spin 1.1: reference documentation

overview, recipes, theory, control reference, registers, test records

Rok Zitko, Jozef Stefan Institute

2026-11-12 (commit 6f77098)

Contents

spin: a pulsed RF transceiver **2**
Connecting to lab equipment: read first . . . 2
Quick start . . . 2
Presets . . . 3
Specifications (1.1, design values unless stated) . . . 4
Hardware setup . . . 5
Files . . . 5

spin recipes **6**
Protecting an amplifier and a probe . . . 6
Without wiring: the emulator . . . 6
A loopback check: OUT1 → IN1 . . . 6
Pulsed NMR with a probe . . . 6
NQR: a frequency search with spin-locked echo trains . . . 7
Pulsed ESR at an IF: the echo and the field sweep . . . 7
Qubit readout at an IF: blobs, the threshold, active reset . . . 7
Triggered experiments . . . 8
With other rpInstruments designs . . . 8

spin: theory of operation **8**
1. Physics for lab users . . . 8
2. Definitions (the contract) . . . 10
3. Version 0.1 . . . 17
4. Version 0.2 . . . 17
5. Version 0.3 . . . 17
6. Version 0.4 . . . 18
7. Version 0.5 . . . 18
8. Versions 1.0 and 1.1 . . . 18

spin commands (TCP) **19**
Configuration . . . 19

spin registers **21**
Block RAMs . . . 21
Control registers . . . 22

Verification record **24**
2026-11-08: 1.1, the averager in common (build 0x33bbab94) . . . 24
2026-11-08: safety rounding and the gates at a guard trip (build 0x5e4b8972) . . . 25
2026-11-08: the bench session (latencies, gates, the duty guard, AO0, COND) . . . 25
2026-11-08: the manuals . . . 27
2026-11-08: 1.0, the release (plan step 1.0) . . . 27
2026-11-07: 0.5, decisions and branching (plan step 0.5) . . . 28
2026-11-07: 0.4, the emulator, experiment templates and fits (plan step 0.4) . . . 28
2026-11-07: 0.3, receivers, averaging, weighted results, records (plan step 0.3) . . . 30
2026-11-07: 0.2, I/Q outputs, shapes, chirps, limits, the duty guard, slow outputs (plan step 0.2) . . . 32
2026-11-07: 0.1 on the wired bench (the loopback and trigger checks) . . . 34
2026-11-06: 0.1, the pulse program, real transmitters, pins, capture (plan step 0.1) . . . 36

spin: a pulsed RF transceiver

spin turns the Red Pitaya into a **pulsed RF transceiver** for two-level systems. It transmits phase-cycled RF pulses up to about 50 MHz, receives with the same oscillators, and averages the echoes on the board. It also measures one I/Q value per shot and decides on the board, setting a pin or branching its pulse program. A pulse program on the board runs the experiments: FID, spin echoes and T2, CPMG, NQR spin-locked echo trains, inversion and saturation recovery, nutation, ESR echo-detected field sweeps, Rabi, Ramsey, T1, single-shot readout, active reset. A built-in emulator, one qubit or a spin ensemble, answers the transmitter, so all of them run without a magnet.

It is meant for:

- teaching pulsed NMR and NQR;
- pulsed ESR at an intermediate frequency, with an external microwave bridge;
- qubit readout at an intermediate frequency, with external mixers.

Status: version 1.1 (1.0 with its averager moved into `common/`, unchanged). Everything in the specifications below is built and board-tested (docs/VERIFICATION.md: 1.1, build 0x33bbab94, passed all 62 board checks twice; 1.0, build 0x5e4b8972, on 2026-11-08, with OUT1 → IN1, OUT2 → IN2 and a jumper D8 → COND A; the emulator's runs equal the bit-exact model's; latencies, gates, the duty guard and AO0 measured with an oscilloscope). Behaviour with real samples (probes, amplifiers, T/R switches, bridges, spins, qubits) is unmeasured: no such hardware was at hand. RECIPES.md has worked setups, docs/THEORY.md the physics (section 1) and the definitions (section 2); the user's manual and the theory manual (with exercises on the emulator) are in docs/manual/.

Amplitudes are volts under the board's DC calibration. On the test board a cable OUT1 → IN1 reads 3.5 % low at DC, 2.6 % high from 0.25 to 2 MHz and up to 10 % high near 20 MHz (docs/VERIFICATION.md). That is the board's analog path, which needs response tables to correct.

Connecting to lab equipment: read first

Read the equipment-safety section of the root README. For spin in particular:

- **Extension connector E1 is 3.3 V LVCMOS and not 5 V tolerant.**
  - spin reads DIO0_P (TRIG), DIO1_P (COND A) and DIO2_P (COND B), all with pull-downs.
  - A 5 V TTL source needs a 50 Ω terminator at the board, which brings it to about 2.5 V. Never connect 5 V directly.
- **Outputs DIO0_N–DIO7_N** (D1–D8) are high impedance after loading and after SAFE. They are driven only when enabled. Never drive an output pin from outside.
  - A released pin has only the FPGA's weak pull-down.
  - Before the FPGA is configured the pins are not defined at all.
  - **Fit an external pull-down (e.g. 10 kΩ) on any line that unblanks a power amplifier or switches a T/R switch**, so the amplifier stays blanked while the board boots or reloads.
- **OUT1/OUT2** carry the transmitters' RF pulses (calibrated 0 V between them and when no program runs; code 0, about 0 V uncalibrated, right after loading). **Never connect an RF power amplifier's input without an attenuator sized for the amplifier's maximum input.** Set each output's **limits** (`analog`: a clamp in volts on every sample) and each transmitter's **duty guard** (`guard`: the longest pulse, the mean duty and a burst allowance; a trip blanks the RF and the gate pins and stops the run) for the amplifier, the probe and the sample, then **lock them** (`protect(n)`, TCP `PROTECT n`): until the FPGA is reloaded no configuration, preset or SAFE can change them. Unset, nothing limits a pulse but the DAC's range (about ±1.1 V). At a trip the gates end as after any pulse: each closes its lag (+ 32 ns) after the cut RF's end at the connector, so a receiver protection switch stays on through the ring-down.
- **E2 AO0–AO3** (0 … 1.8 V through the board's RC filters) are the slow outputs: they hold their idle levels outside runs, also right after loading (0 V).

Quick start

With spin loaded on a board (no Vivado needed; from the SD image: `LOAD spin`), the emulator's FID without any wiring, through a preset and through a template with its fit:

```
import sys; sys.path[:0] = ["common/python", "spin/control"]
from spin import SpinClient, experiments as X, analysis as A
```

```
with SpinClient("192.168.0.102") as dev:
    dev.load_preset("fid-emulator")               # an ensemble at 10.02 MHz, a 20 kHz line
    p = X.pulses(f=10e6, amplitude=0.8, t90=0.25e-6, dwell=0.5e-6, points=256, tr=5e-3, scans=64)
    r = X.acquire(dev, X.fid(p))                  # APPLY, run, collect: volts at the LO
    print(A.fit_fid(r["t"], r["traces"][0]))      # T2*, the offset (20 kHz), the amplitude
```

In Jupyter (on the board at port 9090, or on a PC) `spin.make_gui(dev)` opens the panel: run control, the program as a sequence view, the emulator, the templates with their traces, spectra and fits, a window's live average, the I/Q blobs of the RESULT records, the outputs and the presets. `X.simulate(driver, exp)` runs an experiment in the bit-exact model on a PC.

A program written by hand: a 1 µs pulse of 0.2 V at 10 MHz every 100 µs, its phase cycled through 0, 90, 180 and 270 degrees from a table, a gate on D1 that opens 40 ns before each pulse, a capture of IN1 and window 0 averaging 200 points of 64 ns at TX1's oscillator (a cable OUT1 → IN1 shows the pulse):

```
win = {"rx": 1, "lo": 1, "raw": False, "average": True, "weigh": False, "points": 200,
       "decimation": 8, "region": 0, "region_table": None, "weights": None, "phase": 0.0,
       "threshold": 0.0}
with SpinClient("192.168.0.102") as dev:
    dev.apply({
        "tx": {"1": {"frequencies": [10e6] + [0.0] * 7, "amplitudes": [0.2] + [0.0] * 7}},
        "tables": {"counters": [0, 0, 0, 0], "data": [[0, 16384, 32768, 49152], [], [], []]},
        "timer": 100e-6, "windows": {"0": win},
        "outputs": {"1": {"mode": "gate1"}}, "gates": {"tx1": {"lead": 40e-9, "lag": 200e-9}},
        "capture": {"length": 2000},
        "program": [
            {"op": "loop", "count": 4, "duration": 1e-6, "capture": True, "shot": True,
             "window": 0, "tx1": {"on": True, "phase_table": 0}},
            {"op": "wait", "wait": "timer"},
            {"op": "end", "duration": 1e-6},
            {"op": "stop"}]})
    dev.collect(); dev.clear_average()            # earlier sums out
    dev.run()
    print(dev.state())                            # shots, errors, missed triggers
    print(dev.capture()["in1"][:20])              # volts at IN1, 8 ns apart
    dev.swap(); dev.collect()
    print(dev.window_average(0)["volts"][2:6])    # [re, im] volts per point
```

With `"weigh": True` a window also gives S = Σ w Z (weights normalised so a steady tone reads its amplitude in volts) and a state Re S > threshold, as a RESULT record per shot (`dev.results(n)`: the last n), for branches (`{"op": "branch", "if": {"state": w}}`) and state outputs.

Presets

Preset	What it sets up (RUN starts it)
`off`	all outputs in high impedance, both transmitters off, the program a STOP
`pulse-loopback`	four 1 µs pulses of 0.2 V at 10 MHz, phases 0/90/180/270 with the receiver following (CYCLOPS), the gate on D1, IN1 captured and averaged (a cable OUT1 → IN1)
`fid-emulator`	the emulated ensemble (10.02 MHz, a 20 kHz line, T1 1 ms, T2 300 µs): FIDs after 0.25 µs 90° pulses, CYCLOPS, 64 scans
`echo-emulator`	the same: Hahn echoes, τ = 100 µs
`cpmg-emulator`	the same: CPMG, τ = 100 µs, 32 echoes
`nutation-emulator`	the same: pulses of 0.05 … 1 µs, the nutation curve
`rabi-qubit`	the emulated qubit (10 MHz, T1 20 µs, readout at 12 MHz, ±60°): Rabi over 0.05 … 2 µs with two references
`ramsey-qubit`	the qubit 500 kHz below TX1: Ramsey over 0.1 … 3 µs
`readout-qubit`	1000 single shots each of the ground and the excited state, a RESULT record per shot (the I/Q blobs)
`nmr-fid-in1`	a real probe on IN1 at 10 MHz: 1 µs pulses of 0.2 V, a power-amplifier gate on D1, OUT1 limited to ±0.5 V, the duty guard at 50 µs and 5 %
`esr-iq-echo`	an IQ bridge: I/Q mode (TX1 at a 10 MHz IF), Hahn echoes read on IN1 + i IN2, the gate on D1, the same limits

Saved presets (`save_preset`) keep a whole configuration; limits locked by PROTECT survive every preset.

