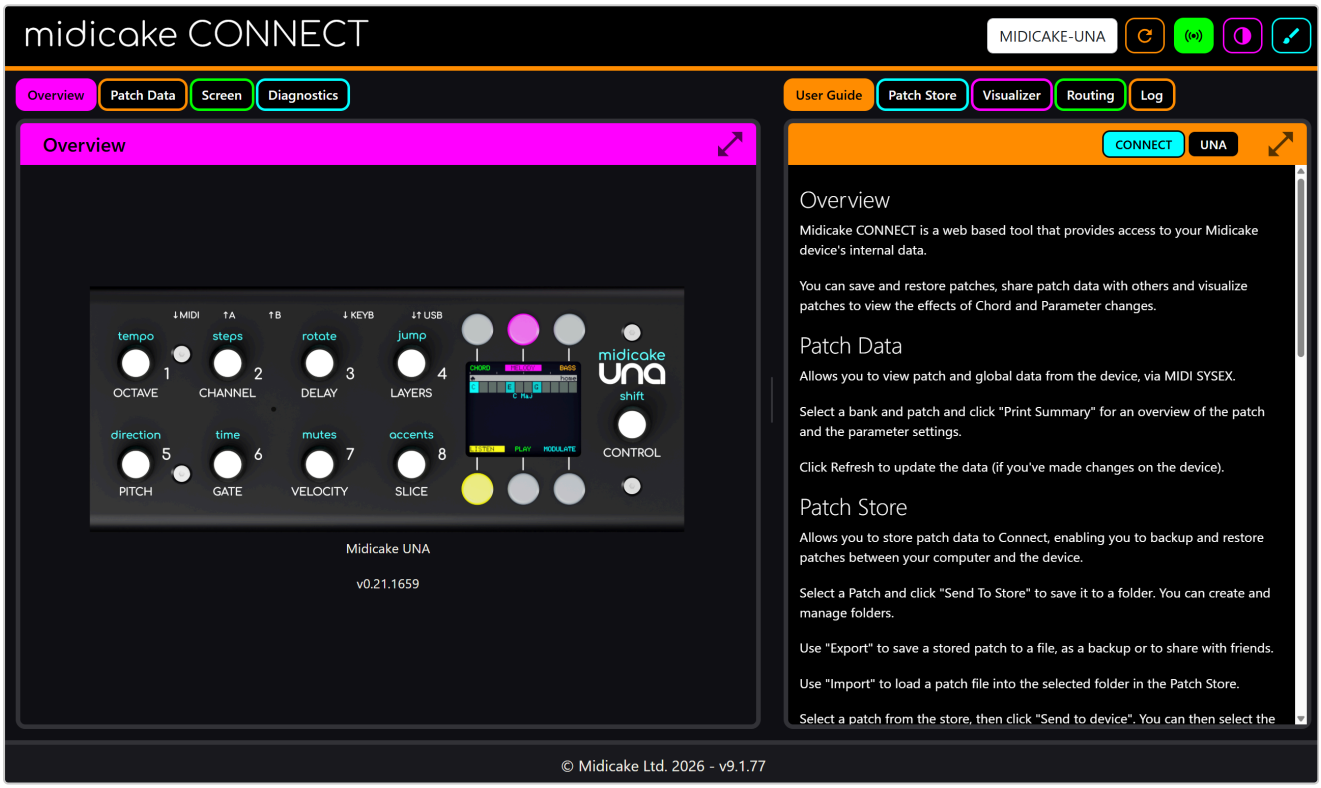
Getting started

Chapter 1 - Welcome & Overview

Welcome to CONNECT.

Midicake CONNECT is the companion app for Midicake hardware. It runs in your browser at connect.midicake.com (https://connect.midicake.com) - no download, no install, no account. Plug your Midicake into your computer, open the page, and you have a patch librarian, a live screen mirror, a MIDI visualizer, and a rig-planning canvas, all in one window.

CONNECT works with **Midicake UNA** and **Midicake ARP**, and treats them the same way: connect the device, press **Go Live**, and every panel that talks to hardware lights up.



What CONNECT does

- **Back up and organise your patches.** Read every patch and bank off the device, keep them in named folders in the Patch Store, rename them, lock them against accidents, and send them back - individually or as a whole bank. Export folders as files to share or archive.
- **Mirror the device screen.** See the LCD live on your monitor, pixel for pixel. Useful on stage, in the studio, when recording tutorials, or any time the hardware is out of arm's reach.
- **Visualize your MIDI.** Five real-time visualizations of the notes your device is playing, coloured per channel.
- **Plan your rig.** A node-graph canvas for mapping the MIDI devices in your setup - who sends on which channel, where conflicts and feedback loops hide.

- **Diagnose problems.** Fetch crash reports and live logs from the device when something misbehaves, ready to send to Midicake support.
- **Read contextual help.** The built-in guide follows the mode your device is in and shows the relevant page as you work.

What CONNECT deliberately isn't

CONNECT is a companion, not a replacement. Sequencing, sound-shaping and performance stay on the hardware, where the knobs are. CONNECT gives you the things a browser does better: big-screen visibility, safe storage, organisation, and a keyboard for renaming things.

There is also no cloud. Your patches, rigs and settings live in your browser's local storage on your machine - nothing is uploaded anywhere. That has one important consequence covered in Chapter 14: **your browser's data is the library**, so exports are your backups.

Where to start

- **Just plug in.** Chapter 2 gets you from cable to live screen in about two minutes.
- **Check your browser first.** CONNECT relies on Web MIDI and Web Serial, which not every browser supports - Chapter 3 has the compatibility table.
- **Understand the connection model.** Chapter 5 explains the two levels of connection - MIDI and Live - and which features each unlocks. Five minutes there saves confusion everywhere else.

Chapter 2 - Quick Start

This chapter gets you connected in about two minutes. The ideas behind each step are in Chapter 5; they'll make more sense once you've seen it work.

Connect

1. **Open connect.midicake.com** (https://connect.midicake.com) in **Chrome** or **Edge** on a desktop or laptop. (Safari and Firefox won't work - see Chapter 3 for why, and for Android.)
2. **Plug your Midicake into the computer** with a USB-C cable - the same cable that powers it.
3. **Check the device selector** in the top bar. CONNECT looks for Midicake devices automatically and picks the first one it finds. If it shows *NO DEVICE FOUND*, quick press the **↻ reconnect** button, and check the cable.

At this point the MIDI-level features are already working: play something on the device and the **Visualizer** and **Log** panels respond.

Go Live

1. **Press the Go Live button** - the broadcast icon in the top bar. Your browser asks which port to use: pick the Midicake entry and confirm.
2. **The button turns green.** You're Live. The **Screen** panel now mirrors the device LCD in real time, and the **Patch Data** panel can read from the device.

> Live unlocks everything that needs a direct line to the hardware: the screen mirror, patch transfers, backups, diagnostics, and the context-following device guide. The browser asks permission once per session - that's the browser's security model, not a CONNECT quirk.

Do something useful in the first five minutes

- **See your screen big:** open the **Screen** panel, then use the view controls to take it fullscreen.
- **Back up your device:** open **Patch Data**, choose **Device Actions → Backup Device**. Every patch, bank and global lands in a locked, date-stamped folder in your Patch Store. Two clicks, and everything on the hardware is safe.
- **Watch your music:** open the **Visualizer**, play, and cycle the modes with the glyph button.

If nothing happens

- Wrong browser is the most common cause - Chrome or Edge, desktop.
- The device selector only lists Midicake hardware. If it's empty, the cable or the port is the suspect - try a different USB port, and a cable you know carries data.
- If the Go Live prompt doesn't list your device, another app may be holding the port - close other MIDI software and try again.
- Full troubleshooting lives in Chapter 14.

Chapter 3 - Requirements & Compatibility

CONNECT is built on two browser capabilities: **Web MIDI** (for notes, clock and the device selector) and **Web Serial / WebUSB** (for the Live connection - screen mirror, patch transfers, diagnostics). Browser support for those two features decides where CONNECT runs.

The compatibility table

Platform	Browser	Works?	Notes
Windows / macOS / Linux	**Chrome**	Yes	Recommended
Windows / macOS / Linux	**Edge**	Yes	Same engine as Chrome
Windows / macOS / Linux	Firefox	No	No Web Serial
macOS / iOS	Safari	No	No Web MIDI, no Web Serial
iPhone / iPad	any browser	No	Apple doesn't allow the required APIs in any iOS browser
Android	**Chrome**	Yes	Live connection uses WebUSB instead of Web Serial - see below

Other Chromium-based browsers (Brave, Opera, Vivaldi) generally work but aren't tested; Brave ships with some device APIs off by default and needs them enabled.

What you need

- **A Chromium browser** - Chrome or Edge, reasonably current.
- **A USB cable that carries data.** The cable powers the device and carries everything CONNECT does. Charge-only cables are the classic silent failure: the device powers up, but no browser will ever see it.
- **Nothing else.** No driver, no install, no account, no plug-in. Midicake devices are class-compliant, so the operating system already knows how to talk to them.

Android

CONNECT works on Android Chrome with one difference under the hood: Android's Web Serial support is limited, so the Live connection uses **WebUSB** instead. In practice the flow is the same - press **Go Live**, pick the device from the browser prompt - but the prompt looks slightly different and you may need an OTG adapter depending on your phone's port.

A tablet running CONNECT fullscreen next to the hardware makes a nice second screen for the mirror and visualizer.

Permissions - why the browser asks

The first time you press **Go Live** in a session, the browser shows a device-picker prompt. That's the browser's security model working as designed: web pages can't open a hardware connection silently, ever. CONNECT can't skip the prompt, and neither can any other web app. Picking the device and confirming is the whole ritual.

The Web MIDI side may also ask once for MIDI access (including SysEx). Same reason, same one-off.

