15 — Markers

> **TraxProbe & Traxcope User Manual** · Version 1.0.0 · Updated 2026-07-28
 · https://embedya.com

Markers measure **named code intervals**: wrap a section in START/STOP and Traxcope computes every statistic on the host — min/max/mean, distribution, periodicity, deadline compliance. The device pays only two 12-byte frames per interval and holds **zero** statistics RAM. Markers are the tool for "does my control loop always finish within 600 µs?" — and with the JUnit export, for making that a CI-enforced requirement.

The same START/STOP frames are also a **Trigger Studio** source: freeze the timeline on marker start/stop, while the interval is open, or when a duration exceeds (or beats) a threshold. See Chapter 17 — Marker conditions.

On the device

```
/* TID registry (trax_marker_tid.h) — range 0x5000–0x5FFF */
enum trax_marker_tid_t {
    MARKER_TID_CONTROL_LOOP = TRAX_TID_RANGE_MARKER_USER_START,
    MARKER_TID_SENSOR_READ,
};

/* Optional metadata: tid, name, color, warn_us, deadline_us (0 = none) */
TRAX_MARKER_DEFINE(MARKER_TID_CONTROL_LOOP, "ControlLoop", TRAX_COLOR_ORANGE,
400.0f, 600.0f);
TRAX_MARKER_DEFINE(MARKER_TID_SENSOR_READ,  "SensorRead",  TRAX_COLOR_TEAL,
150.0f, 200.0f);

void control_loop_step(void)
{
    TRAX_MARKER_START(MARKER_TID_CONTROL_LOOP);

    TRAX_MARKER_START(MARKER_TID_SENSOR_READ);   /* markers can nest */
    read_sensors();
    TRAX_MARKER_STOP(MARKER_TID_SENSOR_READ);

    compute_control();
```

```
    TRAX_MARKER_STOP(MARKER_TID_CONTROL_LOOP);
}
```

Facts worth knowing:

- **Frames are minimal** — header + timestamp + TID, 12 bytes. STOP is the same TID with bit 15 set; the host pairs them.
- **Nesting works**, including the *same* TID nested (the host pairs with a stack); different markers can freely overlap.
- **Deadlines/warnings** declared in `TRAX_MARKER_DEFINE` are in microseconds and seed the compliance analysis; Traxcope can override them in the UI without reflashing (overrides persist in the session schema).
- Without metadata everything still works — the marker shows as `MARKER_0x5000` with no thresholds.
- ISR-safe; the timestamp **is** the measurement, so accuracy equals your timestamp resolution (Chapter 02 §2.2).

In Traxcope — the Marker Statistics view

Open via Toolbox → **Markers**.

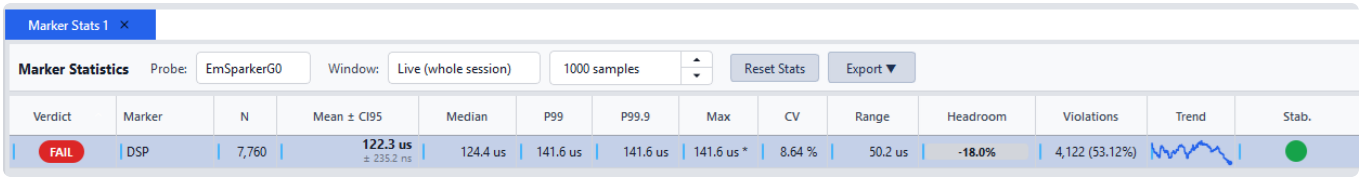
Analysis window

Choose what slice of data the statistics cover:

Window mode	Covers
Live	A rolling window over incoming data
Visible	Whatever the synced timeline currently shows
Last N	The most recent N intervals
Between Cursors	The T1–T2 span shared via view sync — place cursors in the Scope, get stats for exactly that span here

Overview table

One row per marker: count, min / mean / max / latest duration, deadline status. This is the at-a-glance health board for all instrumented sections.

