16 — State Machines

TraxProbe & Traxcope User Manual · Version 1.0.0 · Updated 2026-07-28

Declare your state machine's states once, emit a 16-byte frame on every transition, and Traxcope renders a live state diagram plus per-state statistics. Up to 256 state machines per firmware (TID range 0x7000–0x70FF).

On the device

```
/* trax_config.h — must cover every SM TID index (plain integer literal) */
#define TRAX_CFG_SM_CNT  1

/* State indices — your own enum, 0-based */
enum acq_state_t { ACQ_IDLE, ACQ_SAMPLE, ACQ_PROCESS, ACQ_TRANSMIT, ACQ_ERROR
};

/* TID registry (trax_sm_tid.h) */
enum trax_sm_tid_t {
    SM_TID_ACQ = TRAX_TID_RANGE_SM_USER_START,
};

/* Metadata: tid, name, initial state, then one TRAX_SM_STATE per state */
TRAX_SM_DEFINE(SM_TID_ACQ, "Acquisition", ACQ_IDLE,
    TRAX_SM_STATE(ACQ_IDLE,     "IDLE",     TRAX_COLOR_BLUE),
    TRAX_SM_STATE(ACQ_SAMPLE,   "SAMPLE",   TRAX_COLOR_GREEN),
    TRAX_SM_STATE(ACQ_PROCESS,  "PROCESS",  TRAX_COLOR_ORANGE),
    TRAX_SM_STATE(ACQ_TRANSMIT, "TRANSMIT", TRAX_COLOR_TEAL),
    TRAX_SM_STATE(ACQ_ERROR,    "ERROR",    TRAX_COLOR_RED));

/* Runtime: one call per transition */
TRAX_SM_SET_STATE(SM_TID_ACQ, ACQ_SAMPLE);
```

- `TRAX_SM_SET_STATE` costs the same as a `TRAX_VAR_SET` (single atomic 16-byte frame) and is ISR-safe.
- The state *table* (names, colors, initial state) is compile-time metadata; the wire only ever carries the new state index.

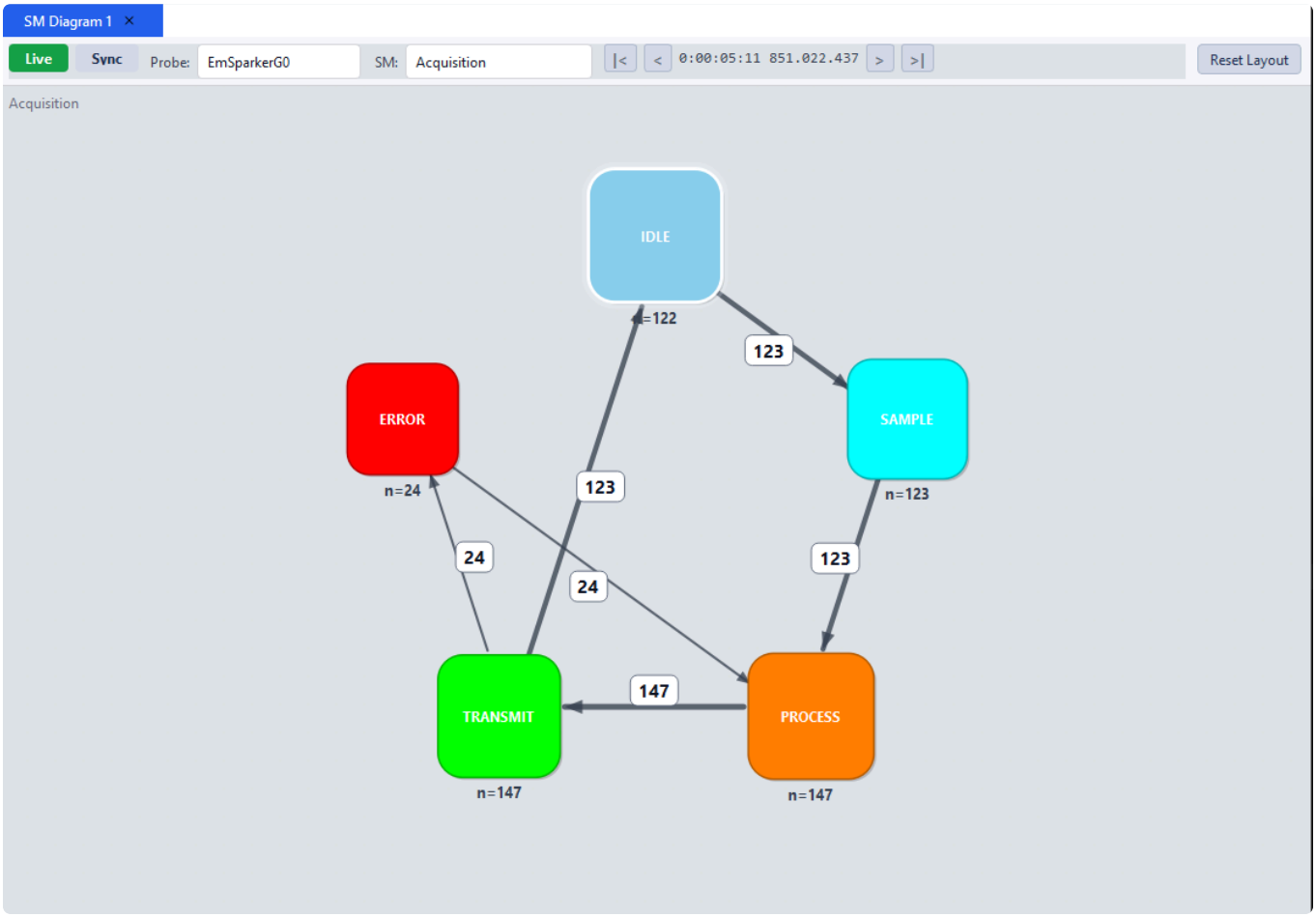
- The cleanest integration point is your state machine's own transition function — one `TRAX_SM_SET_STATE` there covers every transition forever.
- Traxcope derives transitions, durations, and statistics entirely from the sequence of SET_STATE events; the firmware keeps no state-machine bookkeeping for tracing.

