11 — Variables & Scope

> **TraxProbe & Traxcope User Manual** · Version 1.0.0 · Updated 2026-08-25
 · https://embedya.com

Variables (VARs) are the "plot this value over time" feature: every `TRAX_VAR_SET` emits one timestamped sample, and Traxcope's Scope View renders the result like an oscilloscope channel. Variables work with **zero metadata** out of the box; optional metadata adds names, units, expected ranges, colors, conversion formulas, and host-side filters.

For *high-rate* sampled data (kHz–MHz, fixed sample rate), use variable **streams** instead — Chapter 12.

On the device

Quick use — no definitions needed

```
enum { VAR_TID_TEMP = TRAX_TID_RANGE_VAR_USER_START, VAR_TID_PRESSURE };

TRAX_VAR_SET(VAR_TID_TEMP, temperature);     /* works immediately */
TRAX_VAR_SET(VAR_TID_PRESSURE, pressure);
```

Without metadata, Traxcope shows the channel as `VAR_0x2000`, treats the value as `uint32`, and auto-scales the Y axis. Cost: < 2 µs on a 64 MHz MCU, 16 bytes on the wire. Safe from ISRs.

Update macros by type

Macro	Type	Frame size
`TRAX_VAR_SET(tid, v)`	any 32-bit integer	16 B
`TRAX_VAR_SET_FLOAT(tid, v)`	`float` (bit-exact IEEE 754)	16 B
`TRAX_VAR_SET64(tid, v)`	`int64_t` / `uint64_t`	20 B
`TRAX_VAR_SET_DOUBLE(tid, v)`	`double`	20 B

Rich metadata (optional, recommended)

```
/* tid, name, type, min, max, unit, color, plot style */
TRAX_VAR_DEFINE(VAR_TID_TEMP, "Motor Temp", TRAX_DATA_TYPE_INT16,
                -40, 125, "°C", TRAX_COLOR_RED, TRAX_PLOT_AUTO);
```

- **type** — how Traxcope interprets the raw 32-bit word (`TRAX_DATA_TYPE_INT16` , `_FLOAT` , `_UINT32` , ...).
- **min/max** — initial Y-axis scaling.
- **color** — `TRAX_COLOR_RGB(r,g,b)` , a named `TRAX_COLOR_xxx` , or `TRAX_COLOR_AUTO` for palette auto-assign.
- **plot style** — `TRAX_PLOT_AUTO` (line for analog, step for booleans), or force `_LINE` / `_STEP` / `_POINTS` . Step plots are ideal for counters and queue depths.

Metadata lives in the ELF sections (zero RAM, zero runtime cost; flash cost only in `TRAX_META_IN_FLASH` mode).

Conversion formulas (host-side)

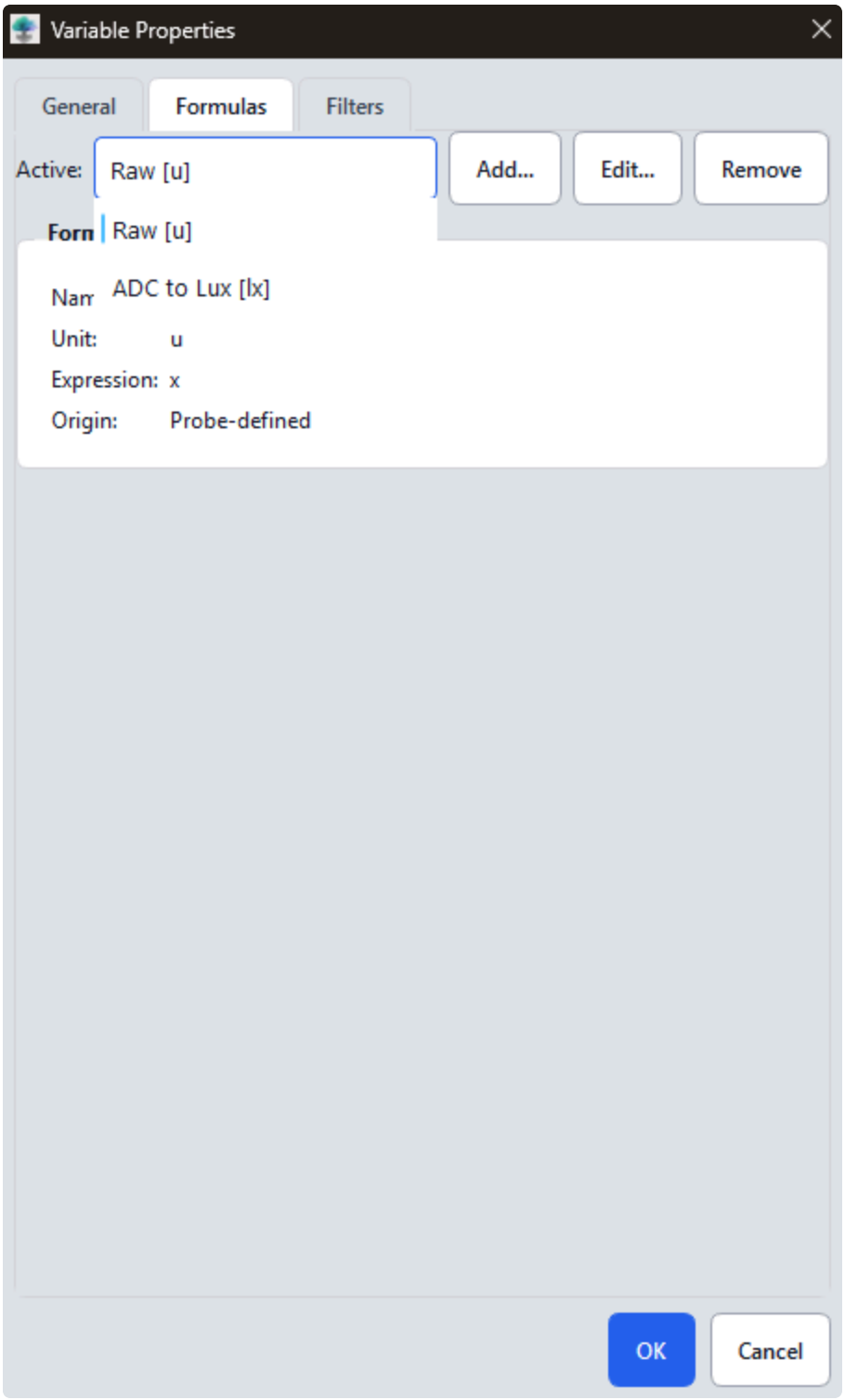
Ship engineering-unit conversions with the firmware; raw values stay on the wire and in storage, conversion happens at display time:

```
/* Simple scale/offset — fixed-point temperature stored as tenths */
TRAX_VAR_FORMULA(VAR_TID_TEMP, "Celsius",    "°C", "x * 0.1");
TRAX_VAR_FORMULA(VAR_TID_TEMP, "Fahrenheit", "°F", "x * 0.18 + 32");

/* 12-bit ADC counts → volts */
TRAX_VAR_FORMULA(VAR_TID_ADC, "Voltage", "V", "x * 3.3 / 4095.0");

/* Several views of the same encoder counter (1000 counts/rev) */
TRAX_VAR_FORMULA(VAR_TID_ENC, "Position", "mm",  "x * 0.05");
TRAX_VAR_FORMULA(VAR_TID_ENC, "Angle",    "°",   "x * 360.0 / 1000.0");
TRAX_VAR_FORMULA(VAR_TID_ENC, "Radians",  "rad", "x * 2 * pi / 1000.0");

/* Derived quantities — voltage across a 100 Ω shunt */
TRAX_VAR_FORMULA(VAR_TID_ADC_VOLTAGE, "Current", "mA", "x / 100.0 * 1000.0");
TRAX_VAR_FORMULA(VAR_TID_ADC_VOLTAGE, "Power",   "mW", "x * x / 100.0 *
1000.0");

/* Nonlinear: NTC thermistor beta model (B = 3950, T0 = 298.15 K) */
TRAX_VAR_FORMULA(VAR_TID_NTC, "Temperature", "°C",
    "1 / (ln(x / (4095.0 - x)) / 3950.0 + 1.0 / 298.15) - 273.15");

/* Logarithmic: raw power ratio → dB */
TRAX_VAR_FORMULA(VAR_TID_RF_PWR, "Level", "dB", "10 * log(x)");
```

