MWV — user manual

Edition for MWV 0.14.0 · rendered from `MANUAL.md`

MWV draws a PCB board file on screen so you can look at a board, inspect its parts and nets, and follow it into the board's schematic PDF.

This is the **user manual**. It is written for the people who open a board with MWV, not for the people who build it. Its source is this one Markdown file; the HTML and PDF editions are generated from it on demand.

> **Status — first draft complete.** All twelve chapters, the glossary and the hands-on walkthrough (**Appendix B**) are written, and the walkthrough carries figures captured from a running MWV. Its HTML and PDF editions are generated by `scripts/build_manual.py`; the chapters and the walkthrough have not been reviewed line by line — see `TODO.md`.

About this manual

- **Audience.** Someone repairing or studying a board at a bench — a repair technician, another technician meeting MWV for the first time, or a skilled hobbyist. They know boards and they have a board file, perhaps with its repair data and schematic; they know nothing about this program's internals and need no development toolchain. The manual is **not** written for a reader who merely wants to look at a board file without repairing it.
- **Voice.** Address the reader as **you**, in imperative steps: *Open a board. Click the part.* The reference chapters stay terse but keep the same person.
- **Scope, and what this manual is not.** It is about *using* MWV. Three other documents cover the rest, and this manual points at them rather than repeating them:
  - `README.md` — how to get, build and run MWV, and the platform prerequisites.
  - `FEATURES.md` — the product documentation: what the viewer draws, what it can read, and *why* it behaves the way it does.
  - `CLAUDE.md` — the working guide to the code.
- **Coverage.** Only what ships today is described. The manual **never** describes or promises an unbuilt feature, gives **no** build, install or run instructions, carries **no** code or file-format internals, and does **not** re-explain what is already obvious on screen.
- **Worked example.** One board runs through the manual: **`820-02016`**. Its schematic is `samples/BVR/820-02016.pdf`, beside `samples/BVR/820-02016_V1.3.bvr`, so the example can be reproduced, and its repair data comes from the bundled archive. **Appendix B** replays a real fault on it, start to finish, with figures; the other chapters illustrate with it where it helps and name no board of their own.
- **Figures.** The walkthrough in **Appendix B** carries figures captured from a running MWV; they live in `docs/manual/`. Everywhere else the manual is text.
- **Applies to.** MWV 0.x. The version in hand is `version.txt`; a rendered edition is stamped with it when it is rendered.

- **Conventions.**
  - Keyboard shortcuts are written `Ctrl+X`. **On macOS, `Ctrl` means the Command key (⌘)**: wxWidgets reports both as the same modifier, so the program spells it one way everywhere. Beyond this one note the manual is **platform-neutral**: it says *open the File menu*, not which platform's menu that is.
  - *Click* is the left mouse button unless the text says otherwise; *drag* is press, move, release.
  - A board has two **faces**, top and bottom. A part is mounted on one **side**; a face is what you are looking at. The manual keeps the two words apart.
  - A **mil** is a thousandth of an inch — the unit board files and this manual use for board measurements.
  - `Monospace` is something on screen or in a file; **Bold** names a control or a feature.
- **Reading order.** Chapters 1 and 2 are the first tour, 3 to 5 are the tools you work with, 6 and 7 configure the program, and chapter 8 onward is reference. The **Glossary** defines the vocabulary, and **Appendix B** is a hands-on walkthrough of one real repair.

Keeping this manual in sync

This file is hand-written, like `FEATURES.md`, and it has to stay true as the program changes. The choices above are the manual's specification — they were settled with the project owner and should not drift without a decision.

- **The rule.** Any change that alters **what a user sees, reads, clicks, types or searches for** updates the matching chapter **in the same change**. A change a user cannot see — a refactor, a performance fix, a test, an internal rename — does not touch this file.
- **One home per fact.** Each fact is written once, in the chapter that owns it, and other chapters cross-reference it. The default keymap lives in chapter 6, the settings file in chapter 9, and so on; the manual deliberately repeats no table, so there is only ever one place to update.
- **Source of truth.** Each chapter opens with a *Source of truth* line naming the section of `FEATURES.md` or of a module `README.md` it must agree with. When that section changes, the chapter changes with it.
- **Rendering.** This Markdown file is the single source. HTML and PDF editions are generated from it on demand by `python3 scripts/build_manual.py`: one self-contained HTML page with its stylesheet inline and its figures embedded, and that page printed to PDF by a headless Chrome or Chromium. A rendered copy is never edited by hand, and nothing in the program's build depends on this file. Any screenshots or diagrams are committed beside this file and referenced relatively, so a rendered copy stays reproducible. The edition is stamped with the version in `version.txt`, or with the version the release pipeline is about to publish.

Chapter map

#	Chapter	Teaches	Source of truth	Status
1	Getting started	Opening a board and taking the first look	`FEATURES.md` → *How a board reaches the window*; `README.md` → *Run*	Written

#	Chapter	Teaches	Source of truth	Status
2	The board on screen	Panning and zooming, turning the board, layers, pad colours, names, hovering and selecting	`FEATURES.md` → *The board on screen*	Written
3	The Inspector	Reading a part or a net, and following a row to the board	`FEATURES.md` → *The Inspector*	Written
4	The schematic PDF pane	Pairing, browsing, searching and linking a schematic	`FEATURES.md` → *The schematic PDF pane*	Written
5	Finding things	The Find dialog, stepping results, and copying a name	`FEATURES.md` → *Hovering and selecting*; *Keyboard navigation and the keymap*	Written
6	Keyboard and the keymap	The default keys and how to rebind them	`FEATURES.md` → *Keyboard navigation and the keymap*	Written
7	Settings	Colours, drawing, text and the Layers pane	`FEATURES.md` → *Settings*; `src/app/settings/README.md`	Written
8	The window	The title, the status strip, the log palettes, the menus and the About box	`FEATURES.md` → *The window*; *Menus and the About box*	Written
9	Your data on disk	Where settings, the log and the caches live, and what is never written	`src/app/logging/README.md`; `src/app/settings/README.md`; `src/app/schematic/README.md`	Written
10	Board file formats	What MWV can and cannot open	`FEATURES.md` → *Board file formats*	Written
11	OBData repair data	Sidecars, the bundled archive, readings on screen, and the Guides pane	`FEATURES.md` → *OBData repair data*	Written
12	Troubleshooting	When a board will not open, a link will not resolve, or a frame is slow	`FEATURES.md`; `CLAUDE.md` platform notes	Written
A	Glossary	The board and repair vocabulary this manual uses	—	Written
B	A repair, start to finish	The worked example replayed on one real fault, with figures	—	Written

Contents

1. Getting started
2. The board on screen
3. The Inspector
4. The schematic PDF pane

5. Finding things
6. Keyboard and the keymap
7. Settings
8. The window
9. Your data on disk
10. Board file formats
11. OBData repair data
12. Troubleshooting

- A. Glossary
- B. A repair, start to finish

1. Getting started

Source of truth: `FEATURES.md` → ***How a board reaches the window***.

Status: Written.

MWV has one view — the board — and no empty start screen. You open a board and everything follows from it. This chapter opens `820-02016`, the board this manual uses throughout, says what happens while a board loads, and takes the first look.

Getting MWV onto your machine is `README.md`'s job; this chapter assumes the program is open and waiting.

Open a board

There are four ways to open one, and all four do the same thing:

1. **File → Open…** and choose the board file. The dialog's **Board files** filter lists every format MWV can read; **Board files, not yet readable** lists the formats it recognises but cannot draw yet, so a board with a familiar extension can still come back with a refusal (chapter 10).
2. **Drag the file** onto the MWV window, or onto the board itself.
3. **Name the file when you launch MWV** — `mwv 820-02016_V1.3.bvr`. On the command line a second file is refused rather than quietly ignored.
4. **Hand it to MWV from your operating system** — *Open With* for the file type, or a drop onto the app's icon. A plain double-click is not wired up yet (chapter 10).

Each way takes **one file**, because a board is one file plus whatever sidecar sits beside it. If you hand MWV several, it opens the first.

What happens while it loads

Reading the file, importing it and preparing it for drawing all happen **off the main thread**, so the window never freezes and you can open something else meanwhile.

- The status line shows `<name> — loading...` while it works.
- Opening a second board **abandons** the first rather than waiting for it: there is no queue and no dialog, and the newer board simply wins.
- When it finishes, the status line names the format that actually read the file — not the extension you gave it — together with the part and net counts and the repair data that was applied.
- The **log** carries the detail: the same summary, and every note the importer and the sidecar made (chapter 8).

The first question, about repair data

The first time you open a board the archive knows, MWV asks about **archived repair data**. The dialog names the board and offers three answers:

- **Apply** — use the archive's part values, not-stuffed flags and per-net readings for this board, and remember that.
- **No, not this board** — leave the archive out, and remember that.
- **Ask me next time** — decide again the next time this board is opened.

The archive is a mirror of OpenBoardData that travels inside MWV; nothing is downloaded, and the archive is never written. The answer is remembered per board, so the question is asked once (chapter 11). A `.obdata` sidecar that you put beside the board yourself is applied **without the archive question**, and it always wins over the archive. Its own identity is checked separately, below.

A sidecar also states **which board it is for** in its header. If that identity disagrees with the board you opened — the classic case is a file copied from another board and left beside this one — MWV asks before applying anything:

- **Apply Anyway** — use it despite the mismatch. The choice is remembered for that exact file content **and** that board; a copy of the same file beside a different board asks again.
- **Skip This Sidecar** — open the board without it. An override that names this board can still apply on its own.
- **Cancel** — do not open the board at all.

A sidecar that states no identity — a converted file, or a guide-only override — is **not stated**, not wrong, and applies without a question. **View → Board data…** after the open shows which sidecar was used, where it came from, its header and identity, and a list of the notes the load made. A note that names a part or net is clickable and takes the board to it. Select any note and press **Cmd+C** (macOS) or **Ctrl+C** to copy its text, which is the easy way to quote it in a bug report.

The first look

The board arrives **fitted** — the whole outline on screen, nothing to hunt for.

- **Move the pointer over the board.** The status line reads whatever is underneath: a pin as `U7.12 — GND`, a bare pad, a test point as `Nail TP12 — GND`, or the coordinates `X 1234.5 Y 678.9 mil` over empty board. Hovering answers before you click.
- **Click a pad** to select that pin *and* its net together. **Click a part's body** to select the part. The two answer different questions, and chapter 2 returns to the distinction.
- The **title** names the open documents in order: the board's file name, then the schematic's when one is loaded, then the version.
- If MWV finds the board's schematic PDF beside it, the two are paired on open and the schematic pane is ready (chapter 4).

Around the canvas

The window is mostly board. The controls this viewer needs are states rather than buttons, so they are drawn onto the canvas or gathered into a strip down its left edge, and the rest lives in dockable panes:

- the **layer strip** down the left edge, and the **Layers pane** (chapter 2);

- the **Inspector**, right-docked, which reads out a selected part or net (chapter 3), and the **Guides** pane beneath it, which reads the repair guides a board's sidecar carries (chapter 11);
- the **schematic PDF** pane (chapter 4);
- the **status strip** along the bottom and the two **log palettes** (chapter 8);
- the **menu bar**, and the **Settings** dialog behind it (chapters 7 and 8).

If a board will not open

A box titled **Cannot open board** names the reason, and the status line reads `<name> — not loaded`. The previous board stays on screen, because that is still what is displayed, and the log carries the full path and the detail. Chapter 12 covers the failures worth knowing.

What's next

Chapter 2 moves the board: panning, zooming, turning it over, and reading what is on it.

2. The board on screen

Source of truth: `FEATURES.md` → ***The board on screen***.

Status: Written.

The window is the board. Everything in this chapter happens on the canvas or on the strip down its left edge — moving the view, turning the board over, deciding what is drawn, and reading what is under the pointer.

