# atomic_clock: long-term stability tests

Testing manual for Allan-deviation measurements of the simulated clock

Rok Žitko
Jozef Stefan Institute, Ljubljana, Slovenia
rok.zitko@ijs.si

Red Pitaya `atomic_clock` design, version 3.2 with stability recordings
October 28, 2026

**Abstract**

The `atomic_clock` design makes a 10 MHz sine on OUT1 whose phase can carry programmed noise with the power-law spectrum of real oscillators (white and flicker phase noise, white, flicker and random-walk frequency noise, drift, a sinusoidal modulation), against an unmodulated 10 MHz reference on OUT2. With OUT1 and OUT2 cabled back to IN1 and IN2, the board's own phase meter records the difference for hours to days. This manual describes how to record, how to analyze the records on a computer (Allan, modified Allan, Hadamard and time deviations, spectra and a fit of the noise coefficients, also live while the data arrive), and a series of tests: for each one the settings with suggested ranges, the duration, the plots to expect and the criteria for a pass, and applications of the meter to external clocks, loops, cables and amplifiers.

## Contents

**1 What is measured** **2**

**2 Safety, equipment and cabling** **3**

**3 Prerequisites** **3**

**4 Definitions: the IEEE 1139 standard** **4**

**5 Reading Allan-deviation plots** **4**
5.1 The deviations . . . 4
5.2 Slopes and levels . . . 5
5.3 Dither . . . 5
5.4 Confidence intervals . . . 6
5.5 Spectra . . . 6

**6 Running a test** **6**
6.1 By hand . . . 6
6.2 With the test runner . . . 7
6.3 The floor . . . 7
6.4 Verdicts . . . 9

**7 The tests** **9**
7.1 Reading the prediction figures . . . 10
7.2 T0a: split floor (meter only) . . . 11
7.3 T0b: loopback floor . . . 11
7.4 T1: white PM . . . 11
7.5 T2: flicker PM . . . 11

7.6 T3: white FM . . . 13
7.7 T4: frequency offset and a phase step . . . 13
7.8 T8: sinusoidal phase modulation . . . 14
7.9 T5: flicker FM . . . 14
7.10 T6: random-walk FM . . . 15
7.11 T7: linear frequency drift . . . 16
7.12 Clock models . . . 16

**8 Oscillator models (M runs) 17**

**9 Measured results 20**

**10 Further estimators and tools 21**
10.1 Long-term estimators: Totdev and ThêoBR . . . 21
10.2 Time interval error: MTIE, TIE rms and ITU-T masks . . . 22
10.3 Data from other instruments . . . 22
10.4 Three-cornered hat: separating three clocks . . . 24
10.5 Dynamic Allan deviation . . . 24
10.6 Quick tables in adev1 format . . . 24
10.7 Drift models . . . 25
10.8 Live estimation: rustle . . . 25

**11 Applications of the phase meter 27**
11.1 Testing a phase-locked oscillator . . . 27
11.2 Measuring the board's crystal . . . 28
11.3 Comparing two external clocks . . . 30
11.4 Cable delay against temperature . . . 30
11.5 Residual phase noise of a distribution amplifier . . . 32

**12 Troubleshooting 32**

**13 Record sheet 33**

**A Model details 34**

# 1 What is measured

**Signals.** OUT1 is the "clock under test". Its phase is $\phi_1(t) = 2\pi\nu_0[t + x(t)]$ with $\nu_0 = 10\,\mathrm{MHz}$, where the *time error* $x(t)$ (seconds) is what the settings program. OUT2 is the reference with $x = 0$. Both come from the same 125 MHz sample clock, so the board's own crystal drops out of the difference: the measurement shows the programmed $x(t)$ plus the floor of the chain (outputs, cables, inputs, temperature; Section 6.3).

**The phase meter.** IN1 and IN2 are demodulated at exactly 10 MHz in windows of $\tau_w$ (default 10 ms). Each window gives the phase of each input; their difference, divided by $\nu_0$, is one sample of $x$ averaged over the window. The fractional frequency is $y = \mathrm{d}x/\mathrm{d}t$. The 25-entry sine/cosine tables sum to zero, so DC rejection is exact; cancellation of the $2f$ product is only approximate with the quantized tables. Amplitudes in volts use the board's input calibration and configured LV/HV ranges, which must match the physical jumpers. Changing the raw meter window restarts its index but preserves old FIFO entries: low-level users must issue METER CLEAR before reading with the new window.

**A recording** keeps two series (Table 1): every window for the first hour (the *fast* series, for $\tau$ from 10 ms to minutes) and one window per second for the whole run (the *slow* series, for $\tau$ from 1 s to days), plus the amplitudes, the board temperature and the complete settings.

| series | spacing $\tau_0$ | used for |
|---|---|---|
| fast (`fast.csv`) | every window, 10 ms | $\tau \leq$ one tenth of its length (6 min for 1 h) |
| slow (`slow.csv`) | first window of each second | $\tau$ from 1 s to one tenth of the run |

**Table 1:** The two series of a recording. Both are averages over one meter window; the slow series is a point sample once per second, not an average over the second.

**Where each noise term is made.** White PM ($h_2$) and white FM ($h_0$) are generated in the FPGA at every sample (8 ns). Flicker PM ($h_1$), flicker FM ($h_{-1}$), random-walk FM ($h_{-2}$), drift and the sinusoidal PM are computed by the board's service and *played back* as 1 ms segments; they contain only Fourier frequencies below $f_{\text{seg}}/2 = 500$ Hz and need `playback.source = "model"`.

## 2 Safety, equipment and cabling

Read "Before connecting equipment" in the repository's `README.md` first. The outputs reach about $\pm 1.13$ V at DC (at 10 MHz the set amplitude is the level into 50 Ω; a high-impedance load sees about twice it); the inputs take at most about $\pm 1$ V with the LV jumpers (the tests use 0.5 V amplitude on both outputs, the power-up level).

**Loopback (all tests except T0a):** OUT1 → IN1 and OUT2 → IN2 with two coaxial cables of equal type and length, connected the same way (no terminator, or one on each input: a 50 Ω terminator shifts that input's phase by about 0.1 cycle; amplitudes read about 0.67 V without, 0.5 V with). Keep them still and away from air flow: a cable's delay changes by tens of femtoseconds per kelvin per metre.

**Split floor (T0a):** OUT2 through a BNC T to both IN1 and IN2 (two equal short cables); OUT1 unconnected. This measures the meter alone.

**Environment:** a closed box or at least a place without drafts; note the room temperature. The recording logs the Zynq die temperature (it follows the room with an offset of 30–40 K).

## 3 Prerequisites

- The board runs `atomic_clock` 3.1 or later with the stability recordings (`REC` commands; the live dashboard needs the 3.2 service with `MEAS STREAM`) and its service `rp-overlay.service`; check with `STATUS?` or the notebook's version line.
- Board calibration done (`common/CALIBRATION.md`); not critical for phase, but amplitudes are reported in volts.
- At least 1 GB free on the board's SD card (a one-day recording takes about 40 MB; the fast hour about 34 MB).
- On the computer: Python with numpy, scipy, matplotlib and allantools, e.g. the conda environment `base-tools`. The tools are in `atomic_clock/tools/`. The live dashboard (Section 10.8) also needs fastapi and uvicorn.
- Ubuntu's scheduled jobs on the board disabled (`common/SETUP.md`, "Stop the scheduled maintenance jobs"): they can stall the service for seconds.
- NTP on the board is *not* needed: times come from the FPGA sample counter.

# 4 Definitions: the IEEE 1139 standard

**The standard.** IEEE Std 1139, *IEEE Standard Definitions of Physical Quantities for Fundamental Frequency and Time Metrology — Random Instabilities*, is the reference document for describing the stability of oscillators and clocks [1]. First issued in 1988 and revised in 1999, 2008 and 2022 (the current edition, IEEE Std 1139-2022, from the IEEE Ultrasonics, Ferroelectrics, and Frequency Control Society), it collects the definitions that grew out of the work of the 1960s and 1970s [2, 3, 4] into one agreed vocabulary.

**What it defines.** Methods of describing *random* instabilities of importance to frequency and time metrology: the model of a signal with phase and amplitude fluctuations; the phase deviation $\varphi(t)$, the time deviation $x(t)$ and the fractional frequency deviation $y(t)$; the one-sided spectral densities of phase, frequency and amplitude fluctuations ($S_\varphi$, $S_y$, $S_x$) and the phase-noise quantity $\mathcal{L}(f) = S_\varphi(f)/2$ (for oscillators, see [5]); the power-law model $S_y(f) = \sum_\alpha h_\alpha f^\alpha$; and the time-domain measures of frequency instability, built on the Allan variance, with annexes on translating between frequency- and time-domain measures, worked examples of computing them, and a bibliography.

**What it asks of a measurement.** Besides the definitions it gives recommendations for *reporting* measurements of frequency, amplitude and phase instability: the experimental parameters (for example the measurement bandwidth, the sampling interval and the amount of data, on which the time-domain measures of phase noise depend), the experimental conditions, and the calculation techniques used. Its subject is the *random* instabilities; deterministic effects such as a frequency offset or a drift are characterized separately (here: fitted and reported apart from the noise).

**Who it is for.** Anyone who specifies, compares or tests oscillators, clocks and frequency-distribution systems: manufacturers writing data sheets, users comparing them, and metrology and research laboratories reporting results. Its value is that a number such as $\sigma_y(1\,\mathrm{s}) = 10^{-12}$ or $\mathcal{L}(10\,\mathrm{Hz}) = -120\,\mathrm{dBc/Hz}$ means the same thing to everyone.

**What it means for this design.** Every quantity used here — the noise settings $h_2 \ldots h_{-2}$, the drift, $S_y(f)$ and $\mathcal{L}(f)$, and the Allan, modified Allan, Hadamard and time deviations of the reports — follows the definitions of IEEE Std 1139. Two consequences for users:

- a clock can be simulated directly from a data sheet or a paper: enter its $h_\alpha$ (or the ADEV levels they give, Table 2) and drift, and the board produces noise with those statistics;
- results measured with this board and its tools can be compared directly with data sheets and the literature, and reported in the standard's terms.

Where the measurement differs from the textbook case it is stated: the meter averages over windows instead of using a sharp bandwidth $f_h$ (Section 5, appendix), the demodulator reads broadband white PM with 1.5 times the variance, and played-back noise reaches only $f_{\mathrm{seg}}/2$. The reports record the parameters the standard asks for (window, sampling intervals, length, confidence intervals, settings, temperature).

# 5 Reading Allan-deviation plots

## 5.1 The deviations

For averaging time $\tau = m\tau_0$:

**ADEV** $\sigma_y(\tau)$, the overlapping Allan deviation [2, 3]: the standard measure of frequency stability, from second differences of $x$. Blind to a constant frequency offset; a linear drift $D$ adds $D\tau/\sqrt{2}$.

**MDEV** $\operatorname{mod}\sigma_y(\tau)$ [6]: like ADEV but averages the phase over $\tau$ first. It separates white PM (slope $-3/2$) from flicker PM (slope $-1$), which ADEV cannot (both $\approx -1$).

**HDEV** $H\sigma_y(\tau)$, the Hadamard deviation [7]: from third differences. Blind to linear drift, so it shows the noise underneath a drifting clock (rubidium, quartz).

**TDEV** $\sigma_x(\tau) = \tau \operatorname{mod}\sigma_y(\tau)/\sqrt{3}$ [8]: the time stability in seconds; a flat TDEV means a phase-noise floor, slope $+1/2$ white FM.

## 5.2 Slopes and levels

Each power law $S_y(f) = h_\alpha f^\alpha$ has its own ADEV slope (Table 2, Figure 1). A measured curve is a sum of straight pieces on log axes; reading the slopes from short to long $\tau$ identifies the terms, and the level of each piece gives its $h$.

| noise | $\alpha$ | ADEV slope | MDEV slope | $\sigma_y^2(\tau)$ | set by |
|---|---|---|---|---|---|
| white PM | 2 | $-1$ | $-3/2$ | $3f_h h_2/(4\pi^2\tau^2)$ | `h2` (FPGA) |
| flicker PM | 1 | $-1$ | $-1$ | $h_1[1.038 + 3\ln(2\pi f_h\tau)]/(4\pi^2\tau^2)$ | `h1` (playback) |
| white FM | 0 | $-1/2$ | $-1/2$ | $h_0/(2\tau)$ | `h0` (FPGA) |
| flicker FM | $-1$ | 0 | 0 | $2\ln 2\, h_{-1}$ | `h_1` (playback) |
| random-walk FM | $-2$ | $+1/2$ | $+1/2$ | $(2\pi^2/3)\, h_{-2}\tau$ | `h_2` (playback) |
| linear drift $D$ | – | $+1$ | $+1$ | $D^2\tau^2/2$ | `drift_per_day` |

**Table 2:** Power-law noise (IEEE 1139 [1]; [3, 7]). $f_h$ is the measurement bandwidth: for this meter $f_h \approx 1/(2\tau_w)$ for white PM (the window averages it), and 500 Hz for the played-back flicker PM. The analysis uses the exact meter model (window filter, the 500 Hz cut-off, the two series' spacing), not these asymptotes.

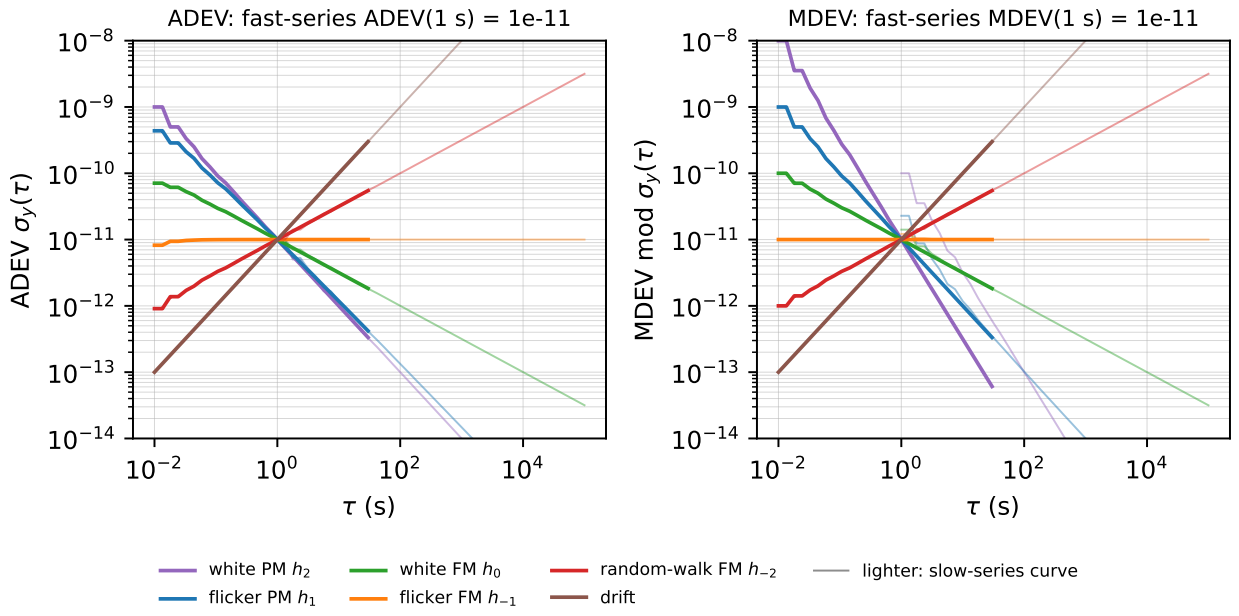



**Figure 1:** Each term alone, normalized separately in each panel to fast-series ADEV (left) or MDEV (right) $= 10^{-11}$ at 1 s. Thus the panels use different $h$ values. Through the meter model, for the fast series ($\tau_0 = 10$ ms, up to 30 s) and the slow one ($\tau_0 = 1$ s, from 1 s), as in the reports. Where they overlap only white PM's MDEV differs: it depends on $\tau_0$.