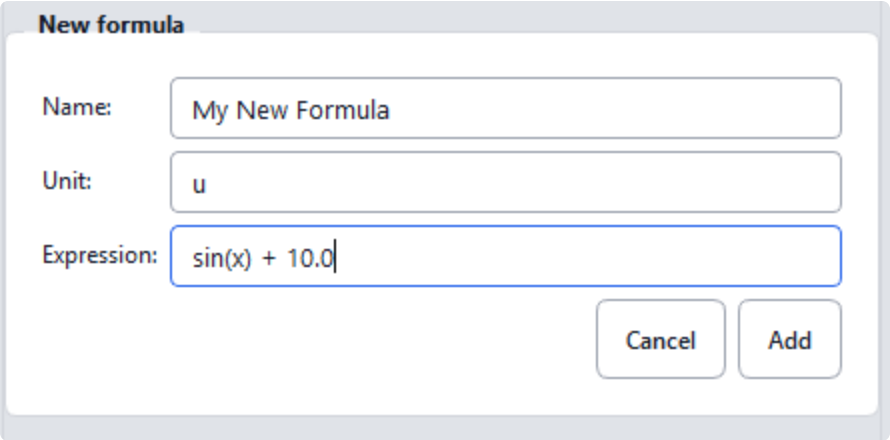
`x` is the raw typed value. Expressions use standard math syntax: `+ - * / % ^` and parentheses, the constants `pi` and `e`, and the usual functions — `abs`, `sqrt`, `exp`, `pow`, `floor`, `ceil`, trig (`sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`, and hyperbolic variants). Note the two logarithms: `ln(x)` is natural log, `log(x)` is base-10.

Multiple formulas per variable are allowed — Traxcope shows them in a per-channel dropdown and you switch at runtime. You can also add your own formulas on the host side in the variable properties dialog (**Formulas** tab → **Add…**: name, unit, expression), without touching the firmware. Since raw data is preserved, you can re-interpret a recording with a different formula later.

A formula always operates on **one** variable (`x`). To combine several channels that were sampled at the **same instant** — voltage × current for instantaneous power, a three-phase sum, a ratio — use a math channel in the Scope View. For windowed IEEE-style power numbers (P, S, PF, displacement factor) see Power analysis.



Screenshot: Formulas tab — switch the active formula (Raw vs a probe-defined conversion), or Add/Edit/Remove user formulas



Screenshot: adding a user-defined formula on the host — no firmware change needed

Host-side filters (IIR/FIR)

Attach discrete-time filters (H(z)) for display-time smoothing. Like formulas, filters are evaluated **entirely in Traxcope** — the MCU ships raw samples and never computes them, so a filter costs zero firmware cycles. They are intended for stream variables (fixed sample rate); on async variables they are accepted but ignored. Filters appear in a per-channel dropdown orthogonal to formulas: you can combine any formula with any filter, and switch both at runtime. Since the raw data is preserved, a recording can be re-filtered later.