Specifications (1.1, design values unless stated)

Quantity	Value
Program	1024 instructions, 16 ns … 34 s each in 8 ns steps; 4 nested loops; 4 tables of 1024 words; instructions of 32 ns or more without limit, about 30 of 16 ns back to back
Triggers	TRIG (DIO0_P) time-stamped to 1 ns; the next instruction 192 ns after the edge, its RF phase referred to the edge within 1 ns (measured against an asynchronous trigger: 0.28 ns rms, 1.0 ns peak to peak); at the connectors a pulse starts 377–384 ns after the edge (one 8 ns clock of spread, measured); a timer; software
Transmitters	2 × 8 frequency registers (0.44 µHz steps, phase-coherent switching), 8 phases (0.0055°) and 8 amplitudes each, or from tables; 7 linear chirps each (3.5 mHz/s … 4.9·10^14 Hz/s); envelope shapes (amplitude and phase, 31 shapes in a shared table of 4096 points, interpolated, 1/65536 … 16 points per sample)
Outputs	two real outputs (OUT1 = TX1, OUT2 = TX2) or an I/Q pair (OUT1 + i OUT2 = TX1 + TX2, each TX's Q path with its own gain and phase), DC levels, lockable clamping limits; amplitudes in volts through the board calibration (unmeasured above DC: on the test board a cable OUT1 → IN1 reads 3.5 % low at DC, up to 10 % high near 20 MHz)
Duty guard	per TX: the longest pulse (8 ns steps; the limits rounded down, never above what is asked), a mean duty with a burst allowance (a leaky bucket); a trip blanks the RF from its sample and stops the run 3 clocks later; the gates close their lag + 32 ns after the cut RF, as after any pulse; lockable
Slow outputs	AO0–AO3 on E2, 0 … 1.8 V (PWM 488 kHz with a second-order ΔΣ, 16 bits), idle levels and 8 registers or tables set per instruction; on the test board AO0 within 28 mV of set (oscilloscope mean, 0.1–1.7 V) with a 488 kHz ripple at the pin of 0.23–0.91 V p-p (largest at half scale): add a low-pass for a field supply or a bias
Outputs D1–D8	2 ns bits, delays 0 … 2 µs in 2 ns steps; sources TTL, gates (whole clocks, rounded up: lead ≤ about 130 ns, lag ≤ 524 µs), capture window, shot, run, a window's state; aligned with OUT1 at the connectors through the board calibration (measured on D7: 1.5 ns early, ±1 ns from the 2 ns grid; a gate on D1 within 2.5 ns)
Capture	1 … 4096 samples of IN1 and IN2 at 8 ns, aligned to the IN connector through the calibration's input delay
Receivers	RX1, RX2: IN1, IN2 or IN1 + i IN2 minus the calibrated offsets, mixed with TX1's or TX2's oscillator (per window), an order-2 CIC with decimation 1 … 16384 (8 ns … 131 µs per point), exact 90° receiver phases per instruction or table, raw mode, an overload flag per window; aligned to the IN connector through the calibration (measured: a receiver and the capture agree within 0.04 %, the points' phase within 0.2°)
Windows	8 descriptors: 1 … 2048 points, a region in the averager (a base plus a table entry), weights, a threshold; one window at a time per receiver, 4 samples between windows
Averaging	per receiver 2 halves × 2048 complex points (35-bit sums, saturation flagged); swaps at a SHOT after N scans or on request, into a free half (no dead time; a released half is cleared in 16.4 µs first); the service reads each frozen half into 64-bit sums every 0.1 s (a half in about 0.15 s, measured)
Weighted results	1024 complex weights per receiver (18 bits), S = Σ w Z at the full rate, normalised so a steady tone reads its amplitude in volts; the state Re S > threshold; a RESULT record per window (shot, time, S, state, overload)
Emulator	one Bloch vector updated every 128 ns (an exact Rodrigues rotation with T1 and T2 ≤ 2 T1) in TX1's frame: TX1's drive (pulses, phases, shapes, chirps; Rabi ≤ 1.95 MHz at full amplitude), the detuning from TX1 (within ±3.125 MHz), a detuning drawn per SHOT (Gaussian or Lorentzian quantiles), AO0 shifting it (a field sweep); qubit: an outcome drawn at each TX2 readout from (1 − z)/2, decay during the readout, a one-pole resonator at ±χ in TX2's frame; ensemble: the transverse magnetization in TX1's frame; Gaussian noise; seeded or free-running; injected digitally into a receiver (its delay 89 clocks); bit-exact with `emulator.py`, within 0.1 % of the float Bloch equations
Decisions	BRANCH on a window's state (Re S > threshold, decided 21 clocks after its last sample), COND A or B (DIO1_P, DIO2_P) or a loop counter's parity, inverted or not; the least durations (checked by the driver, measured on the board to the clock): 29 clocks on COND, (K + 1) R + D + 26 clocks from the window's instruction on a state; outputs D1–D8 can show a window's state; from a window's last sample at IN to a chosen pulse at OUT D + 28 clocks (512 ns nominal, 488 ns on the test board: measured as programmed), to a state output D + 5.5 clocks + the pins' calibrated delay (332 ns nominal, 281 ns measured on the test board)

Quantity	Value
Experiments	templates (`experiments.py`): FID, Hahn echo and T2, CPMG, NQR spin-locked echo trains, inversion and saturation recovery, nutation, ESR echo-detected field sweeps (AO0 from a table); Rabi, Ramsey, T1, single-shot readout, active reset, repeat-until-success; phase cycles (2-step, CYCLOPS), dummy scans, arrays as averager regions; fits (`analysis.py`): spectra with the CIC's droop corrected, phasing, T1, T2, T2*, Rabi, Ramsey, populations, readout fidelity
Records	a ring in DDR through S_AXI_HP0: RUN, ERROR (guard, underflow, a window not taken, records lost), BLOCK, TIME, RESULT; `RECORDS STREAM` on the service; 100,000 RESULTs per second without loss (measured, 0.2 s)

Hardware setup

Pin (E1)	Use
DIO0_P	TRIG (a pulse program waits for it; 1 ns timestamps)
DIO1_P	COND A (a branch condition)
DIO2_P	COND B (a branch condition)
DIO0_N … DIO7_N	outputs D1 … D8 (TTL, power-amplifier gates, T/R switch, windows, states)
DIO3_P … DIO7_P	unused
E2 AO0 … AO3	slow outputs (0 … 1.8 V)

Files

Path	Contents
`src/sp_core.v`	the core: registers, block RAM windows, the blocks below
`src/sp_seq.v`	the pulse program: engine, FIFO, player, WAITs
`src/sp_tx.v`	a transmitter: phase file, NCO with chirps, envelope, I and Q sines and amplitudes
`src/sp_shapes.v`, `src/sp_guard.v`	the shared shape table, the duty guard
`src/sp_pins.v`, `src/sp_cap.v`, `src/sp_trig.v`, `src/sp_ram.v`	the outputs' sources and delays, the capture, the trigger's 1 ns edge, block RAM
`src/sp_rx.v`, `src/sp_rec.v`	a receiver (mixer, CIC, receiver phase, weighted sum), the records; the averager is the common `rp_avg` (`../common/rtl/optional/rp_avg.v`, 1.1)
`src/sp_emu.v`, `src/sp_rod_rom.v`	the emulator, its rotation table (generated by `emulator.py`)
`control/spin/`	driver, client, protocol (the plugin), `gui.py` (the Jupyter panel), `model.py` (bit-exact), `sequence.py` (programs), `worker.py` and `hub.py` (records, the averager's host sums, `RECORDS STREAM`), `emulator.py` (the emulator, bit-exact), `bloch.py` (float Bloch equations), `experiments.py` (templates, `acquire`, `simulate`), `analysis.py` (spectra, populations, fits)
`control/spin.ipynb`	the notebook
`RECIPES.md`	worked setups: the emulator, a loopback, NMR with a probe, NQR, ESR through a bridge, qubit readout, triggers
`docs/manual/`	the user's manual and the theory manual (LaTeX, committed PDFs), the board recordings behind their figures (`data/`)
`tools/`	host tools: the manuals' board recordings (`record_manual_data.py`) and figures (`manual_figures.py`, `theory_figures.py`)
`docs/`	THEORY (physics, definitions), REGISTERS, PROTOCOL, VERIFICATION
`scripts/test_hw.py`	the board test (`make test-hw`)
`tests/`	the driver/protocol/client and model tests, the XSim testbenches and their vectors (`vectors.py`)

spin recipes

Worked examples in Python on a PC (`dev = SpinClient("192.168.0.102")`) or on the board's Jupyter (`dev` from the notebook's first cell): the methods are the same. Times are in seconds and amplitudes in volts at the connectors (through the board calibration). The physics behind each recipe is in docs/THEORY.md section 1, the commands in docs/PROTOCOL.md, the presets in README.md.

```
import sys; sys.path[:0] = ["common/python", "spin/control"]
from spin import SpinClient, experiments as X, analysis as A
dev = SpinClient("192.168.0.102")
```

Read the root README's equipment-safety section first. OUT1/OUT2 give up to ±1 V into a high impedance (about ±0.5 V into 50 Ω); never connect a power amplifier's output to an input. The DIO pins are 3.3 V: a power-amplifier gate or a T/R switch driver needs a pull-down at its input so that it stays off while the board boots or a design loads (the pins are in high impedance then). Before connecting an amplifier, set limits and the duty guard (below) and lock them.

Protecting an amplifier and a probe

```
dev.apply({"analog": {"1": {"dc": 0.0, "limits": {"low": -0.3, "high": 0.3, "protect": True}}},
           "guard": {"1": {"max_pulse": 20e-6, "max_duty": 0.02, "burst": 100e-6}}})
```

The limits clamp every sample of OUT1 (volts at the connector); the guard stops a run whose pulse is longer than `max_pulse` or whose mean duty over the bucket exceeds `max_duty`, blanks the RF from that sample and stops the run; the gates close their lag after the cut RF, as after any pulse, so a T/R switch keeps protecting the receiver. `protect` locks both until the FPGA is reloaded (SAFE, presets and restores keep them). Choose them from the amplifier's and the probe's ratings, not from the experiment.