## 5.3 Dither

Without dither the 14-bit DAC rounding is a fixed pattern that depends on OUT1's phase, so the averaged phase is slightly nonlinear in $\phi$ (about 1 ps). A phase that stays put only sees a constant

| term | MDEV/ADEV | HDEV/ADEV |
|---|---|---|
| white PM $h_2$ | 0.100 | 1.054 |
| flicker PM $h_1$ | 0.438 | 1.047 |
| white FM $h_0$ | 0.707 | 1.000 |
| flicker FM $h_{-1}$ | 0.821 | 0.901 |
| random-walk FM $h_{-2}$ | 0.908 | 0.707 |

**Table 3:** Ratios of the deviations for each term (white and flicker PM at $\tau = 1\,\mathrm{s}$, $\tau_0 = 10\,\mathrm{ms}$; FM terms at $\tau = 100\,\mathrm{s}$), from the meter model. HDEV/ADEV for flicker and random-walk FM were checked against simulated noise (0.90, 0.71).

offset; a moving phase gains short-term error. For white FM at $h_0 = 7 \cdot 10^{-25}$ the measured ADEV without dither is 1.5 times the expected value at 10 ms and 1.05 at 35 s; with `dither` on, 1.00 everywhere (Section 9). The runner turns the dither on in every test except the floors; do the same by hand (`NOISE JSON {"dither": true}`).

### 5.4 Confidence intervals

Each point carries a 68 % confidence interval from the equivalent degrees of freedom [9] (computed with allantools [10], with the noise type identified by the lag-1 autocorrelation [7]). At the longest $\tau$ (a tenth of the record) only a few independent intervals remain: the intervals widen and single points may wander by a factor 2. Judge slopes over at least a decade, never from the last two points.

### 5.5 Spectra

The report also shows $S_y(f) = (2\pi f)^2 S_x(f)$ (1/Hz) and the phase-noise spectrum [11, 12] $\mathcal{L}(f) = S_\varphi(f)/2 = (2\pi\nu_0)^2 S_x(f)/2$ (dBc/Hz at 10 MHz). In $S_y$ white PM rises as $f^2$, white FM is flat, flicker FM falls as $1/f$. Above about $1/(2\tau_0)$ the window filters and aliases.

## 6 Running a test

### 6.1 By hand

Set the noise with the notebook, the GUI, the command-line interface or the Python client, then record. From the computer (`atomic_clock/tools/`):

```
python stability.py start t3a --duration-h 1 --note "white FM 2.6e-22"
python stability.py status            # elapsed time, gaps, the latest point
python stability.py list
python stability.py fetch t3a         # -> recordings/t3a/
python stability.py analyze recordings/t3a --floor recordings/floor -o t3a.pdf
python stability.py delete t3a        # on the board, after fetching
```

Settings with the Python client (board address as an example):

```
import sys; sys.path[:0] = ["common/python", "atomic_clock/control"]
from atomic_clock import AtomicClockClient
with AtomicClockClient("192.168.0.149") as clock:
    clock.reset()                                  # default configuration, no noise
    clock.set_noise(h0=2.6e-22, seed=1, dither=True)   # white FM, dithered DAC
    clock.set_noise(h_1=7.2e-25)                   # flicker FM (needs playback)
    clock.set_playback(source="model")
```

Or over TCP (port 5025): `RESET`, `NOISE JSON {"h0": 2.6e-22}`, `PLAY MODEL`, `REC START t3a JSON {"duration_h": 1}`, `REC?`.

While a recording runs the meter belongs to it: the service's `MEAS? n` averages $n$ consecutive worker windows, also during meter logging or streaming. Measurements use the common acquisition token: a second acquisition is `BUSY`, an accepted control/calibration/range change cancels the pending one (`CANCELLED`), and a reload makes it `STALE`. A direct driver may use a cached single-window reading; multiwindow measurements during a recording require the service.

Changing settings during a recording is allowed and flags the affected data; the runner does so only for the phase step of T4. Changing the meter window ends the recording, preserving the new setting. A recording survives a closed connection but not a reboot or a service restart (it is then closed with "service restart").

The slow interval must be an integer multiple of the fast window after quantization to 200 ns. Each series retains discontinuity flags independently: a gap or playback event between slow rows still flags the next slow row. The acquisition duration uses the board sample counter, including windows rejected for missing signal. Write errors, including errors on final flush, persist in `header.json` and status (`write_error`) and produce a failed stop reason.

### 6.2 With the test runner

```
python run_tests.py --tier quick --dry-run       # the schedule, time and disk use
python run_tests.py --tier quick                  # about half a day
python run_tests.py --tier long --unattended      # about a week
python run_tests.py --tests T3 T8 --prefix check1
```

The runner measures the split floor T0a first (it asks you to recable, then to restore the loopback; `--unattended` skips it), then the loopback floor T0b, then the noise tests with levels scaled to that floor. Before each test it resets the board to the default configuration and applies the test's settings; after the last it resets again. It writes `stability-tests/<prefix>/` with `state.json`, the recordings, one PDF per test and the combined PDF (summary table first). Interrupt with Ctrl-C and start the same command to resume: a running recording continues on the board. Before analysis the runner rejects write errors, invalid sample cadence, mismatched row counts, and less than 95 % acquisition coverage in either requested series. Coverage checks both saved sample count and time span against the requested duration; a "duration reached" status alone is insufficient.

### 6.3 The floor

**What the floor is.** The *floor* is what the board measures when nothing is simulated: the stability of OUT1 relative to OUT2 as seen through IN1 and IN2 with the noise settings at zero (test T0b). It is the noise of the measurement chain, and the limit below which a simulated clock cannot be seen.

**What it is not: the board's crystal.** OUT1, OUT2 and the phase meter are all driven by the same 125 MHz sample clock, derived from the board's crystal oscillator. The quantity measured is the *difference* OUT1 − OUT2, so the crystal's phase and frequency noise, its frequency offset and its aging enter both outputs and the meter alike and cancel. The floor therefore says nothing about the crystal itself: an uncompensated crystal is of order $10^{-10}$ at 1 s and drifts by parts in $10^9$ per day (user's manual, "Oscillator catalogue"), many orders above this floor. What remains of the crystal is negligible: its noise enters only through the small difference of the delays of the two signal paths (nanoseconds), and its ppm-level frequency error only rescales the time axis by parts per million. To measure the board's own clock an external reference is needed; the PPS time-of-day log (README) compares it with NTP only at the millisecond level.

**What it consists of.** The two DAC channels and output stages, the cables, the two ADC channels and the meter's own estimation noise (white phase noise, averaged in each 10 ms window), and, at long $\tau$, the temperature dependence of the delays of the two analog paths and the cables, which

shows as flicker-like or slowly wandering phase. T0a (OUT2 to both inputs through a T) isolates the inputs and the meter; T0b-long and M0 extend the floor to a day and more.

**Over a day.** In T0b-long, over 24 h the time error of OUT1 against OUT2 stayed within 3.0 ps peak to peak (a constant frequency removed), while the Zynq die temperature varied between 70.5 and 76.9 °C (correlation of the time error with it: -0.73). The floor keeps falling at long $\tau$ (Table 4): no flicker or random-walk frequency noise appears within a day. The analog paths' temperature dependence is part of these deviations: the time error follows the die temperature at the picosecond level (an earlier one-day run at a steadier temperature showed no correlation), so note the temperature range with any floor. The fitted power laws describe it closely to an hour; beyond, they lie up to about twice above the measured values, so test levels scaled to the fit there are on the safe side.

**The floor used in this manual.** This manual's numbers use the floor fitted to the recording `long-20261013-T0b-long` (2026-10-13, 24.0 h; measured to 2.4 h with ADEV, 18 h with ThêoBR and Totdev, beyond that the fit is extrapolated): white PM $h_2 = 4.5 \times 10^{-28}$ ($2.9 \times 10^{-14}$ s per 10 ms window), flicker PM $h_1 = 2.9 \times 10^{-26}$, white FM $h_0 = 2.9 \times 10^{-28}$ (Table 4, Figure 2).

| term | coefficient | meaning |
|---|---|---|
| white PM | $h_2 = 4.5 \times 10^{-28}$ | $2.9 \times 10^{-14}$ s rms per 10 ms window |
| flicker PM | $h_1 = 2.9 \times 10^{-26}$ | ADEV $\propto 1/\tau$ (log factor) |
| white FM | $h_0 = 2.9 \times 10^{-28}$ | ADEV $1.2 \times 10^{-14}$ at 1 s, $\propto \tau^{-1/2}$ |
| $\tau$ | ADEV measured | ADEV of the fit |
| 0.01 s | $7.1 \times 10^{-12}$ | $7.1 \times 10^{-12}$ |
| 1 s | $1.2 \times 10^{-13}$ | $1.3 \times 10^{-13}$ |
| 100 s | $2.1 \times 10^{-15}$ | $2.0 \times 10^{-15}$ |
| 8640 s | $8.2 \times 10^{-17}$ | $1.3 \times 10^{-16}$ |
| 10 000 s | $7.1 \times 10^{-17}$ (ThêoBR) | $1.2 \times 10^{-16}$ |
| 64 800 s | $2.9 \times 10^{-17}$ (ThêoBR) | $4.7 \times 10^{-17}$ |

**Table 4:** The board floor: the power-law terms fitted to the floor recording, and its ADEV measured (ADEV to a tenth of the record, ThêoBR beyond) and from the fit. Beyond the measured range the fit is an extrapolation, and the real floor (temperature, aging of the analog paths) may be higher.

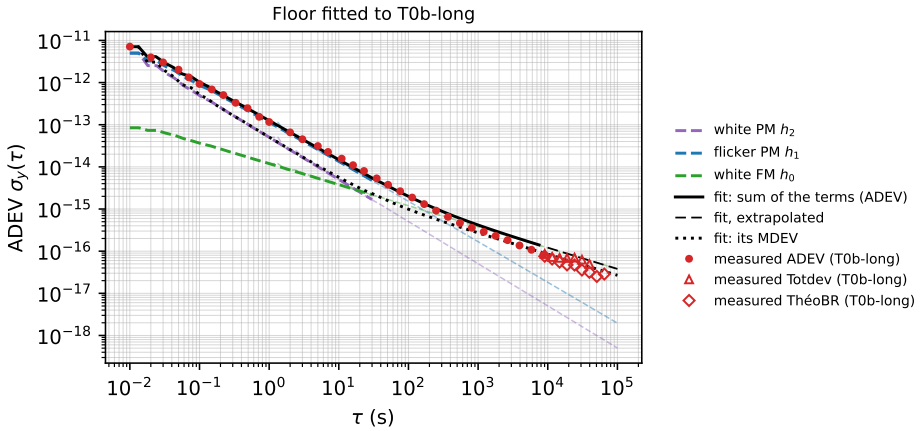



**Figure 2:** The floor of Table 4: each fitted term and their sum (ADEV, and its MDEV), with the measured ADEV, Totdev and ThêoBR.

**Levels relative to the floor.** All noise levels of the tests are chosen relative to the floor $\sigma_{\text{floor}}(\tau)$ measured in T0b: a test at "×10" configures its term so that the expected ADEV is

ten times the floor's at the test's reference $\tau$, "×100" a hundred times. The runner always uses the measured floor of its series and prints the actual settings. To put a newly measured floor into this manual: `python manual_figures.py --floor recordings/<T0b recording>`, then `latexmk -pdf` in `docs/manual`. After a series, `python manual_figures.py --results stability-tests/<prefix>` adds its measured results (Section 9).

### 6.4 Verdicts

The report's first page lists the fit and the checks:

1. **Fit:** a weighted non-negative least-squares fit of AVAR and MVAR (both series) to $\sum_\alpha h_\alpha A_\alpha(\tau) + D^2\tau^2/2 + \text{floor}$, with the meter-model shapes $A_\alpha$; the drift $D$ from a quadratic fit to $x(t)$. Uncertainties are widened for the correlation of neighbouring $\tau$ points.
2. **Each configured term:** fitted/configured within 0.6–1.6, judged only where the fit resolves the term ($1\sigma$ below 30 %); otherwise "not resolved".
3. **Shape:** where the expected ADEV is at least twice the floor, it lies inside the measured 68 % intervals (widened by 1.3) at $\geq$ 80 % of the useful points. At least three are needed; their relative confidence-interval half-width must be at most 30 %.
4. **Exact settings:** the frequency offset (slope of $x$), phase steps and the sinusoidal PM amplitude, fitted directly in $x(t)$ (with the drift, when both are set). A sine's amplitude is judged where the run covers at least two of its periods and the noise at its frequency allows 30 %; otherwise it is reported as not resolved. A phase step needs observations on both sides outside the settling interval in one unbroken unwrapping branch, and a full-rank fit. Across a broken branch its size is not identifiable and the check is inconclusive. A sine at or above the sample Nyquist frequency or at a window null is also inconclusive.
5. **Input amplitudes:** the median amplitude at IN1 and IN2 at least 0.05 V (a missing cable fails here first).

A floor test (T0a, T0b) has no noise configured: only the amplitudes are judged; its deviations are the measurement. For configured noise, a run without a resolved noise coefficient or at least three useful above-floor points dominated sufficiently by that noise is *inconclusive*, even if its amplitudes or deterministic settings pass. Missing phase-step evidence also makes the verdict inconclusive; any failed quantitative check makes it FAIL.

## 7 The tests

Table 5 lists every run with its settings (all but the floors with the dither on); Table 6 the suggested range of each setting, from 3 to 1000 times the floor at the reference $\tau$ (below 3 times the term hides in the floor; far above, the meter's phase range and the playback's resolution are still ample, but the plots stop being informative about the floor). The quick tier takes 9.5 h, the long tier about 6 days; the oscillator models (36 days) are in Section 8.

**Table 5:** Runs and settings (for the floor of Figure 2). Everything not listed is zero; playback is on where a played-back term is set.

| run | tier | hours | settings |
|---|---|---|---|
| T0a | quick | 1.0 | none |
| T0b | quick | 2.0 | none |
| T0b-long | long | 24.0 | none |
| T1-x10 | quick | 0.5 | `h2` $= 1.8 \times 10^{-25}$, dither |
| T1-x100 | quick | 0.5 | `h2` $= 1.3 \times 10^{-23}$, dither |
| T2-x10 | quick | 1.0 | `h1` $= 3.5 \times 10^{-24}$, dither |
| T2-x100 | quick | 1.0 | `h1` $= 3.5 \times 10^{-22}$, dither |

| run | tier | hours | settings |
|---|---|---|---|
| T3-x10 | quick | 1.0 | `h0` $= 4.4 \times 10^{-25}$, dither |
| T3-x100 | quick | 1.0 | `h0` $= 4.4 \times 10^{-23}$, dither |
| T4 | quick | 0.5 | `frequency_offset` $= 1.0 \times 10^{-9}$, dither |
| T8-x10 | quick | 0.5 | `pm_sine_amplitude` $= 7.4 \times 10^{-6}$, `pm_sine_frequency` $= 5.0 \times 10^{-2}$, dither |
| T8-x100 | quick | 0.5 | `pm_sine_amplitude` $= 7.4 \times 10^{-5}$, `pm_sine_frequency` $= 5.0 \times 10^{-2}$, dither |
| T5-x10 | long | 24.0 | `h_1` $= 1.3 \times 10^{-29}$, dither |
| T5-x100 | long | 24.0 | `h_1` $= 1.3 \times 10^{-27}$, dither |
| T6-x10 | long | 24.0 | `h_2` $= 2.6 \times 10^{-34}$, dither |
| T6-x100 | long | 24.0 | `h_2` $= 2.6 \times 10^{-32}$, dither |
| T7-x30 | long | 24.0 | `drift_per_day` $= 5.5 \times 10^{-14}$, dither |

| test | setting | reference $\tau$ (s) | suggested range |
|---|---|---|---|
| T1 | `h2` | $1.0 \times 10^{-1}$ | $1.6 \times 10^{-26}$ – $1.3 \times 10^{-23}$ |
| T2 | `h1` | 1.0 | $3.2 \times 10^{-25}$ – $3.5 \times 10^{-20}$ |
| T3 | `h0` | $1.0 \times 10^{1}$ | $4.0 \times 10^{-26}$ – $4.4 \times 10^{-21}$ |
| T5 | `h_1` | $1.0 \times 10^{3}$ | $1.1 \times 10^{-30}$ – $1.3 \times 10^{-25}$ |
| T6 | `h_2` | $3.0 \times 10^{3}$ | $2.3 \times 10^{-35}$ – $2.6 \times 10^{-30}$ |
| T8 | `pm_sine_amplitude` (cycles) | $1.0 \times 10^{1}$ | $2.2 \times 10^{-6}$ – $7.4 \times 10^{-4}$ |
| T7 | `drift_per_day` | $8.6 \times 10^{3}$ | $5.5 \times 10^{-15}$ – $1.8 \times 10^{-12}$ |

**Table 6:** Suggested ranges: expected ADEV 3 to 1000 times the floor at the reference $\tau$, and at most the hardware's limit (white PM).