A filter is the classic difference equation

[y[n] = \sum*{k=0}^{M} b_k,x[n-k] ; - ; \sum*{k=1}^{N} a_k,y[n-k]]

and there are **two places to define one**:

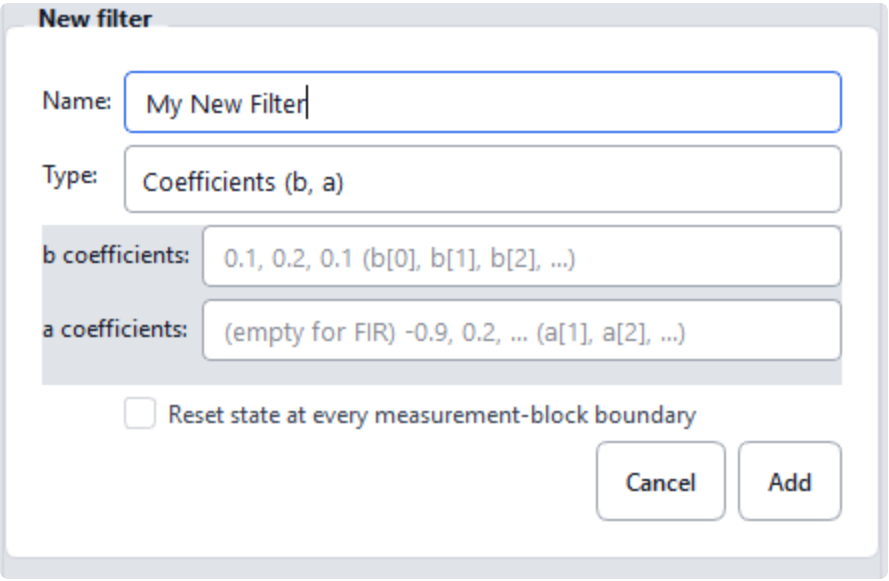
1. In the firmware — ship default filters with the ELF metadata via `TRAX_VAR_FILTER(tid, name, num_b, (b...), num_a, (a...))`:

```
/* 1st-order IIR low-pass, alpha = 0.1:  y[n] = 0.1 x[n] + 0.9 y[n-1] */
TRAX_VAR_FILTER(VAR_TID_ADC, "10Hz LPF", 1, (0.1f), 1, (-0.9f));

/* 5-tap moving average FIR */
TRAX_VAR_FILTER(VAR_TID_ADC, "MAVG-5", 5, (0.2f, 0.2f, 0.2f, 0.2f, 0.2f), 0,
(0.0f));
```

Conventions: `a[0]` is implicit 1.0, so the `_a` list starts at `a[1]` (note the sign — the feedback terms are *subtracted*); a pure FIR uses `num_a = 0` with `(0.0f)` as placeholder. The coefficient lists are parenthesised so the preprocessor treats each as a single macro argument. Multiple filters per variable are allowed.

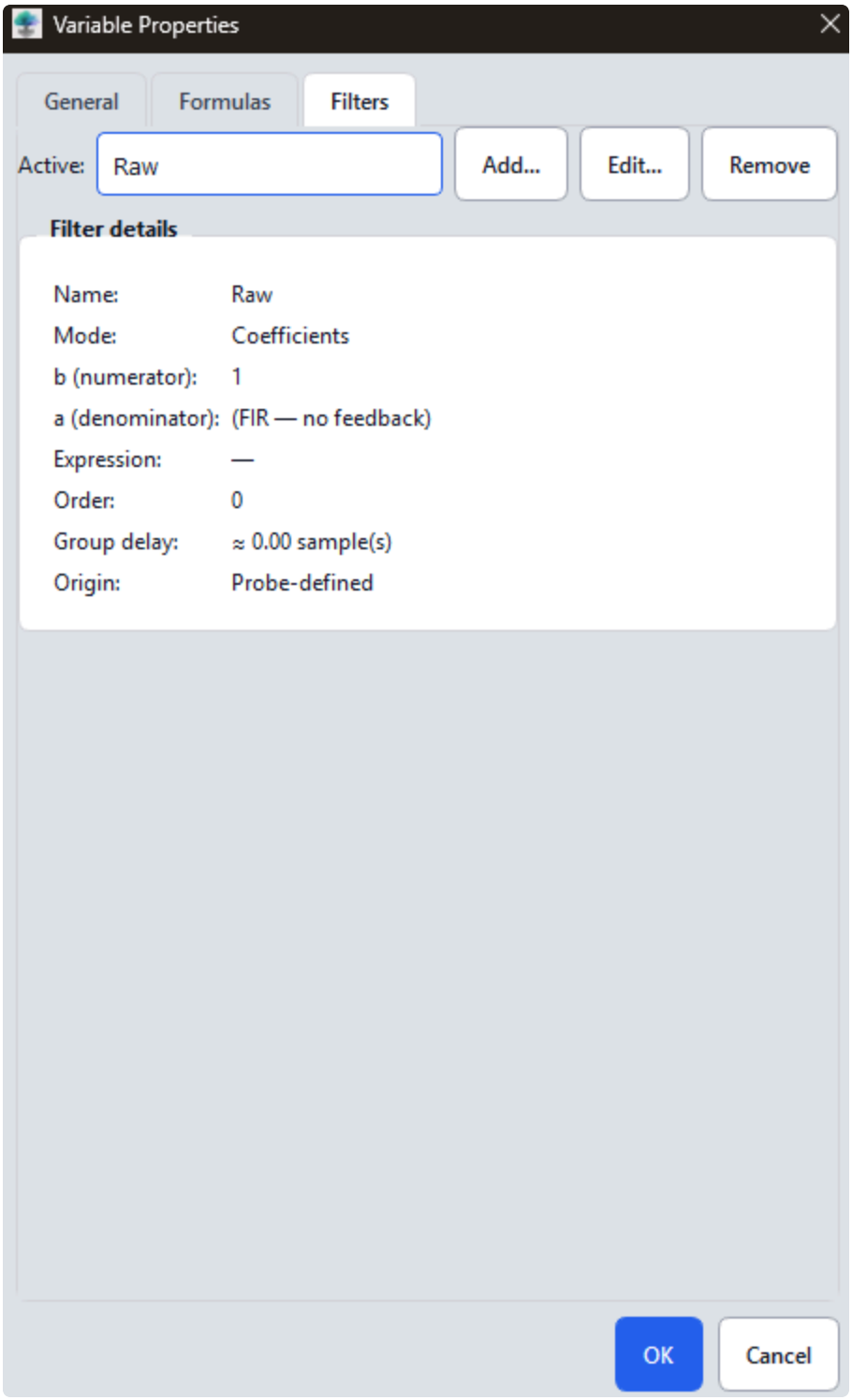
2. In Traxcope — variable properties dialog → filter editor, no firmware change or reflash needed. Two entry modes:

- **Coefficients** — type the `b` and `a` lists directly (same convention as above; leave `a` empty for FIR).
- **Expression** — a free-form formula per sample, where `x` is the current input and `x_n1, x_n2, ...` / `y_n1, y_n2, ...` reference past inputs/outputs, e.g. `0.1*x + 0.9*y_n1`. Useful for nonlinear tricks a linear difference equation can't express; prefer coefficients when it fits, as they evaluate faster.



