MWV — user manual

Edition for MWV 0.25.0 · 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** for MWV. It is written for the people who open a board with MWV, not for the people who build it.

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. It covers only what ships today, 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`**, with its schematic and its repair data. **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** and the repair-job chapters carry figures captured from a running MWV.
- **Applies to.** MWV 0.x. A rendered edition is stamped with the version it belongs to 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.

Contents

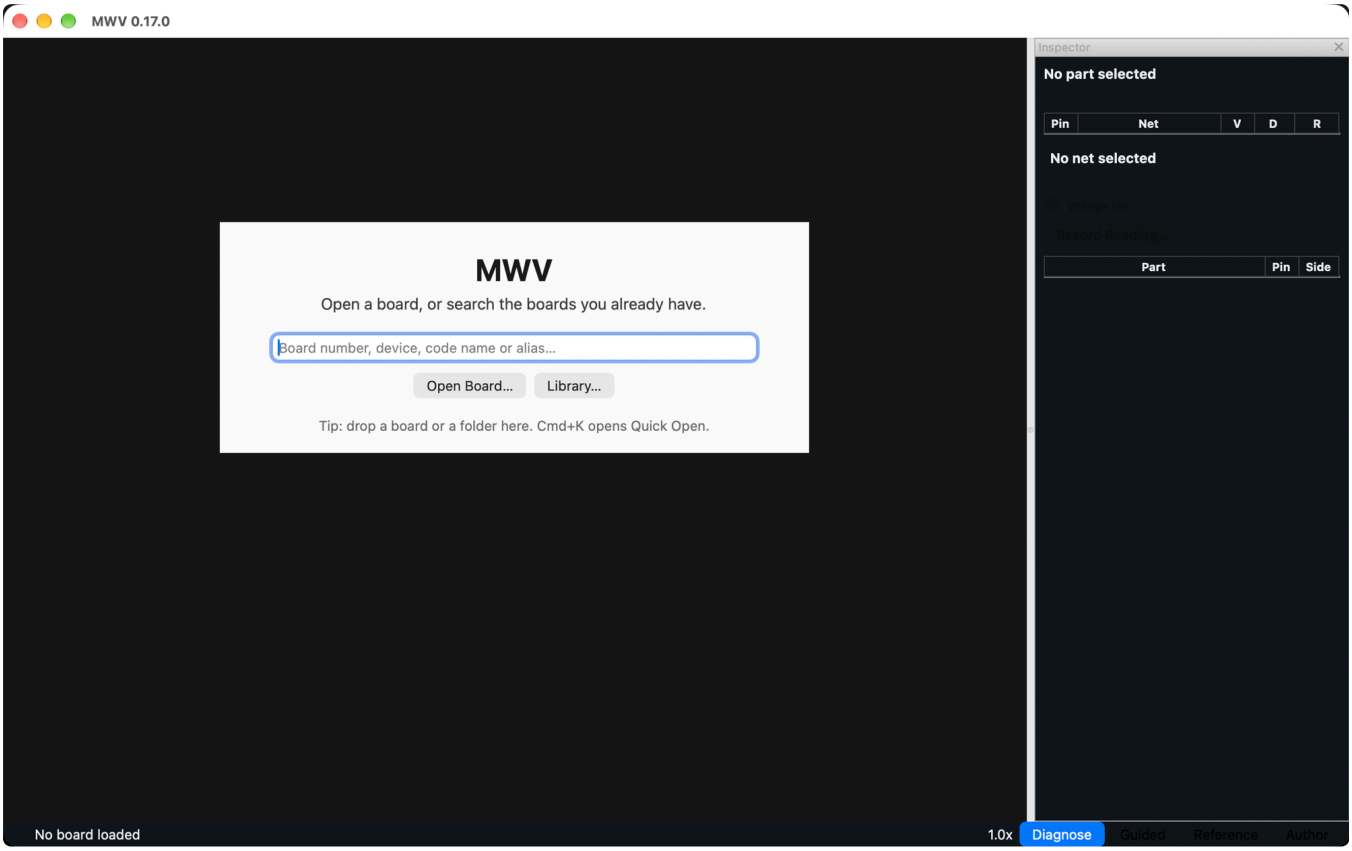
1. Getting started
2. The board on screen
3. The Inspector
4. The schematic PDF pane
5. Finding things
6. Keyboard and the keymap
7. Settings
8. The window
9. Your data on disk
10. Board file formats
11. OBData repair data
12. Repair jobs
13. Troubleshooting
14. The board library

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

1. Getting started

MWV has one view — the board. When no board is loaded it shows a small card with a search field and the two ways in; the card is also where a dropped board, folder or schematic lands. 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.

This chapter assumes MWV is installed and waiting with a board file at hand.



Open a board

There are five ways to open one, and all five 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. A schematic PDF dropped the same way opens in the schematic pane (chapter 4).
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).
5. **File → Quick Open...** (`Cmd+K`) or the empty board's search field, and type part of a board number, device or code name from a folder you have added to the library (chapter 14).

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 pointer shows the **wait cursor** until the board is ready, so a slow machine or a large file says it heard the open.
- The status line shows `<name> — loading...` while it works.
- An open that takes longer than about a third of a second raises a small **Loading** window: the file's name, what the load is doing (reading, importing, applying repair data, building the drawing, preparing the view), a progress bar and the time so far. A fast machine never shows it. **Cancel** — or closing the window, or Escape — abandons the open and leaves the board you had on screen exactly as it was.
- Opening a second board **abandons** the first rather than waiting for it: there is no queue, 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, how long the open took, and every note the importer and the sidecar made (chapter 8). The session's first line names the build configuration, so a **Debug** build — much slower than a release one — is visible rather than guessed at.

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.

The revision is checked too. When the board's file name states a revision (`820-02382-16` is rev 16, `820-00165-A` is rev A, `…_V1.3` is rev 1.3) and the data is for a different revision — or, for the archive, is for **no stated revision at all** — MWV asks the same question before applying it: *"This board is rev 16; the archive entry does not state a revision."* The archive keeps one entry per board number, so this is the usual case for a revisioned board, and the readings may belong to another revision. **Apply Anyway** is remembered for that file content and that board revision; **Open Without the Archive** (or **Skip This Sidecar**) opens the board without the data, and **Cancel** does not open it. A board whose name states no revision is never asked, and a sidecar you placed beside the board that states no revision of its own is treated as yours and applies without a question.

Whenever archive data is in use, the status strip carries a badge for as long as the board is open — `[archive data, rev ?; board rev 16]` — so the readings on screen are never mistaken for ones measured on this exact revision.

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 replaces the board summary with the pointer's reading in fixed columns: pointer X, pointer Y, part, pin and net. Hovering a pin fills all three name columns (`UD3`, `D5`, `+VDDQ_MEM`); a pad, a nail or a part body names what it is in the part column. Over bare board only the coordinates are filled. 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).

The panes remember where you put them, and **View → Focus Board** (`F`) clears them away when you want the whole window for the board. The **Mode** menu and the four names at the right end of the status strip switch between **Diagnose**, **Guided**, **Reference** and **Author**, each with its own remembered arrangement; chapter 8 has the details.

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 13 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

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 100×. 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

A small **badge beside the board** always says which way it is looking: `Top` or `Bottom` (`Top + Bottom` in the butterfly view), then `mirrored` when your own mirror (`M`) is on, then the turn you have given it — `rot 90 cw` after one `R` , `rot 90 ccw` after `Shift+R` , `rot 180` after two. Flip's own left-right reflection is not "mirrored"; it is just what the other side looks like. The badge floats just outside the board's left edge level with its top, follows every pan, zoom, flip and turn, and stays inside the window (over the board's corner when the board fills it). It only reports; clicking it does nothing. It follows the colours of the window chrome.

- **Flip** top to bottom with `Space` (`Ctrl+B` also works). This is how you reach the other side, and it turns the board over the way you would on the bench: the side you see is reflected left to right, with the battery or connector edge kept where it is. 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 no mesh, a flip never disturbs a pan or a zoom, and **Zoom to Fit (`0`) clears the rotation and your own mirror** while keeping the side you are on as that side really looks.
- **The names stay upright.** A part's designator, a net name and a pin number are set across the screen whichever way the board is turned or mirrored, so a name never reads sideways or backwards. A part standing up the board is still named up it.

The layer strip

Down the left edge is a fixed column of thirteen 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 three rows are **modes**, not groups: one switches the **spotlight** that dims the rest of the board while something is selected, the second switches **butterfly view**, and the third switches the **liquid-damage marks** overlay. Each names its state in its tooltip (`Highlight selected: on/off`, `Butterfly view: on/off`, `Liquid damage marks: on/off`). The marks row is a viewing switch only: it hides the chalk without deleting it from the job.

The Traces row is the exception to one click, one group. When the board has inner copper as well as a top and a bottom, clicking it opens a small menu instead of switching every trace: **All traces**, **Show every layer**, **Outer layers**, and then one row for each layer that carries routing. Each row shows the colour that layer's traces draw in and a tick while they are on; an inner layer with no routing is not listed. The topmost layer uses the **Trace** colour from Settings → Colours, and every other layer gets its own colour, spaced evenly around the wheel from it, so no two layers are left as similar shades. **Outer layers** is the inner layers' switch as one control: it is ticked while they are all hidden; ticking it hides them, and unticking it brings them back. The top and bottom rows keep their own switches, so it leaves you with whichever of the two faces is on — top, bottom, or both — rather than forcing both. Unticking a layer row hides only that layer's traces, and ticking one while all traces are off switches them back on. The choices follow the layer, not the board: the same position in the copper stack answers to the same switch on every board, and they are saved with the rest of the settings.

A board with nothing but a top and a bottom keeps the plain one-click toggle: its two faces share the trace colour and there is nothing to separate.

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 board that prefixes its designators with a vendor letter, as Dell and Alienware do (`PC100`, `PRB10`, `PL1234`), is read after that prefix.
- A part your file says is **mounted through the board** is not given a chip package or a tantalum case: a leaded part's pitch is a lead spacing, not a chip land pattern, so it keeps a plain box around its pads. Only a file that states the mount can say this; one that states nothing keeps the guess.

- A large capacitor is drawn as one of the published **tantalum cases** (A to D) where its lands match, with a **grey dot in one corner** at its positive end; 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. A board that states no pad size still names the case its land pitch belongs to: the pitch is what the file gives, and the **Pad scale** setting changes only how large an invented pad is drawn, never which package the part is.

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, and the tantalum's grey positive dot has its own half-transparent colour there. 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

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

A voltage rail is inferred from the net's name — `PP3V3_S5`, `PP0V85_MEMCI_S0_GPU`, `P5V_S4`, the signed `+3V3` and `+1.8V_DDR`, and the planes that spell no voltage such as `PPBUS_AON` — and every pin on it is drawn in the rail colour, a cool teal that cannot be mistaken for copper, ground or a reading. Phone and laptop supply names count too: `VPH_PWR`, `VBAT` / `VBATT`, and `VCC…` or `VDD…` names that carry a voltage (`VDD_1V8`, `VCC3V3`) or the word `MAIN` or `CORE` (`VCC_CORE`, `VDD_MAIN`). A bare `VCC` or `VDD` is not guessed, and neither is a control signal named after a supply: `VCC_3V3_EN`, `VDD_1V8_PG` and `VBAT_SENSE` stay in the ordinary colours. A net is also treated as a rail when the board's repair data (an OBData sidecar or your own local file) states a non-zero voltage for it **and** its name still reads like a supply (it starts `PP`, `VCC`, `VDD`, `VIN`, `VBUS`, `VOUT` or `VBAT`, or has a `DCIN` or `PWR` word, and is not an enable, reset, power-good or active-low `_L` / `_N` signal). A logic signal that merely has a voltage reading, such as `PLT_RESET_L` at 3.3 V, stays in the ordinary colours. The colour is an annotation, not a measurement: it says nothing about the voltage the net actually carries. The flag can be corrected net by net from the Inspector (chapter 3), and your correction wins over both the name and the stated voltage; clearing it returns those pins to their ordinary colours.

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.

Every name is set in **screen** terms: turning or mirroring the view leaves it upright and readable, and a turned name stays turned the same way on screen. The name follows its part around the board; it does not turn over with the picture.

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 — every pad on the net, and the traces that join them, on both faces — 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. A pad's mark is a translucent fill with a glow outside its edge; a trace keeps its own colour and width and gains a glow around the band, so a whole route reads without the traces on one net fusing into a single shape.
- **Click a part's body** to select the **part** instead — its outline, not its pins.

- **Click a trace or a via** on bare board to select its **net**: the whole net lights up as it does for a pad, but no pin is selected. The status line reads `Trace` (with the copper layer's name) or `Via` and the net. A trace is picked within 3 mils of its centre line even when it is drawn thinner, so zoom in on a busy board before you click. Pads, nails and part bodies keep the click where they overlap a trace, and a trace or via that the layer strip or the trace-layer menu has hidden does not answer.
- 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

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 is mostly a reader: it shows, and it reports where you want the board to look. The three things you can change are a net's **reference** reading, from the **Ref V**, **Ref D** or **Ref R** cell of a part's pin list (chapter 11), which is open for editing only in the **Author** and **Reference** work modes; a job reading for the net on screen, with **Record Reading…** (chapter 12), a button enabled only while a repair job is active; and the net's **Voltage rail** flag, which colours its pins (chapter 2).