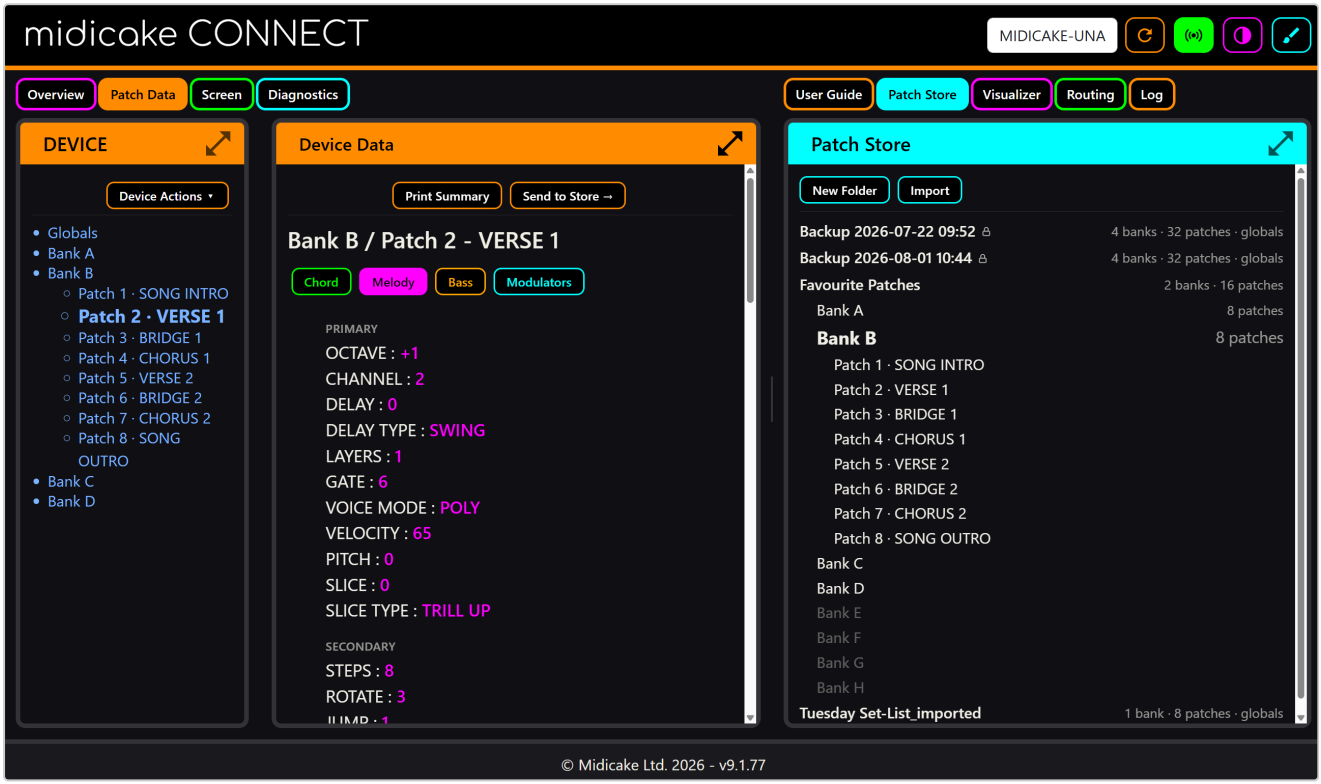
Offline and HTTPS

CONNECT is served over HTTPS, which the device APIs require. It needs the network to *load*, but once loaded, the traffic that matters - screen frames, patch data, MIDI - travels over your USB cable, not the internet. Nothing about your patches or playing leaves the machine (Chapter 14 covers where data lives).

The app

Chapter 4 - The CONNECT Layout

One window, two panes, nine panels. This chapter names the parts so the rest of the manual can refer to them.



The top bar

Left to right:

- **Device selector** - a dropdown listing connected Midicake hardware. CONNECT filters out everything else on your system; if it says *NO DEVICE FOUND*, no Midicake is visible to the browser.
- **↻ Reconnect** - re-scans for devices. Use it after plugging in or switching cables.
- **Go Live** - the broadcast icon. Opens the direct hardware connection (Chapter 5). Green while Live.
- **Theme toggle** - switches between light and dark. CONNECT follows your system preference on first visit and remembers your choice after that.
- **Channel colours** - the brush icon opens a palette with one colour per MIDI channel (1-16). These colours are shared everywhere channels appear - Visualizer notes, Routing wires - so a channel looks the same across the app. Edit any colour with a live preview, or **Reset** to the factory palette.

Two panes, nine panels

The window splits into a **left pane** and a **right pane**, each with its own row of tab buttons. Any left panel can sit beside any right panel - the pairing is yours.

Left pane	Right pane
Overview	User Guide
Patch Data	Patch Store
Screen	Visualizer
Diagnostics	Routing
	Log

A deliberate pattern in the split: device things live on the left, library and monitoring things on the right. **Patch Data** (left, the device) next to **Patch Store** (right, your library) is the pairing you'll use most - transfers flow between the two panes.

The splitter

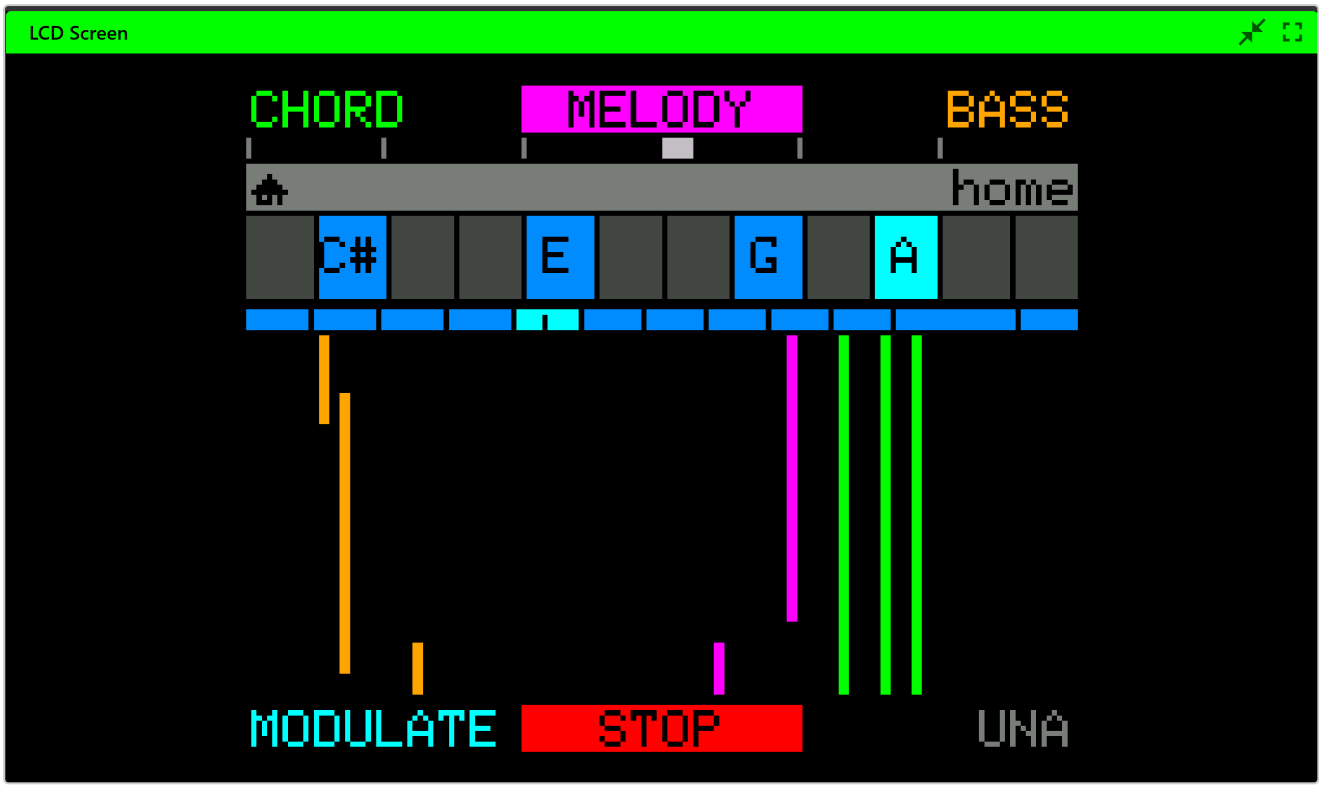
The bar between the panes drags to resize. Double-click resets the split to its default; with the splitter focused, the arrow keys nudge it. The position is remembered between visits.

View controls

Each panel carries a small set of view controls in its header:

- **Expand / Contract** - grow the panel within its pane.
- **Fill screen** - the panel takes the whole browser window.
- **Fullscreen** - the panel takes the whole monitor. Press Esc to leave.

Fullscreen is where the Screen mirror and Visualizer earn their keep - a laptop behind the device becomes a stage display.



Cinema mode

In fullscreen on the Screen or Visualizer panels, a **Cinema** button cycles three layouts: **Screen** only, **Visualizer** only, or a **split** of both. One monitor, LCD mirror on one side, notes raining down the other. The chosen layout is remembered.

What CONNECT remembers

CONNECT saves your workspace as you go - no save button, no setup to repeat:

- which panel is open in each pane, and the splitter position
- theme and channel colours
- visualizer mode and marker choice
- Cinema layout
- your Patch Store folders and Routing rigs (these are your *data*, not just preferences - Chapter 14 explains where they live and how to keep them safe)

Everything is stored locally in the browser. Clear the browser's site data and all of it - preferences *and* library - goes with it. That warning gets a whole section of its own in Chapter 14.

Chapter 5 - Connecting Your Device

CONNECT talks to your Midicake over one USB cable, but through **two levels of connection**. Understanding the split is the key to the whole app - every panel belongs to one level or the other.

Level 1 - MIDI

The moment your device is plugged in and selected in the top bar, the **MIDI connection** is active. No button to press. This is ordinary Web MIDI - the same stream any music app would see.

What it carries: notes, clock, transport - the musical traffic.

What it unlocks:

- **Visualizer** - draws what the device plays
- **Log** - the scrolling message console
- Tempo detection from the device's clock

The MIDI connection is passive and always-on. If the selector shows your device, this level is working.