Screenshot: adding a user-defined filter — coefficients mode with b/a lists and the block-boundary reset option

The editor shows each filter's order, an estimated group delay in samples (the visual lag a filter introduces), and its origin — **Probe-defined** (from the firmware metadata) or **User-defined** (stored in the probe configuration on the host). Probe-defined filters can be edited too, but the edit only lasts until the next connect, when the firmware metadata re-seeds them; make a user-defined copy for permanent changes. (Filter authoring is available on Standard and Professional licenses.)



Screenshot: Filters tab — active filter with mode, b/a coefficients, order, group delay, and origin

Variables from log arguments

`TRAX_LOG_DEFINE_VARS` binds log arguments to VAR TIDs, so an existing log call feeds scope channels for free — see Chapter 10.

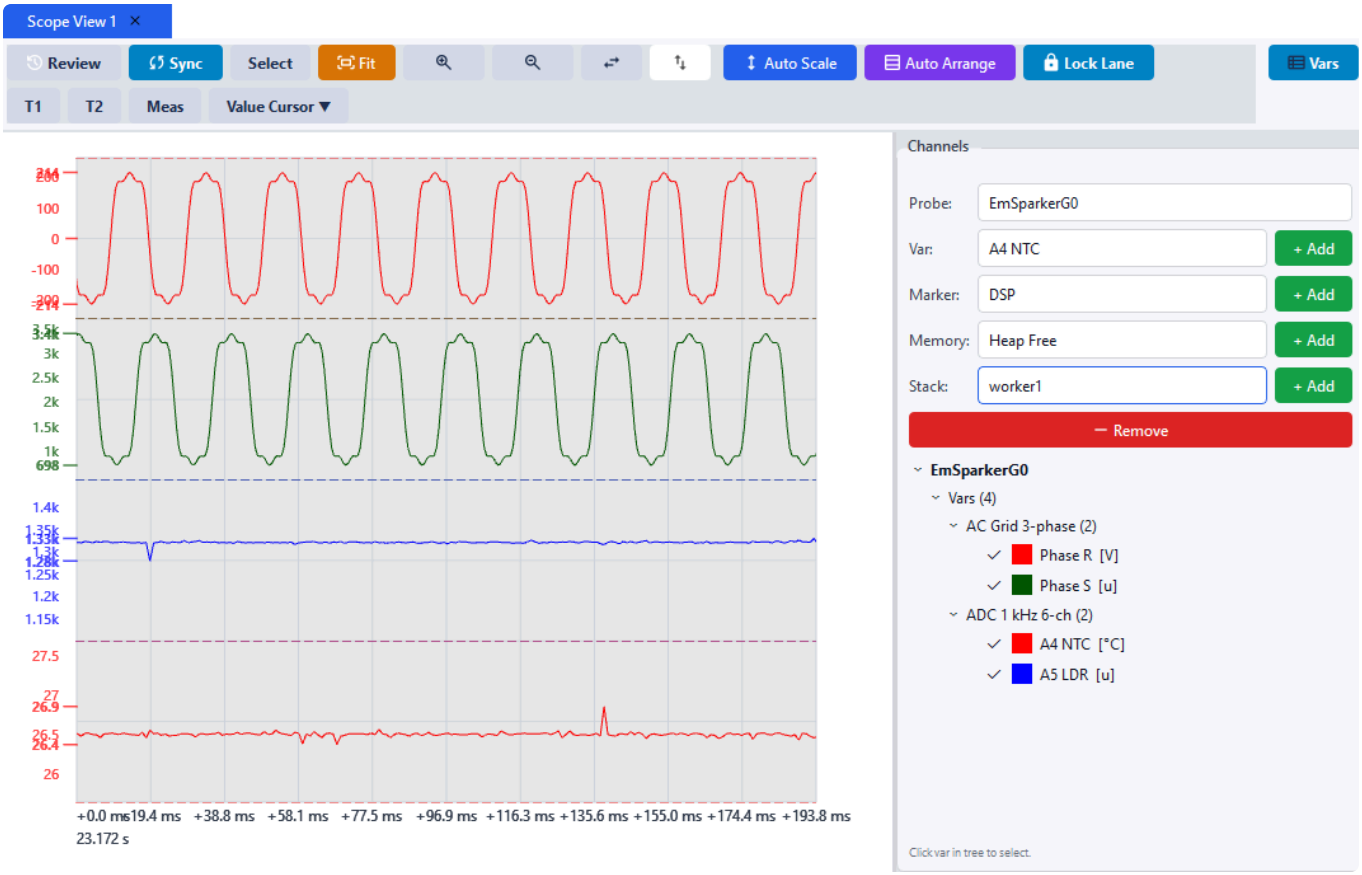
In Traxcope — the Scope View

Open via Toolbox → **Scope**. Multiple independent instances allowed; each instance's channel set is saved in the workspace.