### 7.1 Reading the prediction figures

Each test below comes with a figure of what to expect, made with the board's measured floor. They show more than one curve and more than one set of points:

**Two levels.** Most tests run twice, at ×10 and ×100 the floor (blue and red). Each colour has its *expected* curve (solid): the configured noise through the meter model, plus the floor.

**Two series per level.** A recording has a fast series ($\tau_0 = 10\,\text{ms}$) and a slow series ($\tau_0 = 1\,\text{s}$), and both are analyzed (Table 1). Each expected curve is therefore drawn twice: from 10 ms for the fast series (full colour) and from 1 s for the slow series (lighter). Between 1 s and a few minutes both exist; there they agree, except for white and flicker PM in MDEV, which depend on $\tau_0$ (a real difference, not an error).

**Points.** A *simulated* recording: noise with the configured levels plus the floor, generated through the meter model for the test's duration and analyzed exactly like a measurement. Circles are its fast series, squares its slow series; the bars are 68 % confidence intervals. The points scatter around the lines as a real measurement would, more so at long $\tau$ where few independent intervals remain. A real run should look like this.

**Floor and markers.** The grey line is the floor alone, solid where the floor was measured and dashed where its fit is extrapolated; where the expected curve approaches it, the test can no longer see its noise. The dotted vertical line marks the test's reference $\tau$, where the level was set (T8: the nulls $\tau = k/f_m$).

The reports of real runs (`stability.py analyze`) show one run at a time: circles for the fast series (blue), squares for the slow series (orange), each with its expected curve (solid) and the fitted model (dotted), and the floor (grey dashed).

### 7.2 T0a: split floor (meter only)

| | |
|---|---|
| **Purpose** | The phase meter's own floor: both inputs see the same signal. |
| **Settings** | OUT2 through a BNC T to IN1 and IN2; no noise; `RESET`. |
| **Duration** | 1 h (quick). |
| **Expect** | White PM (ADEV slope $-1$, MDEV $-3/2$) at short $\tau$ from the ADC noise, flattening where the two input channels' temperature dependence takes over. |
| **Pass** | Amplitudes only (the floor is measured). Record the ADEV at 10 ms, 1 s and 100 s in the record sheet. |

**Pitfalls:** unequal cables from the T add a fixed offset only (no noise); a loose BNC shows as jumps in $x(t)$. The amplitudes should be equal.

### 7.3 T0b: loopback floor

| | |
|---|---|
| **Purpose** | The floor of the whole chain: both outputs, both cables, both inputs. All later tests are scaled to it and analyzed against it (`--floor`). |
| **Settings** | Loopback; no noise; `RESET`. |
| **Duration** | 2 h (quick); T0b-long 24 h (long tier). |
| **Expect** | Above T0a by the outputs' contribution; at long $\tau$ a bump or a rise that follows the temperature (compare $x(t)$ with the temperature panel). |
| **Pass** | Amplitudes only (the floor is measured). It is the reference: if it is far above the estimate, all levels scale up. |

**Pitfalls:** drafts and touching the cables; the first minutes after power-up (the board warms up for about 30 minutes; start the floor later).

### 7.4 T1: white PM

| | |
|---|---|
| **Purpose** | The FPGA's white phase noise and the meter's window model. |
| **Settings** | `h2` at $\times 10$ and $\times 100$ (Table 5); playback off. The FPGA's generator allows at most 0.05 cycle per sample, $h_2 \leq 1.6 \times 10^{-23}$; the runner caps `h2` at $1.3 \times 10^{-23}$ and says so ("capped"): $\times 100$ then becomes roughly $\times 60$–$\times 80$, because each meter window averages $\sim 10^6$ samples of it. |
| **Duration** | 30 min each (quick). |
| **Expect** | ADEV slope $-1$, MDEV slope $-3/2$, TDEV slope $-1/2$ (Figure 3). The level follows the meter window: per window the samples carry white noise of $\sigma_x = \sqrt{1.5\, h_2/(2\tau_w)}/(2\pi)$ (appendix). |
| **Pass** | Fitted $h_2$ within 0.6–1.6 and the shape check. |

### 7.5 T2: flicker PM

| | |
|---|---|
| **Purpose** | Played-back flicker phase noise; MDEV separates it from white PM. |
| **Settings** | `h1` at $\times 10$, $\times 100$; `playback.source = "model"`. |
| **Duration** | 1 h each (quick). |
| **Expect** | ADEV slope close to $-1$ (with a logarithmic factor), MDEV slope $-1$ (Figure 4); in TDEV a flat line. |
| **Pass** | Fitted $h_1$ within 0.6–1.6 and the shape check. |

**Pitfalls:** playback must be on, otherwise nothing is added (the report says so). The content stops at 500 Hz: no effect on the 10 ms windows beyond a slight reduction that the model includes.

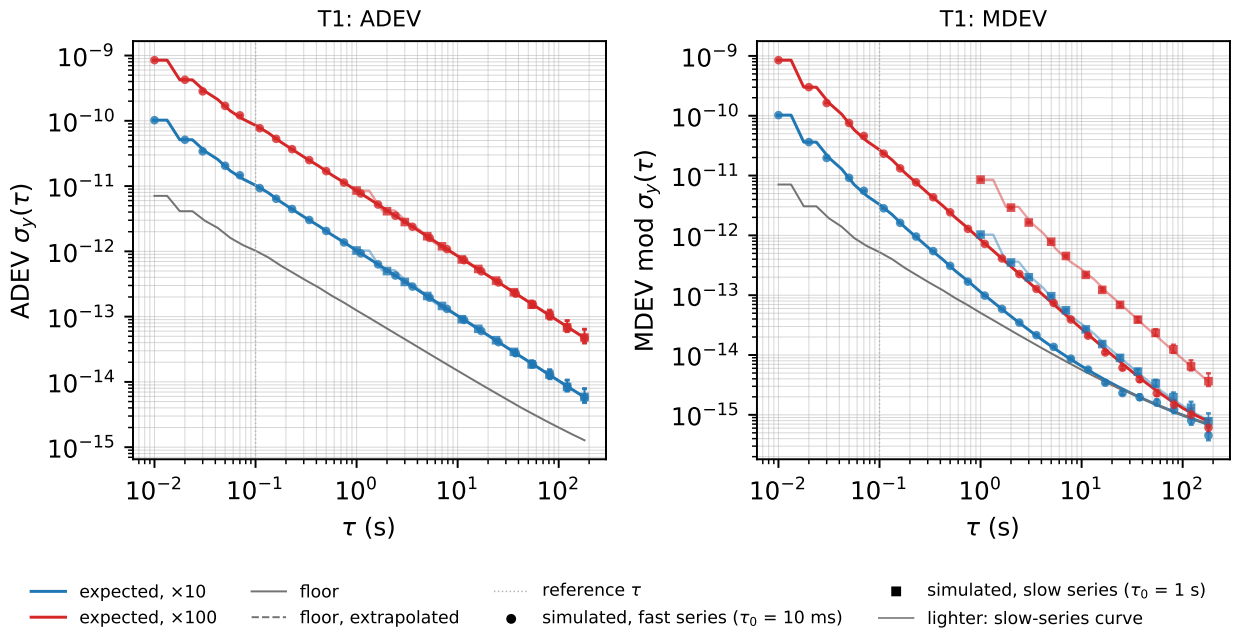



**Figure 3:** T1, predicted (how to read: Section 7.1).

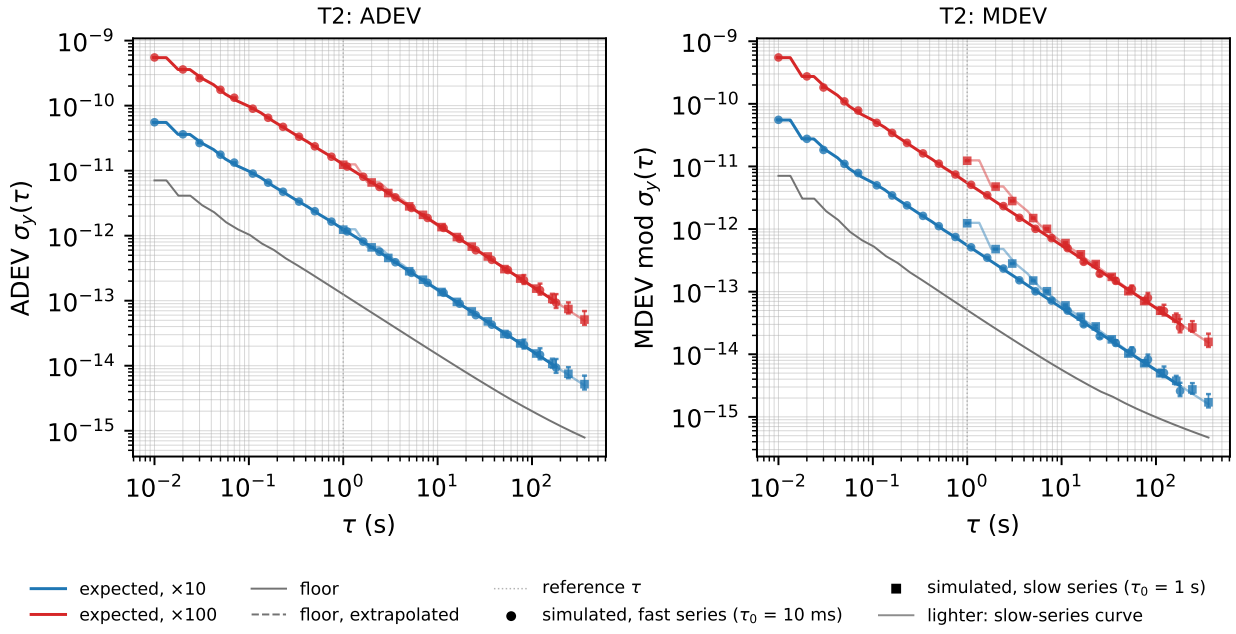



**Figure 4:** T2, predicted (Section 7.1).

### 7.6 T3: white FM

| | |
|---|---|
| **Purpose** | The FPGA's white frequency noise: the classic $\tau^{-1/2}$ of passive atomic clocks. |
| **Settings** | `h0` at ×10, ×100; playback off. |
| **Duration** | 1 h each (quick). |
| **Expect** | ADEV and MDEV slope $-1/2$, MDEV lower by $1/\sqrt{2}$; ADEV = $\sqrt{h_0/(2\tau)}$ (Figure 5). |
| **Pass** | Fitted $h_0$ within 0.6–1.6 and the shape check. |

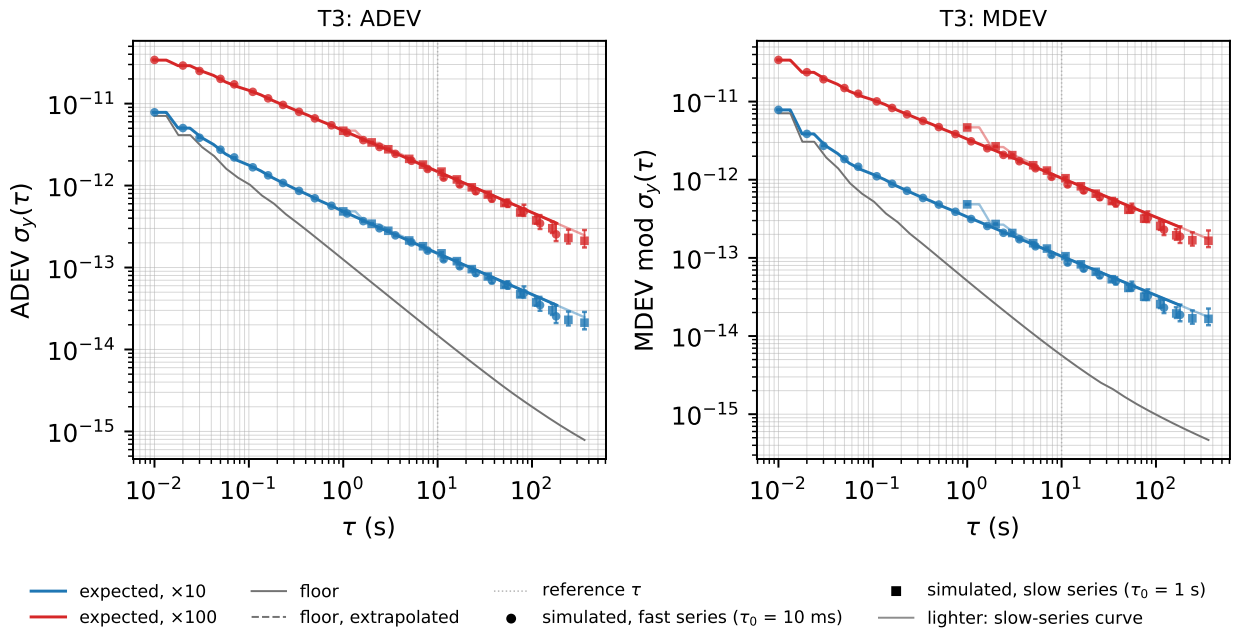



**Figure 5:** T3, predicted (Section 7.1).

### 7.7 T4: frequency offset and a phase step

| | |
|---|---|
| **Purpose** | Deterministic settings appear exactly in $x(t)$. |
| **Settings** | `outputs.1.frequency_offset` = $10^{-9}$; the runner adds a phase step of 0.01 cycle (1 ns) at 900 s, including the accumulated phase in the new target. |
| **Duration** | 30 min (quick). |
| **Expect** | $x(t)$ a straight line of slope $10^{-9}$ (1 ns/s) with a 1 ns step (Figure 6). ADEV is blind to the offset; the step makes a hump. |
| **Pass** | Fitted offset within $10^{-3}$ relative (or the noise limit); step within 10 %. |

**Identifiability:** a phase-setting change can break the meter's unwrapping branch. If the analyzed segment has no observations on both sides, the step check is inconclusive. Adding 0.01 to the saved phase *setting* alone would also remove the accumulated ramp; the runner's `phase_step` uses phase snapshots to set a relative step and records the realized change from the two alignment origins.