Without wiring: the emulator

```
dev.load_preset("fid-emulator")                # an ensemble at 10.02 MHz, a 20 kHz line, T2 300 us
r = X.acquire(dev, X.fid(X.pulses(f=10e6, amplitude=0.8, t90=0.25e-6, dwell=0.5e-6, points=256,
                                  tr=5e-3, scans=64)))
print(A.fit_fid(r["t"], r["traces"][0]))         # T2* from the line, the offset (20 kHz), the
↪  amplitude
f, s = A.spectrum(A.phased(r["traces"][0]), 0.5e-6)
```

The emulator answers TX1 at the receivers whose input is `emulator` (`rx: {"1": {"input": "emulator"}}`), so every template runs without a magnet: nutation, echoes, CPMG, recoveries, Rabi, Ramsey, T1, readout, active reset. Its spins must satisfy T2 ≤ 2 T1; `t2: null` means 2 T1. `X.simulate(driver, exp)` runs the same experiment in the bit-exact model on a PC (a seeded board run equals it when started at the board's START clock).

A loopback check: OUT1 → IN1

A coax cable OUT1 → IN1 (no terminator):

```
dev.load_preset("pulse-loopback")              # four 1 us pulses of 0.2 V at 10 MHz, CYCLOPS
dev.run()
cap = dev.capture()["in1"]                     # 2048 samples of IN1 in volts, from the first pulse
dev.collect(); avg = dev.window_average(0)       # 200 points of 64 ns at TX1's LO
```

The capture starts at the IN connector at the instruction's start (through the calibration's input delay), so a pulse appears at its first sample; the window's points are the pulse's amplitude and phase in volts. On the test board the loop reads 3.5 % low at DC and up to 10 % high near 20 MHz: the board's analog path, which the DC calibration does not correct.

Pulsed NMR with a probe

Wiring: OUT1 → a power amplifier → a duplexer (crossed diodes and λ/4 lines, or a T/R switch) → the probe; the probe → the duplexer → a preamplifier → IN1. D1 (DIO0_N) gates the amplifier (`gate1`: on during TX1's pulses, with a lead and a lag); a T/R switch can take D2 with a longer lag (the receiver protection).

```
dev.load_preset("nmr-fid-in1")                 # 1 us pulses of 0.2 V, D1 the gate, limits and guard
p = X.pulses(f=21.3e6, amplitude=0.2, t90=1e-6, dwell=1e-6, points=1024, dead=5e-6, tr=2.0,
```

```
            scans=8, cycle="cyclops", input="in1")
nut = X.acquire(dev, X.nutation(dict(p, points=8), [0.5e-6 * k for k in range(1, 25)]))
z = A.phased([tr[0] for tr in nut["traces"]])
print(A.fit_oscillation(nut["axis"], z.real))    # f: the Rabi frequency; t90 = 1 / (4 f)
fid = X.acquire(dev, X.fid(dict(p, t90=...)))    # the calibrated 90
```

Find the resonance first with a long tr and a few scans (a FID's offset is its beat against TX1); then calibrate the 90° by nutation; then `t2` (Hahn echoes against τ), `cpmg`, `inversion_recovery` or `saturation_recovery` with `analysis.fit_decay` / `fit_recovery`. Set `dead` from the probe's ring-down and the preamplifier's recovery (watch the first points of a FID without a sample). At low fields (protons below a few hundred kHz) the input's high-pass (a pole-zero near 21 kHz on the test board) bends the baseline; it is not in the calibration. Probe ring-down, duplexer and amplifier recovery are unmeasured.

NQR: a frequency search with spin-locked echo trains

```
p = X.pulses(f=3.6e6, amplitude=0.3, t90=10e-6, dwell=2e-6, points=32, dead=20e-6, tr=0.5,
             scans=16, cycle="2", input="in1")
found = []
for f in [3.55e6 + 10e3 * k for k in range(11)]:
    r = X.acquire(dev, X.slse(dict(p, f=f), 300e-6, 64))      # 64 echoes 600 us apart
    found.append((f, sum(abs(tr[16]) for tr in r["traces"])))   # the summed echo tops
```

Each step's band is about 1/t_p (100 kHz for 10 µs pulses): step by less. The train lasts T2eff (longer than T2 in a powder); sum its echoes. Re-calibrate the pulse at the line found, by the largest echo (a powder's nutation is not a sine).

Pulsed ESR at an IF: the echo and the field sweep

Wiring: OUT1, OUT2 → the bridge's IQ modulator (I, Q) at the IF; the bridge's IQ demodulator → IN1, IN2; D1 the microwave amplifier's gate; AO0 (E2) → the field coil's supply (its control input: 0 … 1.8 V, through the board's RC filter).

```
dev.load_preset("esr-iq-echo")                # I/Q mode, Hahn echoes 90x - 20 us - 180y on IN1 + i
 ↪  IN2
dev.apply({"tx": {"1": {"q_gain": 1.0, "q_phase": 0.0}},     # trim the modulator's image here
           "analog": {"1": {"dc": 0.0}, "2": {"dc": 0.0}}})   # and its LO leakage (DC on OUT1, OUT2)
p = X.pulses(f=10e6, amplitude=0.2, t90=50e-9, dwell=16e-9, points=64, dead=200e-9, tr=1e-3,
             scans=64, cycle="cyclops", input="iq")
sweep = X.acquire(dev, X.field_sweep(p, 400e-9, [0.6 + 0.01 * k for k in range(61)], output=0,
                                     settle=5e-3))
line = [abs(sum(tr)) for tr in sweep["traces"]]               # the echo against AO0's volts
```

Trim the image by a CW tone (q_gain, q_phase per TX, at the frequency used) and the LO leakage by the DC levels, watching the bridge's spectrum. Choose `settle` from the coil supply's step response. In the emulator `ao` (Hz/V) moves the spins with AO0, so the recipe runs without a magnet (the line's centroid lands within 0.01 V of the expected level in `test_experiments`). A real bridge's image rejection, LO suppression and dead time are unmeasured.

Qubit readout at an IF: blobs, the threshold, active reset

Wiring: OUT1 → the drive's IQ mixer (or a real IF), OUT2 → the readout mixer, the readout's down-converted IF → IN1 (or IN1 + i IN2). Here on the emulator (`readout-qubit` preset).

```
dev.load_preset("rabi-qubit")
q = X.qubit(f=10e6, amplitude=0.8, t_pi=0.5e-6, f_read=12e6, amp_read=0.2, t_read=0.5e-6,
            tr=100e-6, scans=500)
ref = X.acquire(dev, X.rabi(q, [0.5e-6]))                     # regions: pi, ground, excited
phase, threshold = X.decision(ref["traces"][1][0], ref["traces"][2][0])
q = dict(q, phase=phase, threshold=threshold)
dev.load_preset("readout-qubit")                              # RESULT records per shot (window 0)
dev.apply({"windows": {"0": {"phase": phase, "threshold": threshold}}})
dev.run()                                                   # ... then dev.results(2000): the blobs
for exp in (X.active_reset(q), X.repeat_until_success(q, attempts=8)):
    z = [tr[0] for tr in X.acquire(dev, exp)["traces"]]
    print(exp.name, A.population([z[0]], z[1], z[2])[0])      # excitation left / reached
```

`analysis.readout(s0, s1)` gives the fidelity and the separation SNR of two sets of RESULTs. The decision's latency (THEORY 2.1, 2.2) is D + 28 clocks from the window's last sample to the chosen pulse; the driver refuses branches that come earlier. On the emulator (T1 20 µs) an active reset leaves 0.03–0.04 of a half-excited qubit excited. A state output (`outputs: {"n": {"mode": "state", "bit": w}}`) shows window w's decision on a DIO_N pin for another instrument.

Triggered experiments

```
dev.apply({"program": [{"op": "wait", "wait": "trigger", "reref": True},
                       {"duration": 1e-6, "tx1": {"on": True}, "shot": True},
                       {"duration": 50e-6, "window": 0}, {"op": "stop"}]})
```

A WAIT on TRIG (DIO0_P, 3.3 V) continues 192 ns after the edge; with `reref` the RF phase is referred to the edge within 1 ns (0.28 ns rms against an asynchronous trigger, measured), so averaging over triggers from another instrument (a field cycle, a laser, awg on a second board) keeps the phase. A branch on COND A or COND B (DIO1_P, DIO2_P) follows an external level read 23 clocks after the branch's start.

With other rpInstruments designs

awg on a second board makes triggers, markers and arbitrary pulse patterns; counter counts photons in gates (optically detected experiments); timetagger time-stamps events. The pin maps, levels, cables and the time and frequency across boards are in guides/combining (the guide's spin section lists its pins).

spin: theory of operation

spin is a pulsed RF transceiver. It transmits phase-cycled RF pulses (up to about 50 MHz, set in volts at the connectors), receives with the same oscillators, averages echoes on the chip, measures one I/Q value per shot with a matched filter, and decides on the board (a state bit on a pin, a branch of the pulse program). An instruction sequencer on the board runs the experiments: FID, spin echo, CPMG, inversion and saturation recovery, nutation, Rabi, Ramsey, NQR echo trains, ESR echo-detected field sweeps, single-shot readout and active reset. Its targets are teaching pulsed NMR, NQR, pulsed ESR at an intermediate frequency (with an external microwave bridge) and qubit readout at an intermediate frequency (with external mixers). A built-in emulator (one qubit, or an ensemble of spins) answers the transmitter, so the experiments run without a magnet.

Status: 1.0 (2026-11-08). Everything above is built and board-tested (VERIFICATION.md); behaviour with real samples, probes, bridges and qubits is unmeasured (no such hardware was at hand). Section 1 explains the physics for lab users. The definitions in section 2 are the **contract** for the bit-exact model (`control/spin/model.py`) and the RTL: they fix what every number means. Sections 3–8 record what each step added.

1. Physics for lab users

This section explains the experiments spin runs and what limits them, with the numbers the board measured on its emulator (docs/VERIFICATION.md). Section 2 defines every register and clock; the templates are in `control/spin/experiments.py`, the fits in `analysis.py`, worked setups in RECIPES.md.