Move around the board

- **Zoom** with the wheel or a trackpad pinch. Both keep the point under the pointer fixed, so you zoom toward what you are looking at.
- **Pan** by dragging: the board follows the pointer rather than fleeing it.
- The **arrow keys** pan, `+` and `-` zoom, and `0` fits the whole board (chapter 6 lists the alternative bindings).
- A short drag — under a few points — counts as a **click**, so selecting survives a shaky hand.
- Zoom runs from a quarter of the fit to 64×. The **status line** reads the zoom as a multiple above a whole board (`12x`) and as a reduction below it (`÷5.1`).

Turn the board over

- **Flip** top to bottom with `Space` (`Ctrl+B` also works). This is how you reach the other side; several formats open on the back, so Flip is often the first thing you reach for.
- **Rotate** a quarter turn with `R` (clockwise) and `Shift+R` (counter-clockwise), and **mirror** left to right with `M`.
- All three are ways of *looking* at the board. They change no geometry and rebuild nothing, a flip never disturbs a pan or a zoom, and **Zoom to Fit (`0`) clears the rotation and the mirror**.

The layer strip

Down the left edge is a fixed column of twelve icons, the full height of the canvas and never moving with a pan, a zoom, a flip or a resize. Ten of them are drawing groups — board, copper pour, part bodies, silkscreen, traces, vias, pads, nails, drills and part names — and **one click toggles one group**. Hovering an icon names the group in a tooltip beside it, and a group that is off is the same icon faded. The icons take their colour from Settings → Colours.

The last two rows are **modes**, not groups: one switches the **spotlight** that dims the rest of the board while something is selected, and the other switches **butterfly view**. Each names its state in its tooltip (`Highlight selected: on/off`, `Butterfly view: on/off`).

The strip is not part of the board: with the pointer over it you are not over the board, so it will not pan, zoom or select, and the status line stops naming copper.

The Layers pane

The strip carries ten of the fourteen element groups. The other four — the part body class colours, net names and the two OBData readings — live in the **Layers pane**, a dockable pane that offers **visibility only** (colour stays on the Colours page). Its important property: **it filters the board, not the pointer.** A group you have hidden still picks and still reads in the status line — hiding it is a decision about what is drawn, not about what exists.

How the parts are drawn

Some board files carry no shape for a part, and MWV draws one from the part's own copper:

- A two-terminal `R`, `C` or `L` is drawn as a **chip package** sized from its pad pitch — a rectangle, because that is what a real chip's copper is, where a viewer that drew a disc read as wrong.
- A large capacitor is drawn as one of the published **tantalum cases** (A to D) where its lands match, as an **aluminium can** with its negative stripe facing the ground pin, or as a chamfered **power coil**.
- **Which of the two a large capacitor is** is your answer in **Settings → Drawing**, never a guess: `Auto` names a tantalum only where the lands match a known case, and invents none for the rest.

Every guessed body is **filled with the Part body colour** as a ground under its pins, so a part reads as a body rather than a hollow outline. The nine **Part body / …** class colours are a second, optional layer inside that ground: turn on **Fill a guessed body with its class colour** in **Settings → Drawing** (or **Part body class colours** in the Layers pane) to show the resistor, capacitor, tantalum, can or coil as its class. A part whose shape the file *did* state is drawn as stated, and the guessed bodies can be turned off in the same place.

Pad colours

Five colours, set on **Settings → Colours**, tell a pad's role apart at a glance: a pad the file described plainly, and the four MWV can recognise — **pin 1** of a multi-pin part, a pin the file marked as connected to **nothing**, a **test point**, and a **ground** pin.

Names over the board

MWV writes a part's **designator** over it, together with its **value** and its **not-stuffed** mark when the Text options ask for them, and a **net's name** where a bridge aliased it. Three rules shape what you see:

- **A name is drawn only when it fits inside the part it names.** Zoomed out, names are held at a ceiling and only the large packages keep one; zoomed in, names come back as the board is examined — a 0402 only under the magnifier.
- **The selected part's pin numbers** are drawn on its pads, but only while none of them overlap, so a dense connector stays unnamed until you are close enough to read it. The selected part's own designator steps outside its body so it is never covered.
- **A part standing vertically is named up it** instead of across it, so the name runs along the part's longer side. Only a part whose orientation the file states clearly is turned this way.

Net names belong to the selected part, and the zoom. The monospaced labels — net names, OBData readings and pin names — are set in a real monospaced font; the designators use Inter, a proportional real face. The net at each pin of the part you have selected in the Inspector — and its OBData voltage or resistance, stacked under the name, when a sidecar carries one — appears on that one part and no other, and only once the view is past **six times the fit**. The net name sits just **outside its pad**, pushed clear of the part body, and a crowded top or bottom edge turns the name to run outward rather than across the part. The OBData readings sit **inside**, opposite the name, their block centred on the pin. A net the file names `UNCONNECTED_n` is a placeholder for a pin that goes nowhere, so its name is not drawn. Zoomed out, the board is the board and no net names are drawn; zoom in on a part and its pins name their nets. The **Net names** layer opens on, and the two OBData groups open off; all three are in the Layers pane.

Which of these are drawn, and how large, is on **Settings → Text**.

Butterfly view

Butterfly view puts both faces of the board in one picture: the top face above, drawn plainly, the bottom face below it turned over, with a gap between. A single camera pans and zooms over the whole scene, a faint line marks the fold, and a click selects the copper of whichever face it lands on. Turn it on from the View menu (`Ctrl+Shift+B`) or from the strip's last row. It is a way of looking, not a property of the board, so it is not saved — a fresh start opens on a single face.

Hover and select

- **Hover** to read: the status line names the copper under the pointer (chapter 1). A part's **body** answers only where no copper does, which is why bare board *inside* a component reads as that component's designator.
- **Click a pad** to select the pin *and* its net: the pin is marked in magenta, the net's copper lights up with it, and the bodies of the parts that carry that net are outlined in hover-cyan beside it — but only on the face where the net has a pin, so a body on the far side never shows through. Every mark is drawn **outside** the copper so it never covers what it names.
- **Click a part's body** to select the **part** instead — its outline, not its pins.
- With the **spotlight** on, everything the selection does not light steps back into the dark, so the selected shape reads at a glance.

Development bridges

A development bridge — a part that joins two different nets, such as a fitted wire or a zero-ohm jumper — is **hidden**, and the nets it joins are **aliased** rather than merged. The status line shows the alias as `(= <canonical name>)`, clicking one side of the bridge lights both, and the Inspector carries the alias on the net (chapter 3).

What's next

Chapter 3 opens the Inspector and reads a selected part or net.

3. The Inspector

Source of truth: `FEATURES.md` → ***The Inspector***.

Status: Written.

The Inspector reads out what is selected on the board. It is one pane with two sections — **Part** and **Net** — because a pin belongs to a part *and* sits on a net, so the part's pins and the net's members are the same question asked twice. The pane has **no buttons**: it shows, and it reports where you want the board to look. The one thing you can change in place is a net's reference reading, from the V, D or R cell of a part's pin list (chapter 11); everything else is read-only.

Open and close it from **View → Inspector** (`Ctrl+I`). It is docked to the right, starts reading from the first board on, and follows the selection wherever that selection came from.

The Part section

- The **header** names the selected part's designator and value, and a **summary line** below it carries its pins, side and position.
- The **pin list** gives every pin its net and any repair readings. A voltage is red, a diode drop green and a resistance grey, each in a light and a dark variant so it reads on either background.

The Net section

- The **header** names the net and its pin count, with the bridge alias when a development bridge joined it.
- Below it are the net's **readings** and its comment.
- The **member list** gives every pin on the net by part, pin and face.

Following a row to the board

Selecting a row takes the board to that pin:

- it **centres the view** on the pin at **4× or more** — a floor, so following a row never shrinks a view you deliberately widened;
- it **selects what the row names**: a part-table row selects its pin, and a Net-section member row selects that member pin;
- a pin that is on the **far face** turns the board to that face *before* centring, so the pin is actually shown.

That is also the keyboard path: the **arrow keys walk the rows**, and the row under the cursor keeps a neutral tint even past the ends of the list. The two lists are app-drawn tables: they take keyboard focus, their headers stay visible when a list is empty, and a header is a label rather than a selectable column.

What's next

Chapter 4 pairs the board with its schematic and links the two.

4. The schematic PDF pane

Source of truth: `FEATURES.md` → ***The schematic PDF pane***.

Status: Written.

MWV pairs a board with the PDF of its schematic and links the two by part reference and net name, in both directions. The pane sits beside the board and docks to any frame edge — right by default — and that choice is remembered.

Pairing

When a board opens, MWV looks **in the board's own directory only** — never further afield — and pairs it with:

- a PDF whose filename **stem matches exactly**;
- a PDF whose stem matches once descriptive suffixes (such as `boardview` or `schematic`) are removed, **when the revision agrees**;
- a lone PDF whose name merely **contains the board number**, as a weaker match that is said out loud.

A PDF for a **different revision** is never paired automatically, and an ambiguous or revision-unconfirmed result leaves the board unpaired and says so in the log rather than guessing.

File → Open Schematic… picks any local PDF yourself. That choice is remembered per board file — a different revision gets its own entry — and wins on later opens. If the remembered file has gone, automatic discovery runs again with a notice. A board that pairs with nothing closes the previous schematic rather than showing it beside the new board.

Reading the schematic

The pane shows **one page at 100%**, with page controls and **Fit page** and **Fit width**. Zoom with the wheel or a pinch and move by dragging, as on the board; the wheel never leaves the page under it. Jumping to a match centres that match in the view, not merely its page. The pane carries no status bar of its own — the toolbar names the page and the current result, and lasting notices go to the log.

With **Follow board colors** on — the default — a dark board canvas turns the page dark, so ink and paper trade places and the two agree about light and dark. A light canvas is never touched.

Board links and text search

The pane can follow the board and be followed from it, but the two searches are separate:

- **Board links** recognise only **complete names from the loaded board**, at token boundaries, and keep part and net identities apart when their strings coincide. A part's matches are **ranked** so the real callout leads — an occurrence the part's value sits under, then the largest text — so a designator lands on the component, not on a BOM-table row or a block-diagram label.

- **Text search** has its own box and its own result sequence; selecting a search result alone never creates a board link.

Auto link is one app-wide switch, **off by default**, toggled in the pane and in the View menu. With it off, a board selection changes only the board (**Find in schematic** jumps to its first match) and a PDF click only selects the PDF name (**Show on board** applies it). With it on, both sides follow immediately. **Next** and **previous match** cycle every occurrence across all sheets, with the current result count and page.

Matches are marked with a **wash**, not an outline — a filled highlight, the current one stronger, in an ink that follows the page from light to dark.

A PDF with **no extractable text** stays viewable and explains why search and links are off. The text index is built in the background and cached under the app's data (chapter 9).

What's next

Chapter 5 finds a part or a net by name.

5. Finding things

Source of truth: `FEATURES.md` → ***Hovering and selecting*** *and* ***Keyboard navigation and the keymap****.*

Status: Written.

Two searches look through a board and its schematic: **Find part or net** on the board, and the schematic pane's own text search. They are separate — one searches board names, the other searches the PDF's text — and this chapter covers the first.

Find a part or a net

`Ctrl+F` opens the **Find part or net** dialog — from the keyboard, or from **Edit → Find part or net…**, which is also what reaches it while another pane has focus. Type into the field and the matches appear below as you type. Matching is on text, ignoring case:

- a **part** matches on its **designator or its value**, so `U7000` finds the part and `10k` finds the resistors. Hidden development bridges are not offered.
- a **net** matches on its **name or any of its aliases**.