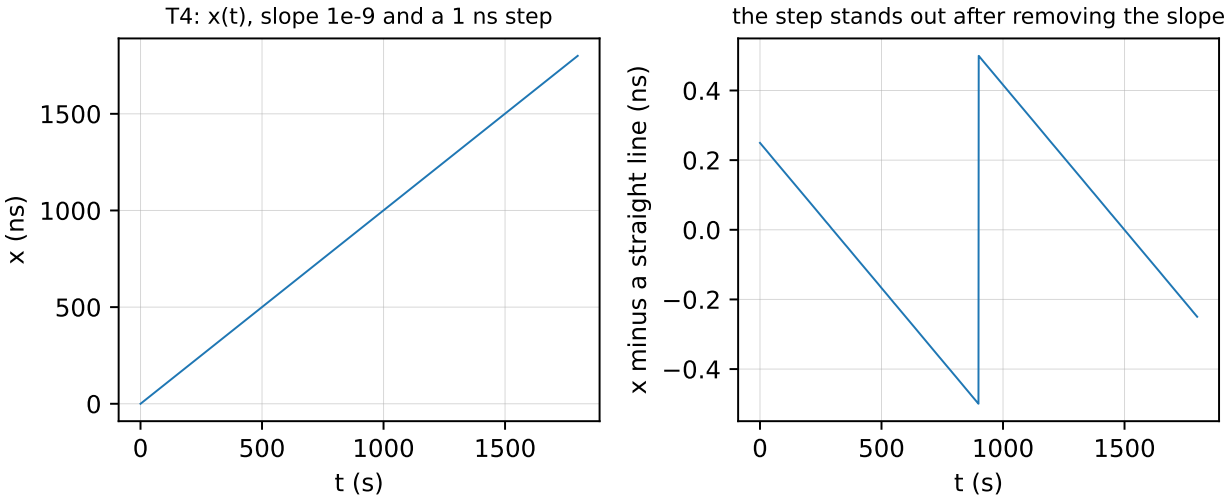



**Figure 6:** T4, predicted $x(t)$.

### 7.8 T8: sinusoidal phase modulation

| | |
|---|---|
| **Purpose** | A deterministic modulation: how a spectral line looks in ADEV (e.g. a temperature cycle or a mains pick-up). |
| **Settings** | `pm_sine_amplitude` at ×10, ×100 (cycles), `pm_sine_frequency` = 0.05 Hz; playback on. |
| **Duration** | 30 min each (quick). |
| **Expect** | ADEV $= 2A\sin^2(\pi f_m \tau)/\tau$ for amplitude $A$ (s): a peak near $\tau = 1/(2f_m) = 10$ s and nulls at $\tau = k/f_m$ (Figure 7). The report's sparse $\tau$ grid may miss the nulls; the time series shows the sine. |
| **Pass** | Fitted amplitude within 10 % and the shape check. |

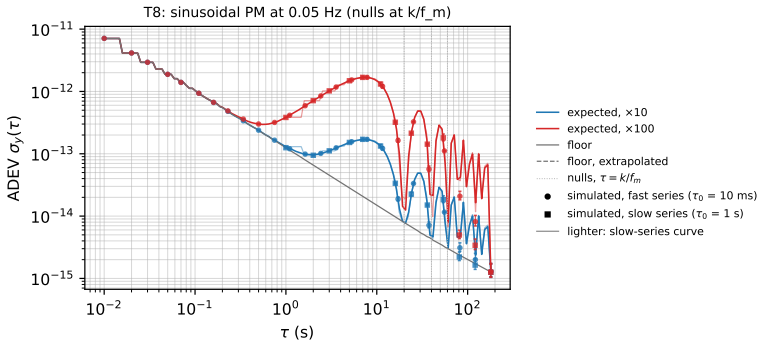



**Figure 7:** T8, predicted (Section 7.1); the expected curves are drawn on a dense $\tau$ grid to show the nulls.

### 7.9 T5: flicker FM

| | |
|---|---|
| **Purpose** | Played-back flicker frequency noise: the floor of real clocks. |
| **Settings** | `h_1` at ×10, ×100; playback on. |
| **Duration** | 24 h each (long). |
| **Expect** | ADEV flat at $\sqrt{2\ln 2\, h_{-1}}$, MDEV flat at 0.82 of it, HDEV at 0.90 (Figure 8). |
| **Pass** | Fitted $h_{-1}$ within 0.6–1.6 and the shape check. |

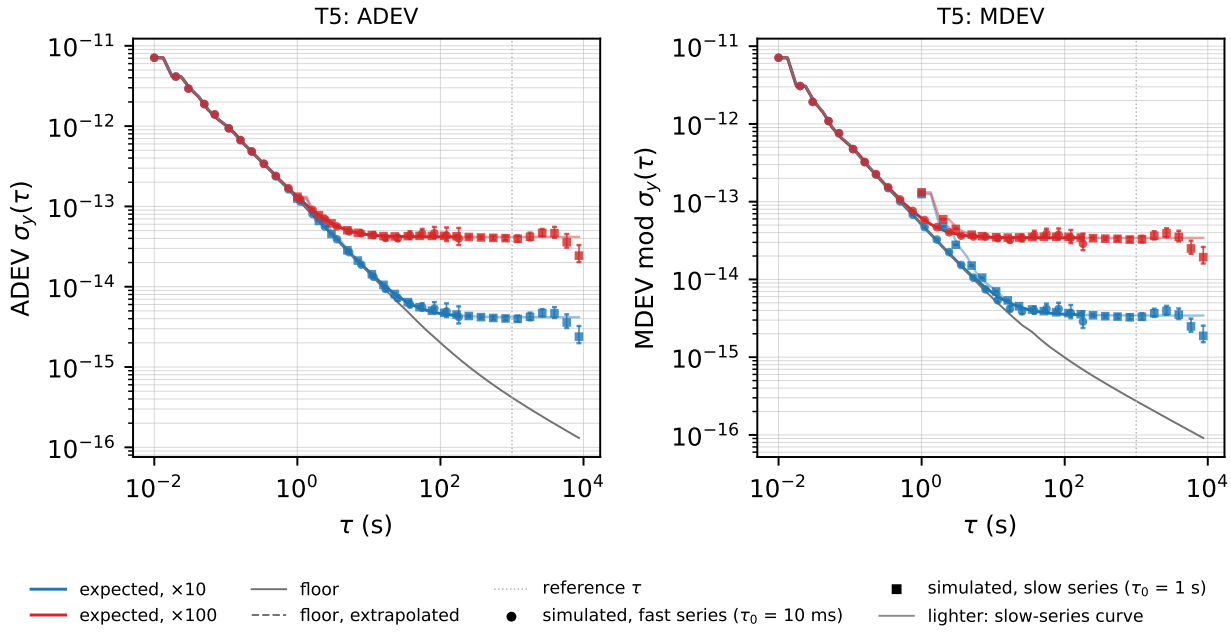



**Figure 8:** T5, predicted (Section 7.1).

### 7.10 T6: random-walk FM

| | |
|---|---|
| **Purpose** | Played-back random-walk frequency noise. |
| **Settings** | `h_2` at ×10, ×100; playback on. |
| **Duration** | 24 h each (long). |
| **Expect** | ADEV slope +1/2, $\sigma_y(\tau) = \sqrt{(2\pi^2/3)h_{-2}\tau}$; HDEV the same slope at 0.71 (Figure 9). |
| **Pass** | Fitted $h_{-2}$ within 0.6–1.6 and the shape check. |

**Pitfalls:** random walk wanders: two days with the same seed and settings differ by a factor 1.5 at the longest $\tau$; that is statistics, not a fault.

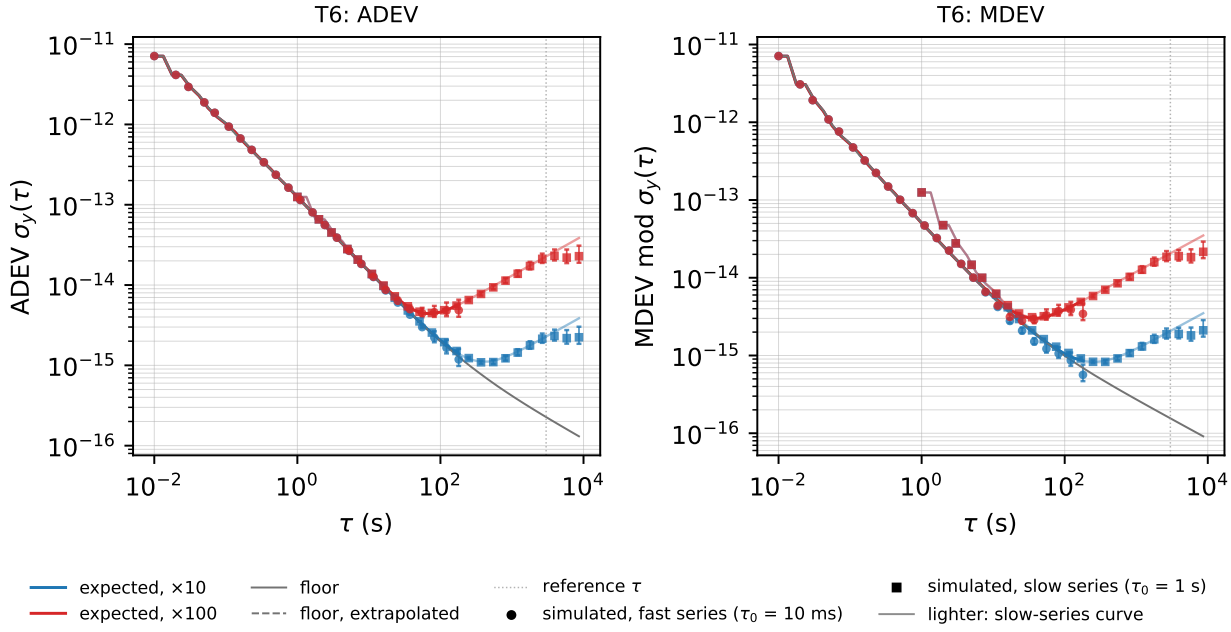



**Figure 9:** T6, predicted (Section 7.1).

### 7.11 T7: linear frequency drift

| | |
|---|---|
| **Purpose** | Drift as in ageing oscillators, and HDEV's role. |
| **Settings** | `drift_per_day` (Table 5); playback on. |
| **Duration** | 24 h (long). |
| **Expect** | ADEV rises as $D\tau/\sqrt{2}$ at long $\tau$; HDEV stays at the floor (Figure 10); $x(t)$ a parabola. |
| **Pass** | Fitted drift (quadratic fit of $x$) within 0.6–1.6. |

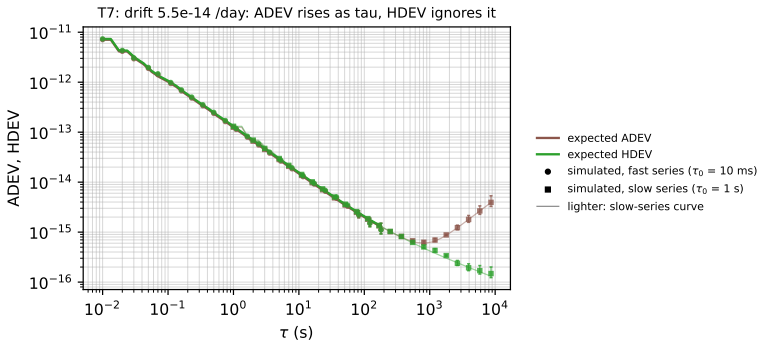



**Figure 10:** T7, predicted ADEV (brown) and HDEV (green) (Section 7.1).

### 7.12 Clock models

Mixes of several terms as in real oscillators, from quartz to hydrogen masers, with real (data-sheet-typical) values, form their own tier: Section 8.

# 8 Oscillator models (M runs)

The *models* tier plays each of the 11 oscillator types of the user's manual (section "Oscillator catalogue": a bare crystal to a hydrogen maser [13, 14], with their typical values and settings) for three days each, after a three-day floor run M0.

**T tests and M runs.** The two tiers answer different questions.

- The **T tests** are acceptance tests of the simulator: each checks one noise term (or one deterministic setting) at a level scaled 10 to 100 times above the measured floor, so the term is fully resolved, against pass/fail criteria (Section 6.4).
- The **M runs** are simulations of real clocks: all terms of an oscillator together, at its real, unscaled values. Where a term lies below the board's floor (the maser, the double-oven OCXO at short $\tau$, the random walk of most models within three days) the board measures its own floor there, the fit reports the term as not resolved, and the verdict confirms only what the floor allows. They demonstrate what each simulated clock looks like on the board's own meter and check the combined playback over days; an external instrument with a lower floor sees more of them.

**D runs.** The *demos* tier plays the six deliberately bad clocks of the user's manual (section "Bad clocks for demonstrations": jitter, a hot crystal, mains ripple, popcorn noise, frequency jumps, a badly installed GPSDO) for one or two hours each, after a one-hour floor run D0: about ten hours in all. Their faults lie orders of magnitude above the floor, so each run shows its signature at once against the expected curves of its settings. `D-mains` uses 5 ms windows (200 samples/s), so the 50 Hz ripple is strictly below Nyquist and its amplitude is identifiable. The other D runs use the default 10 ms window. `stability.py play <key>` plays one without recording.

| oscillator | ADEV at 1 s | at 100 s | at $10^4$ s |
|---|---|---|---|
| bare crystal oscillator (XO) | $1.4 \times 10^{-10}$ | $2.1 \times 10^{-10}$ | $1.7 \times 10^{-8}$ |
| MEMS oscillator | $1.1 \times 10^{-11}$ | $1.9 \times 10^{-11}$ | $1.7 \times 10^{-9}$ |
| TCXO | $5.8 \times 10^{-11}$ | $4.7 \times 10^{-11}$ | $3.5 \times 10^{-9}$ |
| OCXO | $1.4 \times 10^{-12}$ | $1.0 \times 10^{-12}$ | $1.7 \times 10^{-11}$ |
| double-oven OCXO | $1.4 \times 10^{-13*}$ | $1.0 \times 10^{-13}$ | $8.3 \times 10^{-13}$ |
| chip-scale atomic clock (CSAC) | $3.0 \times 10^{-10}$ | $3.2 \times 10^{-11}$ | $1.1 \times 10^{-11}$ |
| rubidium | $2.0 \times 10^{-11}$ | $2.0 \times 10^{-12}$ | $4.0 \times 10^{-13}$ |
| caesium beam | $1.2 \times 10^{-11}$ | $1.2 \times 10^{-12}$ | $1.2 \times 10^{-13}$ |
| caesium beam, high performance | $5.0 \times 10^{-12}$ | $5.0 \times 10^{-13}$ | $5.1 \times 10^{-14}$ |
| active hydrogen maser | $1.5 \times 10^{-13*}$ | $2.7 \times 10^{-15*}$ | $1.0 \times 10^{-15}$ |
| GPS-disciplined OCXO | $1.3 \times 10^{-12}$ | $1.1 \times 10^{-12}$ | $1.6 \times 10^{-13}$ |

**Table 7:** Expected ADEV of each model alone (without the board floor). *: less than twice the board floor there (measured to 2.4 h with ADEV, 18 h with ThêoBR and Totdev; extrapolated beyond), so a run cannot show the model's own value.

**Running the models.**

```
python run_tests.py --tier models                    # all, about 36 days
python run_tests.py --tier models --tests M-ocxo M-gpsdo
```

records the floor M0 for three days, then each model for three days (36 days in all; `--tests` picks some; the runs resume after an interruption). Before M-gpsdo the runner generates the record (`tools/replays/gpsdo.csv`) and uploads it (`REPLAY PUT`); the downloaded run keeps a copy as `replay.csv`, from which the report draws the expected curve. Generated records are reused only after checking their duration, parameters and algorithm identity. Replay variances use the actual

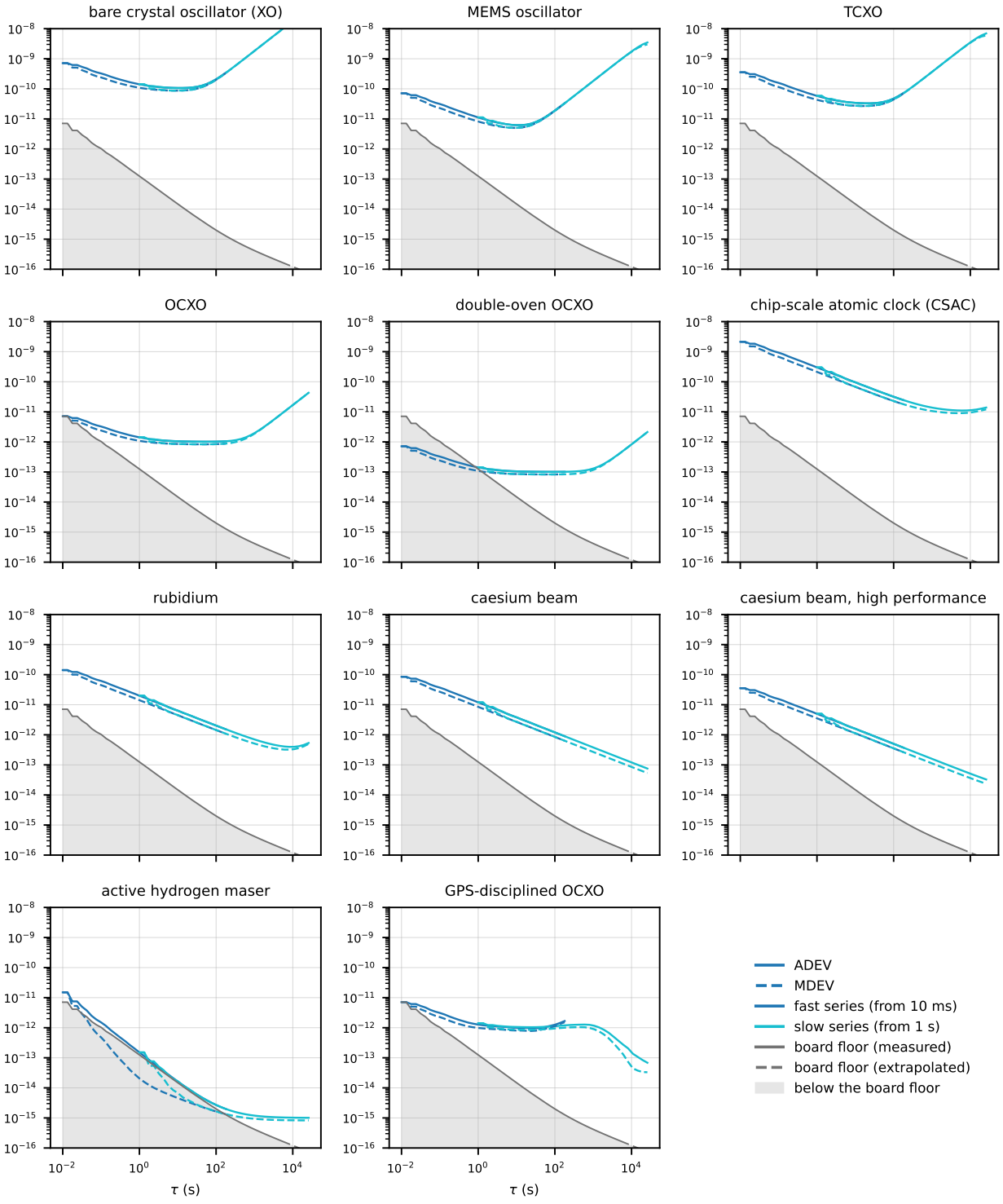



**Figure 11:** Each model's expected ADEV (solid) and MDEV (dashed), for the fast series (from 10 ms, blue) and the slow series (from 1 s, cyan) of a 3-day run, with the board floor (solid where measured, dashed where extrapolated) and the region below it shaded. The XO, MEMS and TCXO rise at long $\tau$ with their daily temperature cycle and aging; the OCXO's random walk and aging show after a few thousand seconds; the GPS-disciplined OCXO follows the OCXO up to the loop time and the GPS noise ($\propto 1/\tau$) beyond.