1.1 The Bloch equations and the rotating frame

A spin ensemble (or one qubit) is described by its magnetization M, a vector of length up to 1 in units of the equilibrium magnetization. In a static field it precesses at the Larmor frequency $f_0 = \gamma B_0/2\pi$. spin describes it in the frame that rotates with the transmitter TX1 (its NCO, with every phase jump): there a spin at f_0 only sees the offset $\Delta = 2\pi$ (f_0 − f_TX1), and a pulse of TX1 is a field B_1 along an axis in the transverse plane set by the pulse's phase (0° = x, 90° = y). The Bloch equations

$$dM/dt = \Omega \times M - (Mx/T2,\ My/T2,\ (Mz - Meq)/T1), \quad \Omega = (\Omega_R \cos\varphi,\ \Omega_R \sin\varphi,\ \Delta)$$

turn M about the effective field with the Rabi frequency f_R = Ω_R/2π, proportional to TX1's amplitude (Hz per volt: set by the coil and the amplifier, measured by nutation, 1.2). A pulse of length t_p turns M by θ = 2π f_R t_p on resonance: a 90° pulse puts M into the transverse plane, a 180° pulse inverts it. T1 is the longitudinal relaxation (Mz → Meq), T2 the transverse; every spin has **T2 ≤ 2 T1**, and the emulator refuses longer T2 (its vector would grow beyond 1). The receivers demodulate with the same NCO, so the measured signal is M's transverse part in that frame, with a fixed phase that `analysis.phase` removes.

1.2 Pulses: hard, shaped, chirped; the 90° pulse

A rectangular ("hard") pulse of length t_p excites a band of about 1/t_p around its frequency; off resonance by more than f_R the spins turn about a tilted axis and the flip angle and phase change (the field-sweep recipe uses this selectivity on purpose). Shaped pulses (envelope tables, amplitude and phase, 2.3) narrow the band or flatten it; a linear chirp sweeps the frequency through the line and inverts it adiabatically when the sweep is slow against f_R^2 ($\Delta f/t_p$ much less than $2\pi f_R^2$), insensitive to B_1 errors.

Calibrating the 90°: the nutation template steps the pulse length and reads the FID's first points: the signal follows sin θ, so its first maximum is the 90° pulse and its zero the 180° (`analysis.fit_oscillation` gives f_R, damped by B_1 inhomogeneity across the sample). The board measured the emulator's Rabi frequency within 0.1 % (0.9990–0.9991 MHz for 1 MHz).

1.3 The FID, T2* and the spectrum

After a 90° pulse the transverse magnetization precesses at each spin's offset and decays: the free induction decay (FID). Field inhomogeneity spreads the offsets, so the FID decays with T2* < T2 (a Gaussian line of rms width σ gives a Gaussian decay $\exp(-(2\pi\sigma t)^2/2)$; a Lorentzian of half width Γ an exponential with T2* = 1/(2πΓ)). The receiver samples it every **dwell** (8 ns × the decimation R), through an order-2 CIC filter (a triangle over 2R − 1 samples): the band is ±1/(2 dwell) around TX1's frequency, the response sinc^2 across it, which `analysis.spectrum` divides out. The spectrum is the Fourier transform of the FID: weight its first point by ½ (no baseline offset), zero-fill, phase it (zero order: one angle; the dead time before the first point adds a phase proportional to the offset). The **dead time** (template `dead`) lets the probe ring down and the receiver recover after a pulse; nothing in spin measures that recovery: it depends on your probe, duplexer and preamplifier.

The board measured the emulator's FID: T2* 7.999–8.000 µs for 8 µs, the offset 50.00 kHz, the amplitude within 0.8 % of the set signal (`analysis.fit_fid`).

1.4 Spin echoes, T2 and echo trains

A 180° pulse at time τ reverses the phases the static offsets accumulated, and they refocus at 2τ: the **Hahn echo** (90x − τ − 180y − τ). Its height decays with T2 (the irreversible part), not T2*: the `t2` template arrays τ and `analysis.fit_decay` gives T2. Diffusion in a field gradient makes the decay faster than exponential at long τ. **CPMG** repeats the 180° pulses (along y, the Meiboom–Gill phase, which corrects flip-angle errors) every 2τ and takes every echo in one scan; the board measured the emulator's T2 by CPMG at 40.04 µs for 40 µs. In NQR the refocusing pulses of a **spin-locked echo train** (SLSE, `slse`: 90° pulses by default) lock the magnetization of a quadrupolar powder, so the train lasts longer than T2 and adds many echoes per scan, which is how weak NQR lines are found. That is a property of spins I > 1/2 in a powder: the emulator's spin-1/2 shows a 90° train's own behaviour (on its line 20 kHz off resonance the train decays faster than CPMG; the user's manual compares it with the Bloch equations).

1.5 T1: inversion and saturation recovery; the repetition time

Inversion recovery (180 − t − 90 − FID) follows M(t) = M_0 (1 − B e^{−t/T1}), B = 2 for a perfect inversion (`analysis.fit_recovery` fits B too); the null at t = T1 ln 2 is a quick estimate. **Saturation recovery** (a comb of 90° pulses, then t − 90) needs no full relaxation before each scan. The repetition time `tr` sets the steady state: with tr ≥ 5 T1 each scan starts at equilibrium; shorter tr gives more scans per second at a lower signal (the optimum for S/N per time is about 1.3 T1 for 90° pulses), but every template's dummies should then precede the scans so the steady state is reached. The board measured the emulator's T1 by inversion recovery at 10.00 µs for 10 µs.

1.6 Phase cycling

A receiver adds its own DC offset and, for I/Q pairs, gain and phase errors between I and Q. Cycling the transmitter's phase with the receiver's cancels them: the **2-step** cycle (0°, 180° on both) cancels offsets; **CYCLOPS** (0°, 90°, 180°, 270°) also cancels the I/Q imbalance and the image. spin applies the receiver phase exactly (90° steps by swapping and negating, 2.4) per instruction or from a table, so the cycle is subtracted on the board before averaging. Echoes can add EXORCYCLE (the 180° pulse's phase cycled too, with the receiver at twice its phase step) through the same tables. The board test checks that a cycle cancels an injected offset (0.3).

1.7 NQR

A nucleus with spin I ≥ 1 has an electric quadrupole moment; the electric field gradient of its crystal site splits its levels in zero field, so a powder resonates at fixed frequencies (e.g. ^{14}N between 0.5 and 5 MHz, ^{35}Cl near 30–40 MHz) with no magnet. The lines are narrow but weak and their frequency drifts with temperature (several kHz per kelvin for ^{35}Cl), so one searches: step TX1's frequency (the band of a hard pulse is ~1/t_p), run an SLSE train at each, and sum the echoes. In a powder the nutation is not a sine (the crystallites see different effective fields): calibrate the pulse by the largest echo, not by a null.

1.8 Pulsed ESR at an intermediate frequency

Electron spins resonate at GHz; a microwave bridge mixes them to and from an intermediate frequency (IF) that spin handles (up to about 50 MHz). In **I/Q mode** OUT1 + i OUT2 drive an IQ mixer whose LO is the bridge's source, so TX1's frequency is the offset from that LO (a negative frequency selects the other sideband); per-TX Q gain and phase and DC levels correct the mixer's image and LO leakage (2.3). The echo returns through the bridge's mixer to IN1 and IN2 (input `iq`). ESR lines are broad (MHz) against the pulse bandwidth and T2 is short, so the dead time hides the FID: one records **echoes**, and the **echo-detected field sweep** steps the magnet's field (`field_sweep`: AO0 drives the coil's supply) and records the echo at each field, mapping the line. The image rejection and LO suppression of a real mixer are unmeasured.

1.9 Dispersive qubit readout

A qubit coupled to a resonator shifts the resonator by ±χ with its state; a readout pulse at the resonator's frequency (TX2) returns with a state-dependent phase: two **blobs** in the I/Q plane, broadened by the amplifier's noise. A weighted integration (a matched filter: the difference of the two mean traces as weights) maximizes their separation; the window's state is Re(S e^{−iφ}) > threshold, with φ along the blobs' difference and the threshold at their midpoint (`experiments.decision`). The assignment fidelity F = 1 − [P(1|0) + P(0|1)]/2 is limited by the blobs' overlap (the separation SNR) and by T1 decay during the readout; the board measured F = 0.985–0.987 on the emulator (SNR 7.6–8.2, T1 20 µs, a 0.5 µs readout). **Active reset** reads the qubit out and, if the state is 1, applies a π pulse: its error is mostly decay between the readout's projection and the π pulse (the decision takes D + 28 clocks after the window: 2.2), about 0.03–0.04 of a half-excited qubit with T1 = 20 µs on the board (0.023 in the model). **Repeat-until-success** prepares until the readout says 1. Thermal population sets the floor of a passive reset; an active reset lowers it.

1.10 The emulator and its limits

The emulator is one Bloch vector updated every 128 ns (a slot) by an exact rotation with T1 and T2, driven by TX1 in TX1's frame (2.8). Its offsets stay within ±3.125 MHz of TX1 (the driver asks 3 MHz), its Rabi frequency below 1.95 MHz at full amplitude. In **ensemble** mode a detuning is drawn from the line at every SHOT: one scan is one isochromat (it does not dephase within the scan), the averaged scans show the line's T2* and refocus in echoes. In **qubit** mode a TX2 readout draws an outcome from (1 − z)/2, the vector collapses and decays with T1 during the readout, and the readout returns through a one-pole resonator at ±χ. TX1's NCO must run before the first pulse of a run (the templates' preludes start it; 2.8). Not modelled: probe ring-down, receiver recovery, amplifier compression, spin I > 1/2 dynamics, couplings, diffusion, field drift. Its outputs are exact against `emulator.py` and agree with the float Bloch equations (`bloch.py`) within 0.1 % for physical T1 and T2.

2. Definitions (the contract)

2.1 Time

- The **sample clock** runs at 125 MHz; sample n is at `t = 8 n` ns, counted from the core's reset (the core's counter `now`). All times are integers of samples unless stated (the trigger's 1 ns timestamp).
- **Connector time.** Every instruction of the pulse program has a start T in connector time: an RF pulse, a digital output and an acquisition window that start in the same instruction start at the same instant at the OUT connector, the E1 pin and the IN connector. Inside, the **player** holds the instruction of clock t ("player time"); the transmitters show its sample on the DAC port in clock `t + L_TX` (`L_TX` = 19: the **run-ahead**; the DAC interface adds 2 clocks, the DAC and the front end their analog delay). A pin's 2 ns word shows its source of player clock `t − 3 − D/4` (D its delay in 2 ns steps, 2.6); the driver sets D from `PIN_ALIGN` (66 steps, from the simulated latencies) and the calibration's `pins_delay_s` (`PIN_ALIGN` − round(pins_delay_s / 2 ns): 80 steps on the test board; measured with an oscilloscope within 3 ns at the connectors: D7 on 2026-11-07, D1 on 2026-11-08). The capture takes the ADC port's sample of clock `t + 1 + delay`; the driver sets the delay

to `L_TX − 1 + round(adc_delay / 8 ns)` (the calibration's converter loop delay: 18 clocks without it), so that its first sample is the IN connector at the instruction's start; the residual below one clock is reported.