Selecting a row **selects what it names, centres it on the board, and — for a part — turns the board to that part's side**: a designator names a body on one face, and leaving the other face in view would show the selection on the side the part is not on. A **net** spans both faces and is centred without turning. The dialog is **modeless**, so the board stays visible while you look — a modal box would hide the very thing you are looking for. Enter steps to the first match, and a double-click selects the row under the pointer; neither closes the dialog.

The dialog is not kept between sessions: **closing it forgets it, and reopening seeds the field with the last query** you typed.

Step through results

- `N` and `Shift+N` step to the **next and previous** result, wrapping past the ends. This works **whether or not the dialog is open**, so you can find once and then step with the dialog closed.
- If a find matches nothing, the status line says `Find: no match`.

Copy a name

`Ctrl+C` copies the **selected part's designator or the selected net's name** to the clipboard.

Walking a selection

The other way through a selection is the Inspector: with a part selected, its arrow keys walk the part's pins and take the board with them (chapter 3), and `Tab` and `Shift+Tab` walk the same pins from the canvas (chapter 6).

What's next

Chapter 6 lists the keyboard's commands and how to rebind them.

6. Keyboard and the keymap

Source of truth: `FEATURES.md` → ***Keyboard navigation and the keymap***.

Status: Written.

The board's commands run from the keyboard while the **canvas has focus**. A left press on the canvas gives it focus, and MWV focuses it once at start-up, so the shortcuts act on the board you just reached for. A key no board command claims is passed back to the system: a focused text field or table keeps its own typing, and the menu accelerators still work. That is the one rule to remember — **if a key does nothing, click the board and try again.** Find is the exception: it also has an **Edit** menu item, so `Ctrl+F` opens it whatever has focus.

A **dialog on top of the board** — the guide editor, Find, Board data — takes the keys for itself, including `Space`: the board underneath does not claim them while you are typing into a dialog.

The default keymap

Command	Primary	Alternate
Pan left / right / up / down	arrow keys	—
Flip board top/bottom	`Space`	`Ctrl+B`
Mirror board left/right	`M`	—
Centre on the selection	`X`	—
Rotate clockwise	`R`	—
Rotate counter-clockwise	`Shift+R`	—
Zoom in / out	`+` / `-`	—
Zoom to fit (reset the view)	`0`	`Ctrl+0`
Find a part or net	`Ctrl+F`	—
Next / previous find result	`N` / `Shift+N`	—
Copy the selected part or net name	`Ctrl+C`	—
Clear the selection	`Esc`	—
Next / previous pin of the selected part	`Tab` / `Shift+Tab`	—
Toggle net names	`A`	—
Toggle OBData readings	`O`	—

Every key is written `Ctrl+X`; on macOS `Ctrl` means the **Command** key, because the system reports both as the same modifier (see *Conventions*).

A part's pins are walked **within that one part**, wrapping round; moving between parts is a click, the Inspector or Find, and there is no key for it. A net selection is folded onto one of its pins so a click and a `Tab` share one cursor.

Two slots, and rebinding

Every command has **two slots** — a primary and a nullable alternate — so a second keyboard layout or a second habit can have its own key. The alternate shows only on the Keys page; the View menu's **Flip Board**, **Mirror Board** and **Zoom to Fit** labels carry the primary alone, so a rebound Flip shows your own key there. A **bare-key** primary (space, a letter) is shown in parentheses, `Flip Board (Space)`, because a key written after a tab would become a global menu shortcut and be taken from every text field in the window — the guide editor's own typing included. Only `Ctrl`/`Cmd` and `Alt` combinations appear as menu shortcuts.

Rebind on the **Keys page** of the Settings dialog (chapter 7): select a slot, press a chord, and it is captured. The **Natural direction** checkbox on that page is not a key — it decides whether an arrow key moves *the board* (the default) or *the camera*.

What's next

Chapter 7 opens the Settings dialog.

7. Settings

Source of truth: `FEATURES.md` → ***Settings***; `src/app/settings/README.md`.

Status: Written.

Settings holds everything about how the board is drawn — the colours, the drawing options, the text and the keymap. It is a native window with four pages: **Colours**, **Drawing**, **Text** and **Keys**. Open it with **Settings…** on the File menu (`Ctrl+Alt+S`, and `Cmd+,` on macOS, where it also sits in the application menu).

Element-group visibility is deliberately **not** here: it is the layer strip and the Layers pane (chapter 2), because those are used while looking at the board, and a settings window would put the board behind the thing being adjusted.

Colours

Colours come in **named themes**. A theme is one complete palette, and exactly one theme is active. Three ship:

- **vanilla** — the board colours MWV has always used, with a light window chrome.
- **graphite** — a near-black canvas with bright pins, a yellow selection and dark chrome: the high-contrast choice for a dim room.
- **paper** — warm light paper with dark ink, for a bench lamp or a printed screenshot.

The dropdown at the top of the page chooses the active theme. **New** starts a theme from the shipped vanilla palette, **Duplicate** copies the active one, **Rename** renames it, and **Delete** removes it; the last theme cannot be deleted. The colour rows below edit the active theme, so each theme keeps its own customisations. A theme name can be anything you like, as long as it is not already taken.

The rows are grouped: the board and its copper, pads and nails, the board body and part bodies, the pointer overlays, the labels over the board, and the window chrome. The chrome rows are the **status bar's background, text and link** and the **panes' background, text and link**; the Inspector's marked row is a neutral tint of the pane text, not the link colour, so the reading colours stay readable. The dialogs, menus and other native controls keep the platform's own appearance.

Two things about the colours are easy to trip over:

- The **hover and selection overlays are the only translucent board colours**, on purpose: an opaque hover mark would hide the very pad it points at.
- The colour **swatch does not show translucency** even though the panel and the board do — a known cosmetic limitation, so judge an alpha by the board, not by the button.

Drawing

The options for **copper the file measured**: **circle segments** (how finely a drawn circle is tessellated), **pad scale**, the **nail, trace and via default sizes**, and a **minimum radius** for the smallest drawn shapes.

Then, under a heading of its own, the options for **a body the renderer invents**: **draw a guessed body** for a part that stated no shape; **fill it with the part body colour** (on by default); **fill it with its class colour** — the class-colour resistor, capacitor, tantalum, can or coil, off until you ask for it; **case sizes** and **polarity marks**; **large capacitor** — `Auto` (a tantalum only where the lands match a published case), `Tantalum` or `Aluminium`; and **can size**, the standard size an aluminium can is drawn at. No board format here states a dielectric, so which a large capacitor is, is your answer rather than the viewer's.

Pad scale is the one to understand. A board file often carries no pad size — a BVR1 pin record states a position and a net and nothing else — so MWV supplies a nominal radius, and on a fine-pitch part those nominal pads merge. The scale shrinks only those **invented** pads; a pad the file measured, a polygon, a box, a stated width or radius, and a two-terminal chip's own lands are drawn as the file described them whatever this is set to. The default (`0.25`) is the value the `820-01700` sample settled on.

Text

What the names over the board **say** — part **values**, **not-stuffed** marks, pin names, and a net's canonical alias — and how **big** they are: a target pixel height, a minimum and maximum height in mils, and the stroke weight. Nothing on this page rebuilds the board; only the text is rebuilt.

Keys

The keymap editor: one row per command with a primary and an alternate column, a slot to capture a chord into, and the **Natural direction** switch (chapter 6).

Apply, OK and Cancel

The dialog holds a **working copy**, and a theme switch or a colour change previews it on the board, the status strip and the panes at once. A preview is not a save.

- **Apply** commits and stays open; **OK** commits and closes.
- **Cancel** puts back the last value you committed — including a value Apply saved earlier in the same visit — so a theme you switched to and cancelled leaves no trace.
- **Restore Defaults** copies the shipped defaults into the working copy and previews them, but does **not** commit, because a button labelled Restore Defaults is not also an Apply.

When settings are saved

Settings are written **on Apply and OK, and on a layer toggle** from the strip or the Layers pane, and **nowhere else** — never per keystroke and never on exit. A preview is never a save. All of those paths land in one save, which is where the rule lives.

What's next

Chapter 8 describes the rest of the window around the board.

8. The window

Source of truth: `FEATURES.md` → ***The window*** *and* ***Menus and the About box****.*

Status: Written.

Around the canvas are a title, a strip of status along the bottom, two log palettes and a menu bar.

The title

The title names the open documents, in order: the **board's file name**, then the **schematic's** when one is loaded, then the **version**. It is set only on a successful open, so a failed open leaves the previous board's name on screen, which is the truth about what is still displayed. A board that pairs with nothing closes the previous schematic rather than naming it beside the new board.

The status strip

The status strip along the bottom is MWV's own rather than the system's, so its background, text and link are the active theme's **status bar** colours (chapter 7). The note count is underlined as well as coloured, so it remains a link even on a theme whose accent sits close to its text. It carries four fields:

- **The board's reading.** With no pointer on the board it is the load state — the format that read the file, and the part and net counts. While the pointer is over the board it is the **copper under the pointer**: a pin as `U7.12 — GND`, a pad, a nail as `Nail TP12 — GND`, the body's designator over a part, or `X 1234.5 Y 678.9 mil` over bare board. On a net a bridge aliased, `(= canonical)` follows the name it really joins.
- **The note count**, when the board has any: a link such as `3 notes` that opens **Board data…**. It stays visible while the pointer changes the reading beside it.
- **The zoom**, glued to the board's reading: a **multiple** above a whole board (`12x`) and a **reduction** below it (`÷5.1`), never `0.2x`.
- **The schematic's reading**, in its own split: the page, the selected part or net, and the match counts.

The board's reading also names the OBData sidecar the open used — `(from the archive)`, `(local override)`, `not applied`, and `(identity mismatch)` when a mismatched file was accepted or skipped. **View → Board data…** carries the detail, including the load-notes list; the note count beside the reading is a link to it.

When a load records a problem note — something dropped, skipped, unreadable or conflicting — the dialog opens once by itself. **OK** leaves the preference unchanged, **Don't show for this board** remembers that file, and **Don't show for any board** switches the prompt off. **View → Open import notes on load** turns it back on.

The log palettes

Two floating panes, hidden at first and toggled from the View menu:

- **Messages** shows only what MWV addressed to you; **Debug** shows everything, including toolkit messages. The text is selectable.

- The log file itself — where every line goes, whether or not a palette is open — is `mwv.log` in the app's data directory (chapter 9).

The menus

- **File**: **Open…** (`Ctrl+O`), **Open Schematic…** (`Ctrl+Shift+O`), **Reload Repair Data** (`Ctrl+R`), **Settings…**, and Quit (`Ctrl+Q`). Reload re-reads the sidecars beside the board and applies them to the open board without reopening it; it is disabled until a board is on screen. On macOS, Quit, About and Settings move to the native application menu.
- **Edit**: **Find part or net…** (`Ctrl+F`), disabled until a board is open. It is the one board command that also reaches you while a pane has focus, because a menu accelerator is frame-wide.
- **View**: the checkable panes — the two log palettes, **Layers**, **Inspector** (`Ctrl+I`) and **Schematic** (`Ctrl+J`) — and **Auto link schematic**, **Guides**, and **Board data…**, then **Flip Board**, **Mirror Board**, **Butterfly view** and **Zoom to Fit**. The Flip, Mirror and Zoom to Fit labels carry your own primary key (chapter 6).
- **Help**: **About MWV** (on non-macOS; on macOS it is in the application menu).

The About box and Licenses

About MWV shows the icon, the name, the version and the copyright, the viewers MWV was inspired by, and the open-source projects it is built with. Its **Licenses** button opens a second window listing every third-party component and the selected one's licence text; components whose text is identical share a single entry. Both windows are modeless, and the Licenses window is parented to the frame rather than to the About box, so it stays open and readable after the About box is dismissed.

What's next

Chapter 9 lists the files MWV keeps on your disk.

9. Your data on disk

Source of truth: `src/app/logging/README.md`; `src/app/settings/README.md`; `src/app/schematic/README.md`.

Status: Written.