Level 2 - Live

Live is the direct line to the hardware - a serial connection over the same cable, opened when you press the **Go Live** broadcast icon and pick the device in the browser prompt. The icon turns green while Live.

What it carries: the screen stream, patch data, diagnostics - the device's internals.

What it unlocks:

- **Screen** - the pixel-for-pixel LCD mirror
- **Patch Data** - reading and writing the device's patches, banks and globals
- **Backups and restores**
- **Diagnostics** - crash dumps and live logs
- **The DEVICE guide** - contextual help that follows the device's current mode
- The live overlay on your device's card in **Routing**

> **Why two levels?** MIDI is a broadcast - anything on your system can listen. The Live line is exclusive and richer: it streams the screen and moves patch data, things MIDI was never built for. Browsers guard that kind of direct hardware access behind an explicit permission prompt, which is why Live is a button and MIDI just happens.

Identify

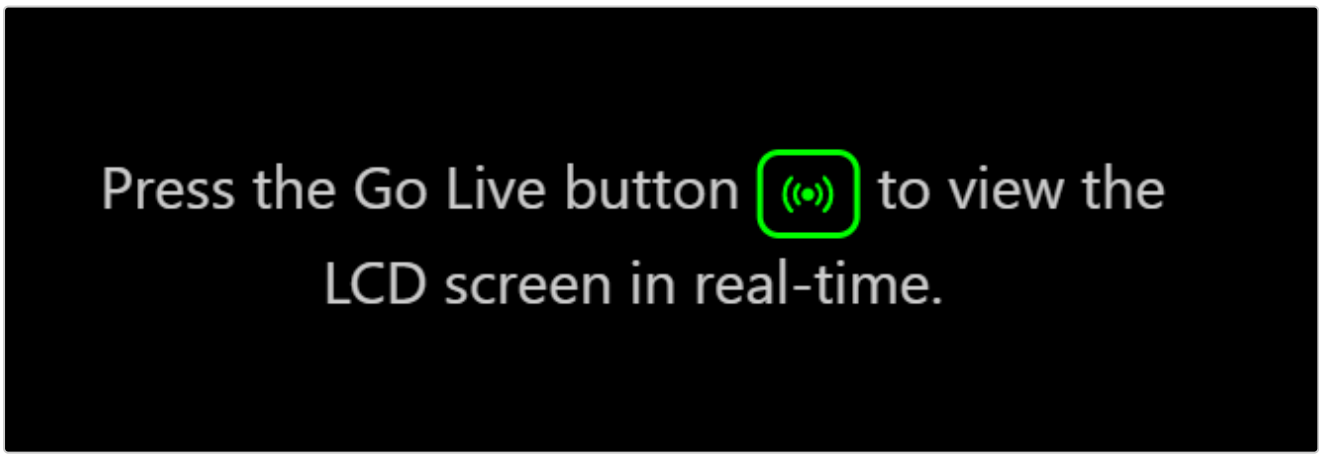
When you go Live, CONNECT introduces itself to the device and asks who it's talking to. The reply carries the device type, firmware version, and its patch geometry - how many banks, how many patches per bank. The **Overview** panel shows the result, and the **Patch Data** tree builds itself to match.

You can re-run this any time from **Patch Data** → **Device Actions** → **Identify** - useful after a firmware update, since the firmware version on Overview comes from this handshake.

Identify also carries the device's **storage layout version**, which CONNECT uses to keep old backups from being restored onto incompatible firmware - the safety guard described in Chapter 8.

Going Live: what to expect

1. Press the broadcast icon.
2. The browser lists eligible devices - pick the Midicake, confirm.
3. The icon turns green; the Screen panel starts mirroring within a moment.



The permission prompt appears once per session. On desktop the connection uses Web Serial; on Android, WebUSB - same button, same result (Chapter 3).

Ending Live is the same button again. Panels that need Live return to their "Press the Go Live button" prompts - several of those prompts include a broadcast icon of their own, which is a shortcut to the same connection.

When the device disappears

Unplugging mid-session, a loose cable, or the device restarting all end the Live connection. CONNECT notices, drops back to the not-Live state, and the MIDI selector may briefly show *NO DEVICE FOUND* until the device re-enumerates. Reconnect is the ↻ button, then Go Live again.

Nothing is harmed by an abrupt disconnect while *reading*. Interrupting a *write* - sending patches to the device or restoring a backup - is the one case worth avoiding; Chapter 8 covers how transfers behave.

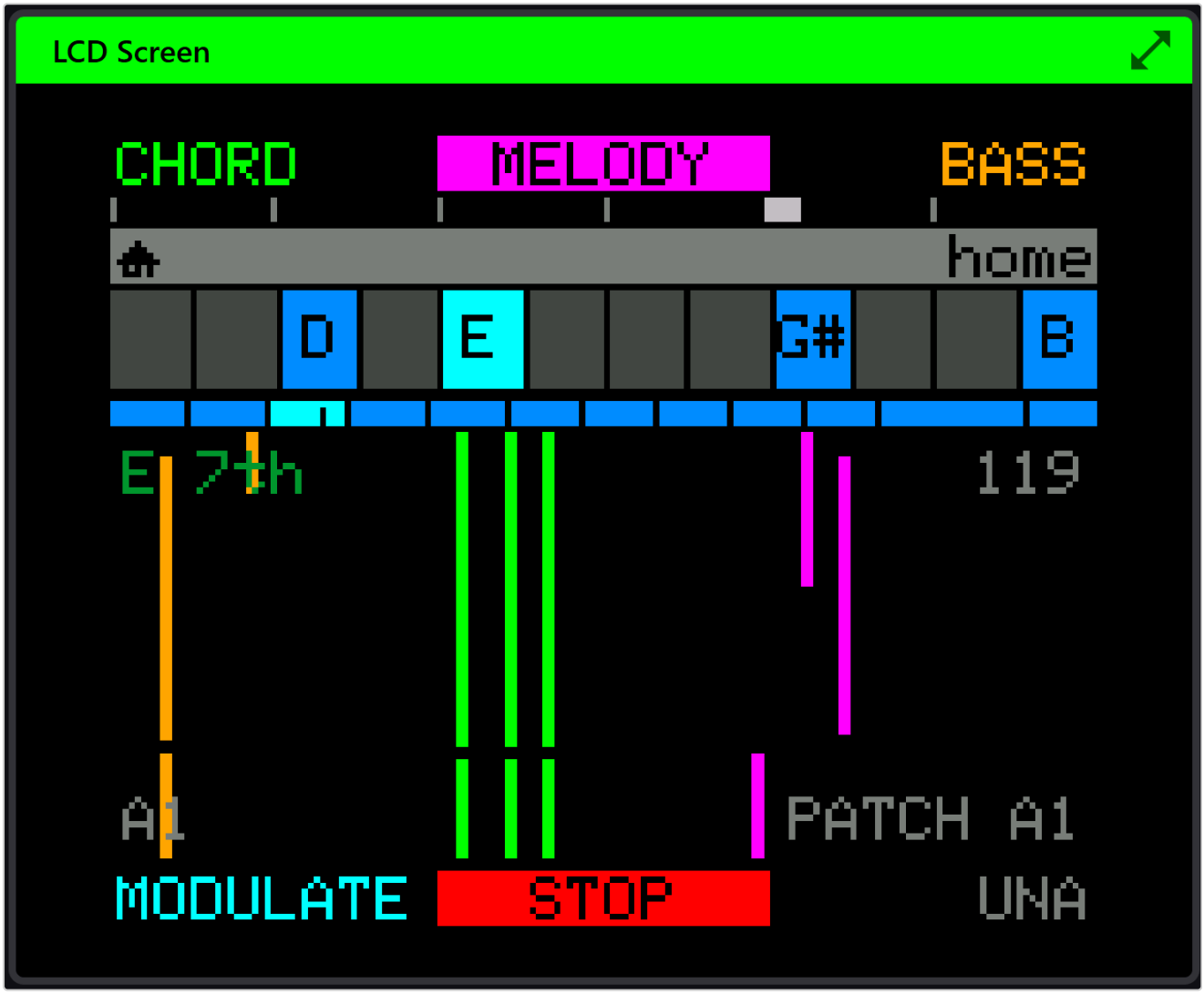
Several devices at once

The device selector lists every Midicake it finds; CONNECT works with the one selected. Switching devices mid-session is fine - change the selector, reconnect, go Live on the new device. The Patch Store doesn't care which device is connected; folders from any device sit side by side, each tagged with where it came from.

The panels

Chapter 6 - Screen Mirror

The **Screen** panel mirrors your device's LCD in the browser, live and pixel for pixel. What the hardware draws, you see - mode screens, parameter edits, the playhead moving through a sequence - at whatever size your monitor allows.



Starting the mirror

The mirror needs the **Live** connection (Chapter 5). Until you're Live, the panel shows a black screen and a prompt - the broadcast icon in the prompt is a shortcut to the same Go Live button as the top bar.

Once Live, the mirror starts by itself. There's nothing to configure: the device streams its display and CONNECT paints it.

What you're seeing

The device's screen is a 160 × 128 pixel panel. CONNECT renders the actual framebuffer - not a reconstruction - so everything appears exactly as on the hardware: the same colours, the same fonts, the same pixel grid. Scaling is done crisply, so at large sizes you see clean sharp pixels rather than a blurred stretch.

The stream is continuous while Live, with one deliberate exception: during patch transfers, backups and diagnostics fetches, CONNECT briefly pauses the mirror so the data transfer gets the full bandwidth of the connection. It resumes by itself when the transfer finishes. A momentary freeze during a Read All is normal, not a fault.