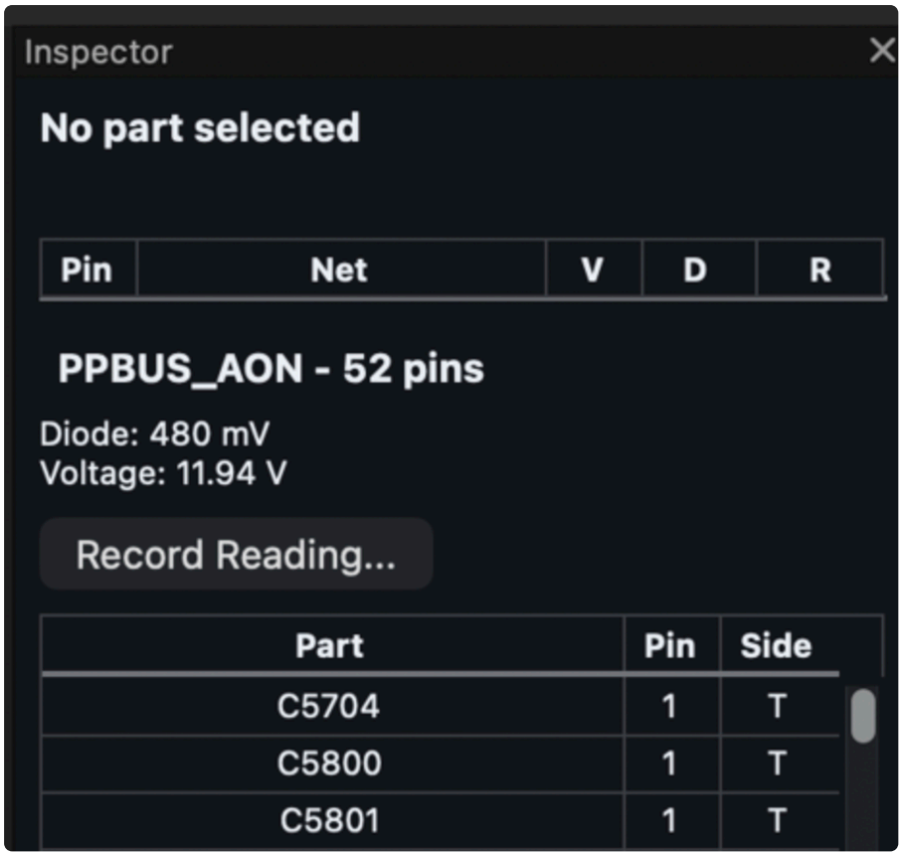
Open and close it from **View → Inspector** (`Ctrl+I`). It is docked to the right, starts reading from the first board on, and follows the selection wherever that selection came from.

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 reference readings, in the columns **Ref V**, **Ref D** and **Ref R**: the known-good values the board's repair data states, not measurements. 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. A measurement you take on the bench is recorded with **Record Reading…**, not typed into these cells.

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.
- Then the **Voltage rail** checkbox — the flag that draws every pin on the net in the rail colour. Its label says where the answer came from: *inferred from the name*, *inferred from its voltage* (the repair data states one) or *set here*. Tick or clear it to correct the guess; the answer is remembered for that board. It is not a reading, and it never appears among the V, D and R values.
- **Record Reading…** below the checkbox starts a job reading on that net (chapter 12); it is disabled when no repair job is active.
- The **member list** gives every pin on the net by part, pin and face.
- A **Test points** list follows it when the net has any: the net's nails, the pins of parts named like test points (`TP101`), and pins themselves named that way. A row takes the board to that test point just as a member row does. **Nearest test point** jumps to the one closest to the selected pin (it is greyed out when the selection is a net rather than a pin, since a net has no place to measure from). The section is hidden for a net with no test points.



The button under the net's readings records a job reading on that net; without an active job it stays disabled. The Ref V, Ref D and Ref R cells in the pin list are the reference readings; they are editable in the Author and Reference modes.

Showing the whole net

Press `X` (Centre on the selection) with a **net** selected and the view **fits the whole net**: the box round all of its pins, with some air, at whatever zoom holds it — zooming out for a long net such as ground, in for a short one, and never closer than the zoom limit. A net of a single pin keeps at least a few hundred mils of context. With a part or a pin selected, `X` still centres on it at 4× or 8×.

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. The boundary between two header cells is a drag handle: drag it to widen or narrow the column, and double-click it to put the column back to its automatic width.

Double-click a part, pin or net cell to copy its contents to the clipboard. It works in every work mode. The part cell is in the Net section's member list; the pin and net cells are in the Part section's pin list. A Ref V, Ref D or Ref R cell opens its editor in the Author and Reference modes; in the other modes it shows a short hint instead (see chapter 11). The side cell is not copied. A short **Copied** badge appears beside the cell so the gesture answers back.

What's next

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

4. The schematic PDF pane

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 first asks the board's local `.obdlocal` sidecar for a `SCHEMATIC` association. The path is relative to the sidecar, so a shared library can move between mount points. If the file is there, it wins without asking.

Otherwise 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.

A guessed pairing is **asked about before anything is loaded**. The question names the PDF's file name and the revision it states (or says it states none) and the board's revision. Accepting it opens the PDF and writes the relative association into the sidecar; declining leaves nothing on screen and the board unpaired. A declined guess is remembered locally so the same candidate is not offered again — you can still pick it with **File → Open Schematic…**.

A **stored** association (or an old hand-picked choice) is yours, so it is never refused; but when the board's name and the PDF's name state **different revisions**, MWV warns first — *"This board is rev 16; the schematic is for rev 15."* — and **Use It Anyway** is remembered for that PDF and board revision. A PDF whose name states no revision is not a disagreement.

File → Open Schematic… picks any local PDF yourself, and a PDF dropped on the window or the schematic pane opens the same way. When it differs from the stored association, MWV asks whether to update it; accepting writes the sidecar. An old hand-picked choice in `settings.json` is migrated into the sidecar the first time it is used. If a 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.

The board library reads the same `SCHEMATIC` key. Its **Schematic** column shows a tick when the associated PDF exists, a warning when the key is present but the file is missing, and nothing when the board has no association. A library scan never invents or stores an association.

Reading the schematic

The pane shows **one page at 100%**, with a compact header of single-glyph buttons: ◄ and ► change page, ⤢ fits the page, ↔ fits its width, ↑ and ↓ step the text-search hits, ‹ and › step the board-link matches, ⌕ jumps to the board selection's first match, and → selects a clicked PDF name on the board. Hover any of them for the words. The search field and the **Link** and **Colours** switches sit beside them; the dock-side choice is on the header's second row. Zoom with the wheel or a pinch and move by dragging, as on the board; the wheel never leaves the page under it. A zoom previews from the page already on screen and redraws it sharply once the gesture settles, so fast scrolling stays smooth. Jumping to a match centres that match in the view, not merely its page, and zooms in when its text is too small to read — never out, and never past the sharp render limit; the mark is padded and edged so it reads on a light or dark page. The status strip names the page and the current result, and lasting notices go to the log.

With **Colours** 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.

Link — **Auto link schematic** in the View menu — is one app-wide switch, **off by default**, toggled in the pane and in the menu. With it off, a board selection changes only the board (⌕ jumps to its first match) and a PDF click only selects the PDF name (→ applies it). With it on, both sides follow immediately. ‹ / › cycle every board-link occurrence across all sheets, and ↑ / ↓ the text-search hits, with the current result count and page in the status strip.

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).

Opening a schematic never freezes the window: the file is read and opened in the background, its first page appears as soon as it is ready, and the index follows. When that takes longer than about a third of a second, the same **Loading** window a board uses appears for the PDF — opening, then *Indexing page n of m*, or reading the cached index. When it follows a board's open it simply carries on from the board. **Cancel** stops the indexing; the PDF stays open and readable, without search and links until it is opened again.

Opening another board while its schematic is already showing keeps the PDF open at the page you were on; MWV relinks it to the new board's names in the background (the Loading window calls this *Linking the schematic to the board*).

What's next

Chapter 5 finds a part or a net by name.

5. Finding things

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

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`
Focus the board (hide the panes)	`F`	—
Mark a glitch (write a diagnostic bundle)	`Ctrl+Shift+G`	—
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**, **Zoom to Fit** and **Focus Board** 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

Settings holds everything about how the board is drawn — the colours, the drawing options, the text and the keymap — and where a glitch mark is written. It is a native window with six pages: **Colours**, **Drawing**, **Text**, **Schematic**, **Keys** and **Diagnostics**. 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 — including the **Liquid damage** chalk — 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 translucent**, on purpose: an opaque hover mark would hide the very pad it points at. The tantalum's polarity dot is the one other translucent board colour, drawn at half alpha so a small bright dot does not out-contrast the case it marks.
- 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. It is a *drawing* scale and nothing more: it never changes which package a part is guessed to be (chapter 2).

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.

Schematic

The **match mark** in the schematic pane: the **minimum match text height** in screen pixels, the **highlight padding**, and the **highlight border** width. When a jump takes the pane to a match whose text is shorter than the minimum height, it zooms in — never out, and never past the page's sharp render limit. Padding and border are **base pixels**, doubled on a HiDPI display, so they keep their proportion on a Retina laptop and an external monitor; the border flips with the page when **Colours** inverts it.

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).

Diagnostics

The **Diagnostics** page holds the folder a **glitch mark** is written to (chapter 13) and the two things a person wants next: see it, and hand it to somebody.

- The picker chooses the folder; **Use the Default** clears it, which puts marks in a `glitch-marks` folder inside the application's own data directory (chapter 9). The line under the picker always says where marks are actually written, default included.
- **Open Folder** shows that folder in Finder, Explorer or your file manager, creating it first if no mark has been taken yet.

- **Send Latest Mark…** packages the newest mark as a **single ZIP file** beside its own folder and shows it, selected, in the file manager. That one file is what you attach to a message or a chat: it holds `mark.txt`, `mark.json` and `view.png` under the mark's own folder name. There is nothing to send before a mark exists, and the message says which key takes one.

Both buttons act on the folder shown on the page, not on the last saved value, so a folder you picked but have not applied yet is the one they use. The same packaging action is on **Help → Send the latest glitch mark…**.

The path is saved with the rest of your settings. Nothing here is uploaded or sent: a mark is a folder of files on your own disk, and nothing is sent anywhere.

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

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 the pointer's reading takes its place, in five fixed columns: pointer **X**, pointer **Y**, **part**, **pin** and **net**. The columns are padded in a monospaced face, so a value changing length never moves the column after it; an empty field is a dash. A pin fills part, pin and net (`UD3`, `D5`, `+VDDQ_MEM`); a pad, a nail, a via or a trace names itself in the part column (a trace adds its copper layer in the pin column); a part body fills the part column alone. 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.

When a CRM is set up and something about the link needs attention, a fifth, short field appears beside them in every work mode: `CRM: 3 pending`, `CRM: offline, 2 pending`, `CRM: 1 not accepted` (chapter 12). It is absent when everything has been sent.

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. Applied archive data adds the badge `[archive data, rev ?]` (with the board's revision when its name states one), and an override kept in the app-data folder reads `(local override, app-data copy)`. **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 arrangement

The panes remember where you put them. Dock a pane to another edge, drag its divider, float it, close it from its title bar, or open it again from the View menu, and the arrangement is written to `settings.json` a moment later, with no dialog and no **Save Layout** command. The next launch opens with the arrangement you left, so a window set up once does not have to be rebuilt every morning.

View → Focus Board (`F`) clears the window down to the board: every pane goes away, and so does the layer strip. The **status strip stays**, because it is the one place that names the part, pin or net under the pointer. Press `F` again, or click the check item, and the panes come back exactly as they were — same edges, same sizes, same floating windows, same open panes. While the board is focused, **the arrangement is not written**, so the empty view never becomes the one that opens next time. The Job menu keeps working while focused; a pane that a load or a job would normally open stays out of the way until you leave the focused view.

View → Auto-save Layout is on by default. Turn it off and changing the window no longer writes the file — useful while trying an arrangement you are not sure about. **View → Reset Layout** returns to the shipped arrangement, forgets every pane position and size **and every work mode's memory**, and writes that state straight away, whatever the auto-save switch says.

Two details are worth knowing:

- **A pane that needs a document appears only when it has one.** On launch the guides, the schematic and the repair job are shown only if the board that opens carries them, so a saved arrangement cannot bring back an empty pane. The Inspector, Layers and the log palettes keep whatever visibility you gave them.
- **A floating pane is never left off screen.** If the window was last used with a second display that is now disconnected, the pane is moved onto the main display rather than lost.

Work modes

A **mode** is a named arrangement of the panes and the surfaces the job in hand needs. It changes nothing about the board itself, and it takes one control away: the Inspector's reference-reading cells are editable only in Author and Reference (chapter 11). The View menu still opens any pane, and the Job menu still works.

Switch with the **Mode** menu or with the four names at the right end of the status strip:

Mode	What it brings up
Diagnose	The board and the schematic side by side, the Inspector, and the Guides and job panes when the board and a job have them. The general-purpose state.
Guided	Built around the repair job: the job pane on its **Checklist**, the Guides, the schematic and the Inspector.
Reference	The schematic and the Inspector alone; the Guides and the job pane are closed, so the reference readings have the column.
Author	The board and the Guides for writing prose, with the schematic and the Inspector; the job pane is closed.

Each mode remembers its **own** arrangement: resize a pane in Guided, go to Diagnose and come back, and Guided is as you left it. The window remembers the last global arrangement as well, and that is what opens next time — a launch always starts on **Diagnose**, whatever mode shaped the layout. **Mode → Reset Mode** returns the active mode to its shipped arrangement without touching the others.