MWV keeps a small number of files in **your own application-data directory** — `~/Library/Application Support/MWV` on macOS, and the platform's standard per-user data directory elsewhere. It writes nothing into a board or a PDF. The only file it writes beside a board is the local `*.obdlocal` you create by editing a reading (chapter 11).

What	Where	What it holds
Settings	`settings.json`	Everything you have changed: colours, drawing options, text options, element-group visibility, the keymap, the schematic pane's own preferences (Auto link, Follow board colors, the docking side, and the schematics you have picked by hand), the per-board answer about the OBData archive, and any remembered "apply anyway" answers for a mismatched sidecar (chapters 4, 6, 7 and 11).
Log	`mwv.log`, with `mwv.log.1` … `mwv.log.5`	Every record of a session. The file is rotated at start-up, so **each session begins empty** and the five previous sessions are kept.
Schematic index cache	`pdf-index/pdf-<key>.json`	The derived text index for a PDF, so a schematic need not be re-extracted each time it is opened. It is a **discardable cache**: delete the directory and the next open rebuilds it.
Local override	`<board stem>.obdlocal` beside the board	The reference readings you edited, in OBData syntax (chapter 11). Never the bundled archive, and never under `samples/`.

The app logs the paths it resolves — the version, the log file, the settings file and the cache directory — as the first lines of each session, so the log tells you where everything is.

What is written, and what is not

- **Nothing is written into a board or a PDF**, and nothing under `samples/`. The application reads your board files and their sidecars; the one file it creates beside a board is the `*.obdlocal` from an edited reading (chapter 11).
- The **OBData archive** that ships with the app is **read-only** and lives inside the installation, not in your data directory (chapter 11). Nothing in the application writes to it, and nothing downloads it — there is no network in the application at all.

Resetting

- Deleting `settings.json` returns every setting to its default on the next start. You lose your colours, options, keymap, the per-board OBData answers and the remembered sidecar-mismatch consents, so those boards and sidecars are asked about again.

- Deleting `pdf-index/` costs only a rebuild of the schematic text indexes.
- Deleting `mwv.log*` costs only history.

What's next

Chapter 10 lists the board formats MWV can read.

10. Board file formats

Source of truth: `FEATURES.md` → ***Board file formats***.

Status: Written.

MWV registers **sixteen** board formats; **ten** of them really import, and the other six recognise their files and decline them with a message.

What opens

Format	Extensions	Opens?
XZZ	`.pcb`	Yes
BRD	`.brd`	Yes
BVR1, BVR3	`.bvr`, `.bv`	Yes
FZ	`.fz`	Yes
KICAD	`.kicad_pcb`	Yes
BDV	`.brd`, `.bdv` (often `.GR` in the wild)	Yes
BDV_ASC	`.bdv`, `.asc`	Yes
CAD (GenCAD)	`.cad`	Yes
TVW	`.tvw`	Yes
BV	`.bv`	No — recognised, not read
ALLEGRO_BRD, EAGLE_BRD	`.brd`	No
CAE	`.cae`	No
MENTOR	`.cad`	No
ALTIUM_PCB	`.pcbdoc`, `.cmpcbdoc`, `.cspcbdoc`	No

Several extensions are claimed by more than one format — `.brd` by four, `.bv` by three, and `.cad` and `.bdv` by two each — because the formats genuinely share them. Which one you get is decided by the file's **content** first, so a real board of one of these reaches the right reader.

The Open dialog

The dialog offers three filters:

- **Board files** — every extension a readable format declares, and nothing else. This is the one it opens on.
- **Board files, not yet readable** — the extensions only a *stub* claims, so a file whose only owner is a stub still reaches that stub's own message in one click.

- **All files** — for a board whose extension is missing or wrong.

An extension a readable format *shares* with a stub stays in the first filter, which is why `.bv`, `.brd` and `.cad` are there even though they can also reach a refusal.

How a file is identified

MWV **sniffs the first 512 bytes** of the file and tries each format's detector. If none matches, it falls back to the **file extension**. Only when both fail does it report `Unknown board file format.`

That order explains most surprises when a board will not open:

- A `.bv` is a Microsoft Access database. It reaches **BV's own refusal**, which says what the file is, rather than being read as a corrupt BVR1 — which is what used to happen, because BVR1 also claims `.bv`.
- GenCAD is reachable **only through its detector**: MENTOR is registered before CAD and also claims `.cad`, so a GenCAD file whose header falls outside the first 512 bytes cannot be recognised and reaches the Mentor stub instead.
- A `.brd` that none of the four `.brd` formats claims falls to the first claimant in registration order, which is **BDV** — so it is reported as a BDV file that could not be read rather than as an unknown format.

When the imported model is invalid

A file can be read from beginning to end and still describe a board MWV cannot use: a coordinate that is not a number, a net that points at a pin which does not exist, bounds that run backwards, or a count so large it can only be corruption. After the importer finishes, MWV checks the finished model once, before repair data is applied or the board is prepared for drawing.

A **structural fault** refuses the open. The **Cannot open board** box carries `The board imported but its model is invalid: …` and the specific fault; the previous board stays on screen. This is deliberate: a partly drawn or unclickable board is worse than a clear refusal.

A **recoverable finding** lets the board open and becomes a `Validation:` line in **View → Board data…** and in the log. Because it is a problem note, the dialog opens once by itself on that load. The two current cases are bounds that do not enclose every part and pin, and a trace or surface naming a layer the board does not have; both have a defined fallback in the drawing.

The limits behind this check are far above real boards — a million parts, ten million pins and net references, twenty million traces, a thousand layers and a hundred million polygon points — so crossing one means a malformed file rather than a large design.

When a board imports but looks wrong

Each format's own quirks — its container, its encoding, the geometry it guesses, and where it departs from its reference — are written down beside its importer, under `src/app/pcbfile/<format>/README.md`. That is the place to look before assuming a board is bad, and `samples/README.md` records which real files this tree can and cannot read.

What's next

Chapter 11 explains the repair data that sits beside a board.

11. OBData repair data

Source of truth: `FEATURES.md` → ***OBData repair data***.

Status: Written.

The repair data a technician wants — part values, not-stuffed flags, and per-net diode, voltage and resistance readings — is not in the board file. It lives in an **OBData sidecar** beside the board, or in the **archive** bundled with the app. A sidecar can also carry **repair guides** — prose about the board — which MWV reads and shows in a pane of their own.

The sidecar beside the board

MWV looks in the board's own directory for `*.obdata` and `*.obdlocal`, and matches one to the board **by name**:

- an exact stem match comes first (`820-00165.brd` → `820-00165.obdata`);
- otherwise the closest name that **relates** to the board's — one stem opening the other (`820-00165-02.pcb` beside `820-00165.obdata`), or a sidecar named for the board number inside a longer board name;
- otherwise **nothing**, rather than a guess.

A candidate that does not relate is refused **even when it is the only one**, because a directory legally holds one board and somebody else's sidecar — and another board's values look right while being wrong. A shared `820-` is not a relation. Every non-obvious choice, and every refusal, explains itself in the log.

A `*.obdlocal` is layered **over** the `*.obdata`, field by field, so it needs to state only what it changes: a local correction to a sidecar, or to the archive's copy.

The bundled archive

With **no sidecar beside the board**, the data comes from the archive that ships inside MWV — a mirror of OpenBoardData, read-only, keyed by the board's own number. The first time you open a board the archive knows, MWV asks:

> *The archive that ships with MWV holds repair data for … : part values, not-stuffed flags and per-net readings. … Apply it to the board you are opening?*

with **Apply**, **No, not this board** and **Ask me next time**. Apply and No are **remembered per board**; Ask me next time applies to that open only, so there is always a way back. A sidecar you place beside the board yourself is applied **without** asking, and wins over the archive either way.

What a sidecar changes

- It fills a field **only where the board left it empty**: the **board file wins a conflict**, and the disagreement is counted, never resolved.

- A **not-stuffed** flag is *set*, because no board format states one — the sidecar is the only source for it.
- A row naming a designator or net the board does not have is **counted, not created**: a sidecar and a board from different sources is the normal case, and inventing empty parts would put hundreds of entries in the viewer and call it data.

Where the readings appear

A **part's** value and not-stuffed mark appear over the board and in the Inspector; a **net's** diode, voltage and resistance readings appear in the Inspector, with a voltage in red, a diode drop in green and a resistance in grey, each in a light and a dark variant.

Editing a net's readings

The Inspector's readings are editable. **Double-click a cell in the V, D or R column** of a part's pin list; the cell becomes a text field with its current value selected. Type the new reading and press **Enter**, or press **Escape** to leave it alone. A reading belongs to the net: every pin on the same net shows the same value, and changing it changes all of them.

- **Voltage** is in volts. `V` and `mV` suffixes are accepted.
- **Diode drop** is in **millivolts**, as the cell shows it: `573` is 0.573 V. `OL` means the meter found no conduction and is a reading, not a blank. A `V` suffix is also accepted if you prefer to type the drop in volts (`0,573 V`).
- **Resistance** is in ohms. A `k` or `M` suffix is accepted (`15k` is 15000), and the cell's own `kΩ` spelling reads back as itself.
- **Both `.` and `,` are accepted as the decimal separator**, whichever your keyboard or locale uses. A comma is a decimal point, never a thousands separator: `1,5` is one and a half, while `1,234` is 1.234.
- **RKM notation is accepted too**, where the multiplier letter carries the decimal point: `4k7` is 4.7 kΩ, `1R5` is 1.5 Ω, `0R47` is 0.47 Ω, `4M7` is 4.7 MΩ, `3V3` is 3.3 V, and `4m7` is 4.7 mV. The same notation is understood in a sidecar's `v`, `r` and `d` rows, so a hand-written local correction reads the way it was written.
- **A blank cell means no reference reading.** If the value came from the imported sidecar, MWV writes an explicit clear (`d ''`, `v ''` or `r ''`) to remove it; if it only ever existed in your local file, the row is removed.

The change is written to the board's local `*.obdlocal` — `<board stem>.obdlocal` beside the board, created on the first edit. **The bundled archive and `samples/` are never written.** Only the rows for the readings you changed are touched: the header, the guides and every other row in the file are kept exactly as they were, including files written with Windows line endings.

The board you are looking at updates at once, without reopening. If the local override on disk is not the one the board opened with — it appeared, vanished, was renamed or was skipped for an identity mismatch — MWV asks you to reopen the board before editing rather than write into a file whose other rows the session has not applied.

Reading the guides

A sidecar can also carry a **DIAGNOSIS** block: prose about the board — a power sequence, a list of rails, troubleshooting notes — written with references to the nets and parts it means. MWV reads it and shows it in the **Guides** pane, docked under the Inspector (`Guides` in the View menu, `Ctrl+G` — `Cmd+G` on macOS). The pane appears when the board you open has guides, and stays out of the way when it does not.

- **Sections and notes.** A chooser picks a section; the list beside it picks a note; the note's text fills the pane. The words are the file's own, line breaks and indentation included. A file that writes one heading more than once — the BRD sidecar's four `Power Rails` blocks — reads as one section holding all of its notes. A section whose heading says what the board *is* rather than what to do about a fault (`Overview`, `General`, `Summary`, `Introduction`, `About`) is listed first; every other section keeps the file's order.
- **Following a reference.** `[n:NET]`, `[p:PART]` and `[p:PART:PIN]` are underlined, and **one click follows one**: the object is selected, the board turns to the pin's face if it is on one, and the camera centres on it. Following a **net** selects it without moving the view, because a net has no one place to centre on. A reference the board does not have keeps its brackets and is drawn **muted**; clicking it tells you in the log that it names nothing on this board.
- **Expected values.** Where a note states what a reading should be — `= 3.3V`, `>= 14.5V`, `80~100K`, `8.2V Air 12.2V Pro` — those appear as rows under the note, headed **Expected**. They are the sidecar's expectations and **not measurements**: a reading you take yourself belongs to a repair job, which MWV does not keep yet. A row names what it is measured on — a net, a part or a pin — and both ends when the guide states them (`[p:C1910:1] to [p:R5321:2]`, or `to GND`). A written tolerance (`±5%`, `±0.1V`) is shown with the value, and the **Mode** column names the quantity with the board state it implies: `(DC volts)` and `(frequency)` are measured powered, `(diode)` and `(resistance)` unpowered. A row is clickable and focuses its first end.
- **Keyboard and copying.** `Left` and `Right` walk the references, `Enter` follows the marked one. Right-click copies the whole note or a single reference name.
- **A corrected note, and the imported text.** A local `*.obdlocal` can replace a note's text. When the imported text has changed since that correction was written, the note is marked **imported text changed** in the list and the imported text appears under the correction, with the file it came from — a correction never quietly wins over prose you have not seen.