Making it big

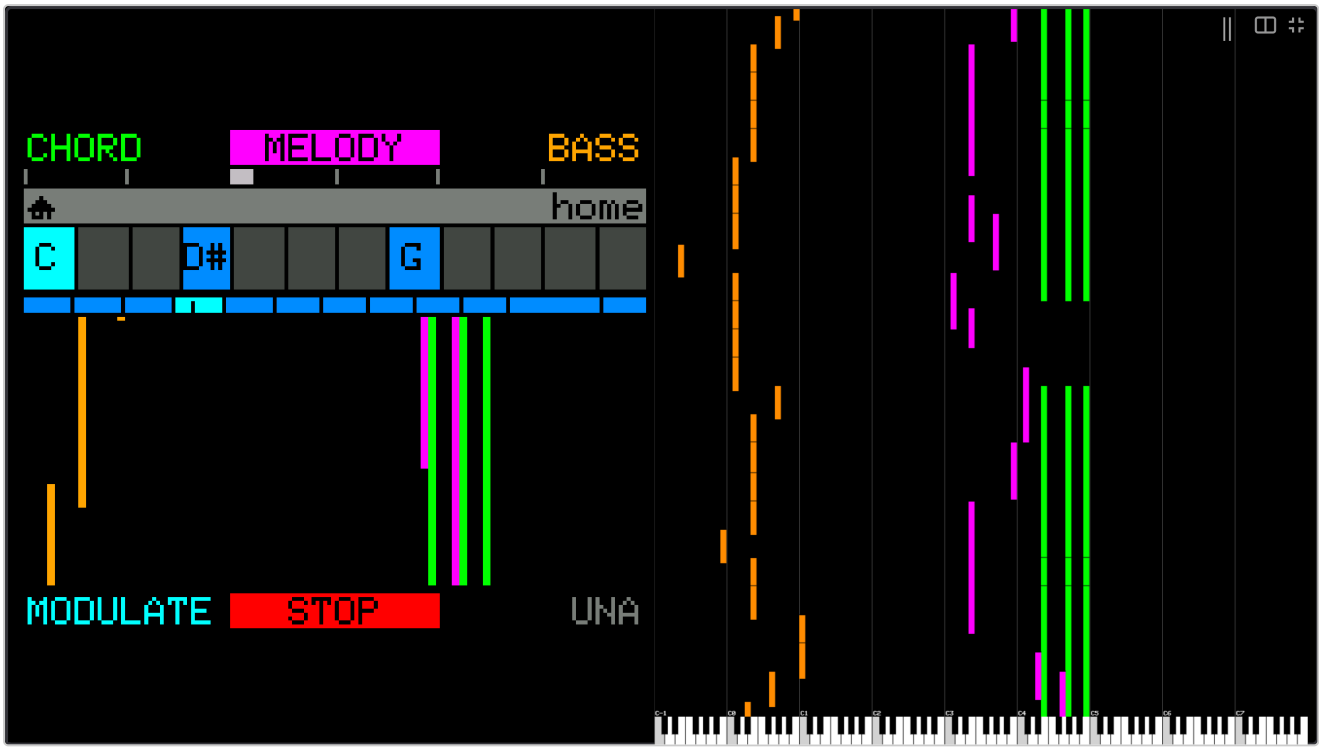
The panel's view controls step up through **Expand**, **Fill screen** and **Fullscreen** (Chapter 4). Fullscreen is the natural home for the mirror:

- **On stage** - a laptop or tablet where the audience or bandmates can see what you're doing.
- **Teaching and streaming** - screen-capture the browser and your viewers see the device UI at broadcast size.
- **In the studio** - the device can sit in the rack while its screen floats on your monitor.

Cinema mode

In fullscreen, a **Cinema** button appears and cycles three layouts:

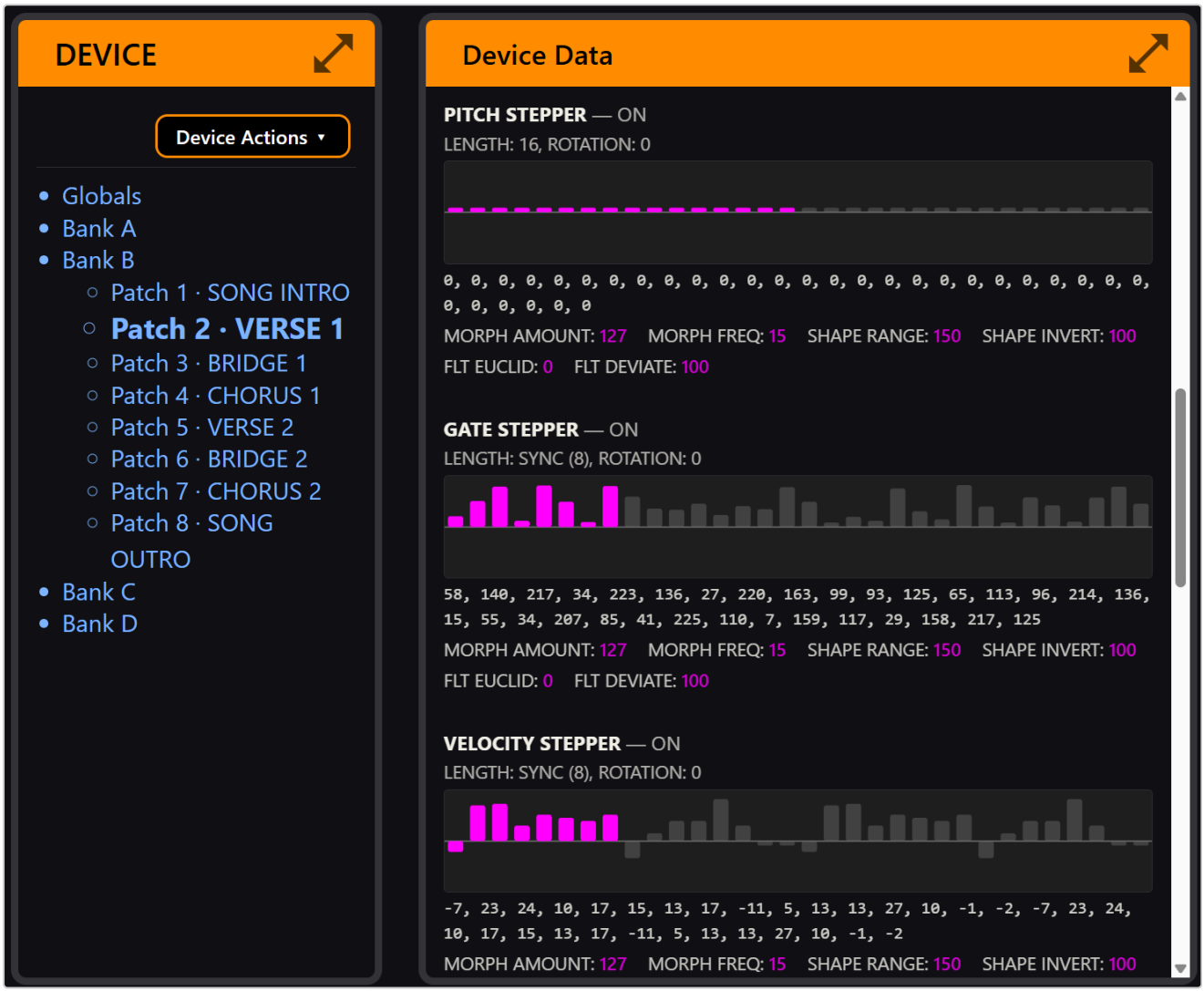
1. **Screen** - the mirror alone.
2. **Visualizer** - the note visualization alone (Chapter 9).
3. **Split** - both side by side: the device UI on one half, the music visualized on the other.



The split is the show layout - one glance covers what the device is doing and what it's playing. Your chosen layout is remembered for next time.

Chapter 7 - Patch Data

The **Patch Data** panel is your window into what's on the device right now: every patch, every bank, and the global settings, readable and decoded. It's the left half of the patch workflow - the right half is your local library, the **Patch Store** (Chapter 8), and moving things between the two is how patches travel.



The device tree

The left column shows the device's contents as a tree:

- **Globals** - the device-wide settings unit.
- **Banks** - lettered A, B, C... built from what the device reports during Identify, so the tree always matches the connected hardware's real geometry.
- **Patches** - the slots inside each bank. Once a patch has been read, its name appears next to the slot.

Until you're Live, the panel shows a prompt instead - like the screen mirror, it needs the direct connection (Chapter 5).

Selecting anything in the tree - a patch, a bank, Globals - shows its contents in the viewer on the right.

Device Actions

The **Device Actions** dropdown at the top of the tree:

- **Identify** - re-runs the handshake: device type, firmware version, bank/patch geometry. Run it after a firmware update.
- **Read** - reads the currently selected unit (one patch, one bank header, or Globals) from the device.
- **Read All** - reads the entire device, unit by unit, with a progress bar. On UNA that's every patch in every bank plus the bank headers and Globals in one pass. Takes a little while; the screen mirror pauses during the transfer and resumes after.
- **Backup Device** - reads everything *and* files it in the Patch Store as a locked, provenance-stamped backup folder. This is the one-action safety net - the full story is in Chapter 8.

> **Read before you browse.** The tree starts as slots and placeholders; the content arrives when you read it. **Read All** first, then browse freely - everything you click is already local.

The patch viewer

Select a read patch and the viewer decodes it into the same structure the instrument uses, spread across tabs:

- **Chord / Melody / Bass** - one tab per track, showing the parameter groups as the device organises them: primary and secondary parameters, locks, and the bass Improv settings on the Bass tab.
- **Modulators** - every modulator slot with its source, target and settings, including modulator step-sequence lanes.

Where a parameter is really a pattern, the viewer draws it rather than listing numbers:

- **Stepper grids** - the 32-step lanes rendered as grids, so a pitch sequence or gate pattern reads at a glance.
- **Chord Rhythm** - the rhythm drawn as a timeline.
- **Chord Chain** - the chain as a grid of links.

The viewer is **read-only by design**: parameters are edited on the instrument, where the knobs are. What CONNECT adds is legibility - seeing a whole patch at once, in a form you can compare and archive - plus the handful of things a keyboard genuinely does better: renaming, organising, and transfers.

The Globals view

Selecting **Globals** shows every device-wide setting, grouped as the device groups them. Two extras earn their place:

- **Filter box** - type to narrow the list to matching settings. The fastest way to answer "what's my clock set to?" without touching the device menu.
- **Print Summary** - exports the current view as a text file. Patch summaries have the same button - useful for documentation, or a plain-text record of a setup before you change it.