Channels

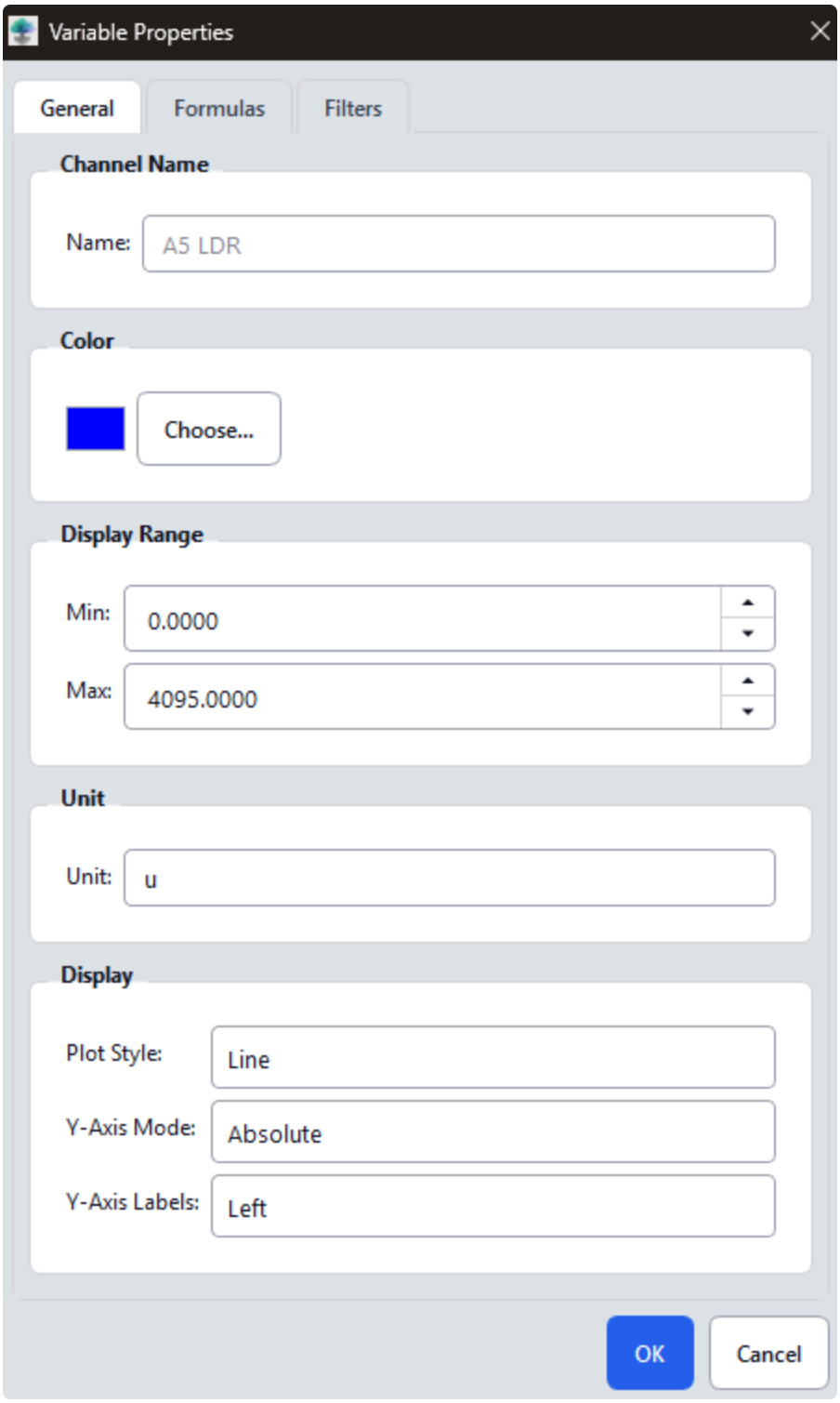
The **channel tree** is where you compose the display. Beyond firmware variables it can also plot:

- **Variables** — everything sent with `TRAX_VAR_SET*` (and stream channels, Chapter 12)
- **Math channels** — host-side combinations of co-sampled stream channels (`a * b`, …) — see below
- **Markers** — interval occupancy from Chapter 15
- **Memory metrics** — heap free / per-task stack trends from the RTOS trace (Chapter 13)



Screenshot: Scope View, channel tree expanded, 3–4 variables plotted in lanes

Per-channel controls include the **formula** dropdown, the **filter** dropdown, color/style overrides, and lane assignment. Channel properties open via the **Variable Properties** dialog — its General tab covers name, color, display range, unit, plot style, and Y-axis mode/labels; the Formulas and Filters tabs are described above. Double-clicking a math-channel output opens the math-channel editor instead of that dialog.



Screenshot: Variable Properties — General tab with channel name, color, display range, unit, and display options

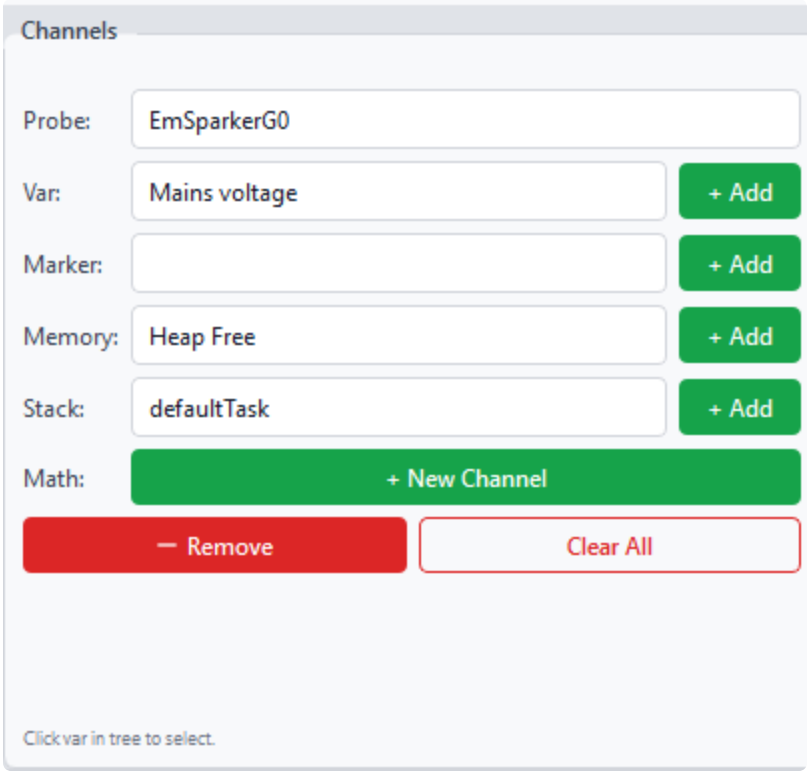
Math channels

A **math channel** is a host-side recipe that evaluates an expression on several stream variables **at the same timestamp** and publishes the result as a new scope channel. The MCU still ships the raw samples; Traxcope pairs them and does the arithmetic. The classic case is instantaneous power: bind voltage to `a`, current to `b`, and write `a * b`.