`replay_scale`, linear interpolation, and the analyzed series' spacing and meter windows, including below the record's original spacing. The cache includes file identity and sampling; missing long-$\tau$ estimates are not extrapolated as a constant. The verdicts are those of Section 6.4: each configured term within 0.6–1.6 where the fit resolves it, the expected ADEV inside the measured intervals, the drift and the daily cycle's amplitude where the run (at least two days) and the noise allow; terms below the floor are reported as not resolved.

# 9 Measured results

The quick tier was run on this lab's board (series `quick-20261007`, recordings from 2026-10-07, 2026-10-09; loopback with two equal cables, 0.5 V, the board's scheduled jobs disabled, die temperature about 71 °C). The loopback floor T0b: ADEV $7.0 \times 10^{-12}$ at 10 ms, $1.2 \times 10^{-13}$ at 1 s, $2.3 \times 10^{-15}$ at 100 s. All noise tests ran with the dither on and their levels scaled to that floor.

**Table 8:** The quick tier: fitted/configured noise coefficient, the expected ADEV inside the measured intervals, the exact settings found in $x(t)$, and the verdict. T1-x100 is capped at the white-PM limit.

| run | dither | fitted/configured | shape | exact settings | verdict |
|---|---|---|---|---|---|
| T0a | no | – | – | – | pass |
| T0b | no | – | – | – | pass |
| T1-x10 | yes | h2 1.02 | 38 of 38 points | – | pass |
| T1-x100 | yes | h2 1.03 | 38 of 38 points | – | pass |
| T2-x10 | yes | h1 1.06 | 42 of 42 points | – | pass |
| T2-x100 | yes | h1 1.00 | 42 of 42 points | – | pass |
| T3-x10 | yes | h0 0.98 | 37 of 37 points | – | pass |
| T3-x100 | yes | h0 0.99 | 42 of 42 points | – | pass |
| T4 | yes | – | – | offset error $1.6 \times 10^{-15}$; step 934.1 of 937.5 ps | pass |
| T8-x10 | yes | – | 14 of 14 points | amplitude ratio 1.054 | pass |
| T8-x100 | yes | – | 26 of 26 points | amplitude ratio 1.021 | pass |

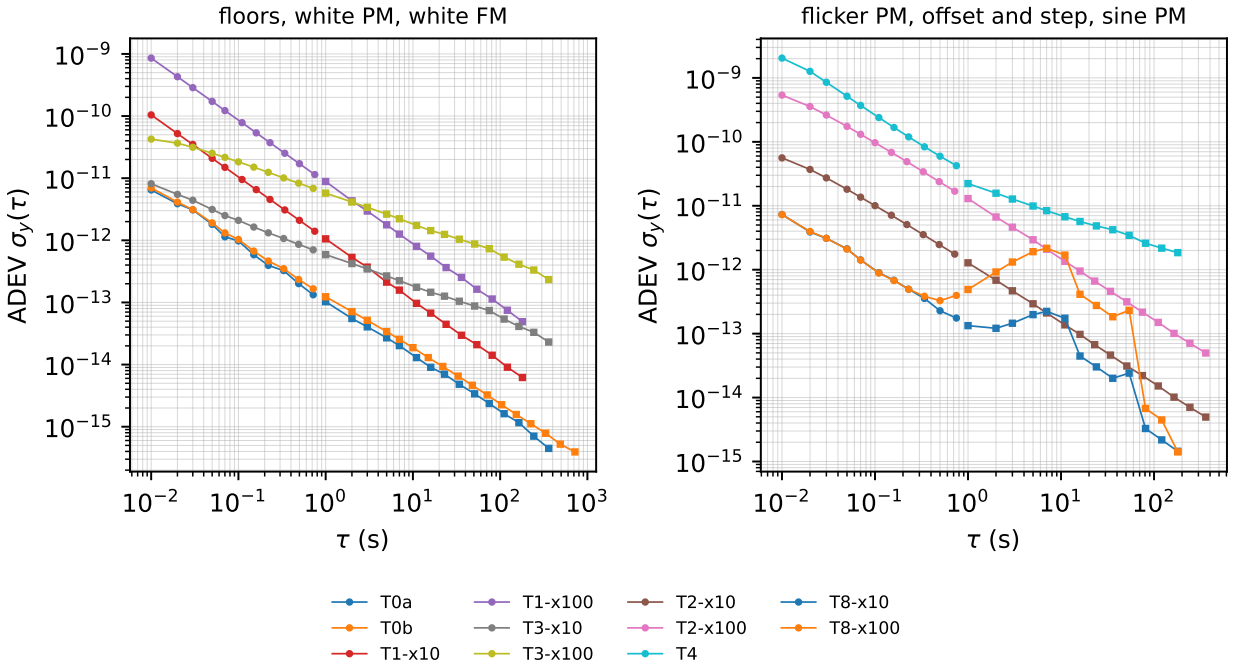



**Figure 12:** Measured ADEV of every run: circles the fast series (below 1 s), squares the slow series (from 1 s), one colour per run. Left: the floors and the FPGA's white PM and white FM at two levels; right: the played-back flicker PM, the sinusoidal PM and T4 (whose ADEV is dominated by its 1 ns phase step, as expected).

**What the results show.**

- Each noise type comes out at the configured level: fitted/configured 0.98–1.07 for white PM, flicker PM and white FM at both levels, every expected point inside the measured intervals.

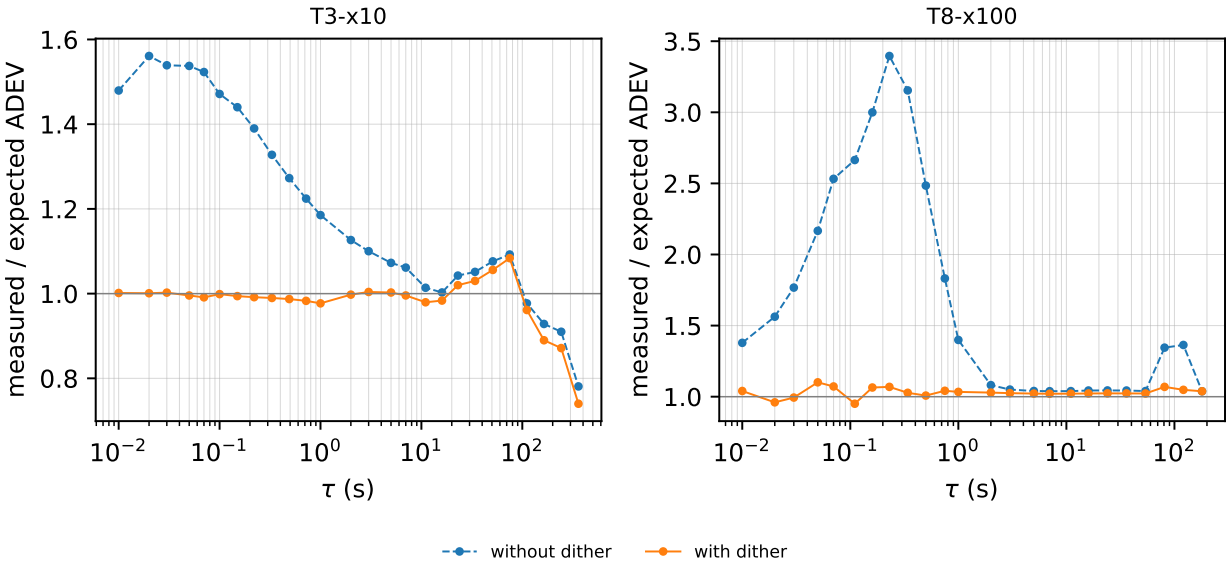



**Figure 13:** The effect of the dither: measured over expected ADEV for white FM (T3-x10) and sinusoidal PM (T8-x100), recorded without and with it. Without it, the DAC rounding adds short-term error to a moving phase (up to 1.5 and 3 times); with it, the measurement follows the model.

- Exact settings appear exactly: the frequency offset to $2 \cdot 10^{-15}$, the phase step within 0.4 %, the sine amplitude within 2–5 %.
- The meter reads the FPGA's broadband white PM with 1.5 times the variance of the window-averaged phase (appendix); the model includes this.
- Without the dither, a moving phase picks up extra error at short $\tau$ (Figure 13): use it whenever OUT1's phase moves.
- Not yet measured: the long tier (flicker FM, random-walk FM, drift) and the oscillator models (Section 8). The floor over a day (T0b-long) is in Section 6.3.

# 10 Further estimators and tools

The report of `stability.py analyze` and the runner's per-test reports (`reports/<run>.pdf`) add, after the five main pages, the estimators below, as found in Stable32 and NIST SP 1065 [7]. They are information for the reader, not part of the verdict. `--no-extras` leaves them out (faster); the runner's combined PDF shows only the main pages.

## 10.1 Long-term estimators: Totdev and ThêoBR

The overlapping ADEV is shown up to $\tau = T/10$ of the longest gap-free stretch $T$, beyond which it has too few degrees of freedom. Two estimators reach further (Figure 14, left):

**Totdev** [15] extends the record at both ends by reflection and reaches $T/2$. For flicker and random-walk FM it reads low by the factor $1 - a\,\tau/T$ ($a = 1/(3 \ln 2)$ and 3/4, SP 1065 eq. 52); the report corrects this, with the noise type taken from the configured model at that $\tau$ (else from the data's lag-1 autocorrelation).

**ThêoBR** Thêo1 [16] reaches $0.75\,T$ with the most degrees of freedom of any estimator; ThêoBR removes its noise-dependent bias by the average ratio of AVAR to Thêo1 at short $\tau$ (SP 1065 eq. 33). With ADEV below $T/10$ it forms the *ThêoH* plot.

Both estimate the Allan deviation, so the expected ADEV curve applies. The 68 % intervals are $\chi^2$ intervals with SP 1065's equivalent degrees of freedom. Long records are reduced by picking every

$k$th phase sample (at most 20 000 points for Totdev, 4000 for Thêo1). This retains white PM at its full level; block averaging would divide its variance by $k$ and bias PM-dominated records low. Only $\tau$ much longer than the resulting spacing are shown. The modified and Hadamard total deviations of Stable32 are not provided: the available implementation of the first is marked preliminary and disagrees with SP 1065's bias table, and the second needs a bias correction that depends on the noise type and the record length.

### 10.2 Time interval error: MTIE, TIE rms and ITU-T masks

Telecommunication and network timing specify clocks by the time interval error (TIE) [17]. For an observation interval $\tau$,

$$\mathrm{MTIE}(\tau) = \max_k \Big[ \max_{t_k \le t \le t_k+\tau} x(t) - \min_{t_k \le t \le t_k+\tau} x(t) \Big], \qquad \mathrm{TIE}_{\mathrm{rms}}(\tau) = \sqrt{\big\langle [x(t+\tau) - x(t)]^2 \big\rangle},$$

the largest peak-to-peak time error in any window of length $\tau$, and the rms time-error change. Unlike the deviations, both include a frequency offset and a drift (MTIE $= |y|\tau$ for a pure offset), and MTIE, as a maximum, grows with the length of the record. The report plots both for the fast and slow series, together with TDEV; with `--mask NAME` it draws an ITU-T wander-generation mask and states whether MTIE and TDEV stay inside it (Table 9). A TDEV mask is evaluated only where the actual analyzed gap-free segment spans at least $12\tau$; time across gaps does not count towards that requirement.

| `--mask` | clock | MTIE limit | TDEV limit |
|---|---|---|---|
| `G.811` | PRC [18] | $0.275 \cdot 10^{-3}\tau + 0.025\,\mu s$ $(0.1 < \tau \le 1000\,s)$ | $3\,ns$ $(0.1 < \tau \le 100\,s)$ |
| | | $10^{-5}\tau + 0.29\,\mu s$ $(\tau > 1000\,s)$ | $0.03\,\tau\,ns$ $(100 < \tau \le 1000\,s)$ |
| | | | $30\,ns$ $(1000 < \tau < 10^4\,s)$ |
| `G.8262-1` | EEC opt. 1 [19] | $40\,ns$ $(0.1 < \tau \le 1\,s)$ | $3.2\,ns$ $(0.1 < \tau \le 25\,s)$ |
| | | $40\,\tau^{0.1}\,ns$ $(1 < \tau \le 100\,s)$ | $0.64\sqrt{\tau}\,ns$ $(25 < \tau \le 100\,s)$ |
| | | $25.25\,\tau^{0.2}\,ns$ $(100 < \tau \le 1000\,s)$ | $6.4\,ns$ $(100 < \tau \le 1000\,s)$ |
| `G.8262-2` | EEC opt. 2 [19] | $20\,ns$ $(0.1 < \tau \le 1\,s)$ | $3.2\,\tau^{-0.5}\,ns$ $(0.1 < \tau \le 2.5\,s)$ |
| | | $20\,\tau^{0.48}\,ns$ $(1 < \tau \le 10\,s)$ | $2\,ns$ $(2.5 < \tau \le 40\,s)$ |
| | | $60\,ns$ $(10 < \tau \le 1000\,s)$ | $0.32\sqrt{\tau}\,ns$ $(40 < \tau \le 1000\,s)$ |
| | | | $10\,ns$ $(1000 < \tau \le 10^4\,s)$ |

**Table 9:** ITU-T wander-generation masks: G.811 (1997) for primary reference clocks (PRC), G.8262 (2007) for synchronous-Ethernet equipment clocks (EEC), constant temperature. The recommendations measure through a 10 Hz first-order low-pass filter with samples at most 1/30 s apart; the meter's 10 ms window is a comparable filter for the fast series.