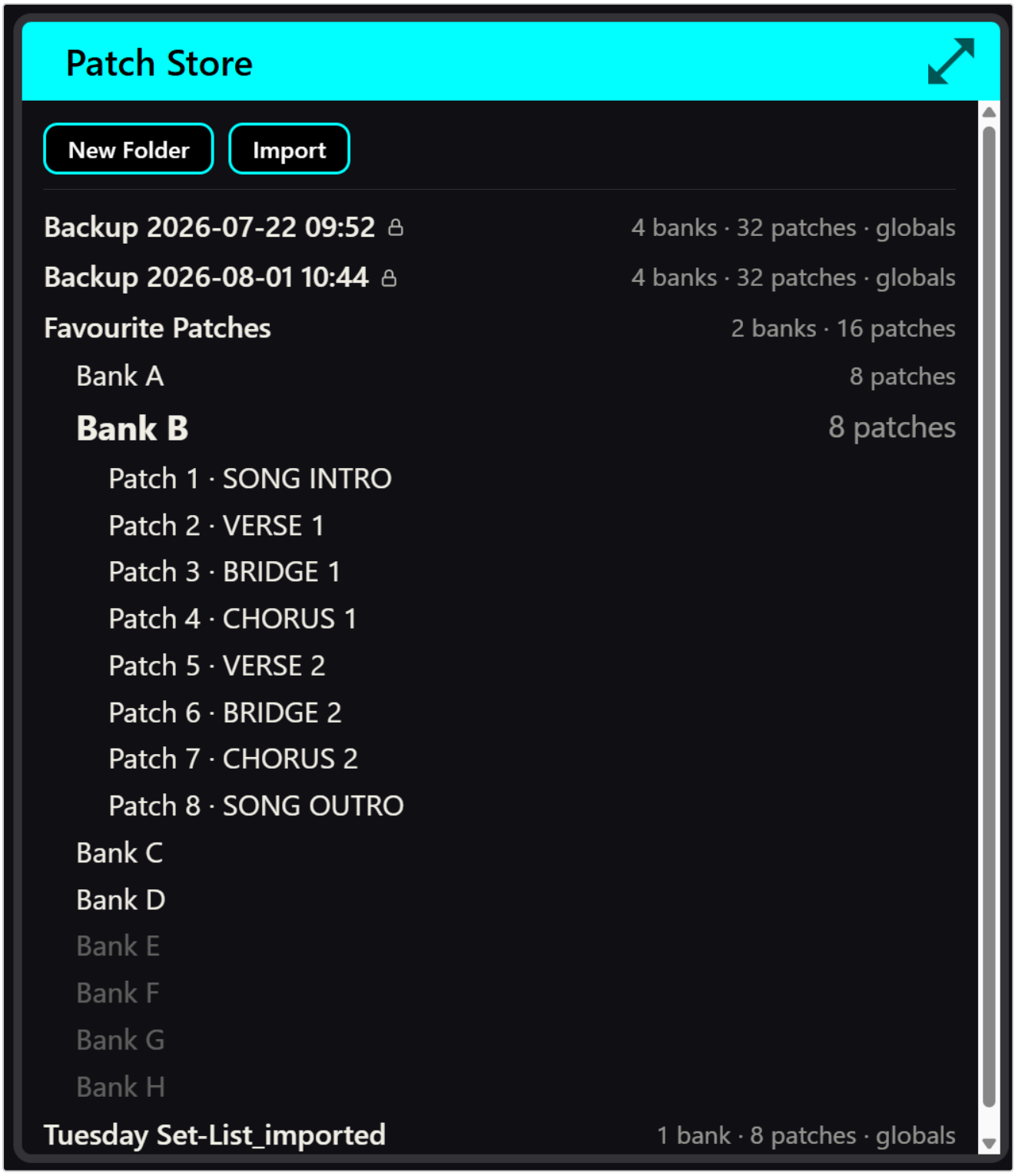
Transfers, briefly

With a patch selected you can **Send to Store** - copy it from the device into a folder in your library. The reverse trip, **Send to Device**, starts from the Patch Store side and is covered with its confirmations and safety guards in Chapter 8. Transfers pause the screen mirror momentarily; interrupting a *read* is harmless.

Chapter 8 - Patch Store

The **Patch Store** is your patch library - folders of patches that live in your browser, independent of what's on the device. It's where backups land, where you stage patches before sending them to hardware, and how patches travel between people as files.

Where **Patch Data** (Chapter 7) shows what's *on the device*, the Patch Store shows what's *yours*. The two panels side by side - Patch Data left, Patch Store right - is the natural working layout.



The shape of the library

Folders → banks → patch slots, mirroring the device's own layout:

- A **folder** is a full device-worth of patches - banks and all. Create as many as you like: one per song, per set, per experiment.

- Inside each folder, **banks** lettered like the hardware's. Banks beyond what current firmware supports appear greyed out - reserved for future firmware, not clickable today.
- Inside each bank, the **patch slots**, named.

The folder-equals-device shape is what makes transfers unambiguous: sending a folder to the device maps bank A to bank A, slot to slot, positionally.

Everyday actions

- **New Folder** - an empty device-worth of slots.
- **Rename** - double-click a folder or patch slot to rename inline. Patch names are real device data: the name is written into the patch itself, in the device's own character set, and travels to the hardware when you send it. Renaming in the Store *is* renaming the patch.
- **Lock / Unlock** - a locked item can't be overwritten or deleted until you unlock it. Backups arrive locked; lock anything else you'd hate to lose.
- **Delete / Clear** - remove a folder, or clear a slot. Locked items refuse politely.
- **Export / Import** - a folder exports as a single JSON file; **Import** brings one in, always as a *new* folder - imports never overwrite existing folders. This is the sharing format: send the file to a friend, they import it, done.

Backups

Patch Data → **Device Actions** → **Backup Device** reads the entire device and files it here as a folder that is:

- **Locked** on arrival - it can't be casually overwritten or deleted.
- **Stamped with provenance** - which device it came from, the firmware version, the storage layout version, and when it was captured.

A backup isn't a special file format hidden somewhere - it's an ordinary folder in your library, restorable in whole or in part, exportable like any other. The lock and the stamp are what make it trustworthy.

> **Back up before you experiment, before firmware updates, and before restoring anything.** Two clicks. The habit pays for itself the first time it matters.

Sending to the device

Transfers to hardware always confirm before they touch anything, inline in the panel - and the confirmation says exactly what will be overwritten:

- **Send a patch** - one patch to one slot. You'll see **WARNING: THIS WILL OVERWRITE THE EXISTING DATA!** and confirm or cancel.
- **Send a folder** - the whole folder, positionally: bank A to bank A, slot for slot. The confirmation summarises the scope, with an **Include Globals** checkbox - off by default, because overwriting device settings is a bigger decision than overwriting patches.
- **Copy to Store** - duplicate within the library, no device involved.

During a write, let it finish - interrupting a *read* is harmless, but pulling the cable mid-*write* can leave a slot half-written. Transfers are quick; there's rarely a reason to.

The safety guard

Every patch records which **storage layout version** it was made under. Firmware updates can change that layout - and a patch written under one layout, restored under another, would corrupt silently.

CONNECT refuses the bad combination outright: if a patch's layout version doesn't match the connected device's, **Send to Device** and **Restore** decline with a clear message rather than writing garbage. Nothing on the device is touched.

If you hit the mismatch message: it means this stored data predates (or postdates) your device's current firmware layout. Check midicake.com/support (https://midicake.com/support) for whether a migration path exists for that combination.

Where the library actually lives

In your browser's local storage, on this machine. No cloud, no account - and no copy anywhere else either:

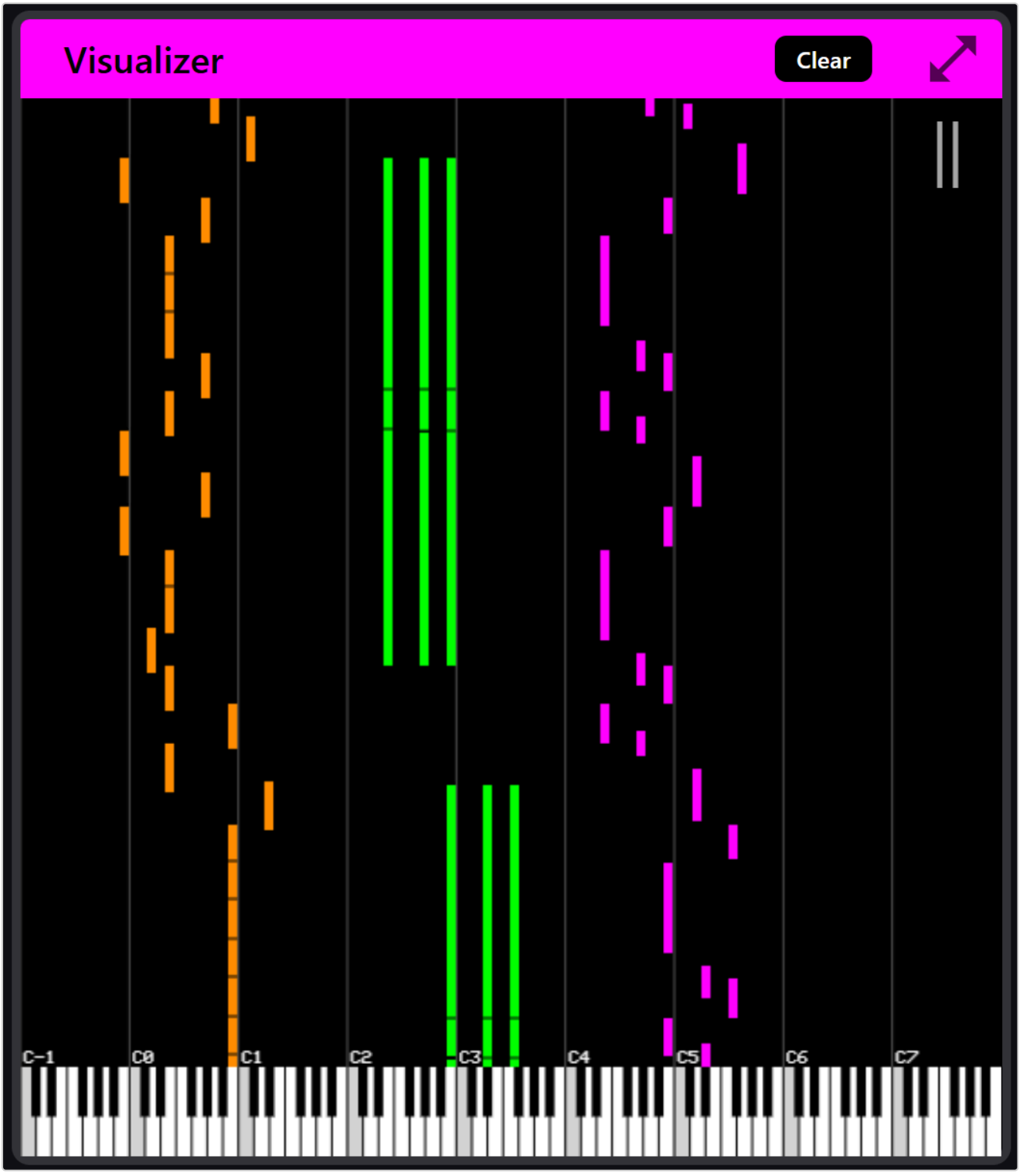
- A **different browser** (or a different computer, or a private window) has a **different library**.
- **Clearing browser site data deletes the library.** Not to a recycle bin - gone.