This is a different job from a per-variable formula. `TRAX_VAR_FORMULA` (and the Formulas tab) converts **one** sample (`x * 3.3 / 4095`). A math channel combines **several** samples that share an epoch (`a * b` , `a + b + c` , `a / b` , …).

Why "same timestamp" is the rule. `a * b` is only meaningful when `a` and `b` were taken at the same instant. That is true for channels of **one stream** (Chapter 12): every sequence carries one sample of every channel together, so index *k* is the same epoch on every input. Async `TRAX_VAR_SET` channels are event-sampled and do not share that contract, so they cannot be math-channel inputs.

Create one from the Scope **Channels** panel: **Math** → **+ New Channel**.

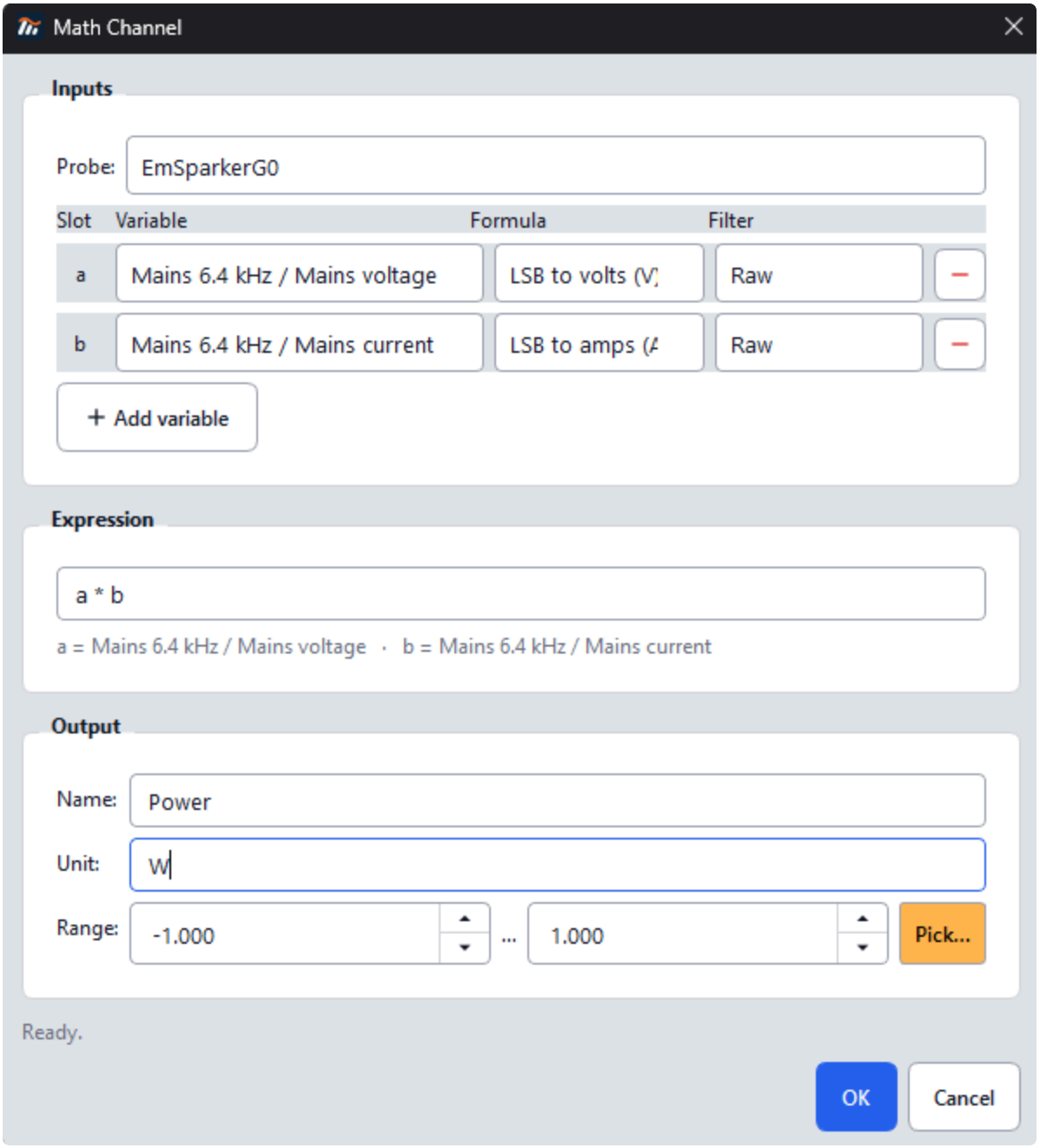


Screenshot: Channels panel — Math has no picker; ***+ New Channel*** *opens the recipe dialog*

The dialog:

1. Picks the **probe** once (not per input).
2. Adds up to **four** stream variables as slots `a` , `b` , `c` , `d` (**Add variable**). Each row can also pick that variable's **formula** and **filter** — the conversion/filter runs *before* the value reaches the expression, so you bind engineering units (`V` , `A`) rather than ADC counts. The chosen formula/filter is stored with the recipe; changing the source channel's active formula later does not silently redefine a running math channel.
3. Writes the **expression** in those slot letters. Same math syntax as a single-var formula: `+ - * / % ^` , parentheses, `pi` / `e` , and the usual functions (`abs` , `sqrt` , `exp` , `pow` , `ln` , `log` , trig, …). `e` is Euler's number, so it is not a slot — that is why the four inputs stop at `d` .