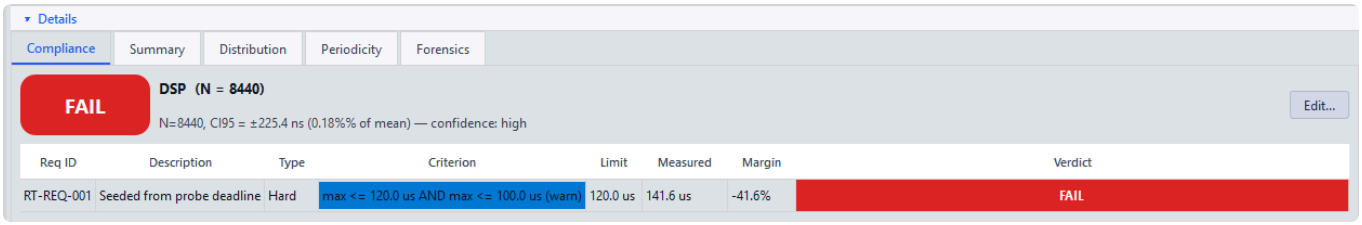


Screenshot: Marker Stats overview, several markers with min/mean/max and pass/warn/fail coloring

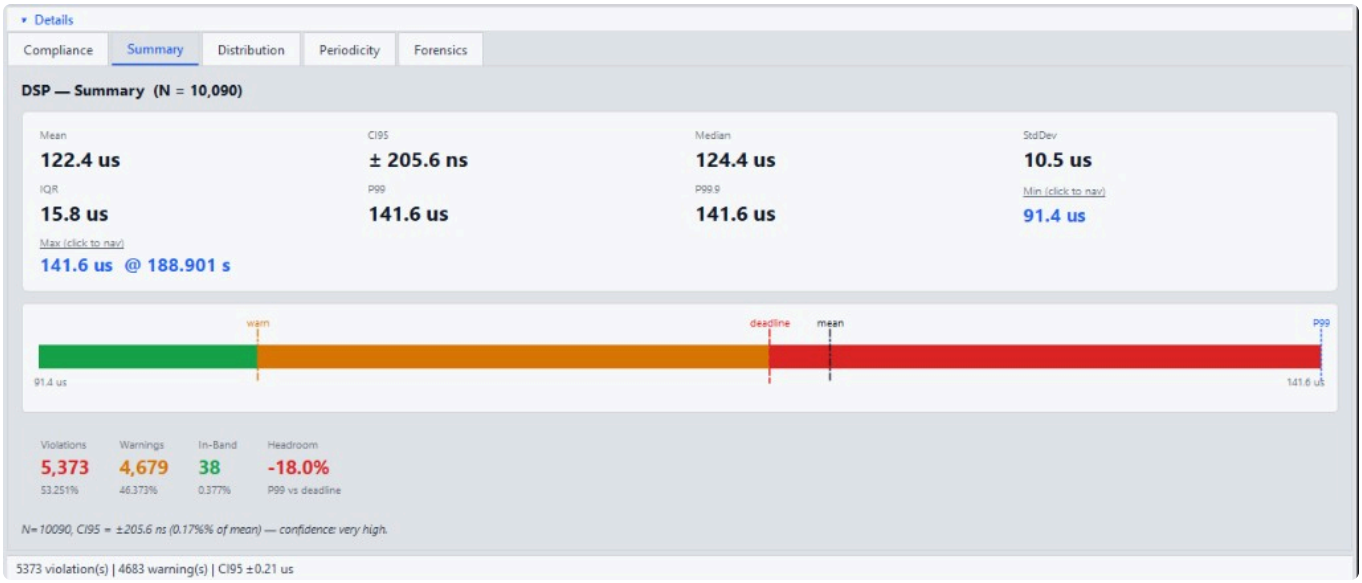
Detail tabs

Select a marker for five analysis tabs:

Tab	Question it answers
Compliance	Does the interval meet its deadline/warning thresholds? Pass/warn/fail counts, margin, worst offenders. Thresholds editable here (requirements editor)
Summary	Full descriptive statistics of the interval durations
Distribution	Histogram — is the timing tight, bimodal, long-tailed?
Periodicity	Interval *cadence* analysis — is the section invoked at a stable rate, what is the jitter?
Forensics	Drill into individual outlier intervals and navigate to them on the timeline