Guided has one action for its checklist, in the **Job** menu and as the **Next Step** button at the top of the checklist: it ticks the first unticked step, selects the next one, and centres its `[n:]` or `[p:]` target on the board (turning to the right side when the target is on one). A step with no reference is still ticked and the next is selected; there is simply nowhere on the board to go.

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. The CRM commands are in **Job → CRM** (chapter 12).
- **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.
- **Job**: **New Job…**, **Open Job…**, **Record Reading…**, **Record Changed Part…**, **Record Liquid Damage…**, **Add Note…**, **Next Step**, **Set Aside** / **Resume Job**, **Done…** and **Rebind to This Board…**, enabled and disabled by the active job (chapter 12).
- **Mode**: the four work modes as a radio group — **Diagnose**, **Guided**, **Reference** and **Author** — and **Reset Mode**.
- **View**: the checkable panes — the two log palettes, **Layers**, **Inspector** (`Ctrl+I`), **Schematic** (`Ctrl+J`), **Guides** (`Ctrl+G`) and **Repair Job** — and **Auto link schematic**, **Board data…** and **Open import notes on load**, then **Flip Board**, **Mirror Board**, **Butterfly view**, **Liquid damage marks** and **Zoom to Fit**, then **Focus Board**, **Auto-save Layout** and **Reset Layout**. The Flip, Mirror, Zoom to Fit and Focus Board labels carry your own primary key (chapter 6).
- **Help**: **Mark a glitch…** (`Ctrl+Shift+G`), **Send the latest glitch mark…**, and **About MWV** (on non-macOS; on macOS About is in the application menu). Both glitch commands are described in chapter 13.

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

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 or a guide (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 window arrangement and its auto-save switch, the schematic pane's own preferences (Link, Colours, the docking side, legacy hand-picked fallbacks and declined guesses), the voltage-rail flags you corrected (chapter 2), the folder a glitch mark is written to (chapter 13), how many days a finished CRM job keeps its local copy (chapter 12), the per-board answer about the OBData archive, and any remembered "apply anyway" answers for a mismatched sidecar (chapters 2, 4, 6, 7, 8 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.
Repair jobs	`jobs/<ulid>.json`	One file per repair job: its number, board identity, readings, changed parts, checklist progress, notes and journal. Never beside a board (chapter 12).
Board library	`library/index.json`	The folders you added, one row per indexed board, the four kinds of name, your aliases, favorites and availability flags, and the recently opened boards (chapter 14). Metadata only: no board geometry.
Glitch marks	`glitch-marks/<stamp>-<board>/`	One folder per mark you took: `mark.txt`, `mark.json` and `view.png` (chapter 13), plus `<stamp>-<board>.zip` beside a folder once you have packaged it for sending. Delete a folder or its archive you no longer need; nothing else refers to them.
Local override	`<board stem>.obdlocal` beside the board	The reference readings and guide corrections you edited, in OBData syntax, with a creation timestamp when MWV made the file (chapter 11). Never the bundled archive.
Local override, read-only board folder	`overrides/<board stem>.obdlocal`	The same file, kept here **only** when the board's own folder could not be written (a USB stick, a share). Read over any override beside the board; View → Board data… says so (chapter 11).

The app logs the paths it resolves — the version, the log file, the settings file, the schematic cache, the repair jobs directory and the library index — 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**. The application reads your board files and their sidecars; the one file it creates beside a board is the `*.obdlocal` from an edited reading or a corrected guide (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. The only thing in MWV that uses the network is the optional CRM link (chapter 12), and only once you set it up.

Resetting

- Deleting `settings.json` returns every setting to its default on the next start. You lose your colours, options, keymap, the window arrangement, the voltage-rail corrections, the per-board OBData answers and the remembered sidecar-mismatch consents, so those boards and sidecars are asked about again. **View → Reset Layout** forgets the arrangement without touching anything else.
- Deleting `pdf-index/` costs only a rebuild of the schematic text indexes.
- Deleting `mwv.log*` costs only history.
- Deleting `jobs/` or one file in it deletes those repair jobs. There is no undo and no in-app delete in this version, so treat the directory as the record.
- Deleting `library/index.json` forgets the library: its folders, aliases, favorites, availability flags and recents. The boards themselves are untouched; add the folders again to rebuild the index (chapter 14).
- Deleting `glitch-marks/` deletes the marks you took. Nothing else uses them, and the next mark simply starts a new folder.

What's next

Chapter 10 lists the board formats MWV can read.

10. Board file formats

MWV registers **eighteen** board formats; **fifteen** of them really import, and the other three 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
ALLEGRO_BRD (Cadence Allegro)	`.brd`	Yes
MENTOR	`.cad`	Yes
VAYOPRO_ASC	`.asc`	Yes
EAGLE_BRD	`.brd`	Yes
ALTIUM_PCB	`.pcbdoc`, `.cmpcbdoc`, `.cspcbdoc`	Yes
BV	`.bv`	No — recognised, not read
CAE	`.cae`	No
CST	`.cst`	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.

An **Allegro** board (v16.0–17.5) opens with its parts, pins, nets, traces, vias, copper layers and silkscreen. Its values and device types come from the board's own text. When the edge is only a drawing frame rather than a closed boundary around the parts, MWV draws a rectangular outline around the parts and says so in the import notes. Allegro 18 and newer are refused with a message asking for a v16–v17 save.

An **EAGLE** board (6.0 and newer, the XML format) opens with its parts, pads, pins, nets, traces, vias, silkscreen and the board's dimension-layer outline. A mirrored footprint's pads are placed on the face they are actually on. EAGLE 5 and older wrote a binary file; it is refused with a message telling you to re-save it from EAGLE 6 or newer.

An **Altium** board (`.PcbDoc`) opens with its parts, pads, pins, nets, traces, arcs, vias, copper fills, silkscreen and the board outline, and its copper stack is read from the file's own layer chain. Copper **regions** (large pours) are not read yet, and a note at the top of the board says so. The ASCII `.PcbDoc` export is recognised and refused: open the binary file Altium writes instead.

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 through its detector: MENTOR is registered before CAD and also claims `.cad` , so a GenCAD file whose `$HEADER` is beyond the first 512 bytes still cannot be recognised and reaches Mentor's importer, which will refuse it. A short binary prefix before `$HEADER` is skipped, so the wrapped GenCAD sample in `samples/MENTOR` opens as CAD.
- 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 has its own quirks — its container, its encoding, the geometry it guesses, and where it departs from its reference. An import that looks wrong usually says why in the board's notes (**View → Board data…**, chapter 8) before you conclude the board is bad.

What's next

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

11. OBData repair data

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. A local file MWV creates opens with a small header: the format version, the board's identity when it can be spelled there, and the UTC time the file was made. The timestamp is for you and for another OpenBoardData-aware tool; MWV does not read it.

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.

The archive's entries state **no revision**. For a board whose name states one, MWV therefore also asks whether to use the data for that revision (chapter 1), and shows `[archive data, rev ?]` in the status strip while it is in use.

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 **Ref V**, **Ref D** and **Ref R** columns hold the board's *reference* (known-good) readings. They are shared repair data, written to the board's local file, so they are an authoring job rather than a measurement, and they can be edited **only in the Author and Reference work modes**. In Diagnose and Guided (the mode MWV starts in) the cells are read-only: a double-click shows a short hint in the status line and next to the cell, and a measurement you take is recorded with **Record Reading…** in the Inspector (chapter 12). Switching mode unlocks or locks the cells at once.

In Author or Reference mode, **double-click a cell in the Ref V, Ref D or Ref R column** of a part's pin list; the cell becomes a text field with its current value selected. While you type, a small note under the cell shows how MWV reads the text (`= 573 mV`, `= 3.3 V`, `= 4.7 kΩ`, or the reason it cannot): it is the same reading Enter will store. 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 shown in **millivolts**. Type what the meter shows: a number with a decimal point is volts (`0.573` is 573 mV), a whole number is millivolts (`573`). A decimal of 5 or more is refused rather than guessed, so write the millivolts as a whole number or give the unit (`0,573 V`, `573 mV`). `OL` means the meter found no conduction and is a reading, not a blank.
- **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 is never written.** If the board's folder cannot be written — a USB stick or a share mounted read-only — the edit is **not lost and nothing is written beside the board**: MWV keeps the override in its own `overrides` folder in the app-data directory instead, and from then on reads that copy as the board's override (it takes the place of one beside the board, which it was started from). **View → Board data…** shows the file's path and says that it lives in the app-data folder. 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.

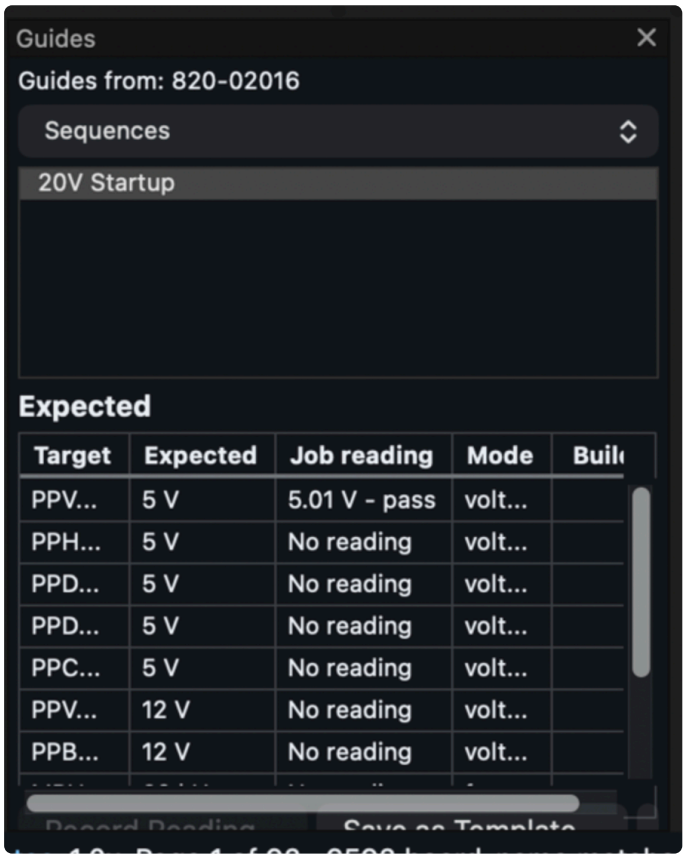
The board you are looking at updates at once, without reopening. If the local override on disk is not the one the board opened with — it appeared, vanished, was renamed or was skipped for an identity mismatch — MWV asks you to reopen the board before editing rather than write into a file whose other 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 (chapter 12), not to the sidecar. 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.

- **A job reading beside an expectation.** With a repair job active (chapter 12) the table adds a **Job reading** column: the latest reading on that target, preferring one whose quantity the expectation can be judged against, its value and its verdict against that row. **Record Reading…** under the table starts a reading from the selected row, with its target, expectation and mode filled in; it is disabled while no job is active. A value stated for a build (`8.2V Air 12.2V Pro`) is marked build-specific and is not compared; the board's notes say so once at load.
- **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.



*With a job active, each expectation carries the latest reading and its verdict. **Record Reading…** fills the dialog from the selected row; **Save as Template…** turns the note on screen into a reusable checklist.*

A guide is reference material: MWV records no progress in one — progress, readings and changed parts live in a repair job (chapter 12). 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. If two headings, or two notes of one heading, that would be written to your local file share a title, **Save** is refused until one is renamed; a duplicate that comes only from the imported guide still merges by first match and does not block an unrelated correction.
- **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**.
- **Insert a reference without typing it.** Hold **Cmd** (macOS) or **Option**/**Alt** (everywhere; **Ctrl**/**Alt** on Linux and Windows) and click a pad, a part's body, or bare copper or a via on the board. The click also selects what it landed on, and `[p:PART:PIN]`, `[p:PART]` or `[n:NET]` is inserted at the prose caret, replacing any selection; the findings re-render. With no note selected there is nowhere to put it, and the status strip says so rather than dropping the click silently.
- **Insert a measurement without typing its grammar. Insert Expectation…** opens a small form for the target, the mode, the value and an optional tolerance. It starts with whatever is selected on the board in the target, which stays editable; a typed name can be a bare net or part, or the `[n:NET]`, `[p:PART]` or `[p:PART:PIN]` spelling. The line is inserted on a line of its own. If the board has no such name, or the reader could not be sure of the value, the form stays open and says why.
- **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: **nothing is downloaded**, and the archive is never written. (The one network feature in MWV is the optional CRM link in chapter 12, which has nothing to do with repair data.) The only board-side file MWV creates is the `*.obdlocal` from an edited reading or a corrected guide.

What's next

Chapter 12 records a repair job — the readings, parts and progress that belong to one attempt at fixing a board. Chapter 13 is what to do when something does not work.

12. Repair jobs

A repair job is MWV's record of **one repair attempt on one board**: the readings you take, the parts you change, the steps you tick and the notes you write. It is deliberately separate from the board's shared repair data. A reading recorded in a job never changes a sidecar, and editing a reference reading in the Inspector never creates a job.

Starting a job

Open the board, then choose **Job → New Job…**. The dialog asks for:

- **Job number** — required. This is the number your workshop already uses for the repair, and it names the job in the pane, the menus and **Open Job…**. It is unique and cannot be changed afterwards, so check it before pressing **OK**.
- **Title** — optional, for a phrase such as "No boot".
- **Customer reference** — optional, and opaque: MWV never interprets it.
- **Template** — optional. **Choose…** starts the checklist from a saved template instead of importing the board's guide lines (see *Reusable templates* below).