4. Names the **output** (name, unit, display range, color). Typical: name `Power`, unit `W`, expression `a * b`.



*Screenshot: Math Channel — `a` = Mains voltage (LSB→V), `b` = Mains current (LSB→A), expression `a b`, output **Power** [W]*

All inputs must belong to the **same stream on the same probe**. Mixing two streams (or two probes) is rejected: their samples are not taken together, and a plausible-looking product would be the wrong answer. A small ADC scan skew between channels of one stream is fine — pairing tolerates up to half a sample period.

The output is a **virtual stream variable**. It plots, zooms, and takes cursors like any firmware channel; you can drop it into the Spectrum Analyzer; you can even use it as a Power-tab input. It is saved in the **workspace** (so every Scope dock sees the same recipe) and is **not recorded**

— a recording would not carry the host-side recipe, so reopen the workspace and let the channel recompute. Double-click the output (or the editor's **Delete Channel**) to change or retire the recipe; removing the trace from one Scope does not delete the channel, because another view may still be plotting it.

(Math channels are available on Standard and Professional licenses.)

Expression	Typical bindings	Result
`a * b`	`a` = voltage [V], `b` = current [A]	Instantaneous power [W]
`a * a / 100`	`a` = voltage across a 100 Ω load	Instantaneous power in a known resistance
`(a + b + c) / 3`	three-phase voltages or currents	Instantaneous average
`a / b`	`a` = voltage, `b` = current	Instantaneous resistance / impedance proxy

A math channel gives you the **waveform**. For the IEEE-style *numbers* over a cursor window (mean P, S, PF, displacement factor) use the Power tab below — you do not need to build `a * b` first.

Navigation & layout

Toolbar control	Effect
Live / Review	Follow incoming data vs free navigation
Fit	Fit all data into view
Zoom In / Out + drag-zoom	Time-axis zoom
Auto Scale	Y-axis fit per channel
Auto Arrange	Distribute channels into lanes
Lane lock	Pin lane layout while data updates

Rendering uses the GPU with level-of-detail decimation, so panning across hours of recording stays responsive.

Cursors & measurements

- **T1/T2 time cursors** — place two vertical cursors; the readout shows each cursor's time, per-channel values at the cursors, and Δt. This is your bread-and-butter latency

measurement.

- **V1/V2 value cursors** — per-channel horizontal cursors for amplitude measurements.
- **Selection mode** — drag-select a time span (also broadcast via sync).

Screenshot: T1/T2 spanning a few 50 Hz periods; the sidebar's Measurements tab shows the statistics over the span plus the @T1/@T2/Δ readouts

Measurements live in the scope's right sidebar, in three tabs:

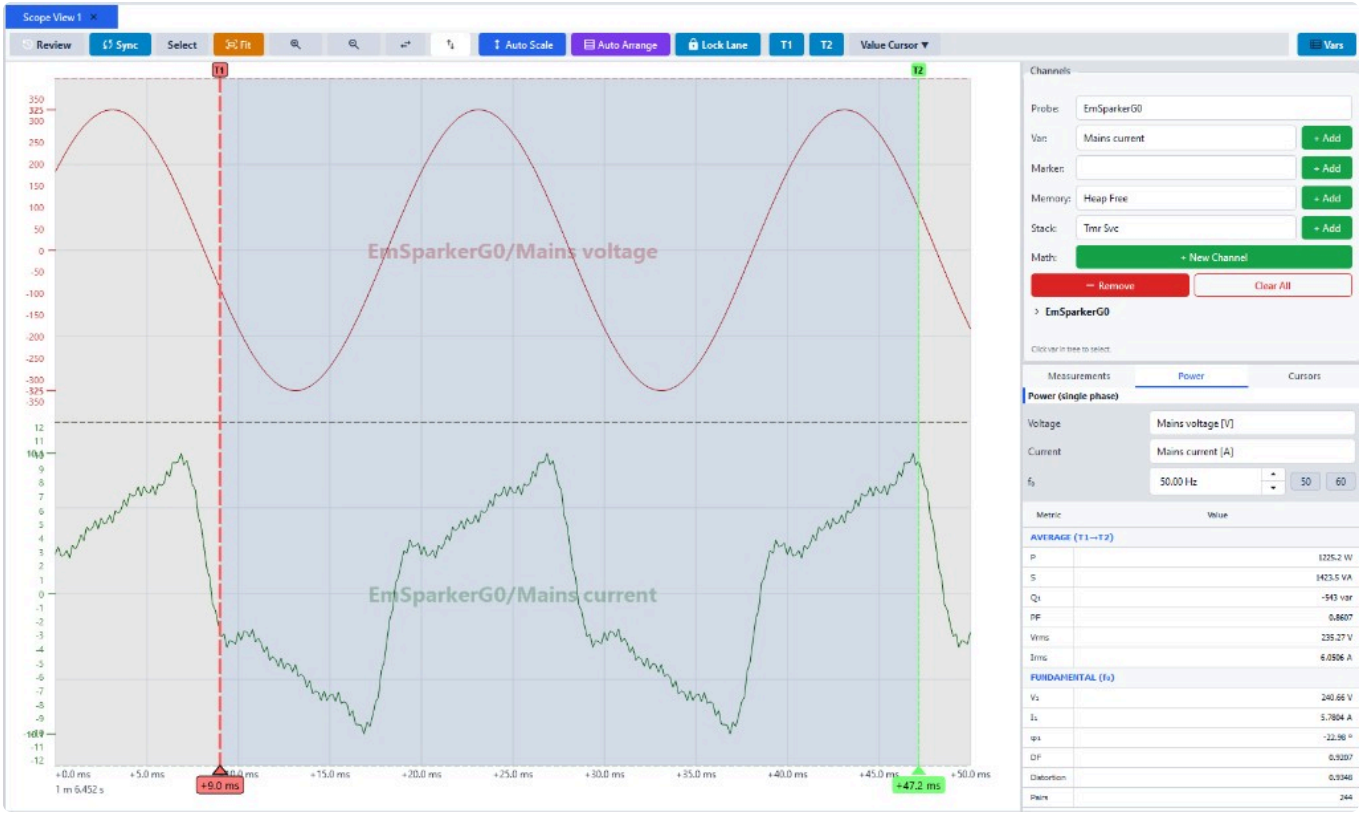
- **Measurements tab** — a metrics × channels matrix. As soon as both T1 and T2 are armed in Review mode, the *Statistics (T1→T2)* section automatically computes Mean, RMS, Min, Max, Pk-Pk, Freq, Period, and sample count over the cursor span, for every visible channel; the *Cursors* section below shows each channel's value at T1 and T2 and the Δ between them. The statistics recompute whenever a cursor moves, and clear back to — when a cursor is disarmed or the view returns to Live.