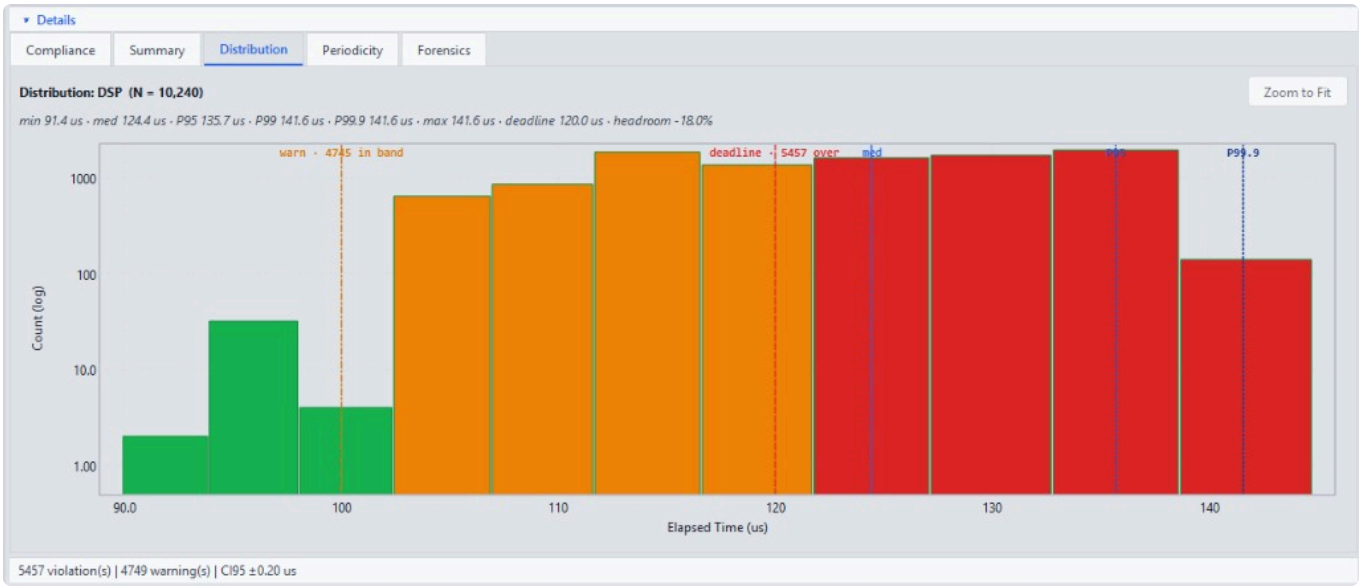


Screenshot: Compliance tab, requirement thresholds and verdict counts visible

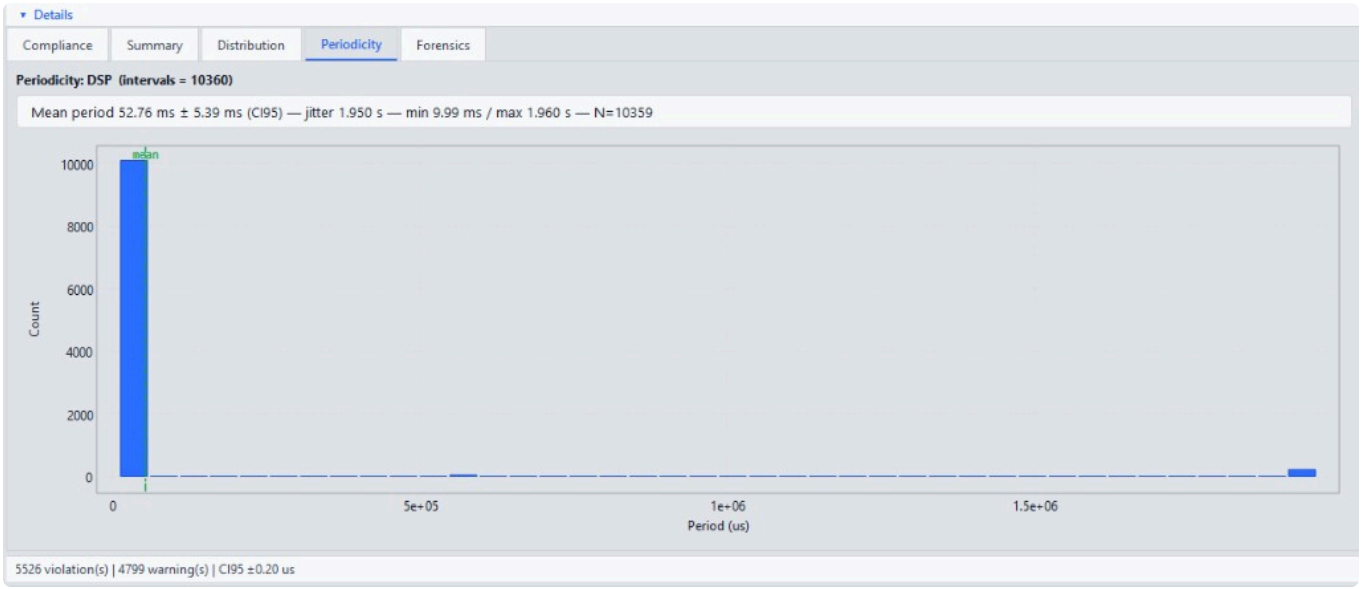


Screenshot: Summary tab — full descriptive statistics (mean ± CI95, median, σ, IQR,

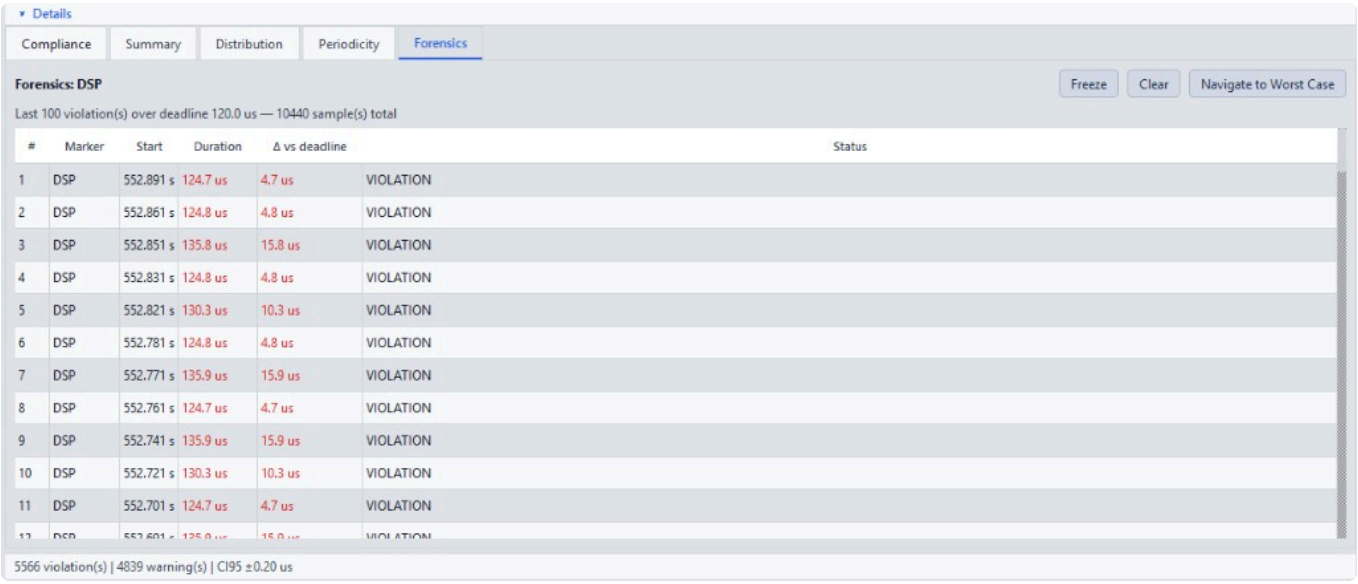
P99/P99.9), a min→P99 range bar against the warn/deadline thresholds, and violation / warning / headroom counters. Min and Max are click-to-navigate: they jump the synced views to that exact interval



Screenshot: Distribution tab — duration histogram (log count axis) colored by threshold band: green below warn, orange between warn and deadline, red over deadline. Median, P99 and P99.9 are marked; this marker's distribution sits mostly past its 120 µs deadline



Screenshot: Periodicity tab — cadence histogram of START-to-START periods with mean period, CI95 and jitter. Outlier bars far from the mean reveal missed or delayed invocations