### 10.3 Data from other instruments

The same analysis and report work on phase or frequency data from another instrument, for example a time-interval counter comparing OUT1 with a reference, or another program's export:

```
python stability.py analyze-file counter.txt --kind phase --units ns \
    --time-column 0 --oscillator ocxo --mask G.811
python stability.py analyze-file freq.dat --kind frequency --units ppb \
    --tau0 10 --settings '{"h0": 2e-24, "h_1": 7.2e-25}'
```

The file holds one value per line or several columns (comma, semicolon, tab or space; `#` comments; a header line is skipped). Phase is time error in `s, ms, us, ns, ps` or phase in `cycles` or `rad` of `--f0` (10 MHz by default); frequency is fractional, `ppm`, `ppb` or an offset in `hz` from `--f0`, summed to phase. The spacing comes from `--tau0` or a time-tag column. Gaps (missing values, time tags jumping by more than $1.5\,\tau_0$, and zeros in frequency data, as in Stable32) split the record; the analysis uses the longest stretch. Expected curves, the $h$ fit and a verdict appear only with settings:

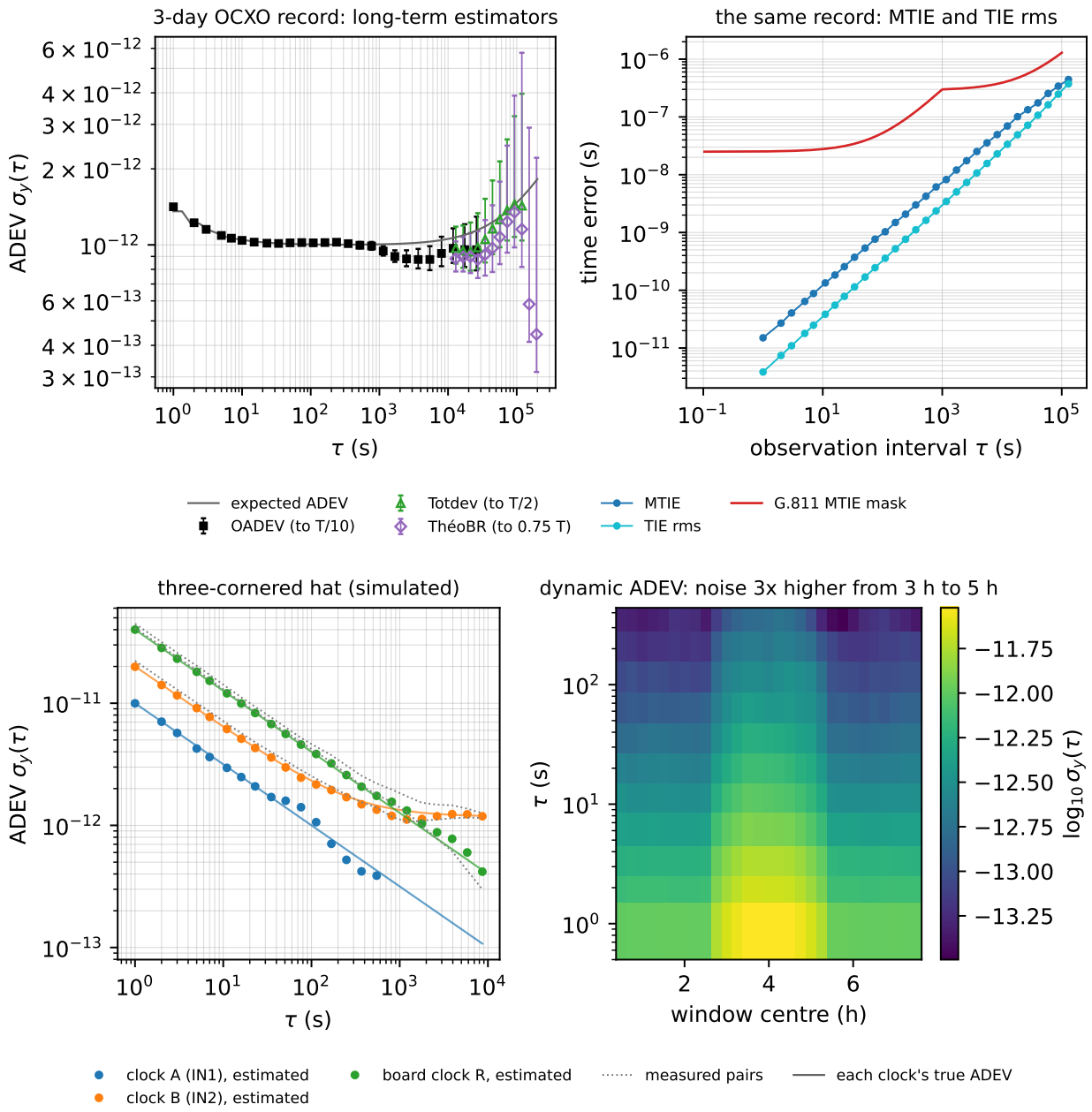



**Figure 14:** Simulated examples (one realization each, 68 % intervals). Top left: a 3-day record of a typical OCXO (white FM, flicker and random-walk FM, point samples at 1 s): ADEV to $T/10$, Totdev to $T/2$, ThêoBR to $0.75\,T$. Top right: its MTIE and TIE rms against the ITU-T G.811 mask; the random walk of frequency makes the time error grow faster than $\tau$. Bottom left: three-cornered hat of three simulated clocks (one day); points: the separated ADEV of each clock, lines: their true ADEV, dotted: the three measured pairs. Bottom right: dynamic ADEV (1-hour windows) of a record whose white FM is 3 times higher between 3 and 5 hours.

`--settings` (JSON with `h2` … `h_2`, `drift_per_day`, `pm_sine_amplitude`, `pm_sine_frequency`) or `--oscillator` (a model of the user's manual's "Oscillator catalogue", with $h$ from the standard Allan-deviation formulas). The values are point samples, not window averages: white PM is taken as flat up to `--bandwidth` (default $1/(2\tau_0)$), which is the measurement bandwidth $f_h$ of Table 2. The white-PM kernels retain the sample covariance $R(k) = h_2 f_h \operatorname{sinc}(2f_h k\tau_0)/(2\pi)^2$. Samples are independent only when $2f_h\tau_0$ is a positive integer; narrower bands and nonintegral ratios require the correlated kernels. Delimited empty fields and missing samples preserve their positions and split the record rather than shifting columns or closing gaps.

### 10.4 Three-cornered hat: separating three clocks

The meter measures each input against the board's own clock R: a recording holds $x_A - x_R$ (IN1), $x_B - x_R$ (IN2) and their difference $x_A - x_B$. With two clocks A and B on the inputs (instead of the loopback), the three pairwise variances separate into each clock's own, if the three are independent [20]:

$$\sigma_A^2 = \tfrac{1}{2}\big(\sigma_{AR}^2 + \sigma_{AB}^2 - \sigma_{BR}^2\big), \quad \sigma_B^2 = \tfrac{1}{2}\big(\sigma_{BR}^2 + \sigma_{AB}^2 - \sigma_{AR}^2\big), \quad \sigma_R^2 = \tfrac{1}{2}\big(\sigma_{AR}^2 + \sigma_{BR}^2 - \sigma_{AB}^2\big),$$

for ADEV and MDEV at each $\tau$ (Figure 14, bottom left). This measures the board's own 125 MHz oscillator, which the rest of this manual leaves unmeasured.

```
python stability.py start clocks --duration-h 24 --note "Rb on IN1, OCXO on IN2"
python stability.py fetch clocks
python stability.py hat recordings/clocks
```

Points to keep in mind:

- *Beat frequency.* The recorder unwraps each input's phase window by window, so a clock must stay within half the window rate of the board's 10 MHz, and a window averages a beat at frequency $f$ down by $\operatorname{sinc}(f\tau_w)$. Keep $|f| \lesssim 0.3/\tau_w$: about ±30 Hz (±3 ppm) with 10 ms windows; check with `MEAS?` before starting. The board's crystal is typically within a few ppm, but this is not measured.
- *The meter's noise* on each input appears in $\sigma_{AR}$, $\sigma_{BR}$ and $\sigma_{AB}$ so that it is attributed to A and B, not to R. A clock much better than the meter's floor (Section 6.3) is not resolved.
- *Negative estimates.* When one clock is much better than the others, or the record short, its variance estimate can come out negative; such points are left out (the report and the printed table show them as missing).

### 10.5 Dynamic Allan deviation

A long run is cut into windows (an eighth of the run, moved by half a window) and ADEV computed in each [21]. Stationary noise gives the same curve everywhere; a change during the run, such as a warm-up, a temperature step or a disturbance, shows as a band (Figure 14, bottom right). The page appears for runs of 4 hours or more; windows containing lost data are left blank.

### 10.6 Quick tables in adev1 format

Many in the time-and-frequency community check a clock with Tom Van Baak's `adev1` [22]: a plain file of phase readings on standard input, and a table of the normal and the overlapping Allan deviation with the number of terms behind each value. `tools/adev1.py` does the same, with the same arguments and raw valid-input output (byte for byte against `adev1.c` compiled on Linux), and needs only Python:

```
python adev1.py period [scale] [bins] < phase.txt
python adev1.py 1 < counter.txt              # 1 s readings in seconds
python adev1.py 1 200e-9 < tsc.dat           # 1 s readings in units of 200 ns
tail -n +2 recordings/run1/slow.csv | cut -d, -f4 | python adev1.py 1 1e-7
```

`period` is the spacing in seconds, `scale` converts the readings to seconds, and `bins` chooses the $\tau$ values (default 5: 1, 2, 5, 10, ...; 1 one per decade, 2 octaves, 3 1-3-10, 4 1-2-4, 10 ten per decade, 29 "round numbers", 0 every $\tau$, any other $N$ that many per decade). As in `adev1`, `#` comments and HP 53131/53132A statistics blocks are skipped and digit-group commas removed. Unit suffixes `s, ms, us, ns, ps` are read; any explicit suffix, including `s`, rejects a non-unit `scale` to prevent double scaling. The table stops when fewer than four overlapping terms remain. The last line above analyzes a recording: column 4 is IN1 − IN2 in cycles of 10 MHz, 100 ns each. The table has no confidence intervals and no expected curves; for those use `stability.py analyze` or `analyze-file`.

### 10.7 Drift models

The frequency drift is fitted to the time error $x(t)$ of the slow series. By default it is linear in frequency, $y = y_0 + Dt$ (a quadratic in $x$, IEEE 1139). With `--drift log` the report fits the logarithmic aging law of quartz oscillators, $y = y_0 + A\ln(1 + Bt)$ [23], i.e. $x = x_0 + y_0 t + A\,[(1 + Bt)\ln(1 + Bt) - Bt]/B$ (with a configured daily sine fitted at the same time). The summary page gives $A$, $1/B$, the aging rate $\mathrm{d}y/\mathrm{d}t = AB/(1+Bt)$ at the start and at the end, and the rms residual of both models. The full fitted nonlinear aging trajectory is removed before computing the deviations used by the noise fit. Its final rate is reported separately, not used as a substitute quadratic drift; the plotted total adds back the aging trajectory's sampled variances. The simulator itself plays a linear drift (`drift_per_day`); the logarithmic model is for data of real oscillators (`analyze-file`).

### 10.8 Live estimation: rustle

`tools/rustle` computes the analysis of this section's main pages while the data arrive: from the board's meter stream (`MEAS STREAM`), a downloaded recording, another instrument's file or a simulated clock. It shows the results in a web dashboard (user's manual, "Watching the noise live"; `tools/rustle/README.md`). It uses the same meter-aware model (`clocklib.model`, on the common `stabilitylib`) and the same Greenhall EDFs as `stability.py analyze`, so its expected curves and intervals mean the same. It is a quick check during a run, not a replacement for the recorded tests and their verdicts. It differs from the report in these ways:

**Incremental deviations.** Overlapping ADEV, MDEV and HDEV are kept as running sums at every $\tau$, so a new window costs the same however long the run is. They equal allantools' `oadev`, `mdev` and `ohdev` (to $10^{-8}$). Runs longer than 8.4 million windows (a day of 10 ms windows) are stored as blocks of $r$ windows. The long $\tau$ then use terms at multiples of $r$, unbiased but with the degrees of freedom of the run sampled every $r$ windows. The plot and fit always use the fixed union of 1–2–5 and octaves, independent of the table's selected bins or earlier fits. Newly requested $\tau$ backfill retained segments without crossing gaps. Stop, end of file and errors flush pending MDEV terms.

**Mixed-noise intervals.** Where several noise types contribute at one $\tau$, the EDF is that of their sum (Satterthwaite's approximation, with the cross terms), not of one dominant type. Where white PM and white FM cross, the few degrees of freedom of the white-FM part set the scatter. For segment term counts $c_i$ and EDFs $\nu_i$, the count-weighted pooled EDF is $(\sum_i c_i)^2 / \sum_i (c_i^2/\nu_i)$, treating segments as independent. It is not generally the sum of their EDFs.

**Fit.** Non-negative least squares on AVAR, MVAR *and* HVAR. HVAR has no drift term, which fixes $h_{-2}$ where a strong drift dominates AVAR. The drift enters as $D^2 - \sigma_D^2$, which can be negative. Subtracting the estimate's own noise avoids counting it twice, because the $h$ terms already hold it. Measured data get a 3 % systematic floor on each point's uncertainty, for how well power laws describe a real clock at all. The fit, model-dependent EDFs and drift debiasing iterate to a consistent result in each analysis, including the first headless snapshot; `fit.converged` and `fit.iterations` report convergence. Display choices and repeated analysis of unchanged data do not refine the fit.

**Segments.** Each flagged or lost window starts a segment. The deviations pool the segments, and the offset and drift fit keeps one phase intercept per segment. The offset and drift errors are those of the least-squares estimate under the fitted noise, for the actual discrete endpoints and sample counts, without rounding the layout. Two-sample segments constrain slopes; singletons constrain their own intercepts. White-PM weight norms are discrete, white/random-walk FM use analytic weight integrals, and flicker uses numerical frequency integrals (continuous midpoint transforms above 64 samples per segment). FM errors use the broad-band point-sample limit; white-PM trend errors retain the independent-sample approximation, while the bootstrap includes explicitly bandlimited sample correlations.

**Bootstrap.** 20 simulated copies of the fitted clock are analyzed in background processes. Through $2^{21}$ timeline samples, including missing windows, the method is `segment-exact`: one realization advances through gaps, and the original segments enter the deviations and trend, with each estimator's actual term counts (including the live MDEV pending tail), count weights, fixed grid and iterative fit. Off-grid gap durations are rounded to the nearest sample for the noise process; the trend retains the actual time tags. Above that limit the labelled `edf-rescaled` approximation uses one or two uninterrupted shorter records, rescaling their variance fluctuations by $\sqrt{\nu_{\text{replica}}/\nu_{\text{run}}}$, and a separate coarser realization for the segmented trend. It does not preserve the original deviation segment boundaries or the joint deviation/trend covariance; short segments can disappear from the coarse trend. Use its errors cautiously for sparse, gap-dominated runs or very few long-$\tau$ terms. Fewer than three successful replicas gives no uncertainty. A measured floor is simulated as its power-law fit, so with a floor bootstrap errors are a lower bound: take the larger fit/bootstrap error. Changing *fit drift* cancels old workers and clears their uncertainties; round keys include model revision, sample count and estimator counts, so finishing an MDEV tail triggers a fresh round even with no new samples.

**Export.** JSON exports include the normalized source specification and kernel arguments, actual segment endpoints/counts, fixed analysis grid, model revision, fit convergence and bootstrap method/count key. Strip-chart data include original block-mean phase, segment IDs and samples per point, as well as detrended phase/frequency. Detrending subtracts the mean quadratic trend over each block. These compressed display data complement the source recording needed for full-resolution reanalysis.

**The stream's phase.** The board unwraps IN1 window by window against OUT1's predicted phase. The service knows it exactly: the frequency offset, every played-back segment and the slew. A window is not sent when either input is below 0.05 V, or when its amplitude fell below 0.9 of the run's level. Such an amplitude loss means the phase rotated within the window: a window that the phase crosses by $r$ cycles keeps $|\operatorname{sinc} r|$ of the amplitude, so the meter cannot measure more than about a quarter cycle per window. A window more than a quarter cycle from the prediction starts a new run. Any change of OUT1's offset, phase, slew or alignment, of the meter window, or of the playback, is flagged. `rustle` treats a flag as a gap.

**Earlier statistical checks** (`tests/test_rustle.py`, `docs/VERIFICATION.md`; these figures predate the segmented/fixed-point review fixes):

- 68 % intervals over 40 simulated runs per noise type covered 60–78 % pooled. Where white PM and white FM cross, single $\tau$ covered 52–62 %.
- The offset and drift errors were 0.7–1.4 times the true scatter.
- The bootstrap errors were within about 20 % of the scatter of repeated runs for the terms the data determine. The fit's own errors were 2–6 times too large.
- In 10-minute to one-hour runs, $h_{-2}$ comes out 10–40 % low (the non-negative fit lets a little flicker FM take its place), within one run's error. Use runs of several hours for random-walk FM.

# 11 Applications of the phase meter

The phase meter and the recorder measure external clocks and devices as well as the simulator: anything at 10 MHz on IN1 and IN2, against the board's clock and against each other. Five applications follow, each with the cabling, the commands and the result to expect. The figures are *synthetic*: made with the simulator's noise models, the oscillator catalogue and the measured floor of Section 6.3, with fixed seeds (`tools/manual_figures.py`, figures `app_*`). None is a measurement.

Two properties of the meter decide what each application can see:

- *IN1 or IN2 alone* is measured against the board's clock, the 125 MHz crystal (Section 6.3): an uncompensated crystal, of order $10^{-10}$ at 1 s and drifting, which hides anything better. *IN1 − IN2* is free of it, down to the floor.
- The inputs must stay within about ±3 ppm (30 Hz) of the board's 10 MHz with 10 ms windows (Section 10, three-cornered hat); check with `MEAS?` before a long recording. Levels within the input range (LV jumpers: ±1 V).