The job is bound to the board by its **name, revision and content**, not by a folder. Opening the same board later finds the job while it is unfinished; moving or copying the board file keeps it. If the board that opens does not match the recorded key — a re-export, an edit, or another board or revision at the path — the job pane says so and offers **Rebind…**; only you can say whether it is the board the job was recorded against.

A new job also imports the board's guide lines as **checklist steps**, in guide order. A step you do not need is removed from the job; the guide itself is not changed.

The job pane

The pane docks along the bottom of the window and is shown when a job is found or started. Its header names the job, its state and the counts, and carries:

- **Set Aside** and **Resume** — one board has one active job at a time; setting a job aside keeps everything and starts nothing, and only an active job can be set aside.
- **Done…** — closes the job with an optional sentence, removes it from the active lookup and keeps it in **Open Job…**, where it appears in a **Close-out** column.
- **Rebind…** — appears when the board that opened does not match the job's recorded board key.

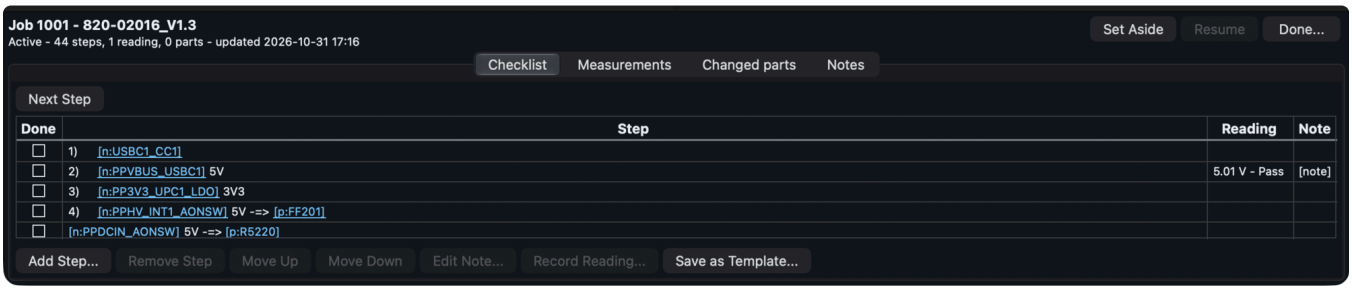
Starting a job and opening an earlier one are in the **Job** menu, as are the pane's actions when it is closed or narrow.

The five pages are **Checklist**, **Measurements**, **Changed parts**, **Notes** and **Marks**; a job linked to a CRM repair also shows a **CRM log** page before Marks. The **Job** menu carries the same actions when the pane is closed. A note is rendered as multiline markdown in the reader beside its row: headings, bullets and emphasis are laid out, and text in backticks is a link to that part or net on the board.

The checklist

A checklist step is a line from a guide or a template, and the page makes it a working list:

- **Tick** a step when it is done; the tick and its time are kept in the job. **Enter** on a selected step does the same and moves to the next line, so a checklist can be worked from the keyboard.
- **Next Step** (also in the Job menu, and most at home in **Guided** mode) ticks the first unticked step, selects the next one and centres its target on the board. It is the one action for working a checklist front to back without hunting for the next line.
- **Edit Note…** writes your own prose about the step — what you saw, which variant applies — and the row is marked `[note]`. It is separate from the guide note the line came from.
- **The Reading cell is the fast way to record a value.** Double-click it, type the reading and press Enter. A bare number is in the mode's unit; a unit the mode can carry is accepted too (`450mV` , `80k` , `24MHz`). The value goes against the target and expectation the step line states (a plain task with no reference uses the board selection), and the row shows its value and verdict. Double-clicking a row that already has a reading corrects that reading — the journal records the correction — rather than adding a second one. Clicking away with the cell empty cancels the edit and records nothing.
- **Record Reading…** opens the full dialog for the selected step when you want the meter and note fields or a different target. The target and expectation the step line states are filled in; the reading is a normal job measurement, and the step shows its value and verdict, or `[reading removed]` if you later delete that reading.
- A step's `[n:NET]` , `[p:PART]` and `[p:PART:PIN]` references are links: click one to select and centre it on the board, just as in the Guides pane. A name the board does not have stays visible and muted; clicking it says so in the log.
- **Move Up** / **Move Down** reorder the steps; **Add Step…** adds one by hand, and **Remove Step** takes one out of the job without touching the guide.
- **Save as Template…** saves the checklist as a reusable file (see below).



*A checklist step holds its own **Reading** and **Note**; the reading is judged against the step's line, and **Enter** ticks the selected step and moves to the next. The buttons wrap when the pane is narrow.*

Readings and changed parts

- **Record Reading…** asks what the reading was taken on (a net, a part, or one pin of a part), the mode, the value, a meter name and a note. The unit follows the mode, and the board state is derived from it: diode and resistance readings are taken unpowered, voltage and frequency readings powered. The verdict is judged by the guide's stated tolerance, or the specification's **±5%** when it states none.

- **Record Changed Part…** asks for the part, the value before and after, the reason and the **donor**. The donor is free text — `C7001 from donor 820-02100-A` — because MWV does not claim a donor board it has not indexed. The part selected on the board pre-fills the designator and its current value; if the part lies inside a liquid-damage mark, the reason pre-fills `liquid damaged`.
- A reading or a changed part is saved as soon as the dialog closes. To fix a mistake, select the row and remove it; MWV asks first, and the journal records the removal.

Liquid-damage marks

Liquid damage is often visible only before the board is washed: corrosion under a part and a dried trail along an edge disappear with the residue that revealed them. A mark records what you saw, so a later probe, replacement or post-cleaning check can return to the spot.

Job → Record Liquid Damage… arms the chalk tool. It is enabled only while an active job is recording this board, because a mark is this repair's observation and is stored in the job — never in the board file, a sidecar or the bundled archive. While the tool is armed:

- **Left-drag paints.** The mark is soft chalk, not a polygon: its density builds where you draw over the same area again, and two marks that overlap both stay visible.
- **Right-drag pans the board.** Every other viewing gesture is unchanged.
- **Esc finishes the mark** and disarms the tool. The Job item toggles it off too.
- **Cmd+Z** (Ctrl+Z on Linux and Windows) or **Backspace** removes the last stroke of the session while you are still painting.

A paint session is one mark. It is dated, records the face it was painted on — a mark on the top does not appear in the bottom view — and can carry a free-text note. Flipping the board or switching butterfly view finishes the session, so a mark never belongs to two faces.

The job pane's **Marks** page lists the job's marks newest first, with their time, face, stroke count, note and a visibility tick. Selecting a row highlights that mark on the board and centres it. **Edit Note…** writes the note after the fact, and **Clear Mark…** removes the mark after asking; both are kept in the job's journal. A mark's visibility tick hides that one mark, while the layer strip's marks row and **View → Liquid damage marks** hide or show all of them together. The overlay switches are session state; the marks themselves stay with the job.

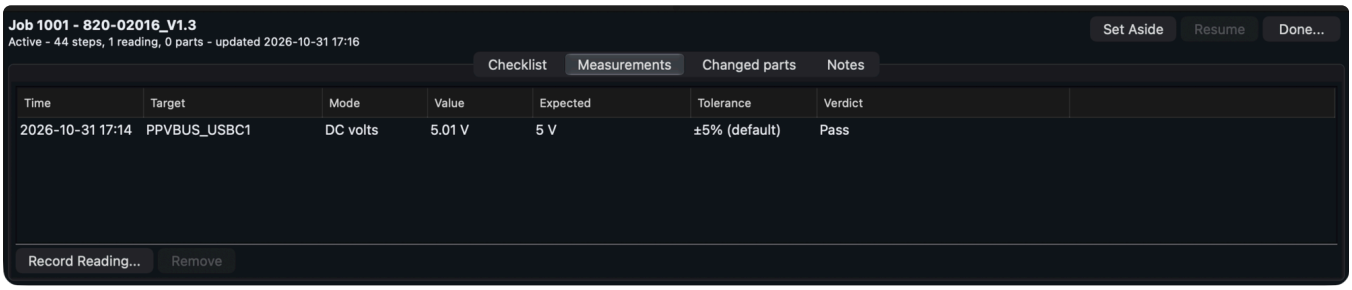
The chalk colour is the **Liquid damage** row on Settings → Colours, so each theme can lend it its own ink.

When you record a changed part whose origin or one of whose pins lies inside a mark on the part's face, the dialog pre-fills its **Reason** with **liquid damaged**. It is a suggestion in an editable field, never an automatic claim.

Comparing a reading with its reference

The **Measurements** page shows, for every reading, the **Expected** value, the **Tolerance** that judged it and the **Verdict**. The Guides **Expected** table shows the same result beside the expectation it came from. A verdict is one of:

- **Pass** — inside the expectation's stated tolerance, or inside the spec's default ±5% when it states none; the tolerance column says which was used. A range is judged by its two ends and a bound by its one, and the column says `range` or `bound` instead of naming a tolerance.
- **Fail** — outside it; the reason names the value it crossed.
- **Not compared** — the expectation and the reading cannot be judged against each other, and the reason says why: a different mode (AC against DC), a different power state, a unit that cannot be converted, a pair expectation, or a value stated for one machine build (Air/Pro).
- **No reference** — nothing on the board or in the sidecar states a value for that target, so there is nothing to judge the reading against.
- **Step gone** / **Reference gone** — the row it was recorded from no longer exists (the step was deleted, the note removed, the sidecar reading changed). The Expected cell then shows the reference text the reading kept, so the row still says what it was judged against when it was taken; nothing is matched in its place.



Each reading shows what it was judged against, the tolerance that judged it, and the verdict with its reason. The journal keeps the as-recorded verdict; the page recomputes the display from the current guide.

A reading is judged against the guide row it was recorded from, or — when it was recorded on its own — the first matching guide expectation, then the sidecar's net reading. **Record Reading…** is available under the Guides table and in the Inspector's net section; both fill in the target and the reference, and a guide row also fills in the mode its expectation implies. The buttons are enabled only while a repair job is active. The verdict is recomputed whenever it is shown, so editing a guide or a sidecar reading changes the display without rewriting what the job recorded. A difference is never by itself a diagnosis.

Reusable templates

Save as Template… under the Guides table writes the note on screen as a reusable checklist file; the job pane's **Checklist** page can save the current job's checklist the same way. The file is an OBData checklist you choose the location of each time: every non-empty line is a step, and it records an optional board key.

New Job… offers **Choose…** beside its Template row. A chosen template replaces the automatic guide-step import, and its steps become the job's editable checklist. A template written for another board is accepted with a warning — its references may name nothing on this board — and a hand-written guide-only OBData file works too. The template file is never changed; the job owns the copy.

Where jobs live

Jobs are **your data, not the board's**: one JSON file per job under `jobs/` in the application-data directory (chapter 9), never beside a board. A save replaces the file atomically. Every change is also appended to the job's journal, which is what the CRM link sends (below). Without a CRM, nothing in this feature uses the network.

Working with a CRM

MWV can work with your shop's CRM — the Repairshop CRM out of the box. It is optional: until it is set up, MWV makes no request at all and jobs work as above.

With a CRM connected, the CRM is the record. A job linked to a CRM repair is a working copy of that repair on this computer: what you record at the bench — readings, notes, finished steps, changed parts, the close-out — goes into the repair's log in the CRM, and opening the repair again rebuilds the job from that log. Jobs you never link stay **local jobs**: they are kept here as before and nothing of them is sent.

Setting it up

Job → CRM → CRM Settings... opens two pages.

- **Connection**: the CRM's **server address** and your **API key**. **Test Connection** checks both and says what it found ("Connected. The CRM lists 11 current repairs", or that the key was rejected, or that the server cannot be reached). Below it, what MWV does for a linked job: read the repair when the job opens, add readings / notes / steps / changed parts / the close-out to the repair log, and fill in the device's board number when the CRM has none. Each can be switched off. **Open in CRM** is the address of a repair in your CRM's own web page or app, with `{{job.external_id}}` standing for the repair number; leave it empty to switch that command off.
- **Requests** (advanced): how each request looks — method, URL, body, headers — and how its answer is read (**Response**: one `field = path` per line). The shipped values fit the Repairshop API; change them only for another CRM. Placeholders such as `{{job.external_id}}`, `{{board.name}}` and `{{log.message}}` are filled in, escaped for where they go (a JSON body, a URL); `{{num:…}}` requires a number and `{{raw:…}}` inserts a value as it is.

The key is stored in MWV's data folder, readable only by you; if the environment variable named on the Requests page (`MWV_CRM_TOKEN`) is set, it wins. The key is sent the way the CRM expects it — for Repairshop as a URL parameter, which MWV removes from every address it writes to the log. **Import...** and **Export...** move the settings between computers; the key is never part of an exported file.

Opening a repair

Job → CRM → Open Repair... lists the CRM's current repairs — number, customer, device, board, title. Type to filter; **Include finished repairs** lists the rest. Choosing one opens its job: the one already here, or a new working copy. If a different job is open, MWV asks whether to **link that job** to the repair or **open the repair's own job** — it never re-links a job silently. **Open Job...** offers the same list through its **From CRM...** button.

A repair whose job has no board yet asks for one: **Choose File…**, **Find in Library…**, or **Use** the board already on screen. When the CRM already names the board, the question says which. The board's name goes to the CRM's device record if it has none; a device that already names a board is left alone.

Job → CRM → Link Job to Repair… links the open job to a repair — everything already in the job is then sent to it. **Unlink from Repair…** makes it a local job again; what was sent stays in the CRM.