In Traxcope

Two dedicated views, plus presence on the shared timeline.

State Machine Diagram

Open via Toolbox → **SM Diag**. Select the probe and the state machine; the view draws the states as nodes and the **observed transitions** as edges (the diagram is learned from the data — an edge appears the first time that transition happens).



Screenshot: SM Diagram in Live mode — states drawn with their firmware-defined colors, the active state (IDLE) highlighted, and the learned transition edges labeled with observed counts. `n=` under each node is its entry count; note the unplanned `TRANSMIT → ERROR` edge (24×) the data has exposed

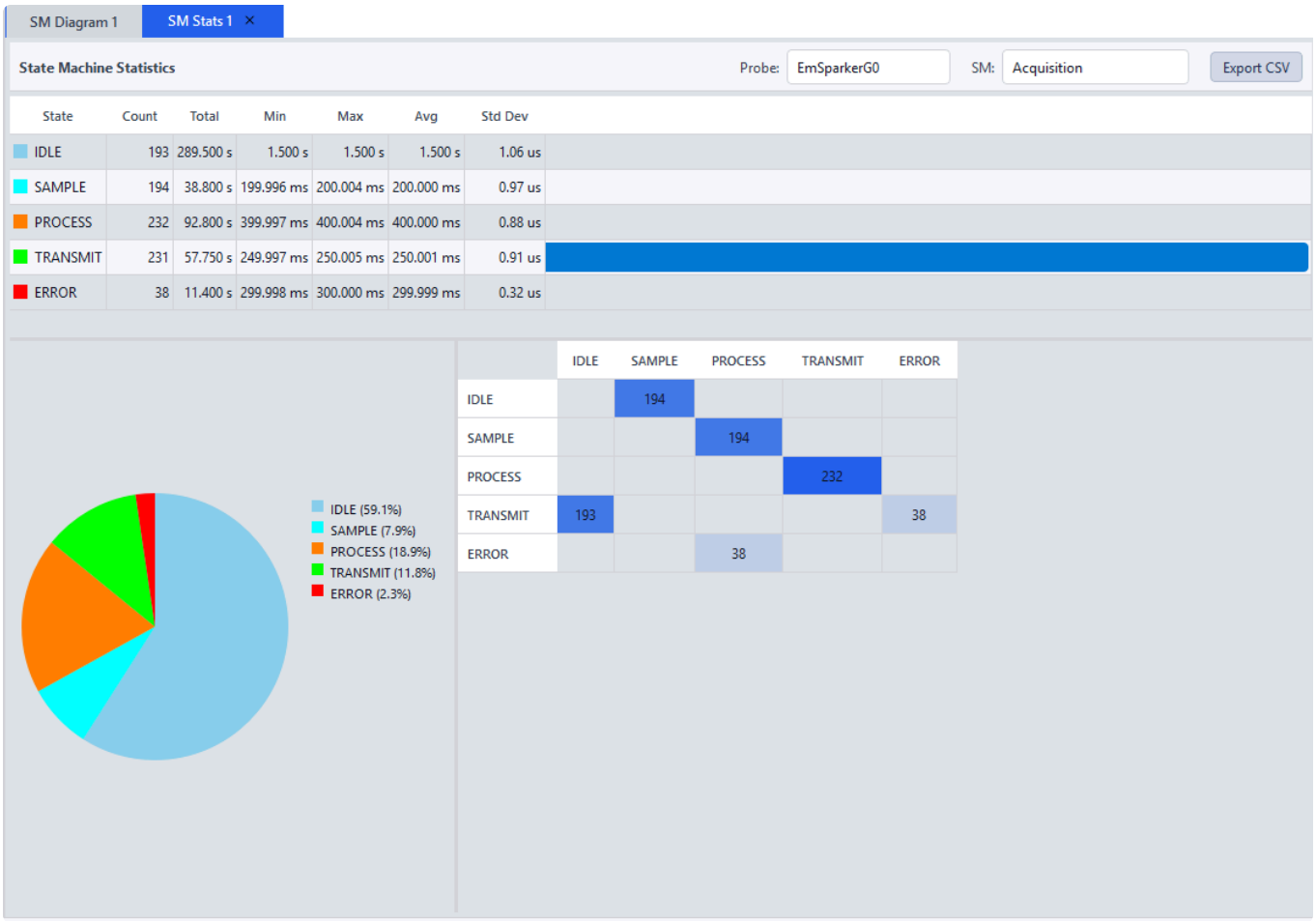
- **Live mode** — the active state is highlighted in real time; great on a second monitor during bring-up.

- **Review mode + Sync** — step the machine through its history transition by transition with the `|< < > >|` navigator (the toolbar shows the exact timestamp), or let a synced view drive it. The diagram shows one moment at a time, so a T1–T2 span selected elsewhere anchors it to the span's start.
- **The drawing itself is quantitative** — each edge is labeled with how many times that transition occurred (thicker = more frequent), `n=` below a state is its entry count, and a state's node grows with its total dwell time. Drag nodes to rearrange, pan/zoom for large machines, **Reset Layout** to re-run the automatic layout.