Export is your real backup. Folders you'd mind losing should exist as exported JSON files somewhere safe - a backup drive, cloud storage, an email to yourself. Chapter 14 covers the full picture of what lives where.

Chapter 9 - Visualizer

The **Visualizer** panel turns incoming MIDI into moving pictures, live. Every note your device plays becomes a shape, coloured by its MIDI channel - so the Chord, Melody and Bass parts are visually distinct voices, and the structure of what you're hearing becomes something you can watch.

It runs on the MIDI level of the connection (Chapter 5) - no Go Live needed. If notes are flowing, it's drawing.

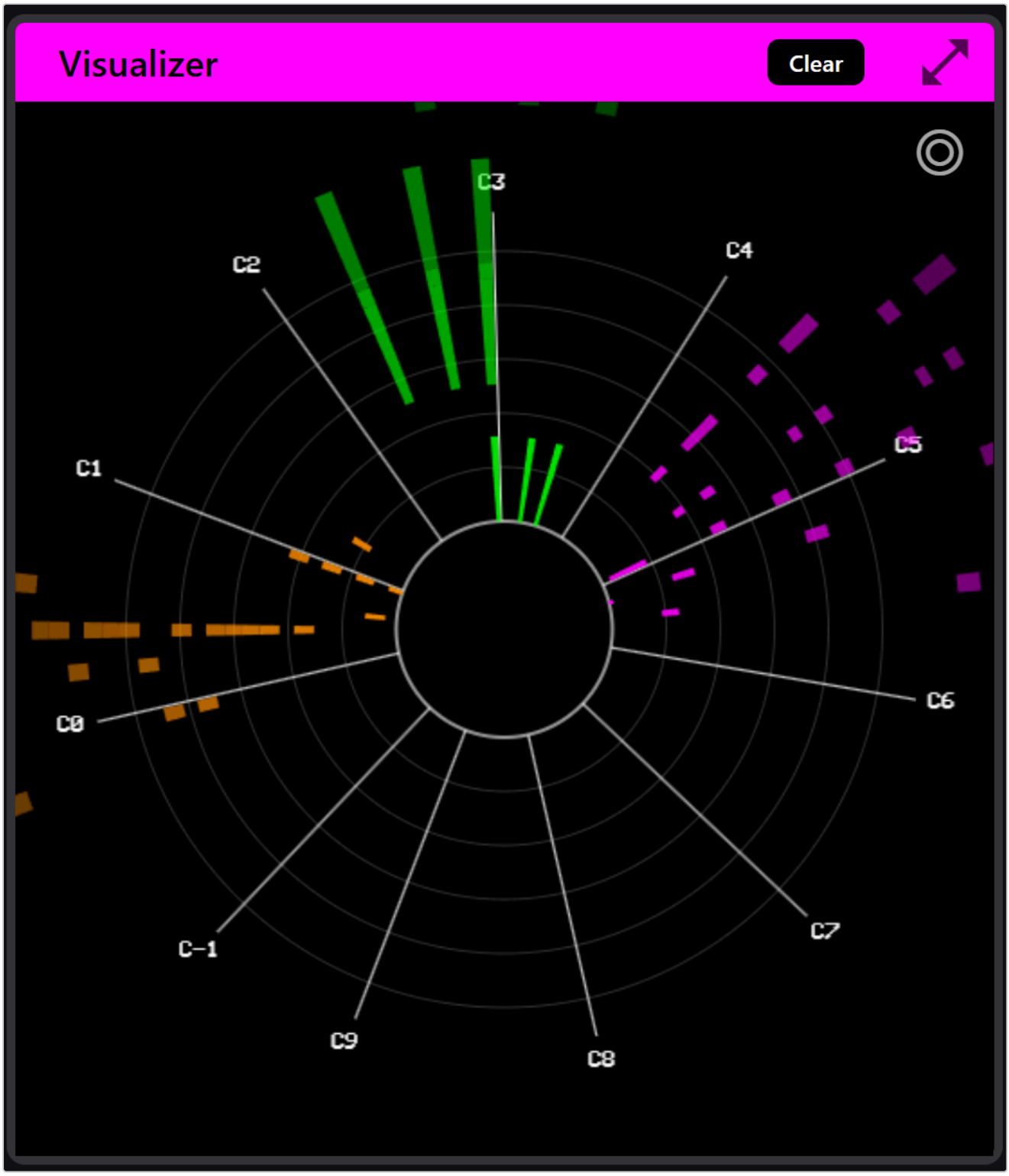


The five modes

The glyph button in the panel header cycles through the modes:

Mode	What it looks like
Up-Scroller	Notes rise from the bottom as coloured columns - pitch across, time upward. The default.
Side-Scroller	The same idea rotated: notes flow horizontally, like a scrolling piano roll.
Starfield	Notes fly outward from the centre as points of colour - more fireworks than notation.
Circle	Notes arranged around a ring - pitch as position, activity as light.
Heatmap	A grid that glows where notes land and fades as they age - the density view.

The cycle actually runs through the list twice: once with **markers** (bar lines and reference marks) and once without. Keep tapping until the combination feels right - your choice of mode and markers is remembered.



The Up-Scroller and Side-Scroller are the analytical modes - you can *read* the music from them. Starfield and Circle are the performance modes - they look wonderful at fullscreen behind a player. Heatmap sits between: it shows where the music lives across the pitch range.

Channels and colour

Colours come from the shared per-channel palette - the brush icon in the top bar (Chapter 4). One colour per MIDI channel, sixteen channels, the same palette the Routing wires use. Set your device's tracks to distinct channels and each track becomes a distinct colour thread in every mode.

Tempo and bars

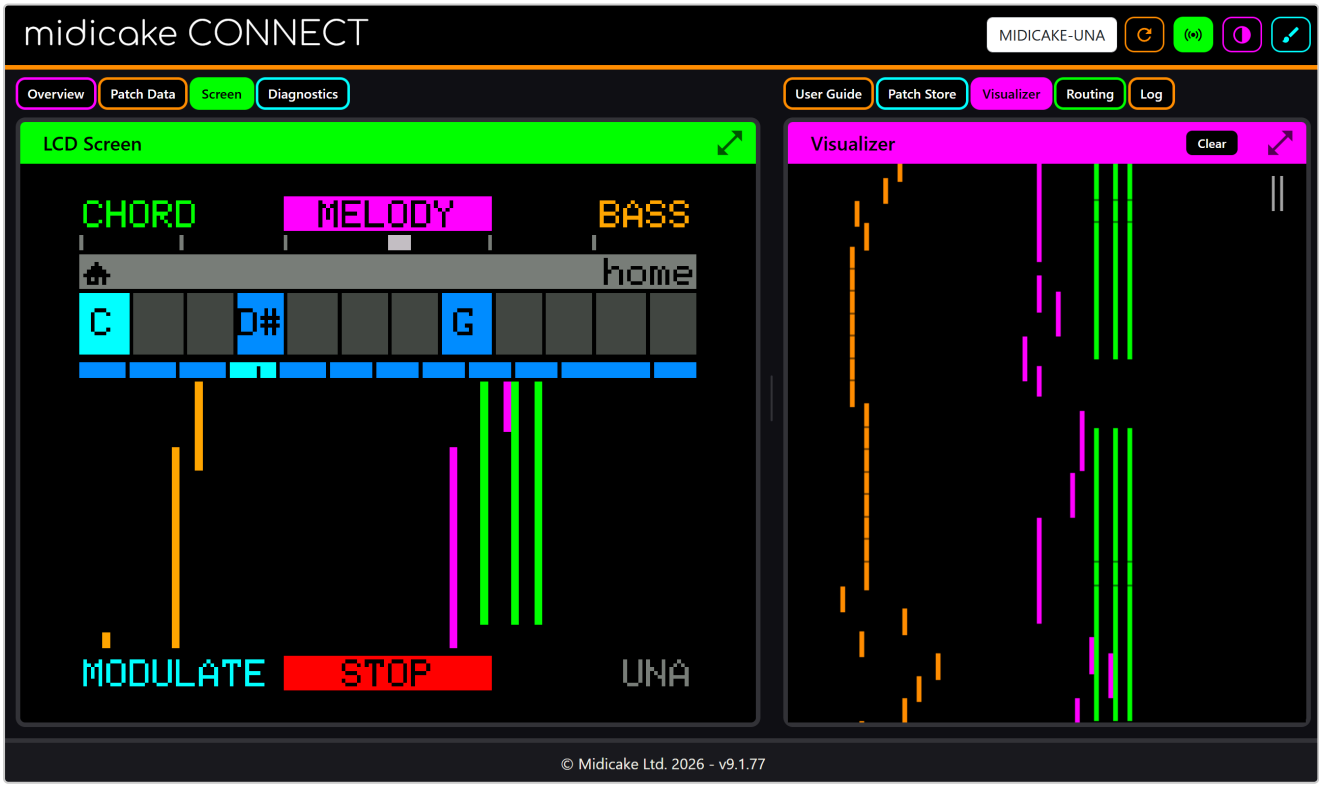
The visualizer listens to MIDI clock and marks bars in the modes that show them. Tempo is detected from the clock stream itself, so what you see stays locked to what the device is playing - including tempo changes mid-performance.