A guide is reference material: MWV records no progress in one. A correction to a note's text lives in the board's local `*.obdlocal`, never in the imported file or the archive.

Correcting a guide

Edit Guides…, under the pane, opens the editor on the section you are reading. The headings are on the left, their notes in the middle and the selected note's text on the right, with the imported text a correction replaces shown underneath.

- **Change the text and Save.** Only your corrections are written, to the board's local `*.obdlocal`; the imported file and the archive are never touched. A note you did not change is not copied into the local file, so a later change to the imported guide still shows through.

- **Add Section** and **Add Note** create local rows. **Up** and **Down** move a heading or a note; a row moved only for its position does not restate its text.
- **Findings and fixes.** Under the text, the editor lists what it can see wrong: a reference that names nothing, a heading or note title used twice, a line that states a value the reader could not read, an empty note. Choosing a finding selects the text that caused it; when a reference looks like a mistyped board name, **Fix** replaces it with the nearest one, and never silently.
- **The review before the write. Save** first lists the rows it will add, change and remove in the local `*.obdlocal`, with a changed note's imported text beside the text that replaces it. Nothing is written until you choose **Write**.
- **Help.** The **Help** button opens a page with the editor's own rules and the expected-value and step syntax, so the grammar does not have to be remembered.
- A note or heading that comes from the imported file **cannot be deleted or renamed** — the file format has no way to say so. Removing an override puts the imported text back.
- Closing with unsaved changes asks before discarding them; saving re-merges the guide and refills the pane.

Reloading repair data

If you correct a `*.obdlocal` outside MWV — or simply want the open board to pick up a file that changed — **File → Reload Repair Data** (`Ctrl+R`) re-reads the sidecars beside the board and applies them to the board on screen. Part values, packages, ratings, not-stuffed marks, net readings and guides all come from the fresh files; the view and the mesh are kept, so nothing is reopened and nothing is re-imported. A changed **package** is the one exception: a guessed body has to be rebuilt, so the mesh is rebuilt for it.

A sidecar that names a different board is refused rather than applied; reopen the board to make that decision, where the identity prompt belongs.

Nothing is downloaded, and the archive is never written

The archive is mirrored rather than queried because the site it comes from is scheduled for retirement: there is **no network in the application at all**. Nothing is downloaded, and the archive is never written. The only board-side file MWV creates is the `*.obdlocal` from an edited reading.

What's next

Chapter 12 is what to do when something does not work.

12. Troubleshooting

Source of truth: `FEATURES.md`; `CLAUDE.md` *platform notes.*

Status: Written.

The log is the first place to look for anything here; its location is in chapter 9, and the app writes its own paths there at the start of every session.

A board will not open

A box titled **Cannot open board** names the reason, the status line reads `<name> — not loaded`, and the log carries the **full path** and the detail. The previous board stays on screen, because that is still what is displayed.

The reason is usually one of:

- **The format is not read yet.** Six registered formats decline every file; chapter 10 lists them, and each has a README under `src/app/pcbfile/`.
- **The content was not recognised and the extension was wrong** — a board renamed to the wrong extension, or a format whose header is not in the first 512 bytes. The extension fallback then hands the file to whichever format claims that extension first, and the message names *that* format. Chapter 10 lists the shared extensions.
- **The board is one of the real `.brd` / `.cst` shapes no importer claims** (see `samples/README.md`).
- **The file parsed but its model is invalid.** The box says so explicitly and names the fault — a non-finite coordinate, an out-of-range index, or a count over the model's corruption guard. Chapter 10, *When the imported model is invalid*, has the rules.

The log also carries the **parser notes** the import produced; **View → Board data…** shows the same notes as a list beside the sidecar provenance. The status strip carries their **count** as a link to that list. A `Validation:` note there is a recoverable model finding, not a failed open; one that names a part or net is clickable and takes the board to it.

A **second open never blocks on the first**: opening another board abandons the one running, so a slow import cannot freeze the window. A board you replaced will not appear — the abandoned import is dropped rather than reported, so the log may show only the board that won.

The schematic will not pair

- Only the board's **own directory** is searched, and a PDF for a **different revision** is never paired automatically — the pane logs why. Pick it yourself with **File → Open Schematic…**; the choice is remembered per board file.
- If MWV had paired a schematic before and its file has since gone, discovery runs again with a notice.
- A board that pairs with nothing closes the previous schematic rather than showing it beside the new board.

A link does not resolve

Board links recognise only **complete names from the loaded board**, at token boundaries, and keep part and net apart. A board link that finds nothing leaves the board selection alone and reports it. Only the loaded board's names are recognised, so a schematic has no link to a name the board does not carry — and a part's *value* can be the anchor as well as its designator.

Search and links are off

A PDF with **no extractable text** — a scan — stays viewable and says why search and links are off. Nothing in it can be searched or linked.

It feels slow

- The **mesh is built once, at load**, for both faces; a flip, a rotation or a mirror rebuilds nothing. A **drawing** option, though, rebuilds the mesh, so changing one on a large board is a visible pause.
- The canvas logs **one warning when a frame takes over 50 ms**, at most one every ten seconds, so a long slow drag is one line rather than many.
- A large board is large — the biggest sample here builds some 8 MB of geometry — and a schematic's *second* open is cheaper thanks to the `pdf-index` cache (chapter 9).

Reporting a problem

Include the log's **session lines** (the version, the build number and the paths), the board file's **format**, and what you did. The log is written to be read afterwards, so it is the right thing to attach.

What's next

That is the end of the manual; Appendix A defines the vocabulary.

Appendix A. Glossary

Source of truth: —

Status: Written.

The board and repair vocabulary the manual uses, defined once. Each entry names the chapter that uses it.

- **Board file** — the file you open: one board, in one of the formats chapter 10 lists.
- **Sidecar** — a data file beside the board (`820-00165.obdata`) carrying repair data the board file does not (chapter 11).
- **OBData** — the sidecar format, and the archive of it that ships inside MWV.
- **Part** — a component on the board.
- **Designator** — a part's name on the board, such as `U7000` or `R2815` ; often called a *reference*.
- **Value** — a part's electrical value, such as `10k` or `100u` .
- **Package** — a part's physical footprint, such as `0402` .
- **Not-stuffed** — a footprint deliberately left empty (also called DNP or no-load). No board file states it; an OBData sidecar does.
- **Pin** — a part's electrical connection point, carrying a number and often a name.
- **Pad** — the copper a pin is soldered to.
- **Nail** — a test point: a pad meant for a probe rather than a solder joint.
- **Net** — an electrical connection: every pin and pad joined to the same signal. A board may also call a supply net a *rail*.
- **Alias** — a second name for a net, recorded when a development bridge joins two nets, or stated by a sidecar.
- **Copper** — the conductive geometry: pads, traces, pours and vias.
- **Trace** — a copper run between two points.
- **Pour** — a large area of copper filled across a region.
- **Via** — a plated hole joining copper on two layers.
- **Side** — where a part is mounted: top or bottom.
- **Face** — which side of the board you are looking at.
- **Facing** — whether the board is showing its top or its bottom face.
- **Revision** — a version of a board; a multi-revision file can carry several, and MWV draws them together today.
- **Development bridge** — a part, such as a fitted wire or a zero-ohm jumper, that joins two different nets; MWV hides it and aliases the nets (chapter 2).
- **Mil** — a thousandth of an inch, the unit board measurements use here.
- **Fit** — the zoom at which the whole board is on screen; shown as `1x` .
- **Zoom multiple** — the current zoom as a multiple of the fit, or as a reduction below it (chapter 2).
- **Layer** — a position in the draw order (chapter 2).

- **Element group** — a set of layers a person recognises and can show or hide, such as *Pads* or *Silkscreen* (chapter 2).
- **Drawing group** — one of the ten element groups on the layer strip.
- **Spotlight** — the mode that dims everything the selection does not light (chapter 2).
- **Butterfly view** — both faces of the board in one picture under one camera (chapter 2).

Appendix B. A repair, start to finish

Source of truth: —

Status: Written — replayed in MWV on `samples/BVR/820-02016_V1.3.bvr` with `samples/BVR/820-02016.pdf`, and captured from the running application.

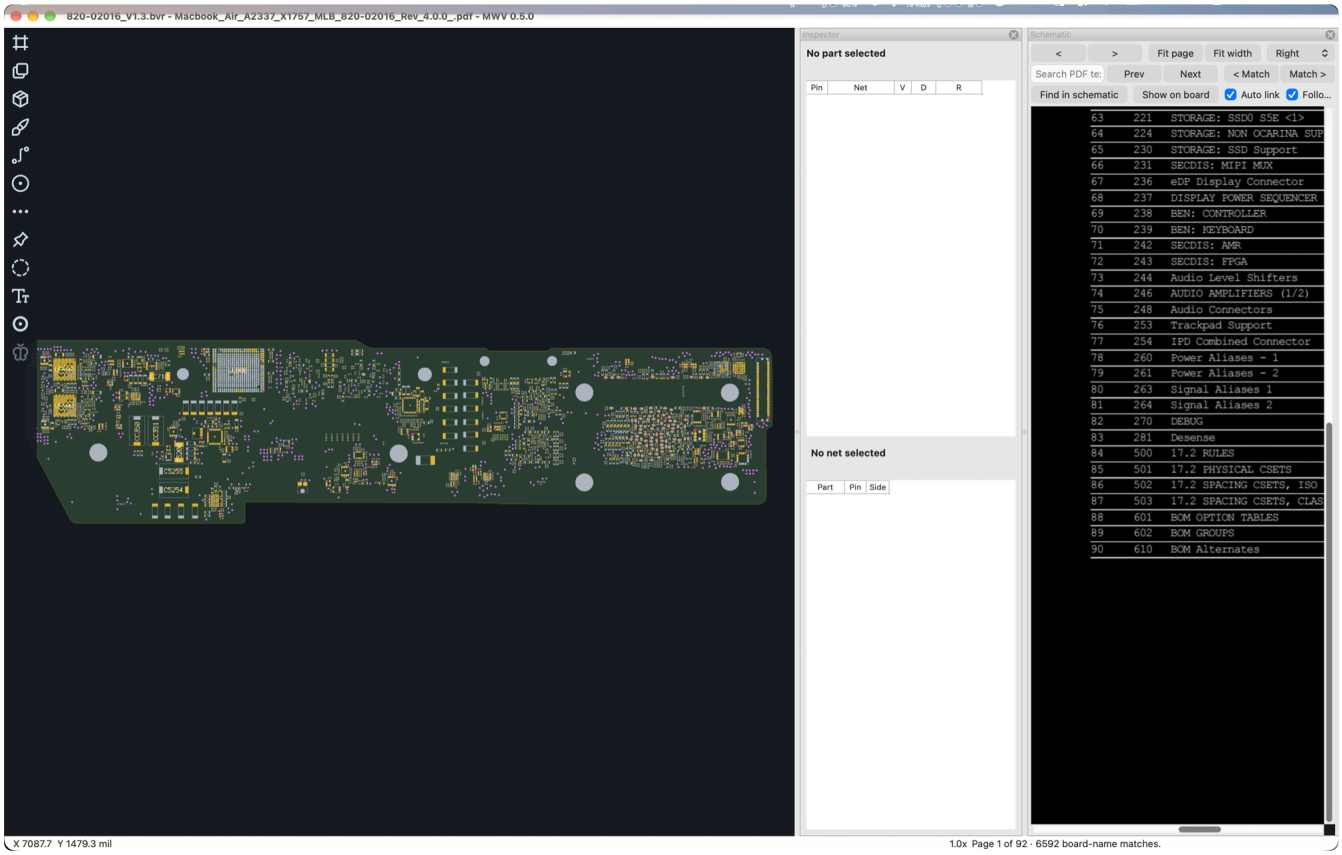
This is the hands-on case: one real board-level fault, followed from the first look to the answer, using only what the earlier chapters describe. It is a replay of a real repair, done in MWV, and each step names the feature it uses.