A transition you never designed showing up as an edge is a bug made visible — e.g. `ERROR → TRANSMIT` without passing through recovery.

State Machine Statistics

Open via Toolbox → **SM Stats**. The quantitative companion:



Screenshot: SM Stats, per-state table, time-share pie chart, transition matrix

- **Per-state table** — entry count, total/min/mean/max dwell time per state.
- **Pie chart** — share of time spent in each state ("why are we 40% in ERROR?").
- **Transition matrix** — counts of every from→to pair; unexpected non-zero cells are design violations, suspicious ratios are logic bugs.

- **CSV export** of the statistics.

On the shared timeline

State transitions are timestamped events like everything else, so they correlate naturally with other views through sync: select a span in the Scope where a signal glitches and check what state the machine was in; jump from an `ERROR` entry to the Trace View to see which task/ISR drove it there.

Troubleshooting

Symptom	Likely cause
Compile error `SM TID index >= TRAX_CFG_SM_CNT`	Increase `TRAX_CFG_SM_CNT` in `trax_config.h` so it covers every SM TID index
SM missing from the selector	No `TRAX_SM_DEFINE` found (check ELF / metadata mode), or no SET_STATE event arrived yet
States named by index only	Metadata not loaded — ELF path missing or stale
"Impossible" transition shown	It really happened — `TRAX_SM_SET_STATE` is called somewhere outside your transition function, or the logic has a hole