The Clear button

Notes are drawn from note-on to note-off. If a note-off goes missing - a cable pulled mid-note, an app switched away - a shape can linger. **Clear** flushes everything and starts the canvas fresh. It hides in fullscreen to keep the show clean.

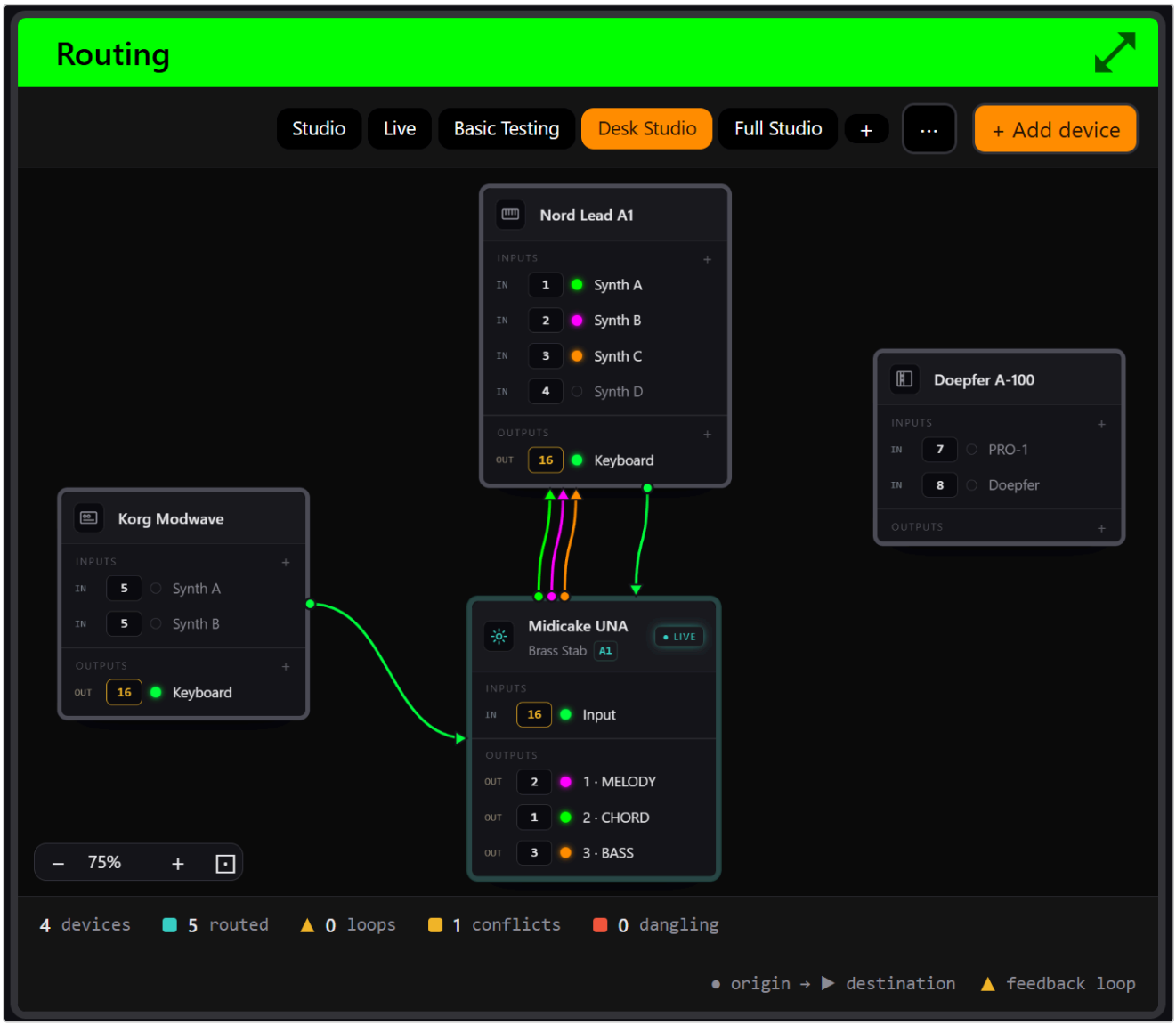
Fullscreen and Cinema

Like the screen mirror, the visualizer is built for fullscreen (Chapter 4) - and it's the other half of **Cinema mode**: in fullscreen, the Cinema button cycles Screen / Visualizer / Split. The split - device UI beside the note rain - is the one people put on projectors.



Chapter 10 - Routing

The **Routing** panel is a canvas for mapping your MIDI setup: every device drawn as a card, every MIDI connection drawn as a coloured wire. It answers the questions every hardware rig eventually asks - *who's sending on channel 3? why is this synth playing two parts? where's the feedback loop?* - by showing the whole picture at once.



The one idea that makes it work

You never draw wires. You tell each device which channels it sends on and which it listens to - and the wires draw themselves. Wherever an output channel matches an input channel, a wire appears, coloured by channel (the same palette as the Visualizer).

That inversion is the point. A wire you drag can lie - the classic out-of-date rig diagram. A wire that's *derived* from the channel numbers can't: if the picture shows a connection, the channel maths says it exists. Fix the channels and the picture fixes itself.

Devices and voices

Each card represents a device, with its **voices** - the channels it speaks on:

- **Add device** opens a dialog: name it, pick an icon, and start from a preset (Keyboard, Mono synth, Multitimbral, Rack, or Custom) that seeds sensible voices you then edit.
- **Set a voice's channel** by clicking it (a 4×4 grid picker, channels 1-16) or just scrolling over it.
- **Flip direction** - each voice is an input or an output; toggle as needed. Add and remove voices freely.
- **Rename** devices and voices inline.
- **Arrange** by dragging cards; pan the canvas by dragging the background, zoom with the wheel, and **Fit** brings everything back into view.

What the canvas flags

The status strip counts devices, routed connections - and problems:

- **Dangling output** (red) - a device transmits on a channel nothing listens to. Silent gear, found.
- **Idle input** - a device listens on a channel nothing sends. Usually fine; occasionally the reason "it's not responding".
- **Conflict** - two outputs on the same channel feeding one input. Sometimes intentional layering, often the answer to "why is it playing both parts?"
- **Feedback loop** (warning triangle) - a channel path that circles back to where it started. The classic MIDI hang, visible before it bites.

None of these block you - the canvas describes, you decide.

Rigs

The tabs across the top hold separate **rigs** - complete setups. *Studio* and *Live* come seeded as examples; make as many as you have configurations. Double-click a tab to rename; duplicate one as the starting point for a variant.

The **⋯ menu** handles the files: **Export rig** and **Import rig** for one setup, **Export backup** and **Restore backup** for all of them. Imports always *add* - an imported rig never overwrites one you have, even with the same name.

Rigs live in the browser's local storage, same as the Patch Store - so the same rule applies: **export what you'd mind losing** (Chapter 14).

The live device card

Your Midicake's card is special. While **Live** (Chapter 5), it shows the channels streamed from the actual device - the real Input, Melody, Chord and Bass channel assignments, locked read-only, marked ● **LIVE**. The rest of the canvas is your description of your rig; that one card is the truth, direct from the hardware.

(The card has its own GO LIVE button, which is the same Live connection as everywhere else - a shortcut, not a second link.)

What Routing is not

It's a *planning* tool - a living diagram, not a MIDI router. It doesn't move or modify any MIDI; it can't, and that's deliberate. The value is that the diagram stays honest: because wires derive from channels, keeping the diagram current is the same act as keeping your channel plan current.

Chapter 11 - MIDI Log

The **Log** panel is the raw feed: a scrolling console of every MIDI message and connection event CONNECT sees, as it happens. It's the least glamorous panel and the one you'll be gladdest of when something misbehaves.

Reading the log

Messages scroll up as they arrive, colour-coded by type so the stream is scannable at speed - note-ons and note-offs in their own colours, clock ticks, system messages, connection/debug lines, and errors standing out from all of them.

What it's for, in practice:

- **"Is anything actually arriving?"** - the first question of every MIDI problem. The log answers it in one glance.
- **"What channel is that on?"** - notes show their channel, which settles most routing arguments.
- **Stuck note hunting** - find the note-on that never got its note-off.
- **Connection events** - the Live link's ups and downs appear here too, which helps when diagnosing drops (Chapter 14).

Filters

Clock alone can flood the view - 24 messages per beat before anyone plays a note. The **Filters** button opens toggles for the message families:

- **Notes** - note-on / note-off
- **Clock** - timing ticks and transport
- **Sysex** - system-exclusive traffic
- **Debug** - CONNECT's own connection and transfer messages

The usual setup: Clock off, everything else on. Turn Clock back on only when the problem *is* timing.