- **Phase reference t0.** Every oscillator's phase is relative to a common t0 (synth's convention): `φ(n) = F (n − t0) mod 2^48` in player time, F the 48-bit frequency word (`f = F · 125 MHz / 2^48`, steps of 0.44 µHz). A **START** in clock w sets t0 = w; the first instruction starts in clock w + 24 (`L_W`). A WAIT can set t0 to its event (2.2): for a TRIG edge in slot j of word m, `t0 = m + j/8` and `φ = floor(F (8 (n − m) − j) / 8) mod 2^48`, so the RF is locked to an external event within 1 ns.
- **Latencies** (0.5; from the clocks and the board calibration, measured with an oscilloscope on 2026-11-08): a window's last sample e at the ADC port is the IN connector at connector time e − 1 − D (D the capture's delay: 36 clocks with the nominal 18-clock loop, 33 on the test board with its 120 ns). A branch on its state starts the chosen pulse at e + 27 at the earliest (2.2), D + 28 clocks later at OUT: 512 ns nominal, 488 ns on the test board. A pin's word of clock c is at the connectors at c − 3 − A/4, A the TTL pins' alignment in 2 ns steps (66 nominal), that is at c − 19.5 + pins_delay_s / 8 ns. A state pin (2.6) changes in its word of clock e + 24 + (its delay)/4, so at connector time e + 4.5 + pins_delay_s / 8 ns: D + 5.5 clocks + pins_delay_s after the last sample, 332 ns nominal, 279 ns on the test board (pins_delay_s −28.6 ns). Measured with DC steps (VERIFICATION, 2026-11-08): the chosen pulse came as programmed (800 ns after the window's input step, + 2.0 ns between the two outputs' cables), and the state pin 281 ns after the last sample as the scope shows it (277 ns referred to the connectors). The driver refuses a branch that would come earlier (2.2).

2.2 The pulse program (sequencer)

- **Program**: up to 1024 instructions of five 32-bit words (ABI 2):

Word	Bits
W0	duration in clocks (values below 2 count as 2)
W1	[2:0] op (0 CONT, 1 LOOP, 2 END, 3 WAIT, 4 BRANCH (0.5), 5 JUMP, 6 STOP), [4:3] loop level k, [14:5] target, [15] SHOT, [16] CAP, [19:17] window, [20] the window starts, [21] duration from a table, [23:22] its table, [31:24] TTL
W2, W3	TX1, TX2: [0] on, [1] keep (the pulse continues: the NCO, the chirp, the envelope), [4:2] frequency register, [5] + table, [7:6] its table, [10:8] phase register, [11] + table, [13:12] its table, [16:14] amplitude register, [17] amplitude from a table, [19:18] its table, [24:20] shape (0 rectangular), [27:25] chirp register (0 none), [31:28] for 0.3 .. 0.5
W4	[9:0] table offset, [10] loop count from a table, [13:11] count register (or table), [15:14] WAIT source (0 TRIG, 1 timer, 2 software), [16] t0 moves to the WAIT's event, [17] a slow output's update, [19:18] its output (AO0 … AO3), [22:20] its AO register, [23] its value from a table ([15:0]), [25:24] that table, [27:26] the receiver phase (90° steps); BRANCH: [30:29] its condition's source (0 a window's state, 1 COND A, 2 COND B, 3 a loop counter's parity), [31] the condition inverted

At power-up every instruction is a STOP.

- **Engine and player.** The engine resolves one instruction per four clocks (the instruction, registered; the tables' addresses; the tables and the resolution, registered; the push and the loops, with the next instruction's address) into a FIFO of 16; a START fetches instruction 0 in its own clock, so the first push comes 4 clocks after it; the player pops one per instruction with its exact duration. Instructions of 4 clocks or more are sustained indefinitely; shorter ones run back to back as long as the FIFO holds them: after a long instruction (a full FIFO) about 30 instructions of 2 clocks, after a START 8 (five entries are ready at the first instruction). If the FIFO is empty when the player needs the next instruction (an **underflow**), the run stops with error 1: timing never shifts silently. The duty guard stops a run with error 2 (TX1) or 3 (TX2) (2.3). The program counter wraps from 1023 to 0.
- **Loops**: 4 nested levels. `LOOP k` is the first instruction of its body and `END k` the last; both have durations, so a loop costs no time. The body runs n times (n from one of 8 count registers or a table; n = 0 skips the body, LOOP included); the counter i_k runs 0 … n − 1 and is 0 outside its loop (an END that does not jump back clears it; a LOOP entered anew sets it to 0, also for its own table reads).
- **Tables** T0–T3, 1024 × 32 bits each, each indexed by one loop counter (TBL_CTR). A field taken from a table reads `T_j[(i_k + offset) mod 1024]` (the instruction's offset). Phases from a table add to the phase register; amplitudes replace it; frequencies add `T · 2^16` to the frequency register (signed: steps of 29 mHz up to ±62.5 MHz); durations and loop counts are the table's word.
- **Operations**:
  - CONT: the instruction's duration, then the next.

- LOOP / END: as above. JUMP: to the target, after its duration.
- BRANCH (0.5): played as a CONT (its duration and outputs; it starts no window). The engine pushes it, then waits until its condition is valid, and continues at the target if the condition holds (W4 [31] inverts it), else at the next instruction. The condition (W4 [30:29]):
  - 0 the **state of window** w = W1[19:17]: valid once every instruction starting window w that the engine pushed before has ended at its receiver (a pending count per window: + 1 at such a push, − 1 when the window is not taken, at its start sample, or when a taken window ends, in clock e + 20, e its last sample). The state is that of the last window w with weights taken in this run (2.5); a window without weights leaves it; 0 after a START;
  - 1, 2 **COND A**, **COND B** (DIO1_P, DIO2_P): the level in the last nanosecond of the input word of clock b + L_TX, b the BRANCH's first player clock (its start at the connectors), read by the engine in clock b + 23;
  - 3 the **parity of loop counter** i_k, k = W1[4:3] (odd: true), valid at once.

  The engine decides in the first clock X after the push with the condition valid (a window's count and state of clock X: changes at e + 20 show in e + 21; a miss at a start sample s in s + 1) and pushes the next instruction in clock X + 4 at the earliest; the player needs it before its decision for the next start, so the instructions in the FIFO after the BRANCH's push must last until X + 5 (else an underflow: error 1). Hence a BRANCH on COND lasts at least 29 clocks; on a state, the instructions from the start of the window's instruction to the BRANCH's end last at least (K + 1)·R + D + 26 clocks (the window's first sample s = t + 1 + D, t its instruction's first player clock and D the capture's delay, 2.5; its last e = s + (K + 1)R − 2, 2.4; so X = e + 21, the next start X + 6). `sequence.py` asks 32 clocks on COND, the driver (K + 1)·R + D + 27 on a state.
- WAIT: its outputs hold until an event: a rising edge on TRIG (DIO0_P; one per 8 ns word, in slot j of word m, seen in clock m + 5), a tick of the timer (every `timer` clocks from the first instruction's clock), or a software trigger (clock m of the register write). The WAIT takes the first event of its source in a word m at or after its own first clock; the next instruction starts in clock m + 24. With "reref", t0 moves to the event (2.1). Other TRIG edges count as **missed** and other timer ticks as missed ticks (both while running).
- SHOT: a flag: counts shots (the emulator's per-shot draws and RESULT records from 0.3/0.4).
- CAP: a flag: starts the raw capture (3).
- STOP: the run ends when it starts; the outputs return to idle (the offsets, the TTL bits low, gates released after their lag, the slow outputs at their idle levels).

- A **run** starts at instruction 0 with all loop counters 0 (START); a START while running restarts; STOP ends it at once.

2.3 Transmitters

- **TX1, TX2**: a 48-bit NCO each with 8 frequency registers. Switching between registers is phase-coherent: each register's phase is kept as `F_r (n − t0)` all the time (a phase file: eight entries, each advanced by 8 F_r every eighth clock, read through two DSP48 at an instruction's start), so a return to a frequency restores its phase. A frequency from a table is phase-continuous (the accumulator keeps its phase and changes its step), not coherent with t0; `keep` continues the NCO unchanged. A transmitter that is off follows its instruction's frequency fields all the same (the receivers' LO from 0.3 use the same accumulators, so transmit and receive stay coherent). A START sets both accumulators to 0 at player clock w.
- **Chirps** (per TX, seven slope registers, 48-bit two's complement, in 2^−4 steps of the frequency word per clock: `rate = S · 125 MHz² / 2^52`, 3.5 mHz/s … 4.9·10^14 Hz/s). The NCO's step F carries four fraction bits: acc(t) = acc(t − 1) + floor(F(t − 1) / 16); F(t) = 16 F_r at a coherent start, 16 (F_r + T · 2^16) at a start with a table frequency, else F(t − 1) + S(t) (S of the instruction's chirp register, 0 for chirp 0 or no instruction), mod 2^52. A chirp thus starts from the instruction's frequency and continues with `keep` (the next instruction may change the slope or stop it: chirp 0 holds the frequency reached). The frequency registers' phases (the phase file) are not affected: a later coherent start returns to F_r (t − t0).
- **A pulse** (per instruction and TX): an amplitude A (one of 8 amplitude registers or a table, 16-bit), a phase p (one of 8 phase registers, plus a table: 16 bits, 2^−16 turn), an envelope (a_e, e_ph) and a chirp. Per player clock t:

- `A_I = floor((A · a_e + 2^15) / 2^16)`, `A_Q = floor((A_I · g_q + 2^15) / 2^16)` (g_q the TX's Q gain, 17-bit, 65536 = 1);
- `θ_I = (acc[47:24] + 2^8 ((p + e_ph) mod 2^16) + 2^22) mod 2^24`, `θ_Q = (acc[47:24] + 2^8 ((p + δ_q + e_ph) mod 2^16)) mod 2^24` (δ_q the TX's Q phase);
- `y_I = sat14(floor((sine(θ_I) · A_I + 2^19) / 2^20))`, `y_Q` likewise with θ_Q and A_Q (`common/python/rpfpga/sine.py`: 18-bit, ±131071), both 0 while the TX is off or blanked by the guard.

So y_I is a cosine of amplitude `A_I · 131071 / 2^20` codes (A = 65535 reaches 8192 and saturates to 8191) and y_Q a sine.