The fault

A MacBook Air M1 (A2337) logic board, `820-02016`, arrives dead. A USB-C ammeter reads **5.15 V / 0.00 A** on both ports with no battery attached: the board draws nothing. The technician's hypothesis is that the main system bus, `PPBUS_AON`, is shorted to ground, forcing the charger into over-current shutdown.

Open the board

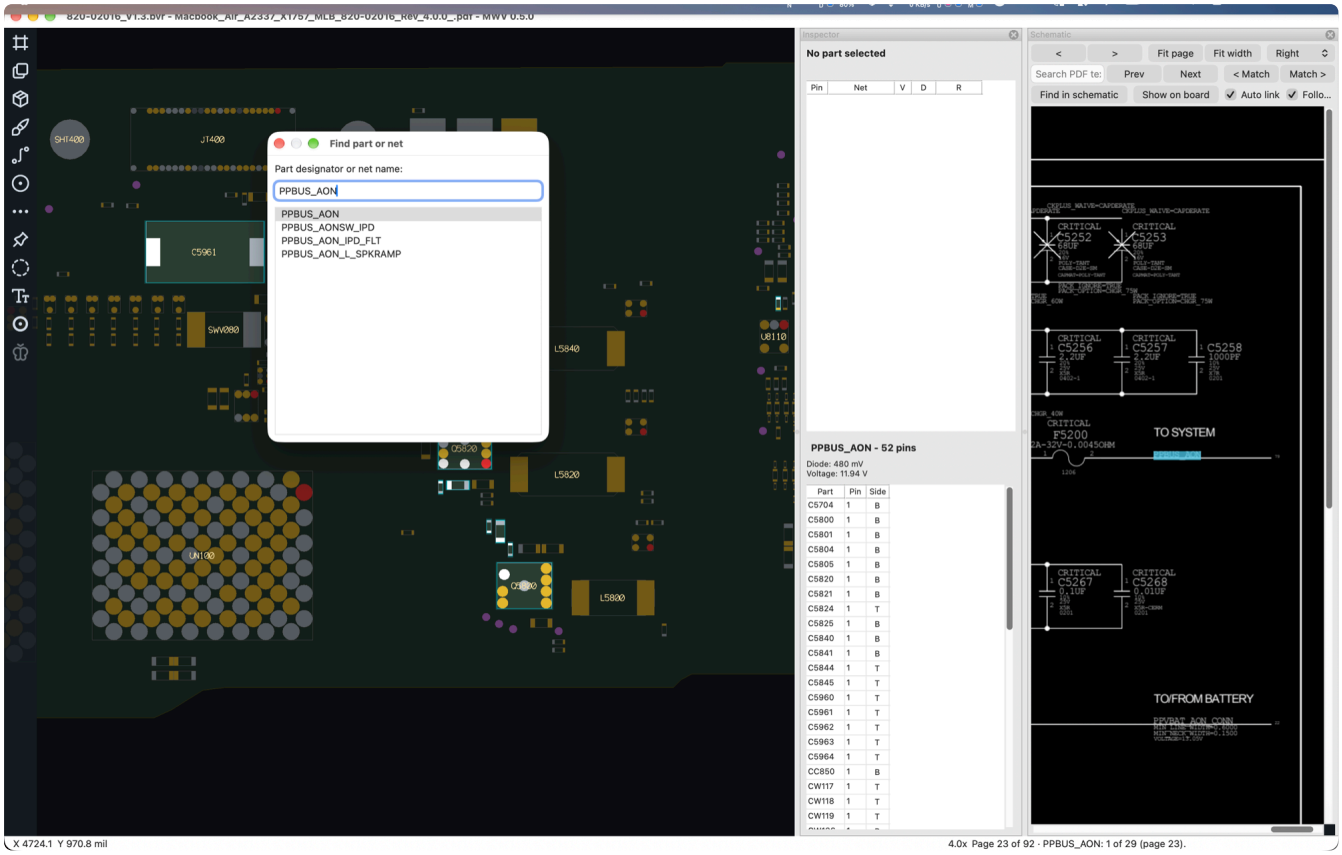
Open `820-02016_V1.3.bvr`. MWV finds `820-02016.pdf` beside it, so the board and its schematic are side by side from the first moment (chapter 1).



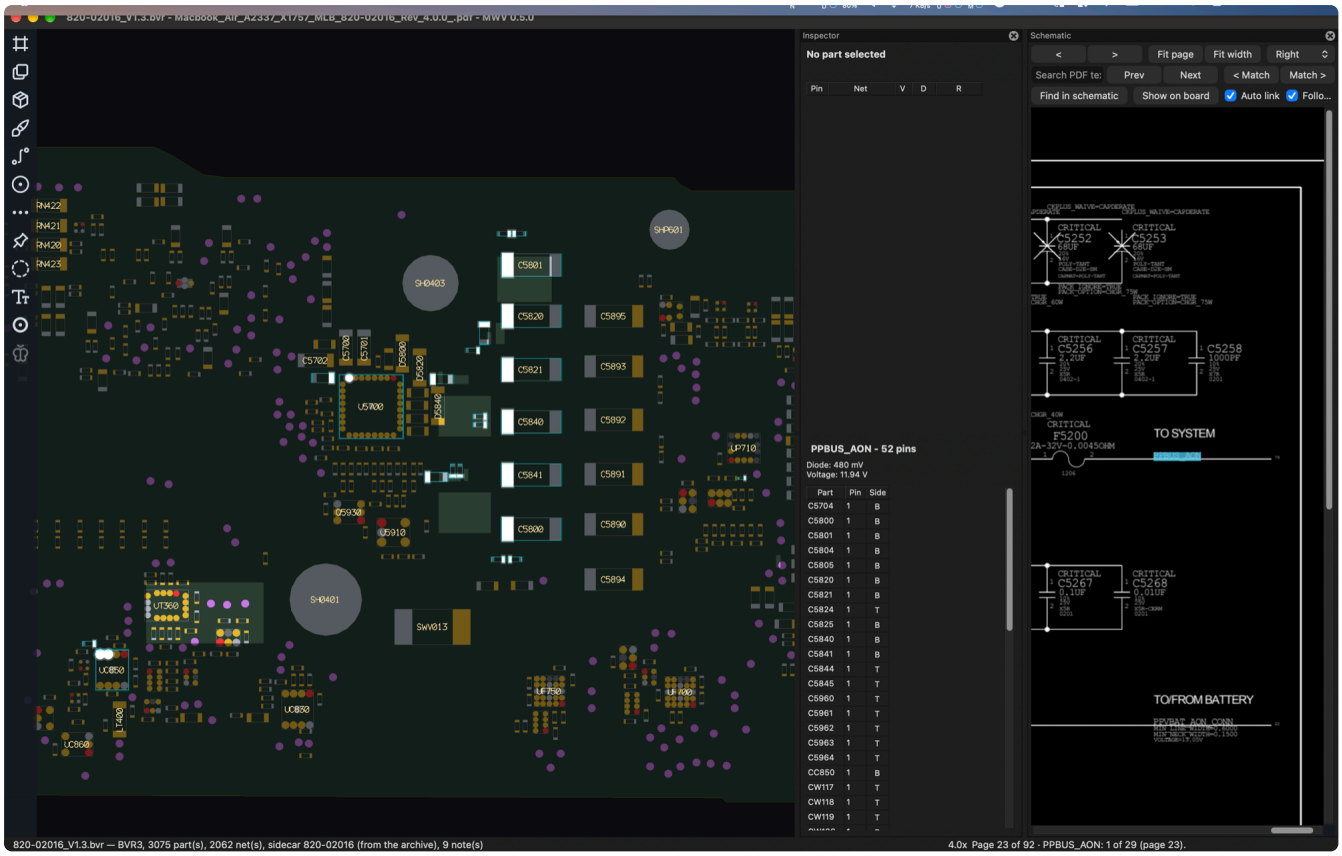
The status strip names the file, the format MWV actually read it as (`BVR3`), the part and net counts, and the repair data applied from the archive.

Step 1 — find the rail, and read its reference

Press `Ctrl+F` and type the rail's name; the matches appear as you type (chapter 5).

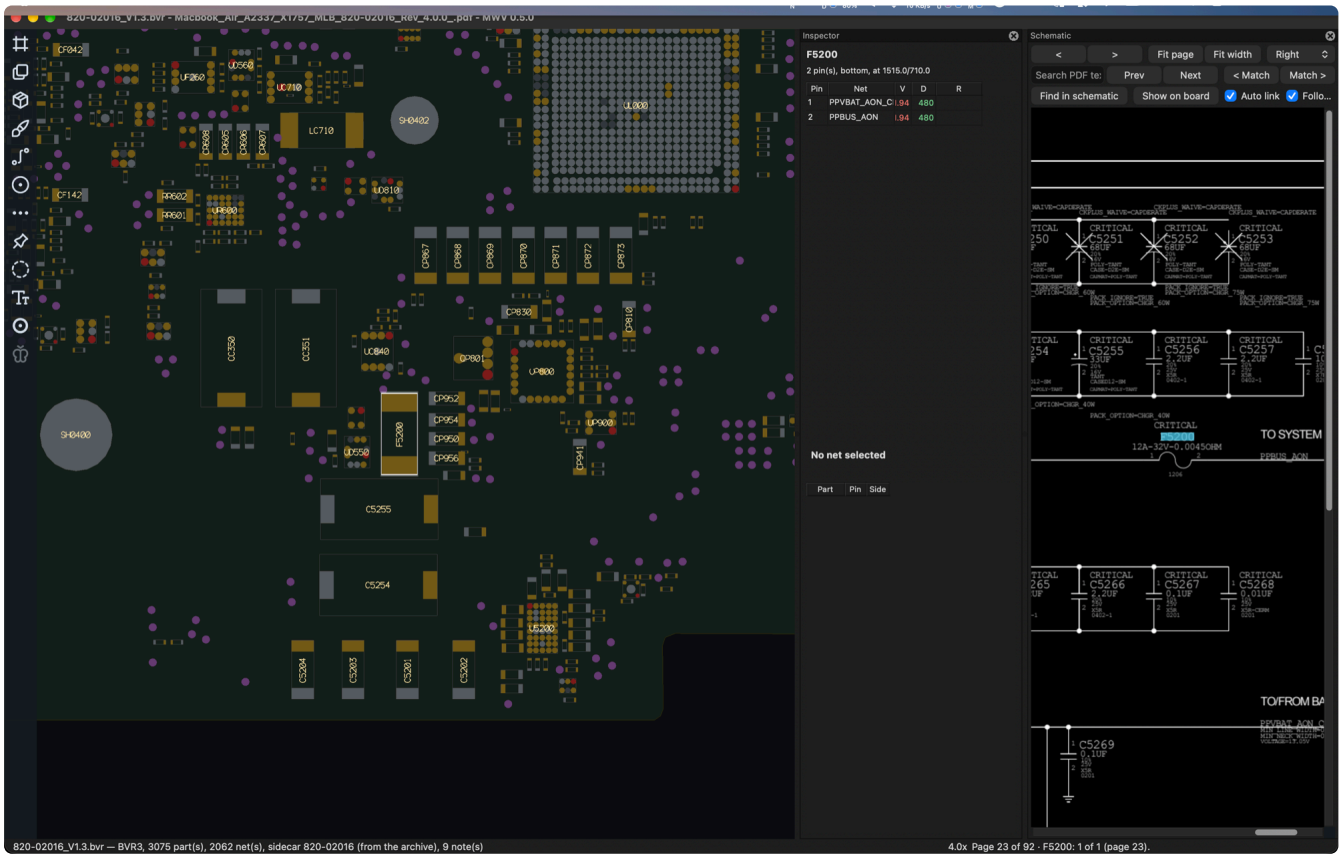


Choose the net. It lights up across the board, the Inspector lists all 52 of its pins with the archive's reference readings, and — because Auto link is on — the schematic jumps to a page where `PPBUS_AON` appears (chapters 2, 3 and 4).