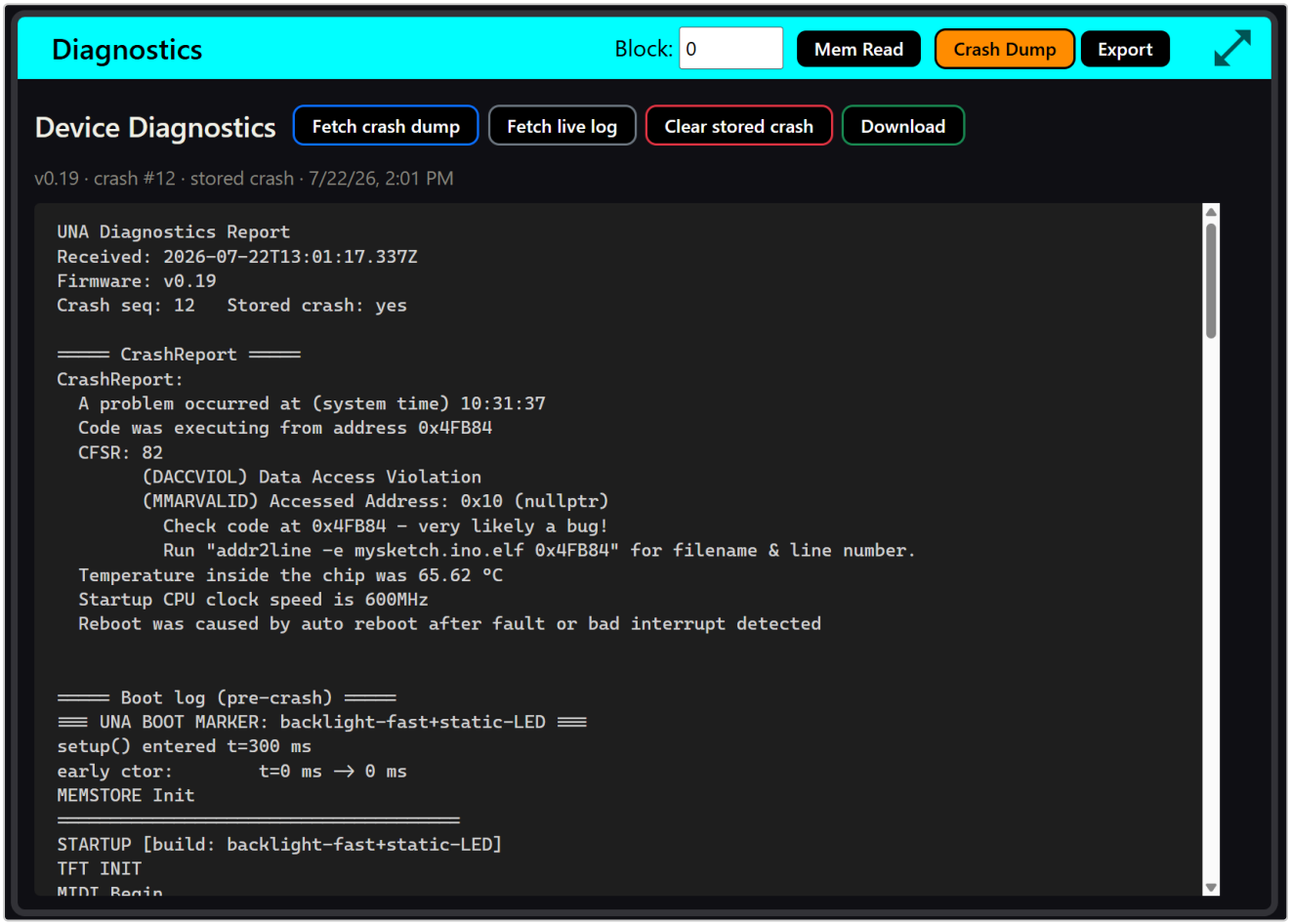
Clear

Clear empties the console. Do it right before reproducing a problem - the log that follows contains only the evidence.

Chapter 12 - Diagnostics

The **Diagnostics** panel is for the rare bad day: the device froze, restarted itself, or did something odd - and you want to know why, or you want Midicake support to know why. The device keeps records of its own trouble; this panel fetches them.

Everything here needs the **Live** connection (Chapter 5); the buttons stay disabled until you're Live.



What the device remembers

When a Midicake crashes, the very next boot preserves a **crash report** - where in the firmware the fault happened, what kind of fault, and the boot log around it. It survives power cycles and waits until you collect it. The device also keeps a rolling **live log** of recent internal events, fetchable any time - useful when something is odd but hasn't actually crashed.

The buttons

- **Fetch crash dump** - pulls the stored crash report. The panel shows the essentials up front: firmware version, when it happened, a crash sequence number, and whether a stored crash exists at all. No stored crash is the healthy answer.
- **Fetch live log** - pulls the recent-events log without needing a crash.
- **Download** - saves everything fetched as a single text file, named and dated.
- **Clear stored crash** - wipes the stored report once you've collected it, so the next fetch answers cleanly about the *next* incident.

The screen mirror pauses briefly during a fetch, like any transfer.

When the device restarts by itself

A spontaneous restart is the device's fault handler doing its job - fault, capture, reboot. The report of what happened is already stored. Go Live, fetch the crash dump, download it, and send the file to midicake.com/support (https://midicake.com/support) with a line about what you were doing at the time.

That file turns "it crashed sometime yesterday" into the exact firmware line that faulted - the difference between a shrug and a fix in the next release.

> **The routine:** Fetch → Download → email support → Clear. Thirty seconds, and the evidence is preserved before anything can overwrite it.

Chapter 13 - The User Guide Panel

The **User Guide** panel is help without leaving the app - two tabs, two kinds of help.

The CONNECT tab

A quick reference for the app itself: what each panel does and where things live. It's the condensed version of this manual, built in - for the fuller story on anything, you're already reading the right document.

The DEVICE tab

The second tab is the clever one. When you're **Live** (Chapter 5), it's labelled with your device's name and shows **contextual help that follows the device**: switch the hardware into Chord Chain Edit, and the panel shows the Chord Chain Edit page. Turn to Stepper Morph, and the help turns with you.

There's nothing to search and nothing to navigate - the device's current mode *is* the navigation. Keep the panel open in the right pane while you learn the instrument, and every screen you land on arrives with its explanation alongside.

Device help is tied to the connection: it's available when a device is connected, because it's the device that tells CONNECT which page you need. No device, no DEVICE tab content - the CONNECT tab and this manual are the offline references.

If you land on a screen the guide doesn't cover yet, it says so honestly rather than guessing - coverage grows with the firmware.

Where the full manuals live

The in-app guide is deliberately brief - explanations, not the full reference. The complete manuals live at manuals.midicake.com: this manual for CONNECT, and the device manuals (with their full parameter references, recipes and troubleshooting) alongside it.

Reference

Chapter 14 - Your Data & Troubleshooting

Two halves: where your data actually lives (read this before you need it), and what to check when connections misbehave (read this when you do).

Where your data lives

CONNECT has no cloud and no account. Everything it stores - your patch library, your rigs, your preferences - lives in **your browser's local storage, on this machine**:

Data	Lives in
Patch Store folders (including backups)	This browser
Routing rigs	This browser
Theme, channel colours, panel layout, visualizer choices	This browser
Your patches and settings on the hardware	The device itself

That design has real virtues - nothing about your music leaves your machine, and there's nothing to sign into. It also has consequences worth stating plainly:

- **A different browser is a different library.** Chrome and Edge on the same computer don't share. Neither do two computers, or a private/incognito window.
- **Clearing browser data deletes the library.** "Clear site data", "clear cookies and site data", aggressive privacy cleaners - any of these erase every folder, every backup, every rig. There is no undo and no recycle bin.

The rule that makes this safe

Anything you'd mind losing should also exist as an exported file. Folder exports (Chapter 8) and rig exports (Chapter 10) produce ordinary JSON files - put them wherever your other precious files live. A library that exists in two places survives anything; a library that exists only in one browser is one settings-menu accident from gone.

The habit: after any session that created something good - export it. Backups of the *device* protect against device problems; exports of the *library* protect against browser problems. They're different risks, covered by different actions.

Troubleshooting

The selector says NO DEVICE FOUND

1. **Cable** - the classic. It must carry data; charge-only cables power the device and connect nothing. Try the cable that came in the box.
2. **Quick press ↻ reconnect** after plugging in - the browser doesn't always notice arrivals.
3. **Try another USB port** - and avoid unpowered hubs while diagnosing.
4. **Check the browser** - Chrome or Edge (Chapter 3). On first use, the browser may be waiting for you to allow MIDI access - look for a permission prompt near the address bar.
5. **Something else holding the device** - a DAW with the device selected can make it invisible to the browser on some systems. Close other MIDI software and reconnect.

Go Live fails, or the device isn't in the port list

- **The port is busy.** Only one program can hold the serial connection - another browser tab already Live is the usual culprit (including a forgotten one). Close other CONNECT tabs.
- **On Android**, make sure you confirmed the USB permission prompt; unplug/replug to trigger it again, and check your OTG adapter.
- **After a firmware update or device restart**, the port can take a few seconds to re-enumerate. ↻ reconnect, then Go Live.

Live keeps dropping

- Nearly always physical: a loose or marginal cable, or an unpowered hub browning out. Swap the cable first, go direct to the computer second.
- Check the **Log** panel (Chapter 11) - connection events land there, and a pattern (drops when you touch the cable, drops under load) points at the cause.
- If the device itself restarted, that's not a connection problem - fetch the crash dump (Chapter 12).

The screen mirror is frozen

- **During a transfer, freezing is normal** - the mirror pauses for patch reads/writes and diagnostics fetches, then resumes (Chapter 6).
- Frozen at any other time usually means Live dropped - check the broadcast icon, reconnect if it's no longer green.

"Layout-version mismatch - refusing to write"

The safety guard, doing its job (Chapter 8): the stored patch was made under a different device storage layout than the connected device is running, and writing it would corrupt. Nothing was touched. This typically means the backup predates a firmware update that changed the layout - check midicake.com/support (https://midicake.com/support) for the migration options for your versions.

The visualizer shows nothing

- The visualizer runs on the MIDI level, not Live - so the device must be selected in the top-bar selector, and actually playing.
- Notes lingering after a disconnection are cleared with the **Clear** button (Chapter 9).

My patches / rigs have vanished

- **Same browser, same machine, same profile?** The library doesn't follow you between browsers, computers, or browser profiles - it's probably still where you left it.
- If browser data was cleared, the library is gone - restore from your exported files, or re-read patches from the device (the hardware copy is untouched by anything that happens in the browser).

Something else

midicake.com/support (https://midicake.com/support) - and if the problem involves the device misbehaving, attach a crash dump (Chapter 12) and say what you were doing when it happened.