- **Power tab** — single-phase V/I metrics over the same T1→T2 window (P, S, PF, displacement factor, …). See Power analysis.
- **Cursors tab** — editable T1/T2 and V1/V2 tables: type exact time/value coordinates instead of dragging, useful for precise placement.

Power analysis

The **Power** tab computes IEEE-style **single-phase** numbers from a voltage/current pair over the armed T1→T2 span. It is the measurement companion to a math channel: `a * b` plots instantaneous power; this tab reports the windowed averages and the fundamental-frequency group.

Setup. Add both traces to the Scope, switch to **Review**, arm **T1** and **T2** around at least one line cycle, then on the Power tab pick **Voltage** and **Current** (the panel auto-picks channels whose name or unit looks like volts / amps). Set $\mathbf{f_0}$ to the declared line frequency — 50 Hz and 60 Hz shortcuts are on the row — used only for the fundamental group, not for P / S / PF.



Screenshot: Review mode, T1→T2 over ~2 cycles of a 50 Hz mains stream. Voltage is a clean sine; current is distorted. The Power tab reports P 1225 W, PF 0.86, and DF 0.92 — PF is lower than DF because the current is not sinusoidal.

How pairing works. Each voltage sample is matched to the nearest current sample within half a sample period. Channels of the same interleaved stream (Chapter 12) share timestamps and lock 1:1. A small ADC scan skew is tolerated; two unrelated async variables usually produce

few or no pairs. If a lane shows `[≈]`, zoom to RAW so the tab sees exact samples rather than a decimated overview.

Two groups fill in independently:

Average (T1→T2) — time-domain only; no spectrum.

Metric	Meaning
P	Active power: mean($v \cdot i$) [W]
S	Apparent power: Vrms · Irms [VA]
PF	Power factor: P / S (1.0 = no reactive or distortion penalty)
Vrms, **Irms**	True RMS over the paired samples

Fundamental (f_0) — a **single-bin Fourier coefficient** at the declared f_0, taken on the same paired samples. This is *not* the Analyzer's FFT and *not* a "tallest bin" tone detector.

Metric	Meaning
V_1, **I_1**	RMS of the fundamental
φ_1	$\angle V_1 - \angle I_1$, wrapped to (−180°, 180°]
DF	Displacement factor: $\cos(\varphi_1)$ — the classical *cos φ*
Q_1	$V_1\, I_1 \sin(\varphi_1)$ [var]; positive is inductive
Distortion	PF / DF — how much of the PF gap is harmonics rather than phase shift
Pairs	How many V/I samples were actually matched in the window

PF and DF answer different questions. **PF** is the full power factor (P/S) and falls when the current is either phase-shifted *or* distorted. **DF** (displacement factor, $\cos \varphi_1$) is only the fundamental phase shift. On a clean sine, PF ≈ DF; on a rectifier or inverter load, PF is lower than DF and **Distortion** = PF/DF drops below 1. You need about one cycle at f_0 (and at least eight pairs) for a stable fundamental group; averages can still be valid on a shorter window, and the tab warns when T1→T2 is less than one cycle.

(Power analysis is available on Professional licenses. The tab is visible on every license; without the entitlement it shows an upgrade page.)

Sync

With **Sync** on (Review mode), the Scope shares its T1/T2 span, cursor, hover, and visible range with all other synced views — place T1/T2 around a glitch and read the same interval in

the Trace View or Marker Stats. See Chapter 03 §5.

Rotary Motor Monitor (bonus)

For variables representing a motor/encoder position, the Toolbox also offers the **Rotary Motor Monitor** — a dashboard that derives shaft angle, RPM, angular velocity, acceleration, and direction from a single position variable. See Chapter 19.

Troubleshooting

Symptom	Likely cause
Channel named `VAR_0x1xxx`, no unit	No `TRAX_VAR_DEFINE` for that TID, or ELF missing/mismatched
Float values look like huge integers	Sent with `TRAX_VAR_SET` instead of `TRAX_VAR_SET_FLOAT`, or type in `TRAX_VAR_DEFINE` doesn't match
Jagged/aliased waveform on a fast signal	Async VARs are event-sampled; use a variable stream for fixed-rate signals (Chapter 12)
Filter dropdown empty	`TRAX_VAR_FILTER` not defined for the channel, or ELF not reloaded after adding it
Math channel rejected / "not a stream channel"	Inputs must be stream variables of the **same stream** on the same probe; async VARs cannot be paired
Math expression uses a slot that is not bound	Bind that letter with **Add variable**, or drop the unused slot from the expression
Power tab empty / "no paired V/I samples"	Review mode, T1 and T2 armed, both traces have data in the window, and they share timestamps (same stream). Zoom to RAW if the lane shows `[≈]`
DF / φ_1 / Q_1 show — but P and PF are filled	T1→T2 is shorter than one cycle at f_0 — widen the cursors; averages do not need a full cycle