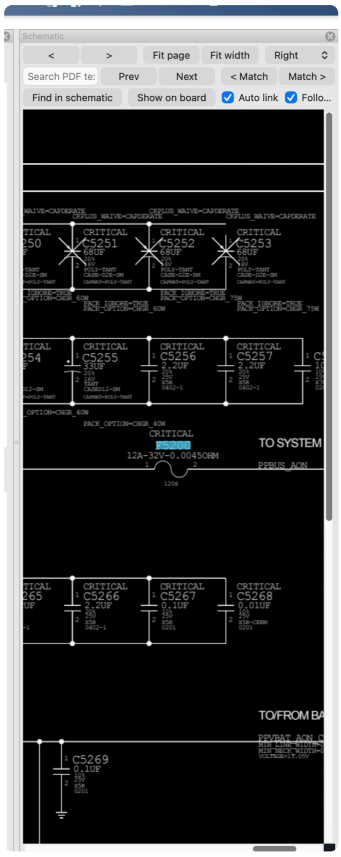


The Inspector reads `Diode: 480 mV`. That is the **reference** for a healthy board; the board in front of the technician measured **0.002 V** — a dead short.

The fuse itself is `F5200`, beside the battery connector `J5150`. Searching a *designator* selects the part rather than the net, and draws its pin numbers. Pin 2 is the rail:

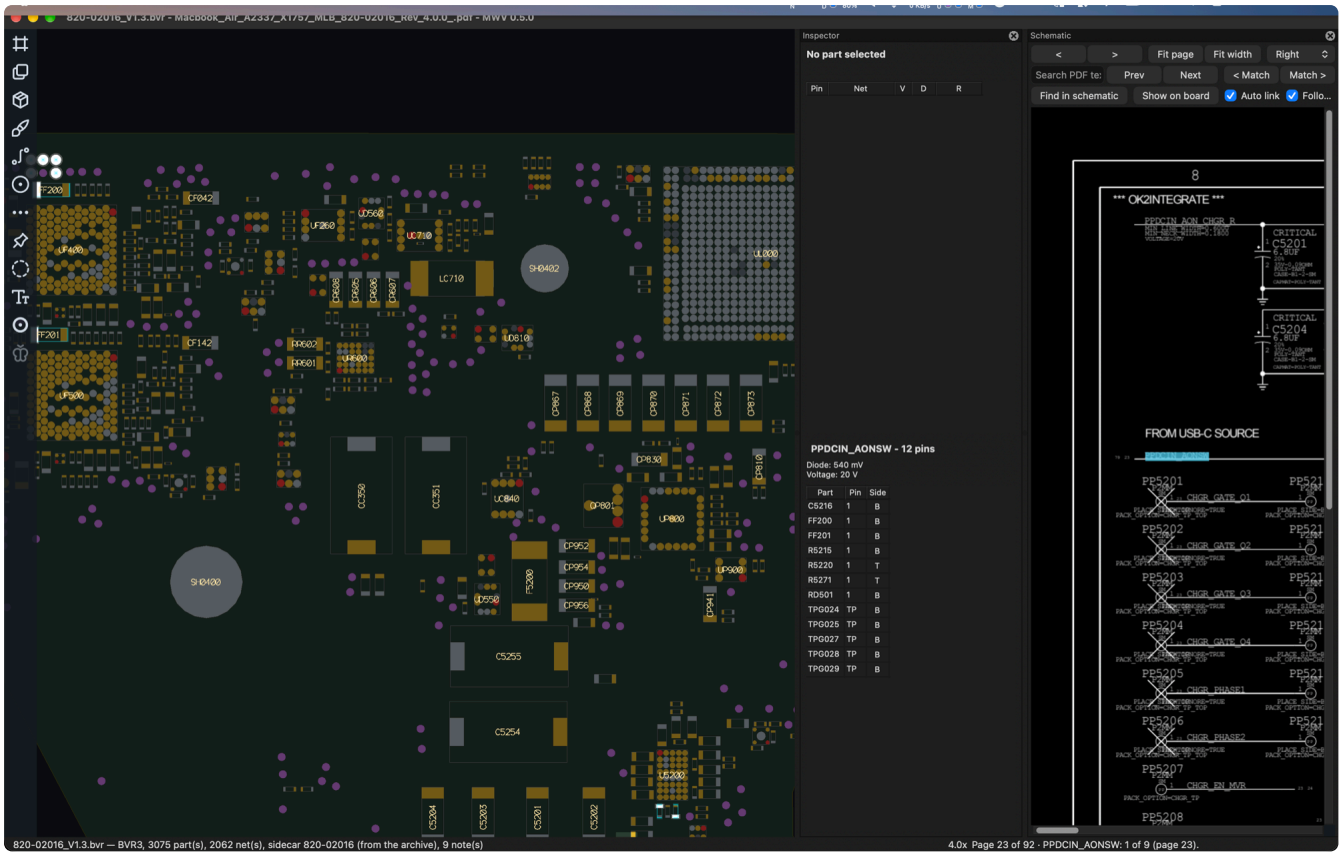


The schematic page the search landed on shows `F5200` in the charger circuit, between `PPVBAT_AON_CHGR_REG` and `PPBUS_AON`:



Step 2 — check the input side

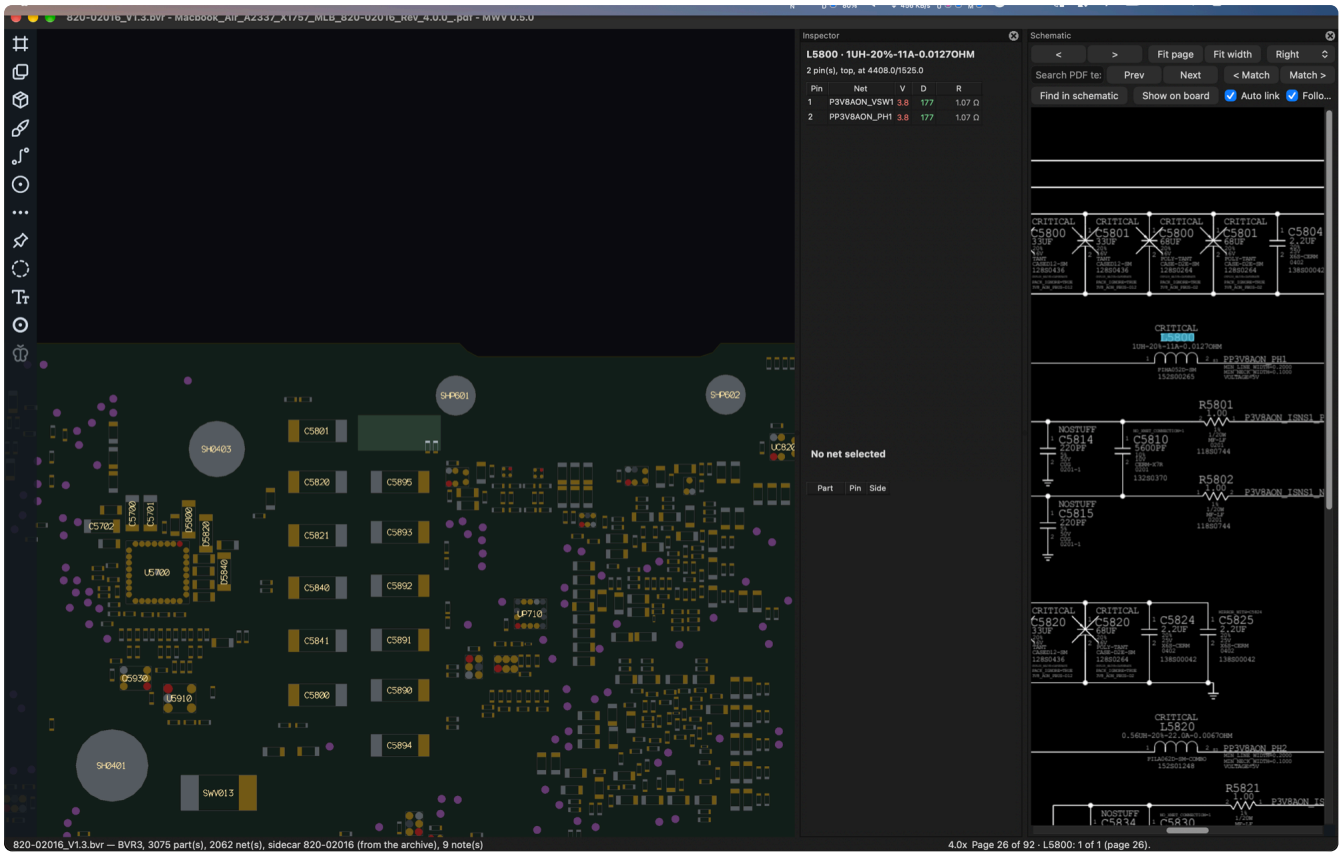
If the short were upstream, the input rail would be shorted too. Search `PPDCIN_AONSW` — the board's name for `PPDCIN_AON`. The Inspector reads `Diode: 540 mV`, normal, and the member list includes `R5220`, the current-sense resistor the technician probed:



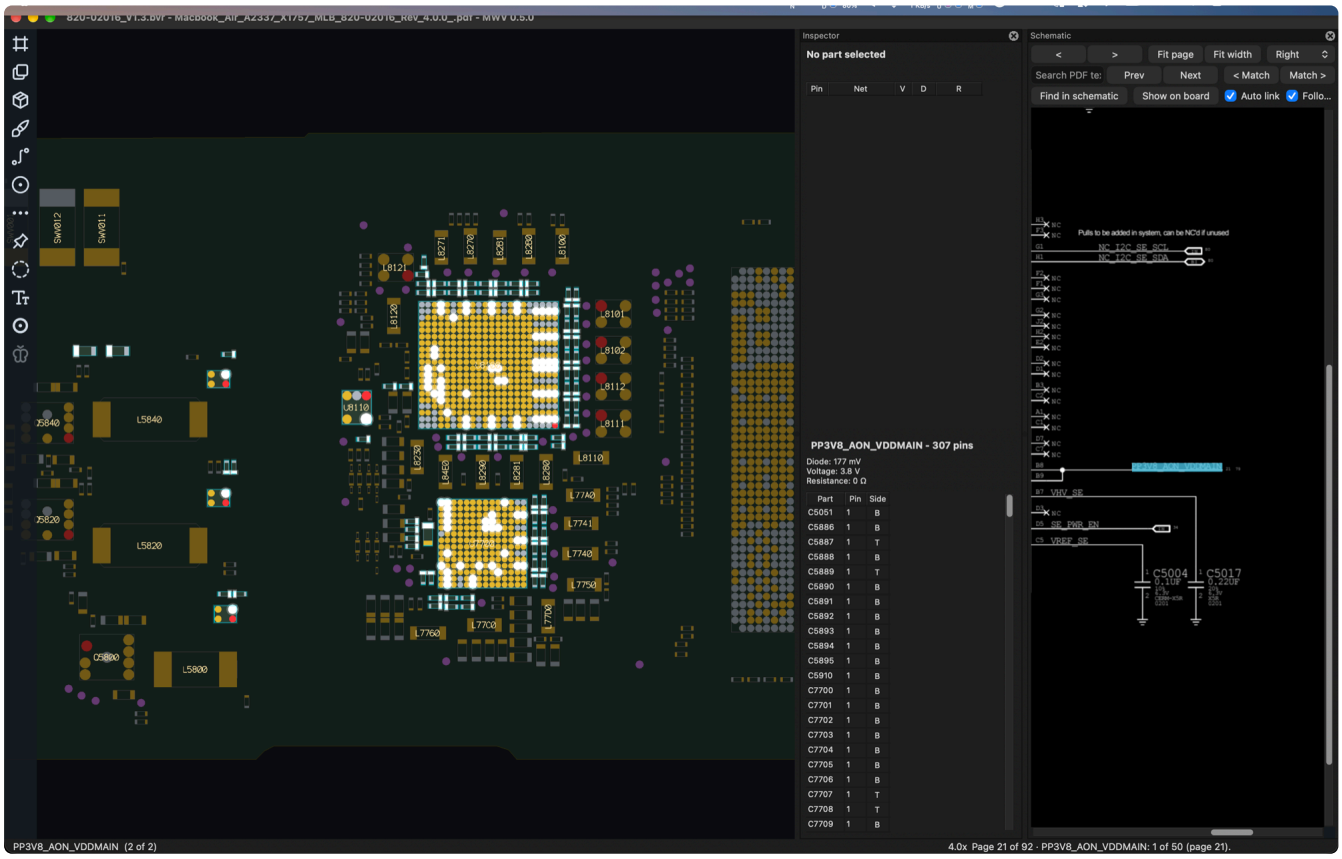
So the input is intact and the short is downstream, on `PPBUS_AON`.

Step 3 — check the 3.8 V rail

The largest load on `PPBUS_AON` is the buck converter `U5700`, which makes the 3.8 V rail through its inductor `L5800`. Searching that designator gives the inductor's value and its two pins — the switch node `P3V8AON_VSW1` and the phase output `PP3V8AON_PH1` — and the schematic jumps to it:



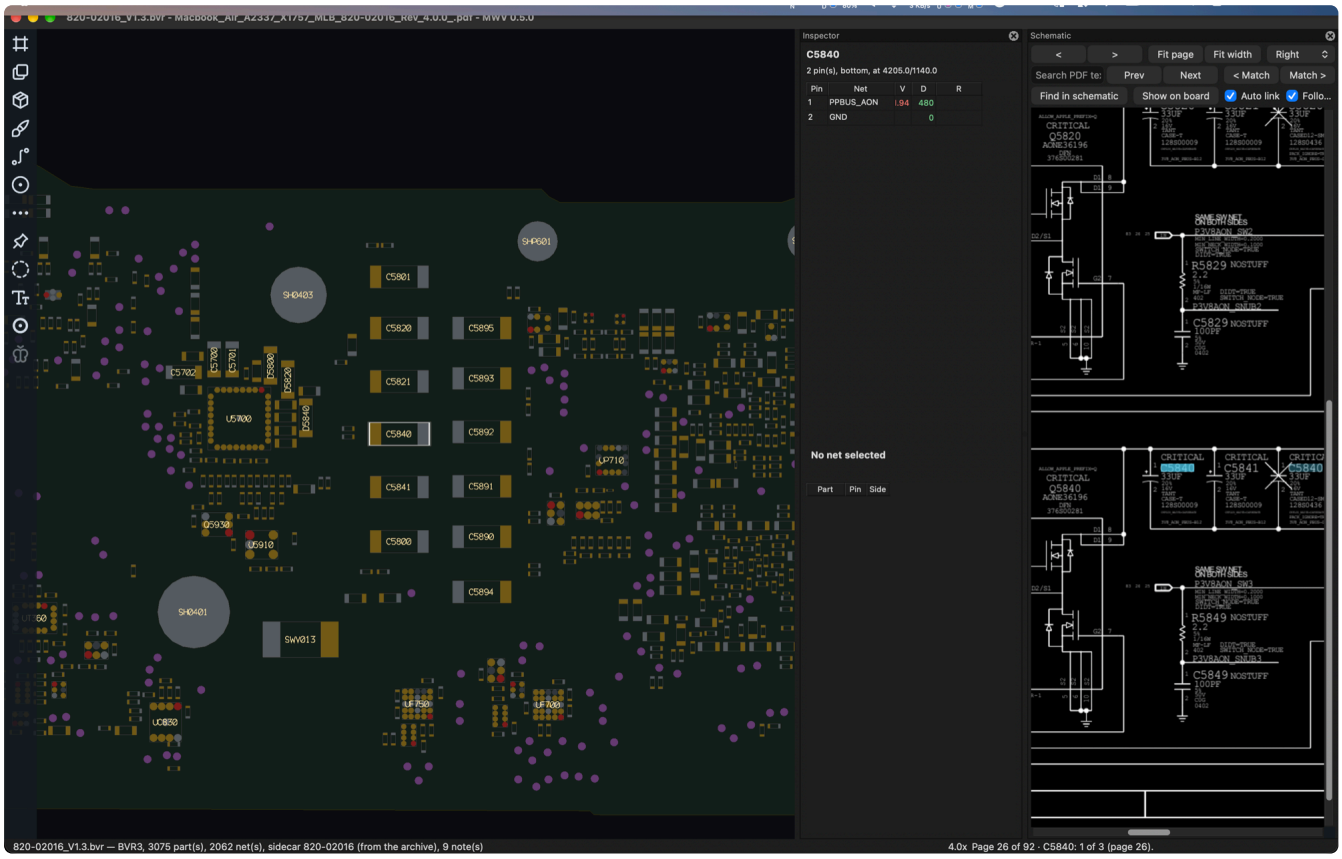
The rail is `PP3V8_AON_VDDMAIN` on this board — its name for `PP3V8_AON`. Searching it shows the rail spanning **307 pins**, with its decoupling capacitors (`C5889` among them), and the Inspector reads `3.8 V`:



The board shows the rail reaching into the master PMU `U8100`. That is the point of the step: if the PMU's high-side switch had failed, this rail would be shorted too. The technician's meter read 0.310 V here — not a short.

Step 4 — isolate the shorted branch

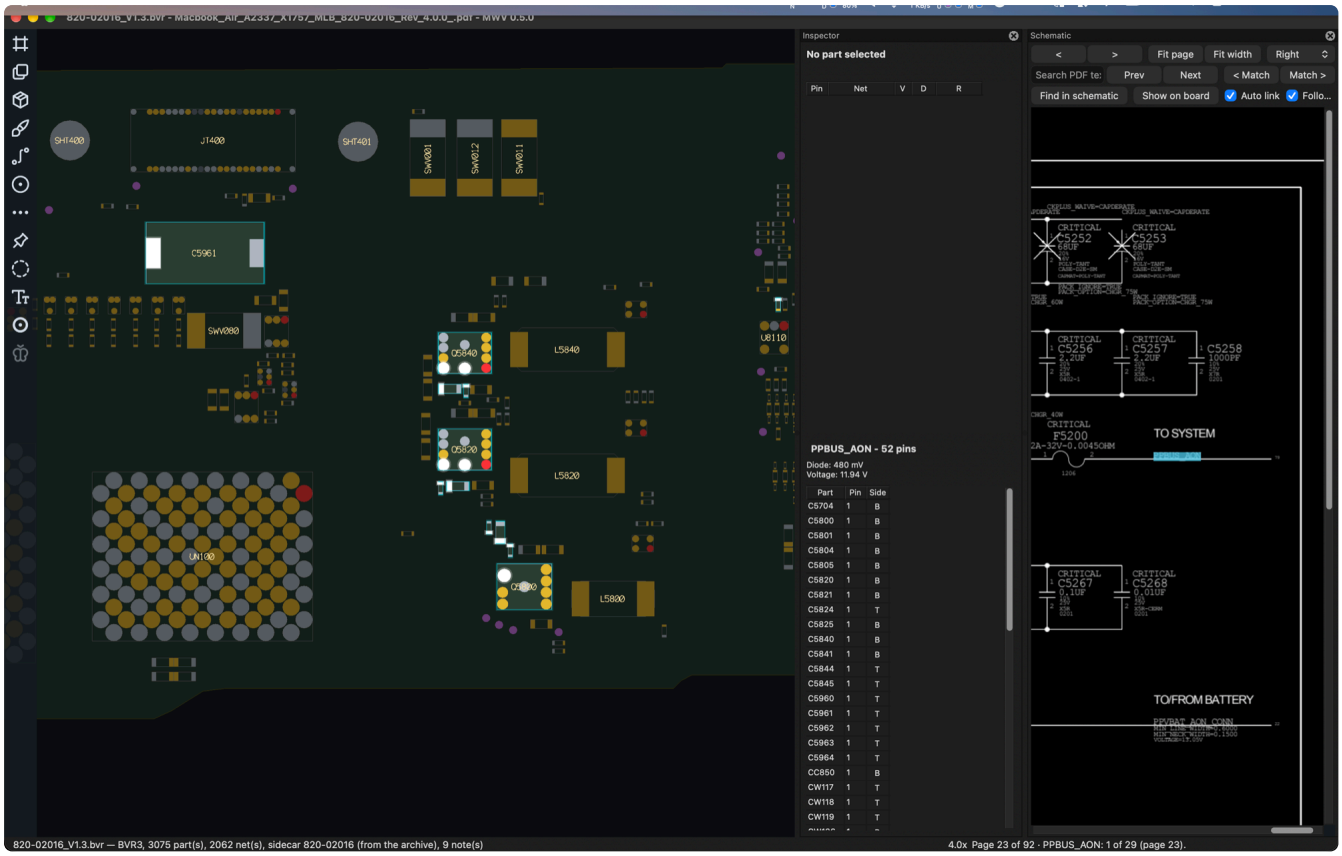
`PPBUS_AON` has dozens of decoupling capacitors, and the Inspector's member list is exactly the list to walk. The suspect is a capacitor; searching its designator selects it, draws its pin numbers, and jumps the schematic to it:



> **A note on the designators.** The repair this appendix replays was recorded on a *different revision* of the board than the one in the repository, and one name in the transcript is simply wrong: the 3.8 V inductor is `L5800`, not `L5700` (there is no `L5700`, on this Air board or the Pro M1). This board's rail names differ too — its `PP3V8_AON` is `PP3V8_AON_VDDMAIN` — and its `C6420` and `J6500` are not present, so the walkthrough uses the names this board actually has (the capacitor in step 4 is `C5840`). The method is identical; only the labels differ.

Step 5 — turn the board over

A rail usually runs on both faces. Press `Space` and the same net is lit on the other side, so the top-side and bottom-side members can both be read (chapter 2):



Step 6 — follow the answer into the schematic

The schematic is not decoration. Every board selection that matches a page jumps to it, and every search you do on the board can be read on the schematic beside it. With Auto link on, the two views stay in step; with it off, **Find in schematic** and **Show on board** move one side at a time (chapter 4).

The conclusion

Remove the shorted capacitor, and diode mode on `PPBUS_AON` returns to **0.525 V** — normal. On power-up the board retests at **20 V / 1.18 A** and boots.

What this case exercised

One repair, and nearly every feature the manual describes: search by part and by net, the Inspector's reference readings, pin-level detail, the layer flip, and the board-to-schematic link. That is the daily loop of chapter 1 — **locate, then read** — with the schematic and the repair data beside you.