- **Shapes** (envelopes): one table of 4096 points shared by both TX, each point {phase [31:16] (2^−16 turn), amplitude [15:0] (2^−16)}, in even and odd banks so that both TX interpolate every clock; 31 descriptors (start, length L = 2 … 4096, step: points per clock, 4.16 bits). An instruction's shape s plays its descriptor: the position rel (16.16 points) is 0 at the instruction's start (without `keep`), else rel(t − 1) + step; i = rel >> 16 and the fraction f = rel mod 2^16 interpolate the points P = start + i and P + 1 (mod 4096): `a_e = a_0 + floor((a_1 − a_0) f / 2^16)`, `e_ph = (φ_0 + floor(d f / 2^16)) mod 2^16` with d = φ_1 − φ_0 as a signed 16-bit difference (the shorter way round). Once i + 1 ≥ L the shape has ended: a_e = 0 until the next start (L points span L − 1 intervals: (L − 1) / step clocks). Shape 0 is rectangular (a_e = 65536, e_ph = 0). The position holds while no instruction is current. The table is written while no run uses it (the driver stops a run first).
- **Output modes**: two real channels (OUT1 = TX1's I path, OUT2 = TX2's) or one I/Q pair (OUT1 + i OUT2 = TX1 + TX2: OUT1 = I_1 + I_2, OUT2 = Q_1 + Q_2, frequency-multiplexed: a TX at a negative frequency is the lower sideband of an IQ mixer). Each TX's Q path has its own gain and phase, so the correction matches each TX's frequency; each output a DC level (an IQ mixer's LO leakage). Per output n, in clock t + 18 and t + 19 (L_TX): `u_n = sat14(sum + offset_n)`, `code_n = min(max(u_n, lo_n), hi_n)`, with sum = y_I1 (n = 1) and y_I2 (n = 2) in real mode, y_I1 + y_I2 and y_Q1 + y_Q2 in I/Q mode; offset_n is the calibrated 0 V plus the DC level, [lo_n, hi_n] the output's limits (14-bit codes; power-up the full range).
- **Volts**: A is the amplitude at the connector into the load of the board calibration. The driver converts volts: A = V / (4 G · 131071/2^20 · response(f) · sinc(f)) for the output's calibration G (volts per 16-bit code), its `dac_response` and the DAC's zero-order hold at the TX's carrier f (frequency register 0, or `carrier`); in I/Q mode with OUT1's calibration, and the Q gain register holds q_gain × (OUT1's / OUT2's volts per unit), so that q_gain = 1 gives equal volts; the Q phase adds 360° f (delay_OUT2 − delay_OUT1) for the outputs' calibrated skew. The limits are volts at the connector, rounded inwards to codes; they and the duty guard of an output's TX are **lockable** (PROTECT, awg's protected bounds): once locked, the FPGA ignores writes to them until it is reloaded.
- **Duty guard** (per TX, from its `on` of player clock t as the instruction says): the pulse length c(t) = on ? c(t − 1) + 1 : 0 trips when c > L_max (if enabled); the bucket b(t) = min(max(0, b(t − 1) + 65536 on − r), 2^48 − 1) trips when b > B_max (if enabled; r = the mean duty allowed × 65536, B_max = the burst allowed × 125 MHz × (65536 − r): a continuous pulse from an empty bucket trips after the burst; the driver rounds L_max, r and B_max down, so the guard never allows more than asked). The bucket keeps draining between runs; a change of the TX's duty settings empties it. A trip of player clock t blanks both TX from player clock t (their outputs 0), stops the run at the end of clock t + 3 (error 2 for TX1, 3 for TX2; G_STATUS keeps which checks tripped). The run's stop ends the TXs' `on` in clock t + 4, so the gates (2.6) end as after any pulse: each closes its lag + 4 clocks after the cut RF's end at the connectors, and a receiver's protection stays on through the probe's ring-down (before 2026-11-08 the gate pins were forced closed from clock t + 5, 116 ns − pins_delay_s ahead of the RF's end: 145 ns on the test board). The blanking lasts until the next START.

2.4 Receivers

- **RX1, RX2.** The input x(c) of ADC clock c (the ADC port's sample in that clock): IN1, IN2 or IN1 + i IN2 (RX_CFG), in 16-bit ADC codes minus the calibrated offset of each input (IN_OFF): `x = code − offset` (17 bits).
- **LO**: the phase accumulator of the window's TX (TX1 or TX2), without the pulse's phase, envelope or Q path: θ(c) = acc[47:24] of player clock `c − 1 − D`, D the capture's delay (C_CFG [23:16]; 6 … 63), so that the LO and the ADC's sample refer to one connector time (the window's first sample is the capture's: 2.1).
- **Mixer**: g_c = sine(θ + 2^22), g_s = sine(θ) (rp_sine), `p(a, g) = floor((a g + 2^13) / 2^14)`; `z_I = p(x_r, g_c) + p(x_i, g_s)`, `z_Q = p(x_i, g_c) − p(x_r, g_s)` (x_i = 0 for a real input): `z ≈ 8 x e^{−iθ}`. **Raw mode** skips the mixer: `z = 8 x`. |z_I|, |z_Q| < 2^19.
- **CIC** of order 2 with decimation R = 1 … 16384, per window from its first sample s (the integrators are 0 before s): point m (0 … K − 1) is `Z_m = Σ_{j=0}^{2R−2} h_j z(s + mR + 2R − 2 − j)` with the triangle

h = 1, 2, …, R, …, 2, 1 (its sum R^2): the samples s + mR … s + mR + 2R − 2, its time label (the triangle's centre) s + mR + R − 1, the dwell 8 R ns. In the FPGA, Z_m = I2(n) − 2 I2(n − R) + I2(n − 2R) at n = s + mR + 2R − 2 (I2 the double sum from s), scaled to 20 bits: `Zs_m = floor(Z_m / 4^r)`, r = ⌈log2 R⌉. The host applies the gain 4^r / R^2 (and adds 1/2 per point and scan for the floor, if it matters). Spectra divide by the filter's response $(\sin(\pi f R T) / (R \sin(\pi f T)))^2$ (T = 8 ns).

- **Receiver phase** k = (the instruction's W4 [27:26] + the window's table word [13:12]) mod 4: `Zr = Zs · (−i)^k`, exact (k = 1: (Q, −I); 2: (−I, −Q); 3: (−Q, I)): the phase cycle's receiver phase.
- **Volts**: a tone A cos(2π f t + φ) at the connector (the LO at f) gives `Zs ≈ 4 K A e^{iφ} R² / 4^r` (K the input's codes per volt): the host reads A e^{iφ} = 4^r Zs / (4 K R^2); in raw mode x = 4^r Zs / (8 K R^2) volts.
- **One window at a time** per RX: from its first sample s to its last point's sample e = s + (K + 1) R − 2, and 3 samples more (the averager's read-modify-write of its last points): a window that would start before e + 4 on that RX is not taken (WIN_MISSED, an ERROR record).
- **Overload**: a window is flagged if a code of its input (IN1 or IN2, as used) reached the converter's limits (−32768 or ≥ 32764) in its samples s … its last.

2.5 Windows, averaging, weighted integration

- **Windows**: 8 descriptors (WIN): the RX, the LO (TX1 or TX2), raw, average, weigh, TRACE, the length K (1 … 2048 points), R (1 … 16384), the region's base (0 … 2047), a table j for the region (its word's [10:0] adds to the base, [13:12] to the receiver phase; read like any table field of the instruction: T_j[(i_k + offset) mod 1024]), the weights' base (0 … 1023) and a threshold. An instruction with W1 [20] starts window W1 [19:17] at its connector time (sample s = t + 1 + D for an instruction starting in player clock t).
- **Averaging**: one bank per RX of two halves × 2048 complex points (I and Q 35-bit signed, saturating; a saturation is flagged). Point m of a window adds Zr_m into the point `(region + m) mod 2048` of the half that was active when the window started. A half becomes active only after it was cleared: a released half clears over 2048 clocks without a point to write (16.4 µs when idle; a clock with a point waits) and is free from then (0.4; 0.3 used a 1-bit epoch per half instead, which let points written two activations earlier and not since reappear).
- **Swaps** (both banks together): at the sample of an instruction with SHOT (t + 1 + D), the halves swap if the other half is free and either the active half holds N scans (AVG_N, if N > 0) or the host requested a swap (AVG_CTRL; without a run the request swaps at once, or as soon as the other half is free). The new active half's scans and saturation clear; then the SHOT counts a scan of the active half. The old half closes; once no window uses it, it is **frozen**: a BLOCK record, readable by the host (other halves read 0) until the host releases it (AVG_CTRL), then clears. A swap that finds the other half not free waits: the active half keeps accumulating.
- **Weighted integration** (per window with "weigh"; one complex MAC per RX at the full rate): `S = Σ_{m<K} w_{(base + m) mod 1024} Zr_m` with 1024 complex weights per RX (18 + 18 bits, signed), S_I and S_Q 48-bit. The window's state is `floor(S_I / 2^16) > threshold` (32-bit signed). The host pre-rotates and normalises the weights, so that a steady signal of amplitude A reads A (volts). A RESULT record at the window's end. The core keeps each window's last state (0.5: for BRANCH and the state pins), from clock e + 21 (e the window's last sample), also when the records are off or full; 0 after a START.

2.6 Digital pins

- **Inputs** (sampled every 1 ns, `common/rtl/optional/rp_pins1ns.v`): TRIG (DIO0_P; rising edges), COND A and COND B (DIO1_P, DIO2_P; levels, for BRANCH at 0.5). Pulled down: open reads low.
- **Outputs** D1–D8 on DIO0_N–DIO7_N (2 ns bits, `rp_oserdes2ns`; high impedance after loading, after SAFE and until enabled; a weak pull-down holds a released pin low). Each has a source: a fixed level or 2 ns pattern; TTL bit b of the instruction; the TX1, TX2 or either **gate** (high while the TX is on and for `lead + lag` clocks after: a stream `lead` clocks early, from which the pin's delay subtracts 4 lead steps, so the gate opens `lead` before the pulse and closes `lag` after it, both whole clocks (the driver rounds them up: minimums); the lead is limited by the pin's delay, about 130 ns: longer leads will be placed by the compiler; measured on D1 with lead 96 ns and lag 200 ns: within 2.5 ns at the connectors, 2026-11-08); **RX protect** is the gate of either TX with its own lag (a dead time after each pulse); the capture's window (its K clocks from the instruction's start); SHOT (the instruction's duration); an instruction current (a run); (0.5) the **state** of window b (the TTL-bit field), as the core holds it (2.5): its own timing, not the player's (the driver sets such a pin's delay without the alignment to the connectors). A pin inverts on request and delays its source by D 2 ns steps (0 … 1023): its word in clock t holds, in slot q, the source of player clock `t − 3 − floor(D/4) − (q < D mod 4)`. A trip of the duty guard ends the pulse: its gates close their lag (+ 4 clocks) after the cut RF (2.3).