## 11.1 Testing a phase-locked oscillator

**Idea.** A device that locks its oscillator to an external 10 MHz reference (a frequency synthesizer, a distribution unit with a clean-up oscillator, a disciplined OCXO) is characterized by its closed-loop phase transfer $H(f)$: how much of a phase modulation of the reference reaches its output. For a type-II second-order loop with natural frequency $f_n$ and damping $\zeta$ [24],

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \qquad s = 2\pi i f,\ \omega_n = 2\pi f_n :$$

unity at low frequencies, falling as $f^{-1}$ above the bandwidth, with a peak in between that grows as $\zeta$ falls. OUT1 is the reference, with programmed modulation, steps and noise; the meter compares the device's output with the reference itself. Three measurements on one setup give $|H(f)|$, the step response and the noise transfer.

**Setup.** OUT1 through a BNC T to the device's reference input and to IN2; the device's 10 MHz output to IN1. IN1/IN2 is then the transfer itself: the board's crystal, the window averaging and equal cable delays cancel in the ratio.

**(a) Sine-PM sweep.** The sinusoidal phase modulation (amplitude and frequency; played back, up to a tenth of the segment rate) is stepped through frequencies during one recording of every 10 ms window:

```
import json, time
import numpy as np
from atomic_clock import AtomicClockClient
freqs = np.logspace(-2, 1.5, 15)                        # 0.01 ... 32 Hz
steps = []
with AtomicClockClient("192.168.0.149") as clock:
    clock.set_noise(dither=True)
    clock.set_playback(source="model")
    clock.start_recording("pll", duration_h=3, fast_duration_s=3 * 3600)
    for f in freqs:                                     # 0.01 cycle = 1 ns
        clock.set_noise(pm_sine_amplitude=0.01, pm_sine_frequency=f)
        start = time.time()
        time.sleep(max(30, 30 / f))
        steps.append((f, start, time.time()))
    clock.stop_recording()
json.dump(steps, open("pll_steps.json", "w"))
```

After `stability.py fetch pll`, the ratio of least-squares sine fits at each frequency (on the last two thirds of each step, after the loop has settled) is $H(f)$:

```
from datetime import datetime
d = np.genfromtxt("recordings/pll/fast.csv", delimiter=",", names=True)
start = json.load(open("recordings/pll/header.json"))["start_utc"]
t0 = datetime.fromisoformat(start.replace("Z", "+00:00")).timestamp()
for f, a, b in json.load(open("pll_steps.json")):
    sel = (d["t_s"] > a - t0 + (b - a) / 3) & (d["t_s"] < b - t0)
    t, w = d["t_s"][sel], 2 * np.pi * f
    m = np.column_stack([np.cos(w * t), np.sin(w * t), np.ones_like(t), t])
    c1 = np.linalg.lstsq(m, d["in1_phase"][sel], rcond=None)[0]
    c2 = np.linalg.lstsq(m, d["in2_phase"][sel], rcond=None)[0]
    h = complex(c1[0], -c1[1]) / complex(c2[0], -c2[1])
    print("%8.3f Hz %7.2f dB %7.1f deg"
          % (f, 20 * np.log10(abs(h)), np.degrees(np.angle(h))))
```

The phase of $H$ includes the difference of the two cable delays, $360° f\,\Delta t$, negligible at these frequencies.

**(b) Phase step.** During a recording, `SLEW 0`, then `PHASE 0.1`: OUT1 jumps by 10 ns on one sample. IN2 shows the step, IN1 the device following it; overshoot and ringing give $\zeta$, the settling time $f_n$. Keep the step within the device's phase-detector range.

**(c) Noise transfer.** With white FM of $3{\cdot}10^{-9}$ at 1 s on OUT1 (`NOISE JSON {"h0": 1.8e-17}`), the Allan deviation of IN1 alone follows the reference's noise above the loop's time constant and the device's oscillator below it:

```
python stability.py analyze-file recordings/pll/fast.csv --kind phase \
    --units cycles --time-column 0 --column 1
```

**What to look for** (Figure 15). The measured $|H(f)|$ follows the loop's curve; the narrow, well-damped loop A has 2 dB of peaking, the underdamped loop B 6 dB, which also shows as ringing after the step and as a bump in the noise transfer near $1/(2\pi f_n)$: the loop amplifies reference noise there. Jitter-transfer limits in telecom standards are masks on exactly this curve.

**Pitfalls.** Above the loop bandwidth IN1 shows the device's oscillator *against the board's crystal*, whichever is worse (Figure 15c: the crystal hides the VCXO): the noise transfer is seen only for simulated noise well above the crystal's, and the device's own oscillator is measured with IN1 − IN2 of two such devices or with an external reference. The sweep's fast series needs `fast_duration_s` longer than the sweep; the lowest frequencies take the longest.

## 11.2 Measuring the board's crystal

**Idea.** The board's own 125 MHz crystal sets the absolute stability of everything the board makes, but cancels from all its self-tests. Two independent, better clocks on the inputs measure it with the three-cornered hat (Section 10). Two free-running OCXOs suffice: each is about a hundred times better than an uncompensated crystal at 1 s, and, unlike two GPS-disciplined oscillators that follow the same satellites, their noise is independent, as the hat requires.

**Setup and steps.** OCXO A to IN1, OCXO B to IN2 (warmed up for a day; levels within the input range; a square-wave output works, the meter uses its fundamental). Then

```
python stability.py start crystal --duration-h 24 --note "OCXO A IN1, B IN2"
python stability.py fetch crystal
python stability.py hat recordings/crystal
```

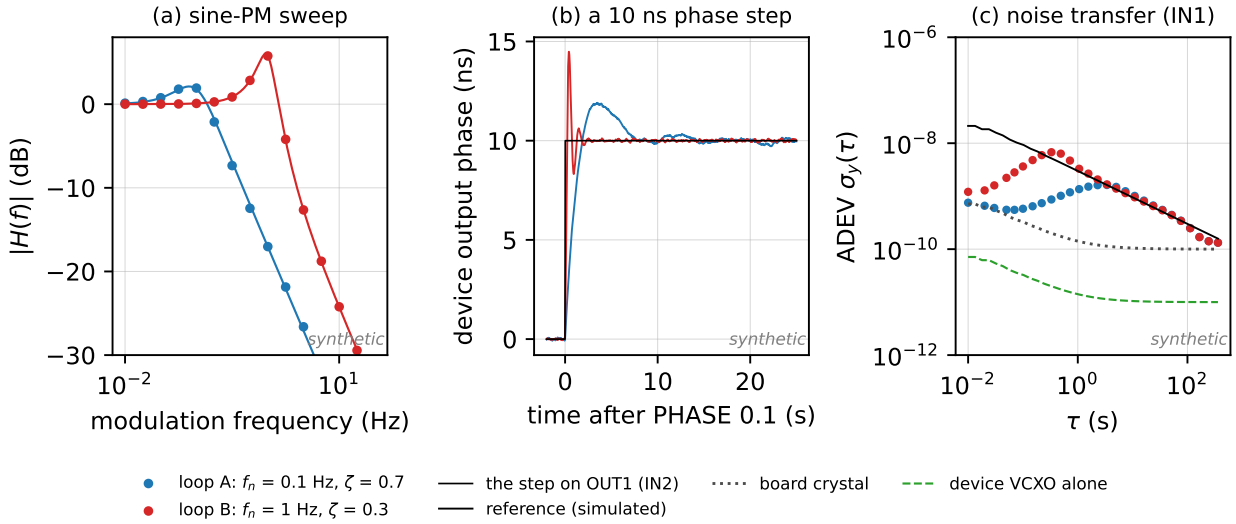



**Figure 15:** Synthetic, for two second-order loops; the device's VCXO has white FM of $10^{-11}$ at 1 s and a flicker floor of $10^{-11}$, the board's crystal is the catalogue's XO. (a) $|H(f)|$: the sweep's points and the loops' curves. (b) Response to a 10 ns step of the reference. (c) ADEV of IN1 with $3 \cdot 10^{-9}$ white FM on the reference.

**What to look for** (Figure 16). The board's crystal R is separated at every $\tau$; for the catalogue's XO, $10^{-10}$ at 1–100 s, rising beyond with its aging and daily temperature cycle. The pairs $\sigma_{AR}$ and $\sigma_{BR}$ are both R; $\sigma_{AB}$, free of R, is the two OCXOs together. Their separate estimates come from small differences of the large R-dominated pairs, so they scatter or come out negative (left out) beyond some tens of seconds; for the OCXOs, $\sigma_{AB}$ is the measurement ($\sqrt{2}$ times one OCXO if both are alike).

**Pitfalls.** Correlated clocks break the hat: whatever noise A and B share is attributed to R. Two GPS-disciplined oscillators share the GNSS time at long $\tau$; their common wander would be reported as the board's. Keep the two references apart (separate supplies, no common distribution amplifier).

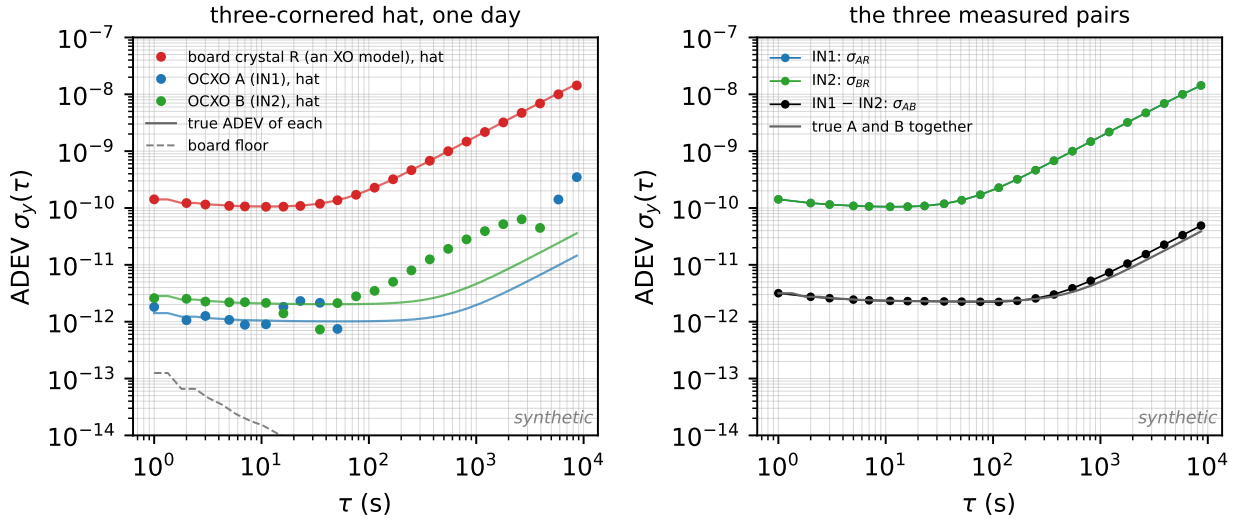



**Figure 16:** Synthetic: one day at 1 s with the catalogue's XO as the board's crystal and two OCXOs (B with twice A's noise). Left: the three-cornered hat (points) against each clock's true ADEV (lines). Right: the three measured pairs; $\sigma_{AR}$ lies under $\sigma_{BR}$.

## 11.3 Comparing two external clocks

**Idea.** With a clock on each input, IN1 − IN2 is their direct comparison: the board's crystal cancels and the floor of Section 6.3, $10^{-13}$ at 1 s and falling, lies far below any commercial standard. The board becomes a two-channel phase comparator for days-long runs, with the reports' ADEV, TDEV and MTIE against ITU-T masks.

**Setup and steps.** A GPS-disciplined OCXO on IN1, a rubidium standard on IN2; three days:

```
python stability.py start gpsdo-rb --duration-h 72 --note "GPSDO IN1, Rb IN2"
python stability.py fetch gpsdo-rb
python stability.py analyze recordings/gpsdo-rb --mask G.811
```

**What to look for** (Figure 17). At short $\tau$ the worse clock dominates (here the rubidium's white FM); between 100 and 1000 s the GPSDO's loop shows as a shoulder where it hands over from its OCXO to GPS; at a day, the rubidium's flicker floor. TDEV, which removes a constant frequency offset, stays inside the G.811 mask as far as it is defined ($10^4$ s). MTIE does not: it includes the rubidium's frequency offset (here $5 \cdot 10^{-12}$, 0.4 µs per day) and aging, and reaches the mask at about a day. A frequency adjustment of the rubidium, or a reference that is itself a primary clock, is needed to pass.

**Pitfalls.** The masks assume a 10 Hz measurement filter and the reference is a perfect clock; here the reference is the other clock, and the result is their sum. For which of the two is which, add a third clock and use the hat.

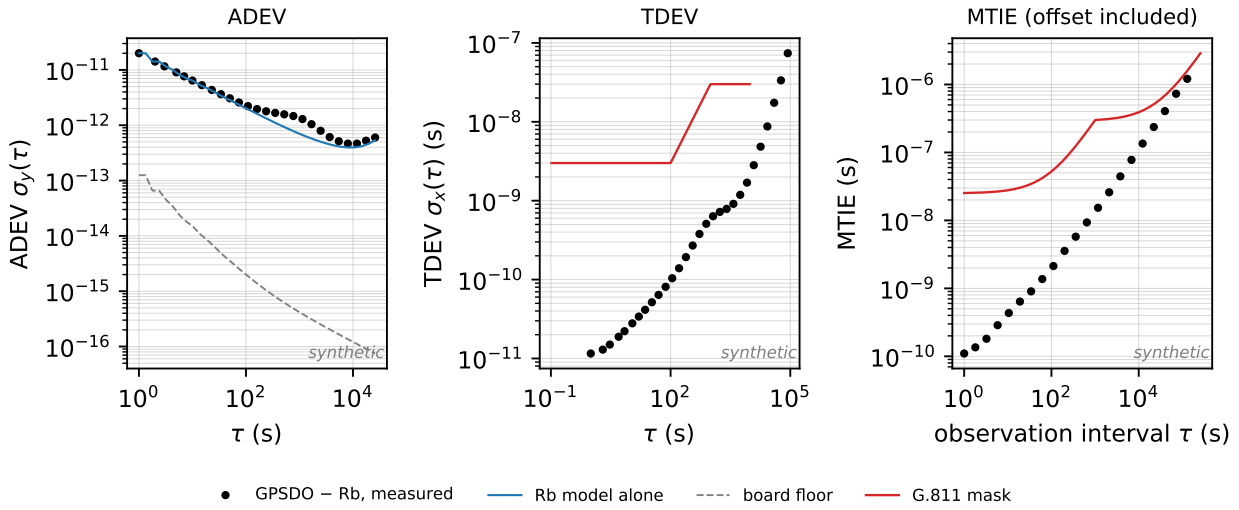



**Figure 17:** Synthetic: three days of a GPS-disciplined OCXO (1000 s loop, 10 ns rms GPS noise) against the catalogue's rubidium with a frequency offset of $5 \cdot 10^{-12}$. ADEV with the rubidium's model and the board floor; TDEV and MTIE with the ITU-T G.811 masks.

## 11.4 Cable delay against temperature

**Idea.** A coaxial cable's delay changes with temperature, typically by tens of ppm per kelvin (more near 19 °C for PTFE dielectric, which has a phase transition there) and by a few ppm per kelvin for phase-stable cables. In timing distribution this is often the largest slow error. The floor's 3 ps peak to peak over a day (Section 6.3) is small enough to measure it for a few metres of cable in an ordinary room.

**Setup and steps.** The loopback of T0b, with the cable under test from OUT1 to IN1 and a short cable from OUT2 to IN2 (the outputs are 50 Ω sources, so the long cable needs no termination at the 1 MΩ input). A temperature logger on the cable, in seconds since the recording's start; the recording's `temp_c` is the Zynq die, not the cable. Record two days, then fit the delay change against the temperature through a first-order lag (the cable follows the air slowly), both with a 3-hour running mean removed, so that the floor's slow wander does not enter:

```
import numpy as np
from scipy.ndimage import uniform_filter1d
from scipy.signal import lfilter
d = np.genfromtxt("recordings/cable/slow.csv", delimiter=",", names=True)
t, x = d["t_s"], -d["difference"] * 100e-9          # delay = -phase/f0, s
T = np.interp(t, *np.loadtxt("cable_temp.csv", delimiter=",", unpack=True))
hp = lambda v: v - uniform_filter1d(v, 3 * 3600, mode="nearest")
for tau in range(0, 3601, 300):                      # lag time constant, s
    Tl = lfilter([1 / tau], [1, 1 / tau - 1], T - T[0]) if tau else T - T[0]
    k = hp(Tl) @ hp(x) / (hp(Tl) @ hp(Tl))
    rms = np.std(hp(x) - k * hp(Tl))
    print(tau, k * 1e12, rms * 1e12)                 # s, ps/K, residual ps
```

The time constant with the smallest residual is the cable's; $k$ there is its coefficient. A longer delay makes IN1 lag, so the delay change is minus the measured phase difference divided by 10 MHz.