While you work

Every change to a linked job goes to the CRM a few seconds after you make it, in the background — the window never waits for the network. The job pane's CRM line says where things stand:

- **in sync** — everything has reached the CRM;
- **N changes waiting** — they are on their way;
- **CRM unreachable, N waiting** — the CRM could not be reached; MWV keeps the changes and tries again (after 15 seconds, then less often, up to every five minutes), and sends them as soon as it can, also after a restart;
- **the CRM rejected the API key** — this one is also shown in a message, once;
- **N not accepted** — the CRM refused a change; the log says why. **Sync Now** tries again.

The same state shows at the bottom of the window, in every work mode, whenever there is something to report: **CRM: 3 pending**, **CRM: offline, 2 pending**, **CRM: 1 not accepted**, **CRM: key rejected**. It covers every linked job, not only the open one, and shows nothing when no CRM is set up or everything has arrived. Deleting a reading, note, step or changed part is sent too, as a line "Reading removed: …" in the repair log, so it does not come back when the repair is opened again or **Sync Now** reads the log.

Sync Now (`Ctrl+Shift+Y`, or the button in the pane) sends what is waiting, retries what was refused, and reads the repair again. The **CRM log** tab lists the entries others wrote in the CRM — a call with the customer, a quote — read-only.

Each entry MWV writes reads as a sentence ("Reading: PP3V3_G3H = 3.31 V (DC volts, Pass)") followed by one line starting with `[mwv:` — that line is how MWV rebuilds the reading when the repair is opened again. Leave it in place.

Upload File… adds a photo, a PDF or another file (up to 25 MB) to the repair, with an optional caption. **Open in CRM** opens the repair in your CRM.

When a job is done

Done… sends the close-out with everything else. Once the CRM has accepted all of it, MWV reads the repair's log back from the CRM and checks that every change is there. Only then does the job's working copy start to age: it stays on this computer for 30 days, and is removed after that (a second check of the CRM's log comes first). The CRM holds the job, and opening the repair again brings it back. The number of days is the `closedJobGraceDays` entry of the `crm` group in the settings file (`0` removes the copy right after the check). A job with a refused change, with a kind of change switched off in the settings, or whose check could not be completed (the strip says **CRM: 1 closed job not verified**) is kept until that is settled.

Quitting MWV gives waiting changes a moment to go out. If some still have not reached the CRM, MWV says how many and asks whether to quit anyway; what is left goes at the next start.

What's next

Chapter 13 is what to do when something does not work.

13. Troubleshooting

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.** Five registered formats decline every file; chapter 10 lists them, and each explains itself in the board's import notes.
- **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.**
- **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. So does declining a guessed pairing.

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.

The CRM does not answer

The job pane's CRM line and the log (every request is logged, with the key hidden) say what happened:

- **The CRM cannot be reached / did not answer in time**: the server address is wrong, the server is down, or the network is. Nothing is lost — changes wait and go out by themselves. **Test Connection** in the CRM settings checks the address.
- **The CRM rejected the API key**: enter a valid key in the CRM settings. If the `MWV_CRM_TOKEN` environment variable is set, it is used instead of the key there.
- **Plain http:// is only...**: MWV talks to a CRM over `https://`. A test server on this computer may use `http://` when **Allow plain HTTP** is on.
- **N not accepted**: the CRM refused those changes; the log names the reason. Fix it (or its settings) and use **Sync Now** to send them again.
- **The CRM commands are greyed out**: Open Repair needs a configured CRM; Sync Now, Upload and Unlink need a job linked to a repair.
- **A secure connection could not be established**: the CRM's certificate is not trusted by this computer. MWV uses the system's certificates (and its proxy settings) on macOS and Windows; the CRM's address must match its certificate.

It feels slow

- The **mesh is built once, at load**, for both faces; a flip, a rotation or a mirror rebuilds no mesh. The names are the one thing re-laid when the view turns or mirrors, because they stay upright on screen. 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.

A glitch mark

The quickest report is a **glitch mark**: press `Ctrl+Shift+G` (`Cmd+Shift+G` on macOS), or choose **Mark a glitch…** on the Help menu, at the moment something looks wrong. Nothing opens and nothing is sent — it writes one folder and logs its path.

The folder is named after the time and the board, and holds:

- `mark.txt` — when the mark was taken, the version and build, the board, its format and counts, the sidecar it used, where the camera was, what was selected, which panes were up, the active job, and the last two hundred lines of the log.
- `mark.json` — the same facts, for a script or a bug tracker.
- `view.png` — the board exactly as it was drawn.

The folder is written where **Settings → Diagnostics** says, or into a `glitch-marks` folder in the application's data directory (chapter 9) when nothing is set there. Nothing is ever written beside your board file.

Sending one

You do not have to find three files. **Settings → Diagnostics → Send Latest Mark…** — or **Help → Send the latest glitch mark…** — packages the newest mark as one ZIP beside its own folder, `<stamp>-<board>.zip`, and shows it selected in your file manager. Attach that single file to a message; it contains the mark's folder with `mark.txt`, `mark.json` and `view.png` inside.

Open Folder on the same page (and the resolved path above it) is there when you would rather look yourself, copy the folder, or delete old marks. With no mark taken yet, there is nothing to send and the message tells you the key that makes one.

What's next

Chapter 14 is the board library for a collection you keep on disk. After that, Appendix A defines the vocabulary.

14. The board library

A collection of boards is easier to search than to browse. The library indexes folders you choose, once, and then finds a board by its number, its device, a code name you know or a name you added. It is **metadata only**: it reads a file's first few hundred bytes to identify its format, reads the repair-data file beside it if there is one, and asks the archive MWV ships with. It never opens the board itself, so a scan of a large collection is quick and a board with a format MWV cannot draw is still listed.

Adding folders

Choose **File → Library…** (`Cmd+Shift+L`), then **Add Folder…**, and pick a folder. Every board file under it is indexed recursively; hidden files and folders are skipped, and so are folders reached through a symbolic link. You can drop a folder on the MWV window or on the empty board's card instead. **Folders…** removes a folder and the boards that came from it; **Rescan** walks the current folders again.

MWV does not walk your folders on its own. Opening the Library window shows the index the last scan wrote, and a scan runs only when you ask for one with **Rescan**, or when you add a folder to index it.

The folder list, the index and everything below are kept in MWV's own data directory (chapter 9). Nothing is written beside a board or into your collection.

What MWV learns from a name

For each board the library collects four kinds of name, and keeps where each one came from:

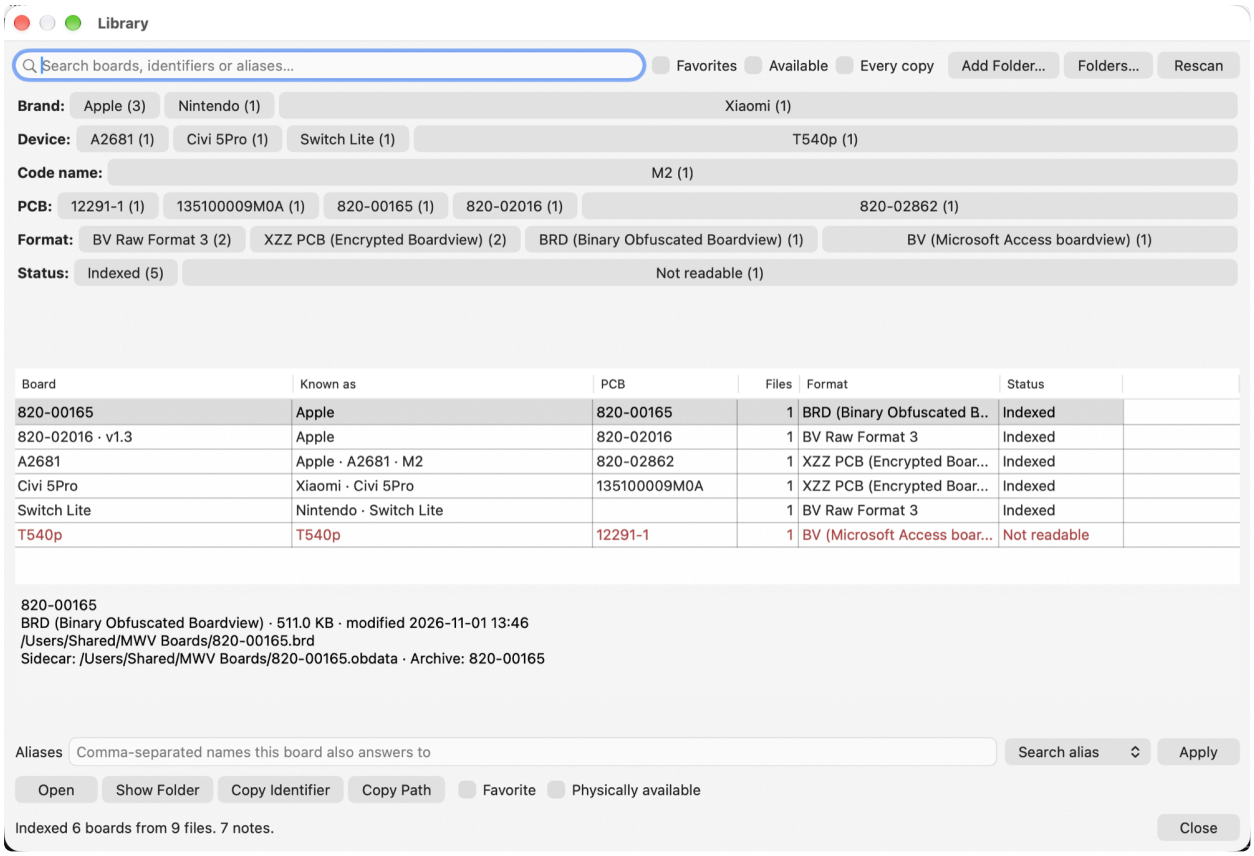
- **PCB identifier** — the board's own number, such as `820-00165` or `12291-1`.
- **Device** — what the board belongs to, such as `MacBook Air` or `Switch Lite`.
- **Code name** — the short name a workshop uses, such as `Rosa` or `M75`.
- **Brand** — the maker, such as Apple, Lenovo or Nintendo.

The sources are the file and folder names, the repair-data file beside the board, the OpenBoardData archive MWV ships with, and the aliases you add. A fact from a repair-data file or the archive wins over a guess from a file name. Packaging words such as *boardview*, *schematic*, *pcb* and *layer* are never treated as a device.

A revision is kept apart from the board number: `820-02016_V1.3` is board `820-02016`, revision `1.3`, and `820-02382-16` is board `820-02382`, revision `16`. Two revisions are two rows; two copies of the same revision are one row with a copy count.

The Library window

The window lists one row per board and revision. Its columns are the board, what it is known as, the PCB identifier, the number of copies, the format, its health and the schematic association:



- **Schematic** shows a tick when the board's sidecar names a PDF that exists, a warning when the association is present but the file is missing, and nothing when the board has no association. A scan reads the association but never invents or writes one.
- **Indexed** — the file is there and readable.
- **Changed** — it changed since the last scan, so the row was refreshed. The next scan makes it **Indexed** again.
- **Missing** — the file was indexed from a folder you still have, but is no longer at that path. The row stays so the name and your notes are not lost.
- **Unreadable** — the file is there but could not be read.
- **Not readable** — MWV recognises the format but this version cannot draw it.

When a board has several copies, the health column follows the copy **Open** would use and names how many other copies differ.

While a scan runs, the progress line counts the folders and files it has seen so far.

Click a column header — **Board**, **Known as**, **PCB**, **Files**, **Format**, **Status** or **Schematic** — to sort by it. Click the same header again to reverse the direction; the arrow in the header shows the active column and direction. The details line below the list follows the selected row, not the sort.

Below the list, the details line names the path, size, modification time and the repair data or archive entry the library matched. Select a row and:

- **Open** opens the preferred copy. **Enter** or a double-click does the same.
- **Show Folder** opens the containing folder.

- **Copy Identifier** copies the board number (or the display name when there is none), and **Copy Path** copies the full path.
- **Favorite** pins the board to the top of an empty search.
- **Physically available** marks the board as one you actually have. It is kept for the donor-part search a later version will build; it changes nothing else today.
- **Aliases** adds other names the board answers to, separated by commas. Choose *Search alias* to make them searchable only, or one of the four kinds to put them in that filter as well. **Apply** saves them.