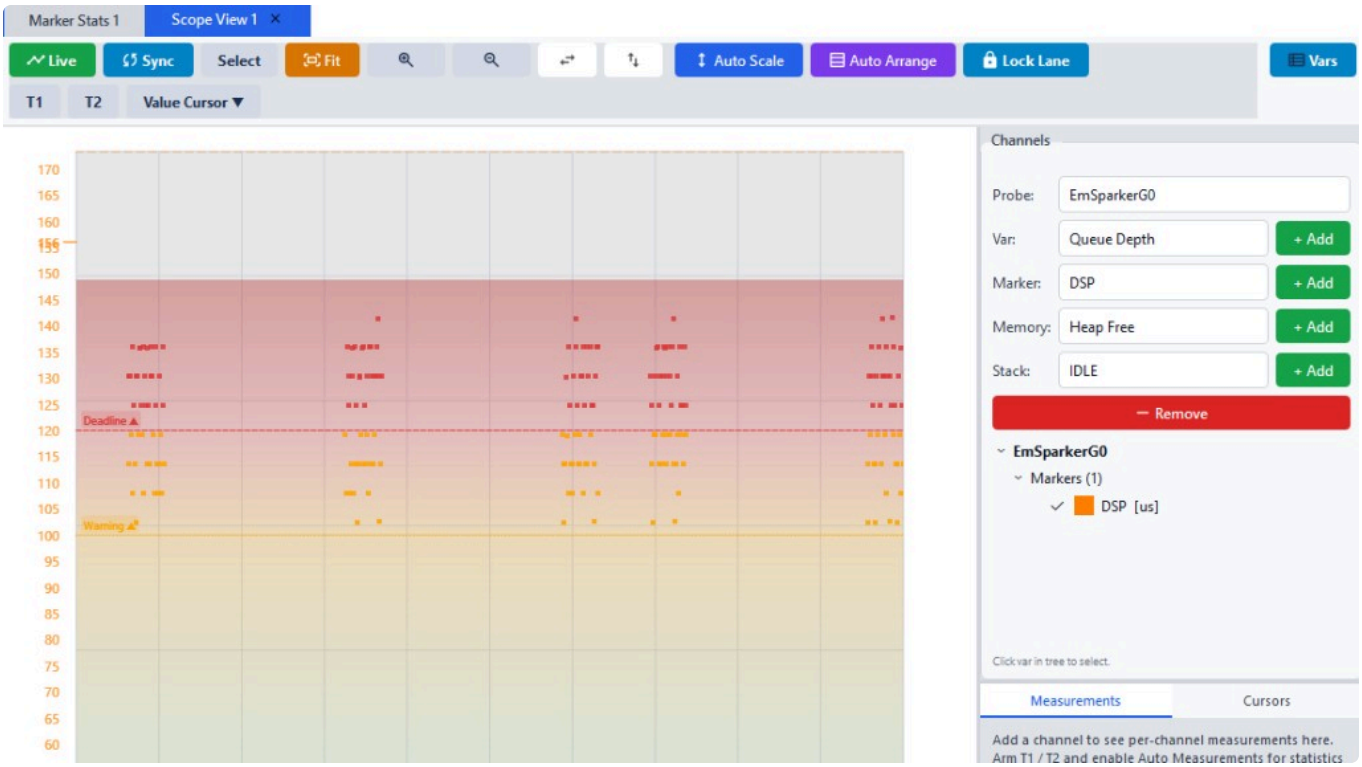


Screenshot: Forensics tab — the last 100 violations as individual rows (start time, duration, overshoot vs deadline). Freeze the list to inspect while data keeps streaming, or jump straight to the worst case