**What to look for** (Figure 18). The delay follows the air with the cable's lag. For an assumed 10 m cable with 2.5 ps/K (50 ppm/K of its 50 ns), a daily swing of ±1.5 K and air-conditioning cycles of ±0.25 K, two days give 3.31 ± 0.21 ps/K and a lag of 21 min (true: 2.5 ps/K, 10 min). The uncertainty comes from fitting separate 6-hour stretches: the floor is not white noise, so the formal least-squares error is several times too small. Even so it is somewhat optimistic, and one run can be unlucky: over 16 further simulated two-day runs the coefficient scatters by ±0.44 ps/K about 2.58 ps/K (median stretch-based error 0.21 ps/K), and the lag lies between 9 and 17 min. Quote the scatter of repeated runs, or of a longer one, rather than a single run's error.

**Pitfalls.** A larger, deliberate temperature change (a heated box, a climate chamber) is more precise than the room's. Keep the board and the short cable out of the air flow that drives the cable under test, or their delays enter too.

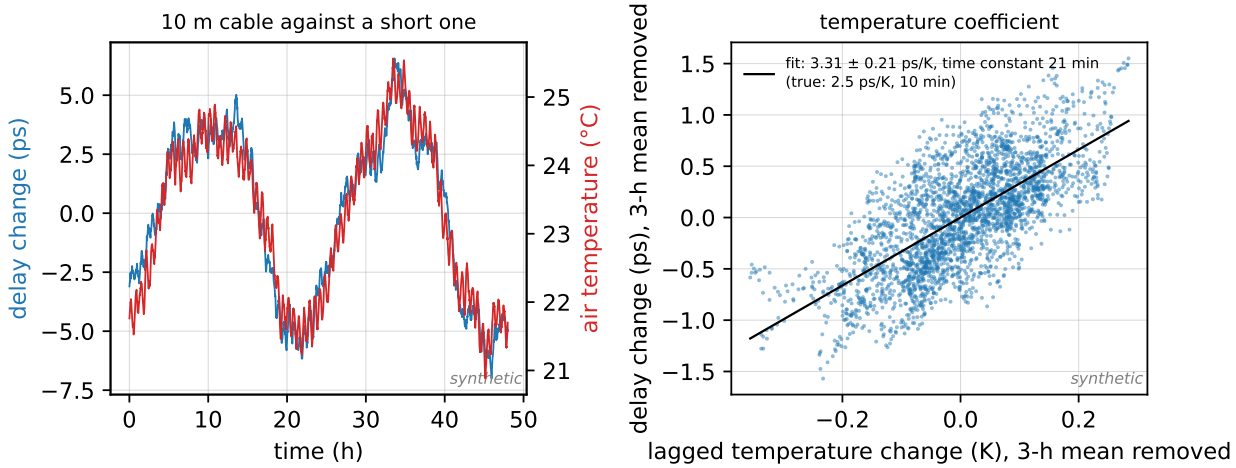



**Figure 18:** Synthetic: a 10 m cable at 2.5 ps/K with a 10-minute thermal lag, on the board's floor, two days at 1 s (1-minute averages shown). Left: the delay change and the air temperature. Right: both with a 3-hour running mean removed, against each other, with the fitted coefficient and time constant.

### 11.5 Residual phase noise of a distribution amplifier

**Idea.** A distribution amplifier, a frequency divider or an isolation transformer in the reference path adds phase noise and slow wander: its *residual* noise, measured with the same clock at its input and at the reference. Inserted in OUT1's path of the loopback, its contribution adds to the floor.

**Setup and steps.** First the floor: T0b with the cables and adapters that will carry the amplifier. Then the amplifier between OUT1 and IN1 (output level within the input range; unused outputs terminated), cold, and a recording from the moment it is switched on:

```
python stability.py start amp --duration-h 6 --note "DA in OUT1 path, cold"
python stability.py analyze recordings/amp --floor recordings/T0b
```

**What to look for** (Figure 19). The floor's flicker PM corresponds to $\mathcal{L}(1\,\mathrm{Hz}) \approx -118\,\mathrm{dBc/Hz}$ ($h_1$ of Table 4); an amplifier with residual flicker PM well below that (here $-125\,\mathrm{dBc/Hz}$) does not show, one above it (here $-110\,\mathrm{dBc/Hz}$) raises the ADEV by its share. The board's strength is the long term: the warm-up (here 20 ps in the first hour) and the temperature coefficient of the delay (here 1 ps/K against 0.1 ps/K), which show in the time error and in the ADEV beyond 100 s, where phase-noise analyzers usually stop.

**Pitfalls.** Phase noise at offsets above some tens of hertz averages away in the 10 ms windows; for it use a phase-noise analyzer. Amplitude changes do not enter the measured phase, but a level outside the input range does (clipping).

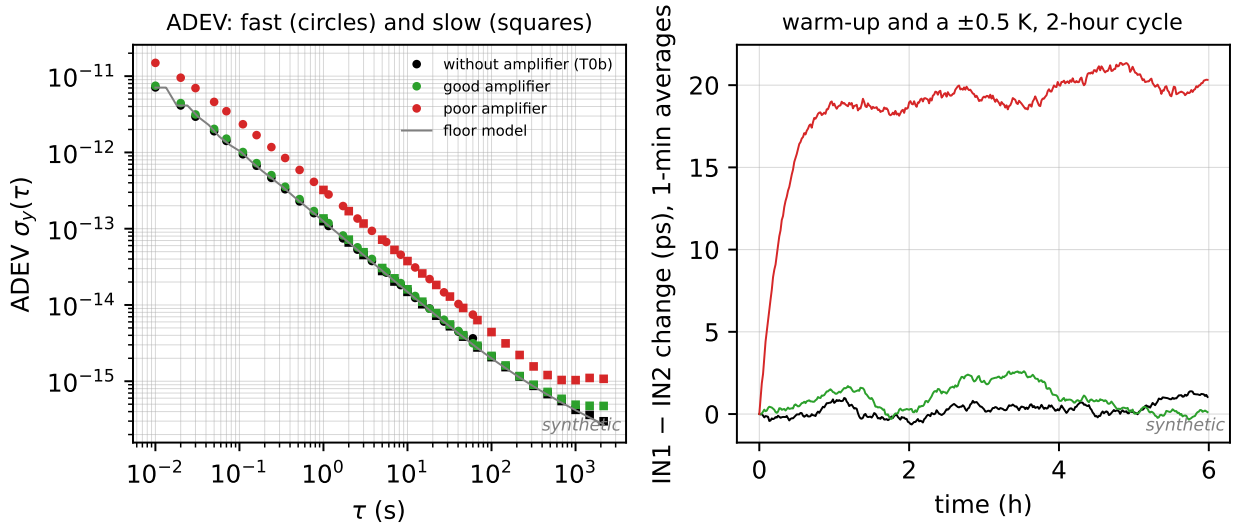



**Figure 19:** Synthetic: the floor alone and with a good and a poor amplifier in OUT1's path (residual flicker PM of $-125$ and $-110\,\mathrm{dBc/Hz}$ at 1 Hz; 0.1 and 1 ps/K; the poor one with a 20 ps warm-up), during a ±0.5 K temperature cycle. Left: ADEV of the fast (10 minutes) and slow (6 hours) series. Right: the time error, 1-minute averages.

## 12 Troubleshooting

**Gaps (lost windows):** the report counts them and analyzes the longest gap-free segment; red lines mark them in the time series. Many gaps mean the board's service is overloaded (check `top` on the board, other clients polling fast); the journal shows `worker stalled` lines with the time each part took.

**Scheduled jobs on the board:** Ubuntu's `apt-daily`, `apt-daily-upgrade` and `man-db` timers load the CPU and the SD card for a minute or more and can stall the service for seconds; the played-back phase then moves while the FIFO is empty. Before long runs: `sudo systemctl disable --now apt-daily.timer apt-daily-upgrade.timer man-db.timer` (enable them again afterwards, or update by hand).

**"resync: underflow" in the journal:** the playback ran dry and restarted from the same trajectory, but while the FIFO was empty the hardware held the last segment's rate, so the played-back phase may have moved (with flicker PM by up to a nanosecond). Recordings flag the row after a resync (`flags` value 2); the analysis splits there and uses the longest clean segment.

**A flat ADEV where a slope is expected:** the term may be below the floor (compare with T0b) or playback is off for a played-back term (the summary page says so).

**Steps or spikes in $x(t)$:** touched cables, connectors, or a setting changed during the run (`RESET` restarts the phase).

**Slow wander that follows the temperature:** the analog paths; improve the thermal environment or shorten the analysis to calm periods.

**The fit attributes noise to the wrong term:** a term must dominate over at least a decade of $\tau$ to be resolved; use the single-term tests T1–T7 first.

**Recording refused:** another recording runs (`REC?`), the name exists, or less than 1 GB is free (`stability.py list`, `delete`).

# 13 Record sheet

| run | date | ADEV(1 s) | ADEV(100 s) | verdict | notes (temperature, cabling) |
|---|---|---|---|---|---|
| T0a | | | | | |
| T0b | | | | | |
| T1-x10 | | | | | |
| T1-x100 | | | | | |
| T2-x10 | | | | | |
| T2-x100 | | | | | |
| T3-x10 | | | | | |
| T3-x100 | | | | | |
| T4 | | | | | |
| T8-x10 | | | | | |
| T8-x100 | | | | | |
| T0b-long | | | | | |
| T5-x10 | | | | | |
| T5-x100 | | | | | |
| T6-x10 | | | | | |
| T6-x100 | | | | | |
| T7-x30 | | | | | |

# A Model details

**The meter's view.** The phase samples are averages of $x$ over each window: the transfer function $|H(f)|^2 = \mathrm{sinc}^2(f\tau_w)$. Expected variances are

$$\sigma_y^2(\tau) = \frac{8}{\tau^2}\int_0^{f_c} S_x(f)\,|H(f)|^2 \sin^4(\pi f\tau)\,\mathrm{d}f,$$
$$\mathrm{mod}\,\sigma_y^2(\tau) = \frac{8}{\tau^2}\int_0^{f_c} S_x(f)\,|H(f)|^2 \frac{\sin^6(\pi f\tau)}{m^2\sin^2(\pi f\tau_0)}\,\mathrm{d}f,$$
$$H\sigma_y^2(\tau) = \frac{32}{3\tau^2}\int_0^{f_c} S_x(f)\,|H(f)|^2 \sin^6(\pi f\tau)\,\mathrm{d}f,$$

with $S_x = S_y/(2\pi f)^2$ and $f_c = 500\,\mathrm{Hz}$ for played-back terms, 62.5 MHz for the FPGA's. White PM is treated as independent window errors of $\sigma_x = \sqrt{1.5\,h_2/(2\tau_w)}/(2\pi)$ (the factor 1.5: the I/Q demodulator's $2f$ product passes half again the variance of phase noise that reaches the 62.5 MHz Nyquist frequency; simulated 1.54, measured in T1 1.52): $\sigma_y^2 = 3\sigma_x^2/\tau^2$, $\mathrm{mod}\,\sigma_y^2 = 3\sigma_x^2/(m\tau^2)$. The window acts like the averaging of a Π-type counter [25]. The model was checked against the standard asymptotes and against simulated noise (power-law noise generated by FFT spectral shaping [26]) (`atomic_clock/tests/test_stability_tools.py`).

**Files.** `header.json` (settings, versions, start and stop, gaps, write errors), `fast.csv` and `slow.csv` with columns `t_s, in1_phase, in2_phase, difference, in1_amp, in2_amp, flags` (phases in cycles, unwrapped; `flags` value 1 for broken continuity, value 2 for playback resynchronization, a start or stop, or a model change, combined by bitwise OR) and `temp_c` in the slow series. Flags are pending independently in the two series; the next slow row retains intervening events.

# References


[1] IEEE standard definitions of physical quantities for fundamental frequency and time metrology — random instabilities. IEEE Std 1139-2022, 2022. Earlier editions 1988, 1999, 2008.

[2] D. W. Allan. Statistics of atomic frequency standards. *Proc. IEEE*, 54(2):221–230, 1966. The two-sample (Allan) variance.

[3] J. A. Barnes, A. R. Chi, L. S. Cutler, et al. Characterization of frequency stability. *IEEE Trans. Instrum. Meas.*, IM-20(2):105–120, 1971. The power-law model and its time-domain measures.

[4] J. Rutman. Characterization of phase and frequency instabilities in precision frequency sources: fifteen years of progress. *Proc. IEEE*, 66(9):1048–1075, 1978.

[5] D. B. Leeson. A simple model of feedback oscillator noise spectrum. *Proc. IEEE*, 54(2):329–330, 1966. Why oscillators show the power laws they do.

[6] D. W. Allan and J. A. Barnes. A modified "Allan variance" with increased oscillator characterization ability. In *Proc. 35th Annual Frequency Control Symposium*, pages 470–475, 1981.

[7] W. J. Riley. *Handbook of Frequency Stability Analysis.* NIST Special Publication 1065. National Institute of Standards and Technology, 2008. The practical reference for the deviations, their confidence and noise identification.

[8] D. B. Sullivan, D. W. Allan, D. A. Howe, and F. L. Walls, editors. *Characterization of Clocks and Oscillators.* NIST Technical Note 1337. National Institute of Standards and Technology, 1990. The classic papers in one volume.

[9] C. A. Greenhall and W. J. Riley. Uncertainty of stability variances based on finite differences. In *Proc. 35th Annual Precise Time and Time Interval (PTTI) Meeting*, 2003. The confidence intervals used in the reports.

[10] A. E. E. Wallin et al. AllanTools. Python library for Allan deviation and related statistics, `https://github.com/aewallin/allantools`.

[11] E. Rubiola. *Phase Noise and Frequency Stability in Oscillators*. Cambridge University Press, 2008.

[12] E. Rubiola and F. Vernotte. The companion of Enrico's chart for phase noise and two-sample variances. *IEEE Trans. Microwave Theory Tech.*, 71(7):2996–3025, 2023. A compact modern summary of the relations.

[13] F. Riehle. *Frequency Standards: Basics and Applications*. Wiley-VCH, Weinheim, 2004.

[14] J. Levine. Introduction to time and frequency metrology. *Rev. Sci. Instrum.*, 70(6):2567–2596, 1999.

[15] D. A. Howe. The total deviation approach to long-term characterization of frequency stability. *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, 47(5):1102–1110, 2000.

[16] D. A. Howe and T. N. Tasset. Thêo1: characterization of very long-term frequency stability. In *Proc. 18th European Frequency and Time Forum*, 2004.

[17] Definitions and terminology for synchronization networks. ITU-T Recommendation G.810, 1996. TIE, MTIE, TDEV.

[18] Timing characteristics of primary reference clocks. ITU-T Recommendation G.811, 1997.

[19] Timing characteristics of synchronous Ethernet equipment slave clock. ITU-T Recommendation G.8262/Y.1362, 2007.

[20] J. E. Gray and D. W. Allan. A method for estimating the frequency stability of an individual oscillator. In *Proc. 28th Annual Symposium on Frequency Control*, 1974. The three-cornered hat.

[21] L. Galleani and P. Tavella. The dynamic Allan variance. *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, 56(3):450–464, 2009.

[22] T. Van Baak. adev1 – Allan deviation statistics. `http://www.leapsecond.com/tools/adev1.htm`, 2006. Command-line tool and C source (2000).

[23] Oscillator, crystal controlled, general specification for. MIL-PRF-55310, U.S. Department of Defense. Logarithmic aging model $y = y_0 + A\ln(1 + Bt)$.

[24] F. M. Gardner. *Phaselock Techniques*. Wiley, 3rd edition, 2005.

[25] E. Rubiola. On the measurement of frequency and of its sample variance with high-resolution counters. *Rev. Sci. Instrum.*, 76(5):054703, 2005.

[26] J. Timmer and M. König. On generating power law noise. *Astron. Astrophys.*, 300:707–710, 1995. Spectral shaping of random Fourier amplitudes.

[27] D. W. Allan. Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators. *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, 34(6):647–654, 1987.