flOMNI real-time loop settings (beamline scientists / local contact only)

Not for users. `flomni.rt_settings` changes how the FZP scanner, the detector trigger and the laser tracker work, and `save_as_default()` makes a change the default of every following loop start. A wrong combination can spoil the data of a beamtime. The user guide only says that these settings exist, are set by the local contact and are stored with every scan; the details are here.

Background and reference (flOMNI control repository, `OMNY/flOMNI` on gitea):

- `FeedbackLoop/loopsettings/README.md` — every setting with the loop parameters it sets and its production value, the rules, the profiles; `FeedbackLoop/loopsettings/flomni.json` — the definition (help texts, rules, profiles)
- `FeedbackLoop/tuning_guide.md`, `FeedbackLoop/feedback_report.md` — characterization, measurements, why the values
- RT-system `docs/loop-settings.md` — design, the web HMI side, installation
- the FZP how-tos next to this file (`local_contact_only-howto_calibrate_the_fzp_scanner.md`, `local_contact_only-howto_limit_the_fzp_scanner_range.md`) — move-beyond, FZP range, calibration steps as profiles

What can be set

Group	Settings
FZP scanner (only between scans)	`fzp_enable` (both axes or none), `fzp_range_V_x` / `_y`, `feedforward_enable`, `fzp_ff_priority` (feedforward first, 1 in production and all scan profiles since 2026-10-26), `move_beyond_um_x` / `_y`, `exposure_during_motion`, `abbe_enable`, `abbe_limit_um`
Sample stage moves (only between scans)	`target_shaper_scale`, `jump_max_um_x` / `_y`
Detector trigger	`trigger_tolerance_nm`, `trigger_hold_ms`, `trigger_min_delay_ms`, `trigger_on_move`
Trigger source (only between scans)	`trigger_soft` (never saved, set by the Fermat scan), `soft_exposure_ms`, `soft_latency_ms`
Laser tracker and thresholds	`tracking_target_y` / `_z`, `psd_threshold_low`, `interferometer_threshold`, `interferometer_threshold_rot`
FZP calibration (expert, only between scans)	`sample_hold_fzp_V`, `sample_hold_err_um`, `sample_hold_zero`

`flomni.rt_settings.help("all")` explains every setting, `help("<setting>")` one setting (what it does, range, live and saved value, the loop parameters it sets, its rules).

Commands

- `flomni.rt_settings.show()` — all settings with live and saved value, violated rules, scan state (`show(expert=True)` also the FZP calibration group)
- `flomni.rt_settings.set(name=value, ...)` — change; refused with the reason if a rule objects (e.g. the feedforward must be off while the FZP loop is off); `set(..., adjust=True)` changes the dependent settings the rule names (never the ones given)
- `flomni.rt_settings.check(name=value)` — dry run

- `flomni.rt_settings.profiles()`, `apply_profile("production" | "fzp_off" | "fzp_calibration" | "high_res" | "high_res_1v5" | "normal" | "low_res")` (scan modes 2026-10-26: flOMNI `FeedbackLoop/scan_modes.md`)
- `flomni.rt_settings.unsaved()` — what differs from the loop's parameter files
- `flomni.rt_settings.save_as_default()` — writes the live values into the loop's parameter files: the next loop start uses them. Only for a change that should stay (a backup of each file is kept on the loop machine).
- `flomni.rt_settings.help()`, `help("all")`, `help("<setting>")`

Changes are live only until the next start of the control loop, unless saved. FZP settings can only be changed between scans (refused while a scan runs); trigger and tracker settings any time.

Detector trigger

In step scans the detector is triggered when the relative position sample vs FZP (interferometer minus FZP cap sensor) stays within the tolerance in x and in y (each averaged over 0.5 ms) for the hold time.

- `flomni.rt_settings.trigger_show()` — tolerance, hold time, minimum delay, trigger after a move
- `flomni.rt_settings.trigger_set_tolerance_nm(10)` — about 30 for a fast low-resolution scan, 10 standard, 2 for the highest resolution (a few ms more per point); 0 = the former criterion (on-target flags of the FZP loops)
- `flomni.rt_settings.trigger_set_hold_ms(3)` — hold time in ms
- `flomni.rt_settings.trigger_set_after_move(True)` — also one trigger after every position change of the sample stage (e.g. `umv(dev.rtx, ...)`, `dev.rtx.controller.move_xyz(x, y, z)`) once on target; `False` (default): scan points only

The former `flomni.interferometer_trigger_*` were removed 2026-10-23.

Soft trigger (working without detector)

With the soft trigger no trigger leaves the loop (the detector trigger output stays 0) and the loop emulates the detector: every trigger starts an exposure of `soft_exposure_ms` after `soft_latency_ms` (1 ms). Scans then run without detector or trigger PLC. The loop always starts with the hardware trigger; `trigger_soft` is never saved as default and is in no profile (flOMNI `FeedbackLoop/soft_trigger.md`).

`flomni_fermat_scan` sets the trigger source at every scan start (2026-10-26):

- **At the beamline → hardware.** At the beamline means: `ddg1` is configured with a real (not simulated) device class (only `ddg1` decides, 2026-10-27). If this cannot be confirmed (web HMI not reachable, refused) the scan is aborted: with the soft trigger the detector would not be triggered while flOMNI exposes internally. A failing check also counts as beamline.
- **Otherwise (simulated or no `ddg1`) → soft** with `soft_exposure_ms` = `exp_time` × `frames_per_trigger` × 1000. If that fails the scan continues with a warning (alarm).

By hand: `flomni.rt_settings.trigger_set_soft(True, exposure_ms=100)`, `trigger_set_soft(False)`; `trigger_show()` shows both groups.

Laser tracker

`flomni.rt_settings.tracking_show()`, `tracking_set_target_y(v)` / `tracking_set_target_z(v)`, `tracking_set_interferometer_threshold(v)`, `tracking_set_psd_threshold_low(v)` and the arrow-key tweak `flomni.rt_laser_tracker_tweak()` use the same settings (`tracking_target_y` / `_z`,

`interferometer_threshold`, `psd_threshold_low`). The former `flomni.laser_parameters_*` were moved here 2026-10-26.

Stored with every scan

The baseline device `flomni_loop_settings` is read at the start of every scan: one value per setting, `available` (0 when the loop machine could not be reached — the scan continues), the number of settings not saved as default or inconsistent, violated error rules, the definition with its sha1, and a JSON of all settings. `dev.flomni_loop_settings.show()` prints what a scan would store now.

Connection

`flomni.rt_settings` sends its requests through the device `flomni_loop_settings` (device server). The device talks to the web HMI on the loop machine (`http://mpc2844:8080`, REST `/api/settings`; port 80 as fallback) with the API token in its device config (`ptycho_flomni.yaml`, `token:`; the line of `~/loop-data/webhmi/api_token` on mpc2844). Without that device (e.g. a simulated config) the client talks to the web HMI directly with `~/.flomni_webhmi_token`. The web HMI tab *Loop settings* shows and changes the same settings with the same rules.