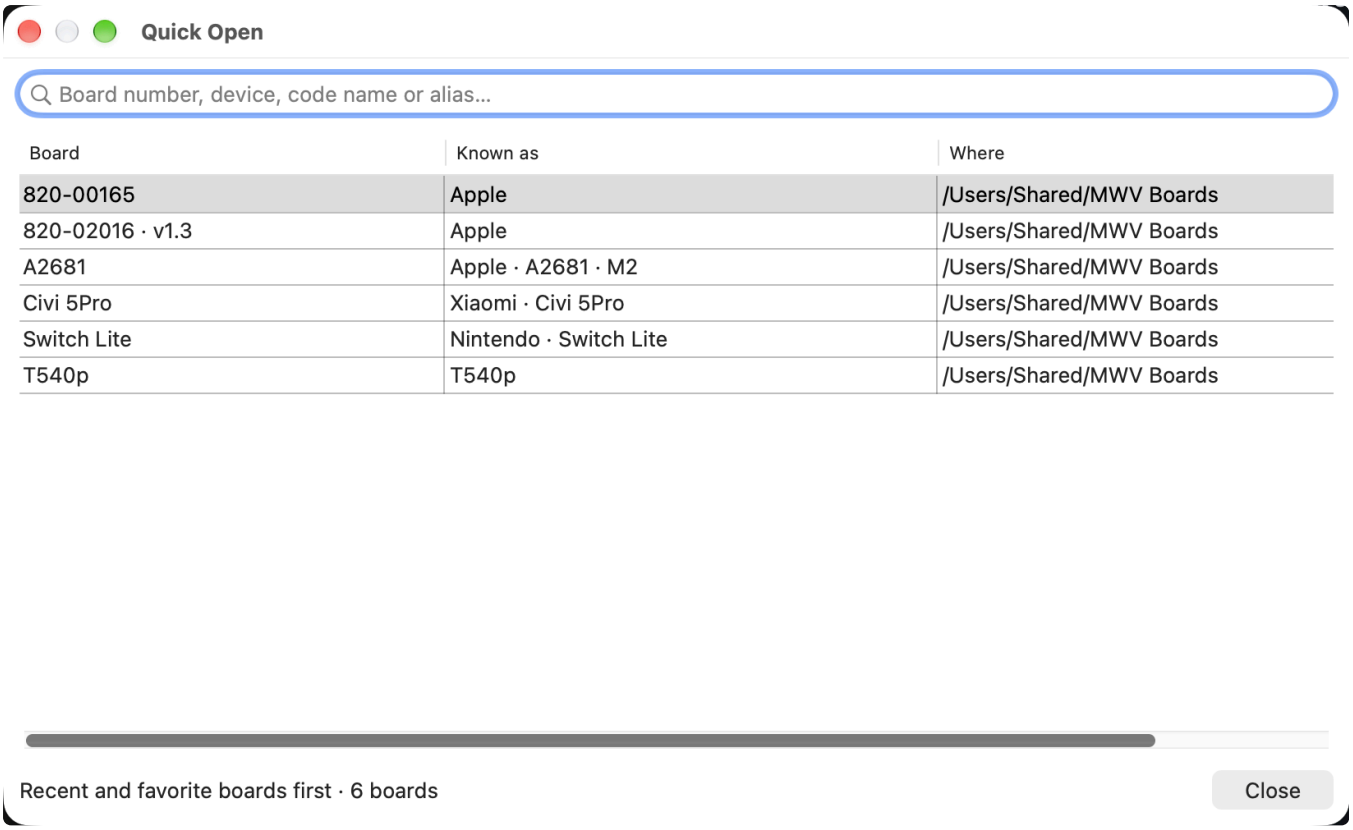
Searching and filtering

Type in the search box. Punctuation and spaces do not matter, so `820-00165`, `820 00165` and `82000165` are the same query, and a part of a name is enough: `165` finds the board, and `mcbk` finds a MacBook by its letters. Every word you type must match. An exact board number or code name comes first, then a prefix, then a word inside the name, then a close spelling.

The chips above the list filter by brand, device, code name, PCB identifier, format and health. Click one to keep only the matching boards, click it again to remove it. Two chips of the same kind widen the result; chips of different kinds narrow it. **More** shows the rest of a long list. The **Favorites** and **Available** boxes are the same filters for those two flags, and **Every copy** switches from one row per board and revision to one row per file.

Quick Open

File → Quick Open… (`Cmd+K`) is the fast way in. Type any part of a board number, device, code name or alias, choose a result with the arrow keys and press **Enter**. It uses the same search and the same preferred copy as the Library window, but it does not scan: it reads the index the last scan wrote. If the library is empty, it offers **Manage Library…**.



The empty board's card, shown when no board is loaded, has the same field: type a name and press **Enter**, or use **Open Board...** or **Library...** (chapter 1).

When a folder moves or a board disappears

A board that is no longer at its recorded path stays in the list as **Missing** while its folder is still in the library; **Open** is disabled and the details line says what is wrong. A folder that is gone is reported in the status line, and its boards are marked missing too. **Rescan** after reconnecting a drive is the way to bring them back; until you run it, the list and the health column show what the last scan found. If you remove a folder with **Folders...**, its rows are removed with it.

The index itself is one file in MWV's data directory. If it cannot be read, the Library window says so and will not overwrite it until you answer **Replace** on **Add Folder...**. A newer version's index is refused rather than guessed at.

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

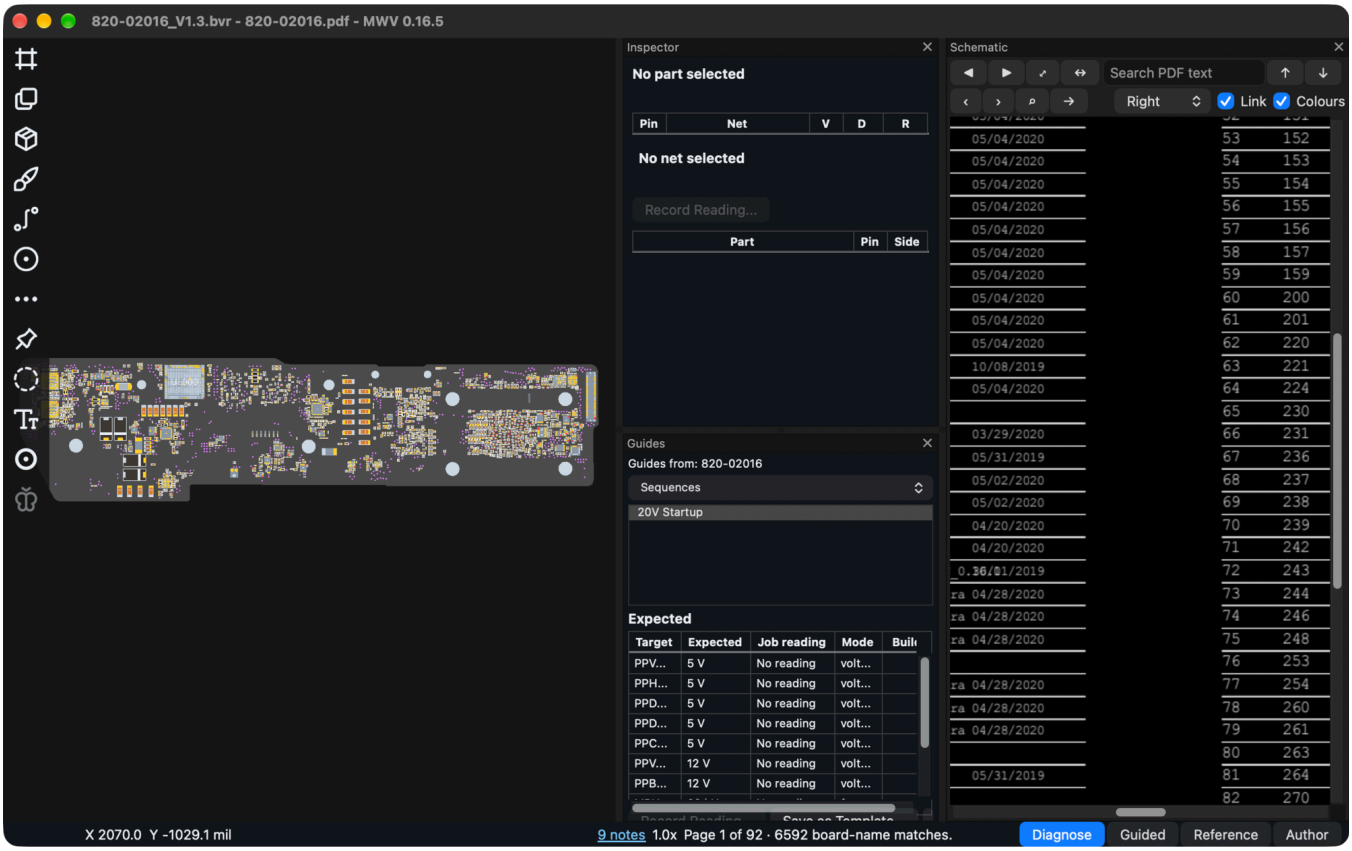
This is the hands-on case: one real board-level fault, followed from the first look to the answer, using only what the earlier chapters describe. It is a replay of a real repair, done in MWV, and each step names the feature it uses.

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

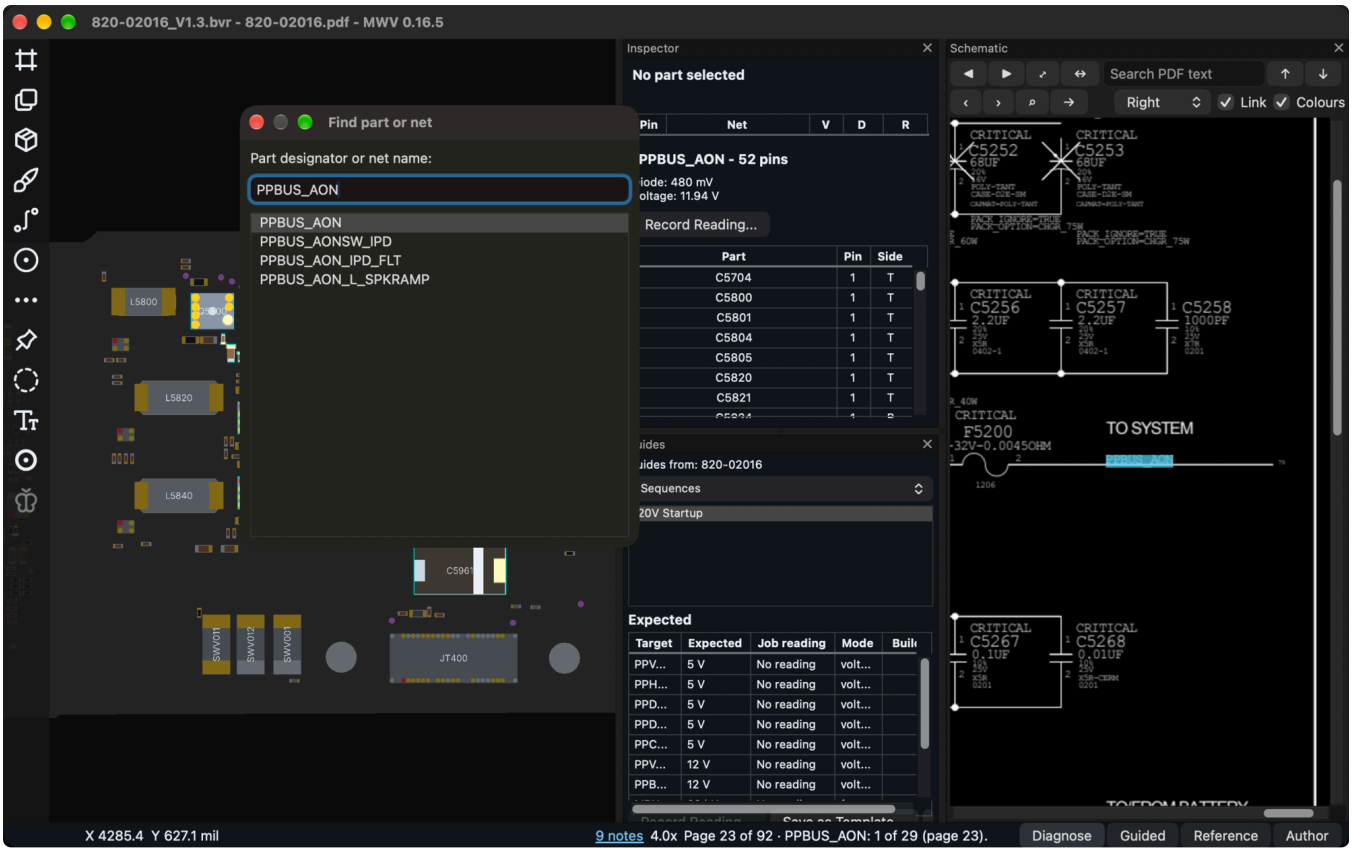
Open `820-02016_V1.3.bvr`. MWV finds `820-02016.pdf` beside it, so the board and its schematic are side by side from the first moment (chapter 1).



