14 — ISR & Main-Loop Tracing

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

Interrupt handlers get their own swimlanes in the Trace View with two macros per ISR. On bare metal, two more macros give the main loop its own lane — so even RTOS-less firmware gets a real execution timeline.

On the device

ISR tracing

```
/* TID registry (trax_isr_tid.h) — user ISRs start at 0x4010 */
enum trax_isr_tid_t {
    ISR_TID_TIM7 = TRAX_TID_RANGE_ISR_USER_START,
    ISR_TID_DMA1_CH1,
};

/* Optional metadata: name + NVIC priority (for correct lane ordering) */
TRAX_ISR_DEFINE(ISR_TID_TIM7, "TIM7_SampleTick", 3);

void TIM7_IRQHandler(void)
{
    TRAX_ISR_ENTER(ISR_TID_TIM7);

    /* ... handler body ... */

    TRAX_ISR_EXIT(ISR_TID_TIM7);
}
```

- `TRAX_ISR_ENTER` / `TRAX_ISR_EXIT` emit minimal timestamped frames; the host pairs them into execution slices. Nesting across different ISRs renders correctly (preemption is visible).
- **Priority**: pass the raw NVIC value exactly as you programmed it via `NVIC_SetPriority()` — do not translate or invert. Traxcope knows the platform's priority direction (for Cortex-M, lower = more urgent) and sorts the lanes accordingly.
- Without `TRAX_ISR_DEFINE` the lane is named `ISR_0x4010`-style; everything still works.
- The RTOS kernel tick (SysTick under FreeRTOS) is traced automatically by the port under the reserved TID `TRAX_TID_ISR_TICK` — don't instrument it yourself. Bare-metal projects

may reuse that TID for their own SysTick handler (with their own `TRAX_ISR_DEFINE`).

Keep the macros as the very first and very last statements of the handler so the slice covers the true execution window.

When a FreeRTOS ISR ends with `portYIELD_FROM_ISR(woken)`, close it with `TRAX_ISR_END(tid, woken)` instead of `TRAX_ISR_EXIT`. That uses the same yield policy as the kernel tick: with `TRAX_CFG_ISR_YIELD_TO_SCHEDULER` (default `1`) Traxcope draws ISR → Scheduler → new task and omits the 2–5 µs Cortex-M resume of the preempted task. Details in Chapter 02 §5.

Main-loop tracing (bare metal)

Without an RTOS there are no context switches to delimit "the application ran here". Bracket your superloop instead:

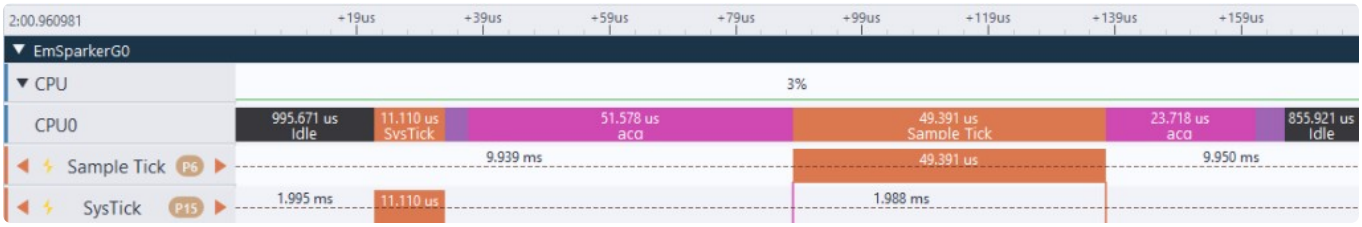
```
while (1) {
    TRAX_MAIN_LOOP_BEGIN();

    poll_sensors();
    run_control_step();
    update_outputs();

    TRAX_MAIN_LOOP_END();

    trax_process();
}
```

Each iteration becomes a slice in the Trace View's **Main** lane. Time spent in ISRs naturally appears as the corresponding ISR lanes, so you can see at a glance how interrupts carve up the loop period and how the iteration time evolves.

Under an RTOS both macros are no-ops — code that migrates from bare metal to FreeRTOS does not need editing.

In Traxcope

ISR and main-loop activity render in the **Trace View** (Chapter 13) alongside any task lanes:



Screenshot: CPU lane with two ISR lanes below it — "Sample Tick" (priority 6) and "SysTick"

(priority 15). Each slice shows its measured duration (e.g. 49.391 µs in the Sample Tick handler, 11.110 µs in SysTick), and the gaps between slices read directly as the interrupt periods (~9.9 ms / ~2 ms). The CPU lane shows exactly how the ISRs carve into the interrupted context.

- ISR lanes are ordered by hardware priority, named from `TRAX_ISR_DEFINE` metadata.
- Hover/select a slice for its exact duration — measure your worst-case ISR runtime directly, or use T1/T2 in a synced Scope to measure entry-to-entry periods.
- ISR enter/exit events also appear as rows in the **Event View** for exact ordering and parameter inspection.

Patterns to look for:

Pattern in the view	Meaning
ISR slice riding on top of a task/main slice	Preemption — the interrupted context resumes after EXIT
Back-to-back slices in one ISR lane	Interrupt storm / re-trigger before completion
Main lane iteration slices growing over time	Loop period creep — something inside is slowing down
Long gap in all lanes	Either idle (fine) or trace data loss (check the diagnostics card)

For measuring a *specific code section inside* a handler or the loop, use markers (Chapter 15) — ISR/main-loop slices show the envelope, markers measure named intervals against deadlines.

Troubleshooting

Symptom	Likely cause
Compile error "TID is outside the ISR range"	TID not anchored at `TRAX_TID_RANGE_ISR_USER_START` (0x4010)
ISR lane missing	Handler never ran, or ENTER/EXIT not both reached (early `return` between them breaks pairing)
Lanes sorted unexpectedly	Wrong priority passed to `TRAX_ISR_DEFINE` (must be the raw NVIC value)