TACH Sensing Progress Report (2026-01-09)

Scope

Goal: add an interrupt-driven TACH input (open-collector fan tach) to the TM4C1294XL firmware, with a command to print implied RPM periodically on UART0 (ICDI) while preserving the existing PWM behavior.

This report documents the state of implementation and the key lab caveats discovered so far.

Summary of what's implemented

1) GPIO TACH input + interrupt pulse counting

- TACH input is configured as a GPIO input with internal weak pull-up (`GPIO_PIN_TYPE_STD_WPU`, pulls to 3.3V).
- Falling-edge interrupts increment a pulse counter.
- Default pin chosen for convenience on the LaunchPad header: **PM3** (GPIOM3).

2) Periodic RPM reporting every 0.5s on UART0

- A `TACHIN` command on UART3 enables/disables periodic reporting.
- When enabled, firmware prints a line every 0.5s on UART0 (ICDI):

  `TACH pulses=<n> rejects=<n> rpm=<n>`

- RPM conversion used (per the project model):
  - `RPM = pulses_per_sec * 30`
  - Over a 0.5s window: `pulses_per_sec = 2 * pulses_in_window`
  - Therefore: `RPM = 60 * pulses_in_window`

3) Non-blocking timebase (SysTick)

- SysTick is enabled at 1ms tick to support clean 0.5s scheduling.
- A derived 32-bit cycle counter (based on SysTick) is provided for short delta measurements.

4) Glitch/noise mitigation added after abnormal lab readings

Observed in lab: extremely large implied RPM values (~1,000,000 RPM equivalent) that do not track PSYN intuitively.

Interpretation: tach input was likely seeing **PWM-coupled noise** or **fast glitches**, causing the edge counter to count events near tens of kHz rather than the true tach signal.

Mitigation implemented:

- ISR-level **minimum edge spacing** filter.
- Any edge arriving closer than `TACH_MIN_EDGE_US` (default 200µs) is rejected and counted in `rejects`.

This should suppress coupling from the ~21.5kHz PWM regime (period ~46µs).

Files and integration points

- `tach.c/.h`
  - `tach_init()` configures PM3 + interrupts
  - `GPIOMIntHandler()` counts pulses (with glitch reject)
  - `tach_task()` prints to UART0 every 0.5s when enabled
- `timebase.c/.h`
  - SysTick ms counter + `timebase_cycles32()`
- `commands.c`
  - Adds `TACHIN [ON|OFF]` command
- `main.c`
  - Calls `timebase_init()` and `tach_init()` once
  - Calls `tach_task()` inside the active session loop
- `TM4C1294XL_startup.c`
  - Hooks vectors for SysTick and GPIO Port M

Electrical and measurement caveats (must-read)

1) Voltage level / pull-up

- TM4C GPIO input is **3.3V tolerant** (do not apply 5V).
- If the fan tach is open-collector and was previously pulled up to **+5V** via 4k7, do **not** feed that directly into PM3.
- Preferred approach for bring-up:
  - Remove +5V pull-up
  - Use internal WPU (3.3V) or add external pull-up to **3.3V** (often stronger than internal)

2) EMI coupling from PWM

- PWM at ~21.5kHz can capacitively/inductively couple into a high-impedance open-collector tach line.
- Long wires and breadboard jumpers can worsen this.
- Cheap scopes can show "aliasing/mangling" that makes tach edges appear distorted.

3) What the diagnostics mean

- `pulses` is the number of accepted falling edges in the last 0.5s.
- `rejects` is the number of edges rejected for being too fast (likely noise).

Expected qualitative behaviors:

- If `rejects` is very large: noise dominates → consider stronger 3.3V pull-up and/or RC filtering.
- If `pulses` stays near 0: no edges → check wiring, common ground, pin selection, or fan tach availability.

Next lab checklist (for 2026-01-10)

1) Verify PM3 wiring, and ensure **no +5V pull-up** is present.
2) Confirm fan ground and LaunchPad ground are common.
3) Run `TACHIN ON`, then step PSYN through a few values.
4) Record a few lines of UART0 output including both `pulses` and `rejects`.

5) If needed, tune the glitch filter (e.g., try 100–1000µs) by changing `TACH_MIN_EDGE_US`.

Additional Notes

New lab notes (2026-01-10) indicate the physical IBM PS tach line is *bursty* (bursts of ~21.5kHz pulses followed by a low tail), which can alias badly with naive fixed-window counting. See: LEEME_MOSA_TACH_ANALYSIS.TXT

Plan (based on lab notes)

Add a `TSYN ON|OFF` mode that drives **PM3** with a clean, synthesized "tach-like" burst waveform whose parameters are interpolated from the lab table vs the currently requested `PSYN n`. Implement the 21.5kHz carrier using a hardware timer output, and only use interrupts at burst boundaries (low interrupt rate). While TSYN is ON, disable tach capture interrupts on PM3 to avoid self-triggering. —

Generated: 2026-01-09