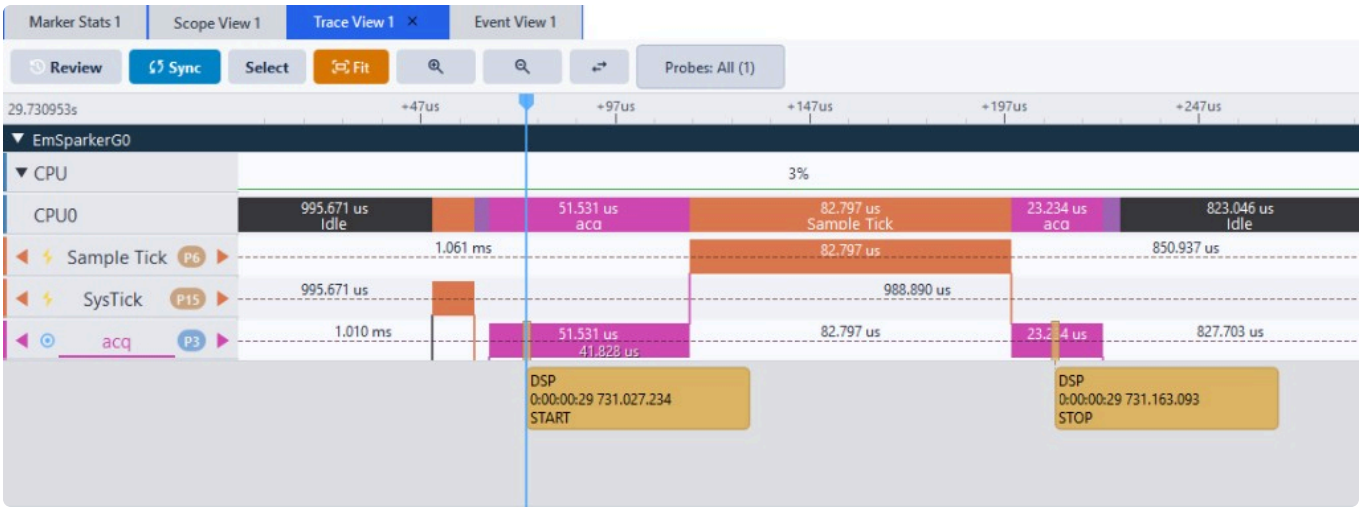
Cross-view navigation: from an outlier here, jump the synced Trace View to that exact interval and see *why* it ran long (preempting ISR? blocked on a mutex?).

Markers elsewhere

- **Scope View** — add a marker as a channel and each interval's **duration is plotted over time**, alongside your variables (Chapter 11). The plot shades the warning and deadline bands from the marker's thresholds, so violations are visible at a glance as dots inside the red band — the time-series companion to the statistics view.
- **Trace View** — marker START/STOP events appear as cards on the timeline, in context with task/ISR execution.



Screenshot: the DSP marker added as a Scope channel. Every interval becomes a point (duration in µs on the Y axis); the amber "Warning" line and red "Deadline" band come from the marker's thresholds, making deadline breaches and their clustering over time immediately visible

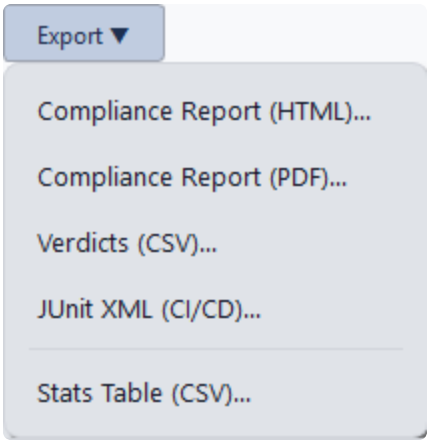


Screenshot: the same DSP marker on the Trace View — START and STOP cards pin the interval's exact boundaries onto the execution timeline. Here the story behind a long interval is plainly visible: the `acq` task's slice is split by the Sample Tick ISR (82.797 µs) running between the two cards — the marker ran long because it was preempted, not because its own code slowed down

Exports & CI integration

The Marker Stats view has the richest export set in Traxcope:

Format	Use
HTML report	Shareable timing report
PDF report	Formal documentation / sign-off
CSV (verdicts)	Per-interval results for your own tooling
JUnit XML	Feed into CI — each compliance requirement becomes a test case, so a firmware change that blows the control-loop deadline fails the pipeline
Stats table CSV	The overview table as data



Screenshot: Export menu showing HTML / PDF / CSV / JUnit options

Troubleshooting

Symptom	Likely cause
Marker missing from the view	No START/STOP pair arrived yet, or TID outside 0x5000–0x5FFF (compile error normally prevents this)
Intervals absurdly long	STOP never reached on some path (early return between START and STOP)
Compliance tab empty	No deadline set — define one in `TRAX_MARKER_DEFINE` or in the requirements editor
Stats differ from expectation	Check the window mode — "Live" vs "Between Cursors" cover very different data