2.7 Slow outputs

AO0–AO3 on E2: 0 … 1.8 V through the board's RC filters (PWM at 125 MHz / 256 = 488.28 kHz with a second-order ΔΣ, `common/rtl/optional/rp_ao.v`, bit-exact reference `rpfpga.ao`; volts through `CAL SLOW`). Each output has an idle level (outside runs and at a run's start) and the run's value: an instruction's update sets it (from one of 8 AO registers or a table's [15:0]) from the clock after the instruction's start. Output k takes its value in clock 256 j + k; per period, v = 0 is off (duty 0, the errors cleared), else `y = clamp(v, 256, 65280) + 2 e_1 − e_2`, `d = (y + 128) >> 8`, `e = y − 256 d`, and the pin is high in clock c when ((c − 8) mod 256) < d (the mean duty is v / 65536). Uses: field sweeps, magnet current, pre-polarisation, attenuators (a level settles through the RC filter, a few microseconds).

2.8 The emulator

A digital two-level system answering the transmitters (EMU_CTRL): one qubit read out through TX2, or a spin ensemble whose transverse magnetization is the signal. One Bloch vector serves both; an ensemble's spread comes from the detuning drawn at each SHOT, so averaged scans decay (free induction, echoes) while a single scan does not (the plan's cut of eight isochromats to one, taken at 0.4 for the slices). `control/spin/emulator.py` is the bit-exact reference; `control/spin/bloch.py` integrates the floating-point Bloch equations it is checked against.

- **Physics.** A Bloch vector M = (x, y, z) (|M| ≤ 1; z = +1 the ground state), in the rotating frame F of TX1's integrated frequency: `dM/dt = Ω × M − (x/T2, y/T2, (z − z_eq)/T1)`, `Ω = (Ω_R cos ψ, Ω_R sin ψ, Δ)`, Ω_R ∝ TX1's amplitude after its envelope (k_R), ψ its phase, `Δ = 2π (f_q − f_TX1) + δ_shot + k_AO (AO0 − ½)`. A drive along x turns M from +z towards −y. |M| stays ≤ 1 only for T2 ≤ 2 T1, as for every spin: the driver refuses a longer T2, and `t2` null with a T1 means 2 T1 (relaxation alone; null with no T1: no relaxation). v's components are clamped to ±1 (2.8 below), which is exact inside the unit ball (1.0: a test with T2 > 2 T1 had let M grow to 1.32 and a pulse then clamped a component).
- **Frame.** F(t + 1) = F(t) + f(t), f(t) TX1's 48-bit step of player clock t (2.3, with a chirp's); TX1's phase is acc(t) = F(t) + J(t), J changing only at TX1's coherent starts (by the start's phase minus the continued one). The drive's phase in F is `ψ(t) = (phase + envelope phase)·2^8 + J(t)[47:24]` (24-bit turns); the qubit's frequency f_q (EMU_FQ, 48-bit like a frequency register) precesses in F at `f_q − f(t)`. A START sets TX1's step to 0 until its first coherent start (2.3), so before it the spin precesses in F at f_q itself (w_z clamped to ±0.4 turn per update); a pulse in a slot that also holds such clocks turns about a tilted axis. Start TX1 (on or off, without keep) at least one slot (128 ns) before the first pulse: the templates' preludes do (1.0; before, a run's first 90° pulse came out near 76°, which the 0.4 checks' 3 % amplitude tolerance hid).
- **Drive** of player clock t: `(dx, dy) = round(a C(p) / 2^16), round(a S(p) / 2^16)` with a TX1's amplitude (0 … 65535; 0 when TX1 is off or blanked), p = ψ's top 12 bits and C, S the cosine and sine of `2π (p + ½) / 4096` from rp_sine_rom's table (the drive's phase is quantized to 1/4096 turn, offset by half a step).
- **Slots** of 16 clocks (t ≡ 15 mod 16 the last, t the core's sample clock: the slots follow the core's clock, not the run's START): the vector is updated at the end of every slot (every 128 ns) with that slot's sums, in both modes. With a seed the draws repeat for runs started at the same clock modulo 16; the model reproduces a board run from its START clock (S_START; `experiments.simulate(..., start=)`).
- **Rotation vector** w (signed 18-bit, units of 2^−18 turn) of an update:
  - `w_x, w_y = ⌊(k_R D + r) / 2^22⌋`, D the drive's sum (Σ dx, Σ dy), r the remainder of the previous update (22 bits, carried; kept in a qubit's readout slots). k_R < 2^17 and |D| ≤ 16 · 131070 bound |(w_x, w_y)| below 2^16 (a quarter turn), so no clamp;
  - `w_z = clamp(⌊(Z + r_z) / 2^30⌋ + δ + round(k_AO (AO0 − 32768) / 2^16), ±104857)` (±0.4 turn), Z the slot's sum of `f_q − f(t)` (48-bit, signed), r_z its remainder (30 bits, carried); δ the shot's detuning; AO0 the slot's last value.
- **Update** `v ← v + N v_t + d_z ẑ`, v's components 32-bit (2^30 = 1), clamped to ±1, `v_t = clamp(round(v / 2^13))` (18-bit). N = E R − I, R the exact rotation by w (Rodrigues), E = diag(1 − g2, 1 − g2, 1 − g1) the relaxation over the update (g1, g2 at 2^−31, EMU_G1/G2; d_z = (1 − e1) z_eq at 2^−30, EMU_DZ), in units of 2^−(23 + e) (signed 25-bit):
  - `θ² = w·w` (< 2^32 + 104857² < 912·2^24 by the bounds above: θ < 0.472 turn, so |M|, |N| < 1.985 · 2^23 and nothing in N needs a clamp); a, b interpolated linearly in θ² from the table ROD (1024 entries at θ_i² = i 2^24: `a_i = round(sinθ/θ · 2π 2^14)`, `b_i = round((1 − cos θ)/θ² · (2π)² 2^12)`, θ in radians), with i = θ²[33:24] and the fraction θ²[23:7] (17 bits);
  - the block exponent e: the largest of 0 … 7 with θ² < (33377 · 2^−e)² and max(g1, g2) < 2^(30 − e) (|N| < 2^(1 − e));
  - `aw_k = round(a w_k / 2^(9 − e))`, `bw_k = round(b w_k / 2^9)`, `q_jk = round(bw_j w_k / 2^(16 − e))` for jk = xx, yy, zz, xy, xz, yz; M = R − I: `M_xy = −aw_z + q_xy`, `M_yx = aw_z + q_xy`,

M_xz = aw_y + q_xz, M_zx = −aw_y + q_xz, M_yz = −aw_x + q_yz, M_zy = aw_x + q_yz, M_xx = −(q_yy + q_zz), M_yy = −(q_xx + q_zz), M_zz = −(q_xx + q_yy);
  - N_ij = M_ij − round(g_i^(17) M_ij / 2^17) − δ_ij round(g_i / 2^(8 − e)), g^(17) = g >> 14;
  - v_k ← v_k + round(Σ_j N_kj v_t,j / 2^(10 + e)) (+ d_z for z), clamped. Rounding is half up ((x + 2^(n−1)) >> n). Accuracy against bloch.py: 10^−4 … 10^−3 of |M| over runs to 50 ms (the norm within 10^−4); a strong drive with a short T2 (1 MHz Rabi, T2 = 20 µs) 3·10^−3 (relaxation applied after the rotation within an update).
- **Limits.** Ω_R/2π ≤ 1.95 MHz at TX1's full amplitude (k_R < 2^17); |Δ|/2π ≤ 3.125 MHz (w_z's clamp: a larger detuning precesses at 0.4 turn per update, so a spin far from TX1's frequency is not emulated, and the driver refuses a TX1 frequency more than 3 MHz from the emulator's). Rotations near the limits are coarse (θ ≈ 3 rad per 128 ns); for accurate dynamics keep |Δ|, Ω_R/2π up to about 1 MHz. A pulse that starts or ends inside a slot is applied with the slot's mean drive and detuning (an error of order Ω_R Δ T^2 per edge).
- **Draws.** xoroshiro128+ (rp_xoro's) gives a 64-bit word r(t) per clock; at a START it is reseeded from EMU_SEED if EMU_CTRL [3] is set (r(w + 1) its first word), else it runs freely. Fields: noise [9:0] and [19:10], outcome/decay [43:20], detuning [53:44].
  - A SHOT at player clock t: δ = DET[r(t)[53:44]] (a host table of 1024 quantiles, units of w per update) from the next slot on.
  - Qubit readout: TX2 on (not blanked) is a readout. At its start (TX2 starting without keep, or switching on) s = 1 if r(t)[43:20] < (1 − z)/2 · 2^24, z the state after the last slot ended before t (a drive in the readout's first slot, before its start, is not seen: leave a slot, 128 ns, between the last drive and a readout). A slot in which TX2 reads is not rotated: v = (0, 0, 1 − 2s) at its end; from the second such slot on, s = 1 decays to 0 when r(t_end)[43:20] < P1 (EMU_P1 = (1 − e^(−16 T/T1)) 2^24), t_end the slot's last clock.
- **Output** e(t) (complex, ADC codes, signed 17-bit, clamped) of player clock t:
  - qubit: a one-pole readout resonator ρ ← ρ + ⌊(target − ρ) / 2^κ⌋ (8 fraction bits; EMU_KAPPA), target = ⌊g (C, S)(p_2) / 2^9⌋ while TX2 reads (else 0), g = round(a_2 G / 2^16) (TX2's amplitude, EMU_GAIN), p_2 the top 12 bits of (phase + envelope phase)·2^8 ∓ χ·2^8 (TX2's, χ for s = 0/1, EMU_CHI); e = ⌊ρ / 2^8⌋. Its LO is 0 (TX2's frame);
  - ensemble: after each update e = round(G ⌊v / 2^13⌋ / 2^17) (x and y; G codes at |M| = 1), held during the next slot. Its LO is J[47:24] (TX1's frame);
  - plus Gaussian noise (NOISE[r[9:0]], NOISE[r[19:10]]) (a host table of 1024 quantiles in codes); off: 0, LO 0.
- **Injection.** A receiver whose input is the emulator (RX_CFG 3) takes x = e(c − 1 − D) and the LO of player clock c − 1 − D at ADC clock c (its window's LO bit is ignored; no overload). The emulator's latency fixes D = D_EMU (the capture's delay; the driver sets it while the emulator is on): windows on IN1/IN2 then start D_EMU − D_cal clocks later than calibrated.
- **Reset.** With EMU_CTRL [2] set, every START puts the vector at (0, 0, z0) (EMU_Z0) and clears the sums, remainders, δ, the readout and the resonator; otherwise the state runs on between runs (relaxing, precessing in TX1's last frame).