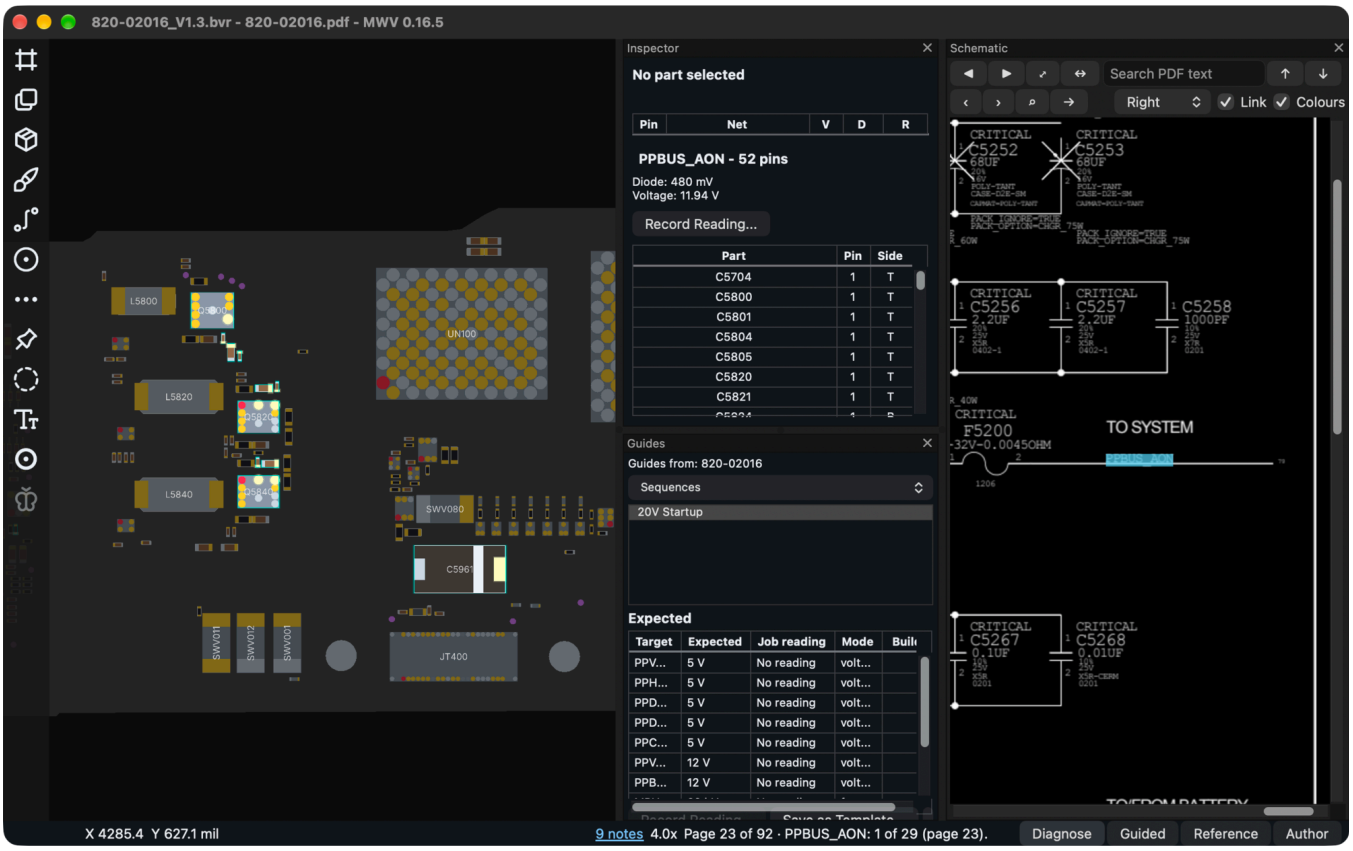
The status strip names the file, the format MWV actually read it as (`BVR3`), the part and net counts, and the repair data applied from the archive.

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).

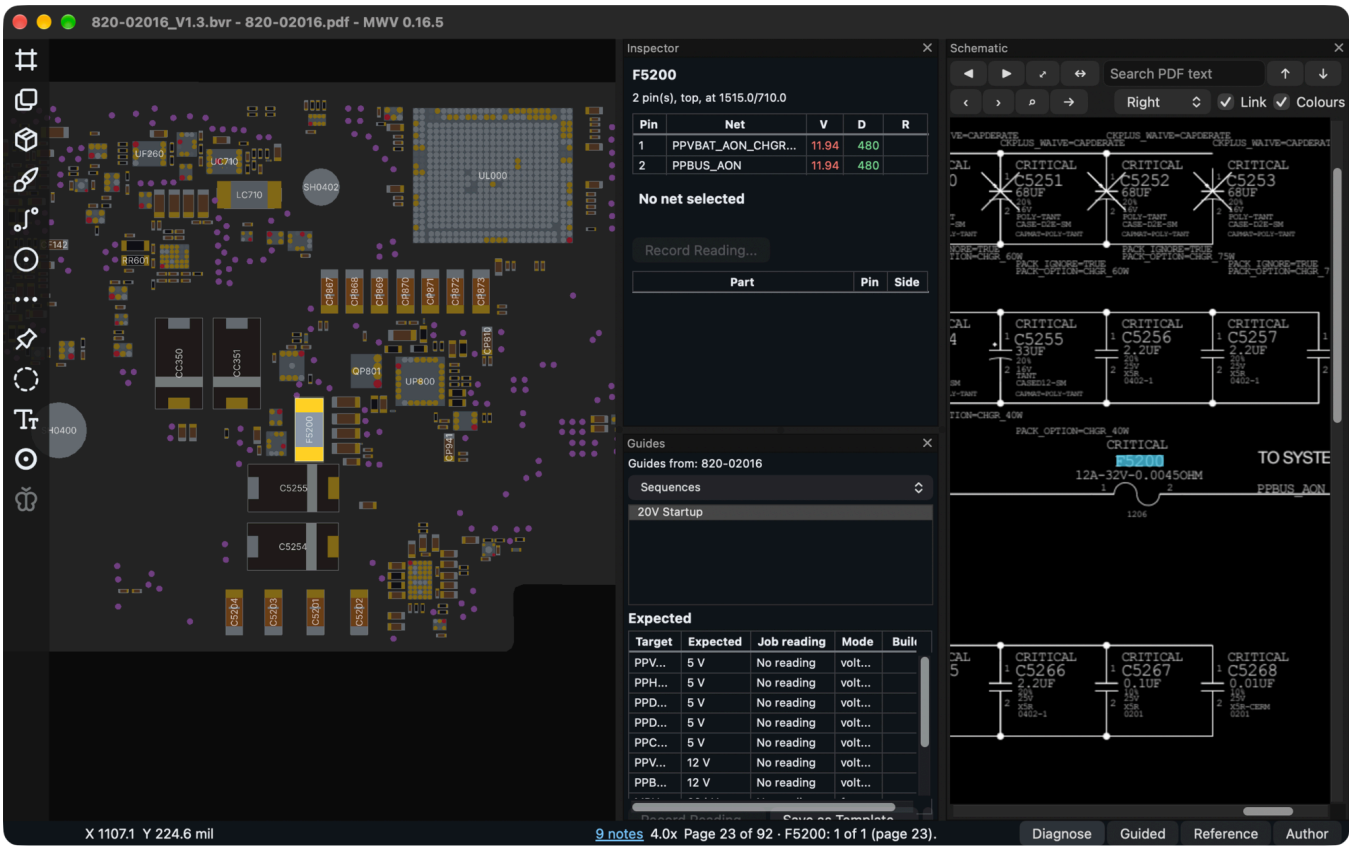


Choose the net. It lights up across the board, the Inspector lists all 52 of its pins with the archive's reference readings, and — because Link is on — the schematic jumps to a page where `PPBUS_AON` appears (chapters 2, 3 and 4).

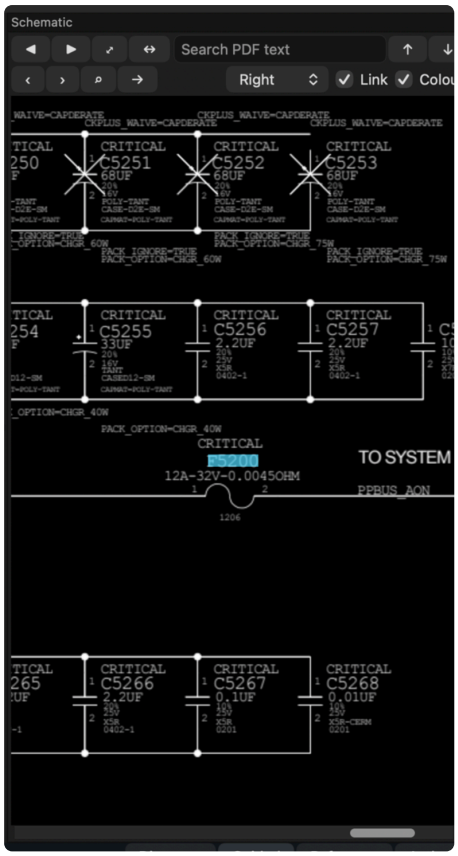


The Inspector reads `Diode: 480 mV`. That is the **reference** for a healthy board; the board in front of the technician measured **0.002 V** — a dead short.

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:

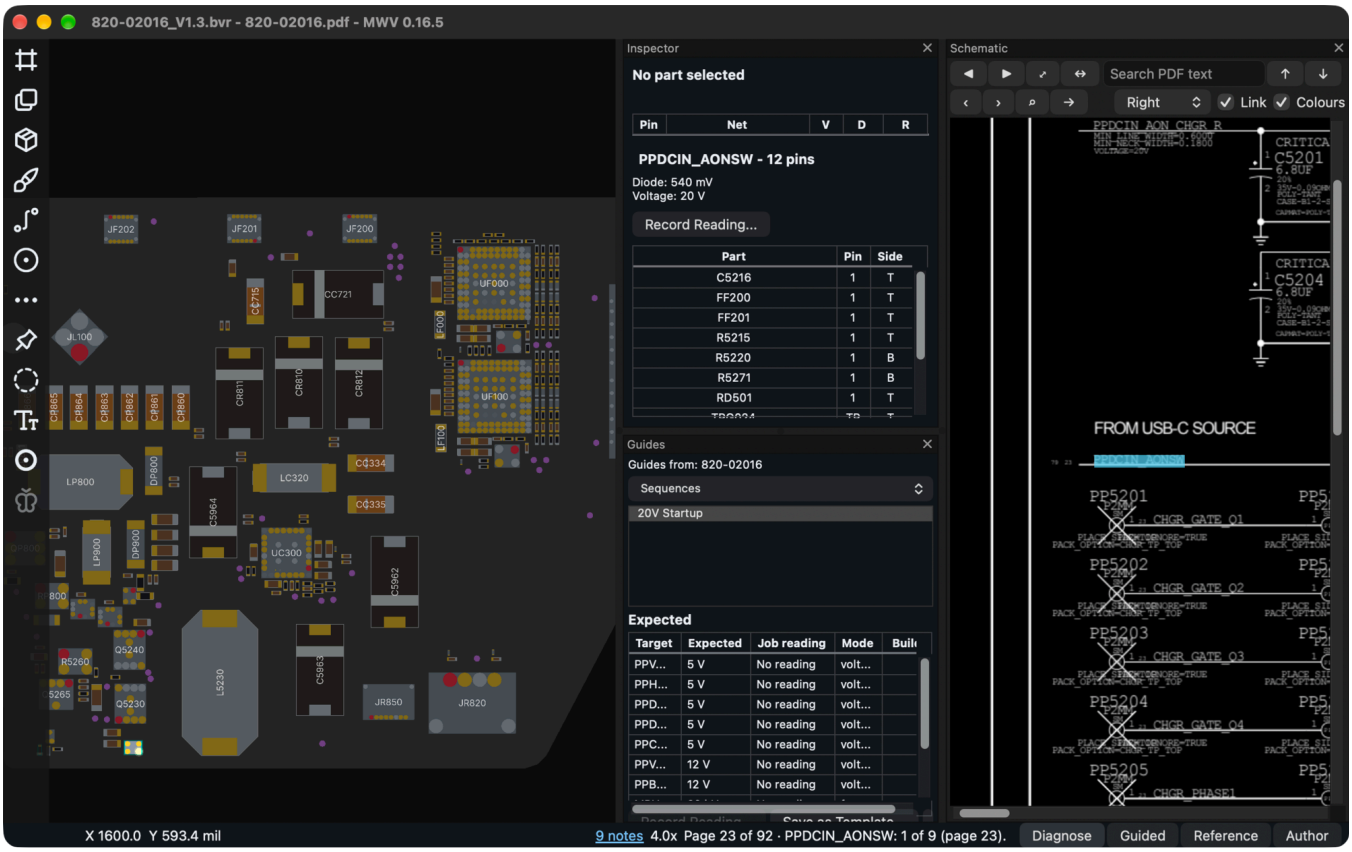


The schematic page the search landed on shows `F5200` in the charger circuit, between `PPVBAT_AON_CHGR_REG` and `PPBUS_AON`:



Step 2 — check the input side

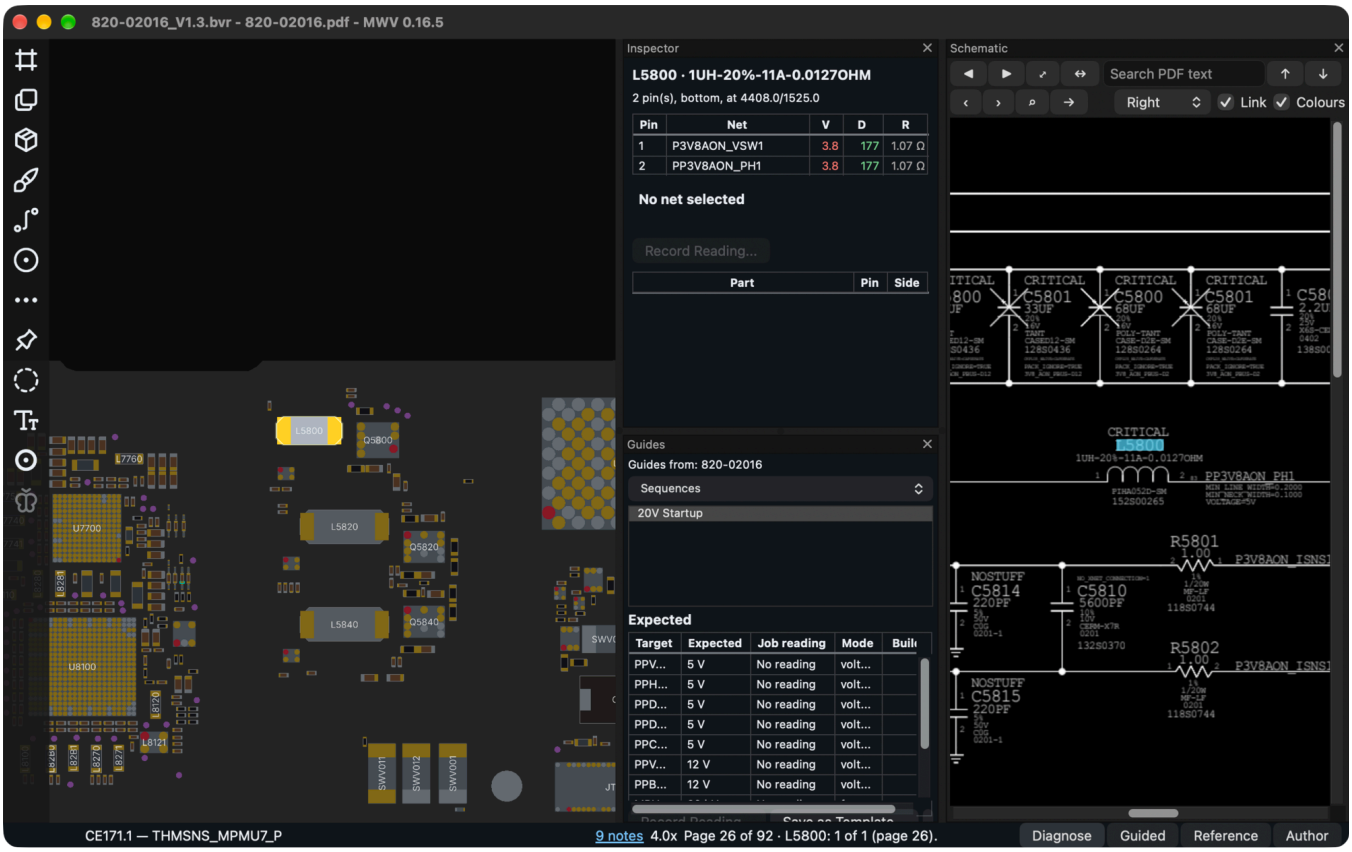
If the short were upstream, the input rail would be shorted too. Search `PPDCIN_AONSW` — the board's name for `PPDCIN_AON`. The Inspector reads `Diode: 540 mV`, normal, and the member list includes `R5220`, the current-sense resistor the technician probed:



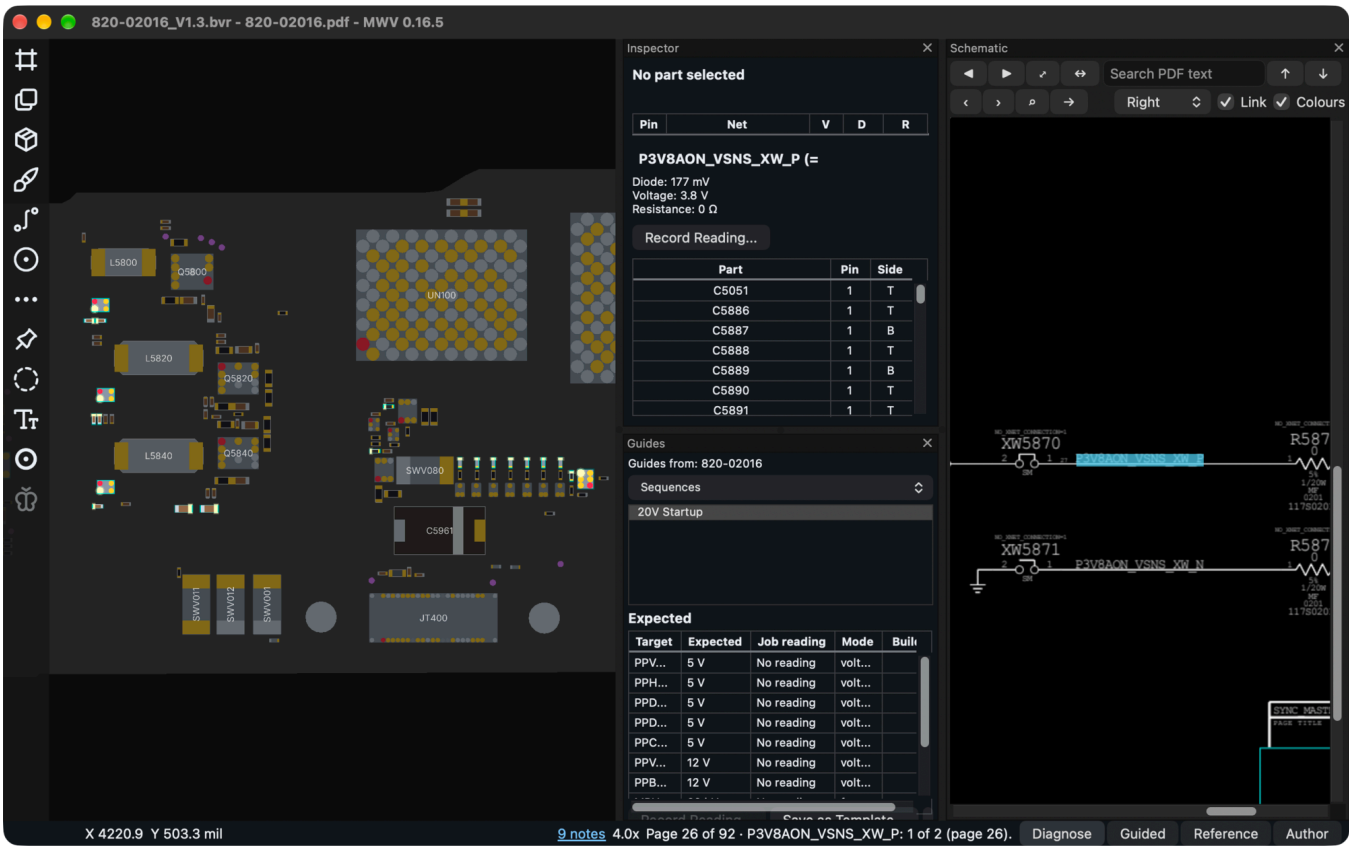
So the input is intact and the short is downstream, on `PPBUS_AON`.

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:



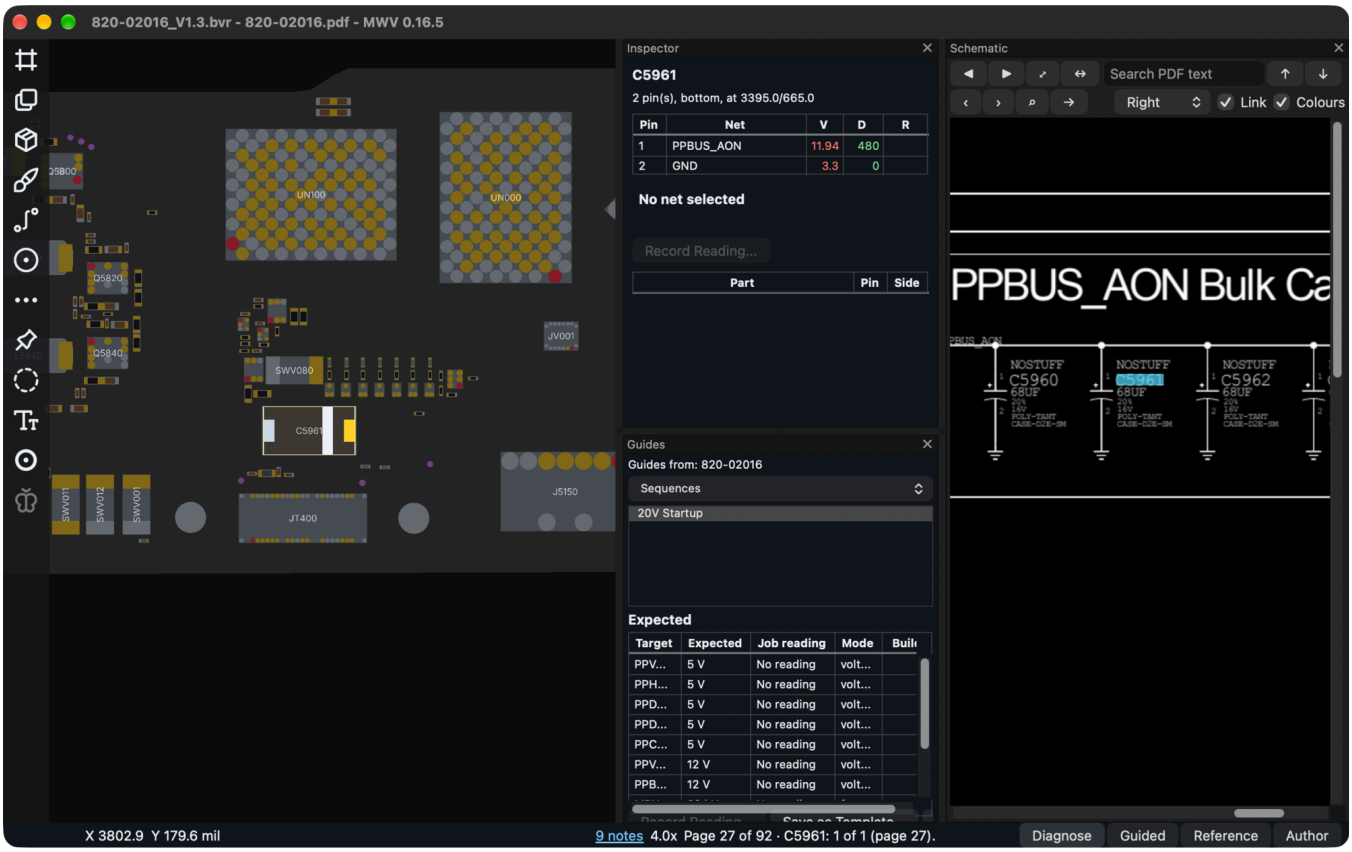
The rail is `PP3V8_AON_VDDMAIN` on this board — its name for `PP3V8_AON`. Searching it shows the rail spanning **307 pins**, with its decoupling capacitors (`C5889` among them), and the Inspector reads `3.8 V`:



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

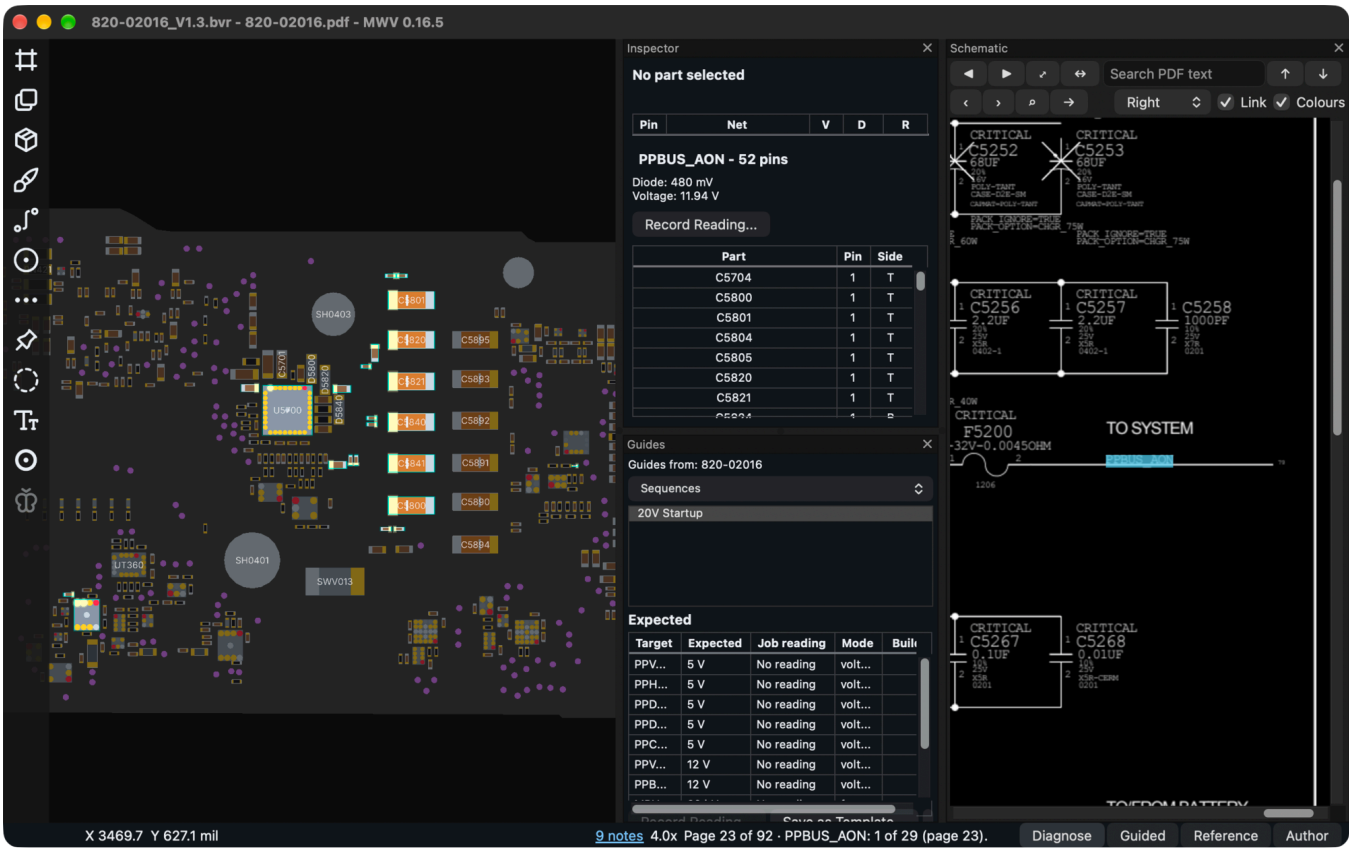
`PPBUS_AON` has dozens of decoupling capacitors, and the Inspector's member list is exactly the list to walk. The suspect is a capacitor; searching its designator selects it, draws its pin numbers, and jumps the schematic to it:



> **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 **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.