2.9 Records

32-bit words in a ring in DDR (common/rtl/optional/rp_ring_writer.v, rpfpga.ringworker): each record is a header (bits 31..27 the type, 26..20 the number of words that follow, 19..0 type-specific) and its words. Times are the low 32 bits of the sample counter (8 ns; a TIME record every 2^31 clocks gives the upper bits). Records are made while REC_CTRL [0] is set; a record that finds its source's last one not yet written is lost (REC_DROPS, an ERROR record).

Type	Header [19:0]	Words
1 RESULT	[2:0] window, [3] RX, [4] state, [5] overload (saturation: the BLOCK records)	shot (the SHOTs counted at the window's first sample), time (its last sample), S_I [31:0], S_Q [31:0], {S_Q [47:32], S_I [47:32]}
2 BLOCK	[0] the half frozen, [1] saturation	scans, time
3 ERROR	[3:0] 1 underflow, 2 TX1's guard, 3 TX2's guard (with the run's end), 4 records lost, 5 a window not taken	time, a count (lost records, windows not taken; 0)
4 TRACE	(reserved: not built, the plan's first cut; [2:0] window, [3] RX, [15:4] the point m)	(Zr_I, Zr_Q)

Type	Header [19:0]	Words
5 RUN	[0] 1 start, 0 end, [7:4] the error	time [31:0], time [63:32], shots
6 TIME	0	time [63:32], time [31:0]

Type 31 is reserved: 0xF8000000 is the writer's padding (NOP). TRACE (each scan's points) is not built: the plan's first cut, taken at 0.3 to leave the slices for the emulator and the decisions; it may return at 1.0 if they leave room.

Losses. Each source holds one record. A source that fires again while its record waits replaces it (the waiting one is lost); while its record is being copied, the new one is lost; a receiver's result that arrives while its last one is being copied is lost. Every loss counts in REC_DROPS (several in one clock each count), and an ERROR record 4 follows with REC_DROPS's total at its time; a loss in the clock that record takes the total is reported by the next one. When several sources wait, they leave in the order RUN, ERROR, BLOCK, TIME, RESULT of RX1, of RX2; nothing else about the order across sources is defined.

3. Version 0.1

- **Program** (2.2): engine, FIFO and player; loops, tables, WAIT on TRIG (1 ns, t0 to the edge), the timer or software; SHOT and CAP flags; JUMP; STOP; underflow detection. BRANCH is a CONT until 0.5.
- **Transmitters** (2.3) in real mode: OUT1 = TX1, OUT2 = TX2, rectangular pulses, eight frequency, phase and amplitude registers each, tables; the offsets set the calibrated 0 V.
- **Capture**: an instruction with CAP takes K = 1 ... 4096 samples of both ADC ports (16-bit codes, {IN2, IN1} per word) from clock `t + 1 + delay` (2.1), unless a capture is still taking samples; the host reads them at 0x28000 (volts through the calibration). Its window is a pin source. It stays beside the receivers (0.3), which the board test compares with it.
- **Pins** (2.6): sources, inversion, delays, gates with lead and lag; TRIG, COND A, COND B sampled every 1 ns (levels and sticky edges readable).
- **Ring**: the record ring's writer with test words (records from 0.3).
- Resources and timing of the build: VERIFICATION.md.

4. Version 0.2

- **Transmitters** (2.3): chirps (seven slope registers per TX), polar envelope shapes (the shared 4096-point table, 31 descriptors, `keep` continuing a pulse), the I and Q paths (a second sine per TX) with the Q gain and phase.
- **Outputs**: real (as 0.1) or an I/Q pair (OUT_MODE); DC levels; clamping limits; PROTECT.
- **Duty guard** (2.3): pulse length and bucket per TX; blanking, the run's stop (errors 2, 3), the gates closing their lag after the cut RF; G_STATUS and the buckets' levels.
- **Slow outputs** (2.7): AO0–AO3 on E2, idle levels and instruction updates.
- **Latency**: unchanged (L_TX = 19): the amplitude products lost their M register to make room for the outputs' sums and limits. A coherent start's phase now passes through a three-stage delay line: 0.1's hold register lost the phase of a coherent start followed by another within 2 clocks (found by 0.2's scenario C, VERIFICATION.md).

5. Version 0.3

- **Receivers** (2.4): RX1 and RX2 on IN1, IN2 or IN1 + i IN2 minus the calibrated offsets, the complex mixer on TX1's or TX2's oscillator (per window), the order-2 CIC (R = 1 ... 16384), the receiver phase, raw mode, the overload flag; one window at a time per RX and 4 samples between windows.
- **Windows and averaging** (2.5): 8 descriptors; two banks of two halves × 2048 points with epochs, swaps at SHOTs (N scans or a request), freezes with BLOCK records, the host's reads and release. The service's worker reads each frozen half into 64-bit host sums every 0.1 s (`collect`); `window_average` gives a window's points in volts.
- **Weighted integration** (2.5): 1024 complex weights per RX, the MAC at the full rate, the state; a RESULT record at each window's end, 20 clocks after its last sample reaches the receiver.
- **Records** (2.9): RUN, ERROR, BLOCK, TIME and RESULT through the ring, `RECORDS STREAM` on the service; TRACE is not built (the plan's first cut).
- **Registers**: the indexed register files (FREQ, CHIRP, PHASE, AMP, COUNT, AO_REG) are LUT RAM (no change to the map); the transmitters' latency is unchanged (L_TX = 19).

6. Version 0.4

- **The emulator** (2.8) in `sp_emu`: player clock t's TX values reach it 8 clocks later (TX1's amplitude, phase word, step and J; TX2's amplitude, phase and start; SHOT, AO0), its output for t leaves 82 clocks after that, so a receiver reading it uses D = D_EMU = 89. The drive (one rp_sine_rom lookup, two DSP48) and the frame's detuning are summed per slot at the full rate; the readout path (TX2 delayed by 73 clocks, by when the slot's outcome is known; the ±χ phase, one rp_sine_rom lookup, the resonator) and the output (noise, the LO) also run at the full rate.
- **The slot processor** updates the vector once per slot over a fixed schedule of 62 clocks: four updates are in flight, each step owning its cycle (a one-hot phase of the slot). Two DSP48 (A, B) form w, θ^2, the table's interpolation, a w, b w and the q terms; a third (C) the relaxation's g M and the ensemble's output; a fourth (L) the loop v + N v_t row by row. The table ROD is one block RAM (1024 × 34 bits, two ports for entries i and i + 1). Each component of v is written back in its own clock (x, y, z three clocks apart); the qubit's outcome is drawn from the z of the previous update (written 2 clocks before) and the readout path reads it per slot from a 4-entry ring. A START with reset advances an epoch: the updates in flight from before it write nothing.
- **Driver**: `emulator` (mode, frequency, rabi, t1, t2, excited, detuning (shape, width), ao, signal, chi, ringdown, noise, reset, seed) compiles to the EMU registers and the NOISE and DET tables (written when they change); the receivers' input `emulator`; `emulator_state()`; `model_settings()` (a configuration as a model run).
- **The averager's halves are cleared** (2.5): a released half clears over 2048 clocks before a swap can make it active. 0.3 marked a half's points with a 1-bit epoch instead, so a point written two activations earlier and not since counted again (found on the board at 0.4, when the experiments' regions moved between runs; `tb_avg` now reuses a half a third time).
- **Experiments and analysis** (`experiments.py`, `analysis.py`): templates as configurations (a SHOT 256 ns before each acquisition, regions per array point, phase cycles with the receiver's phase, qubit readouts 256 ns after the drive with two reference regions), `acquire` on the board and `simulate` in the model; spectra with the CIC's droop, populations, least-squares fits of T1, T2, T2*, Rabi and Ramsey, single-shot fidelity.

7. Version 0.5

- **BRANCH** (2.2): op 4, played as a CONT; its condition in W4 [31:29] (a window's state, COND A, COND B, a loop counter's parity; inverted). `sp_seq`'s engine waits in a new state (BW) after pushing it: per window a pending count (+ 1 at a push starting the window, − 1 when the start is missed or the window ends: the receivers' d events, `sp_rx`), COND armed by the FIFO entry's flag at the branch's pop and read 23 clocks later (`L_COND`), the parity at once. The least durations (29 clocks on COND; (K + 1)·R + D + 26 on a state) were found by sweeping the model: the first derivation (K·R + D + 27) held only for R = 1. The bench pins them both ways (tb_seq I at them, J and K one clock short); the driver refuses shorter ones (the COND floor in `sequence.py`, 32 clocks; the state rule with one clock to spare).
- **Window states** (2.5): `sp_core` keeps the last state of each window with weights (from its end's d event, clock e + 21; 0 at a START); a window without weights leaves it.
- **State outputs** (2.6): PIN_CFG [17] is the source's bit 3; source 8 shows window [5:3]'s state (no alignment shift: the user's delay only). ABI 6.
- **The run epoch**: a bit carried with each window start through the delay line to the receivers; a START flips it, so starts still in flight from the previous run are not taken (checked on the board by restarts: no bench scenario restarts with starts in flight).
- **Templates** (`experiments.py`): `decision(ground, excited)` (the decision window's phase and midpoint threshold from the two references), `active_reset` and `repeat_until_success` (window 1 decides with 16 weighed points, window 0 reads the result; references as before).

8. Versions 1.0 and 1.1

- **1.1:** the averager (2.5) is the common `rp_avg` (`../../common/rtl/optional/rp_avg.v`, moved from spin's `sp_avg` when boxcar needed it) with spin's parameters (AW 11, IN_W 20, SUM_W 35, BWAIT 31, per-half scans, direct host reads), and its model the common `rpfpga.avg.Averager`: the same logic and the same register layout (ABI 6); the benches, the model's vectors and the board checks are unchanged. Nothing else changed.
- **1.0: no RTL change** from 0.5 but the sysid version (ABI 6).
- **Templates**: `slse` (NQR spin-locked echo trains: CPMG with refocusing pulses of a chosen flip, 90° by default) and `field_sweep` (ESR's echo-detected field sweep: a slow output's level from a table per point, in volts, coded through the board calibration when applied: `config_for`).
- **The templates' preludes start TX1's NCO** (a coherent start without output, 256 ns before each acquisition's

first pulse): after a START TX1's step is 0 until its first start, so the emulator's frame stood still and the run's first pulse shared a slot with a large detuning (2.8; a 90° pulse came out near 76°, hidden in 0.4's 3 % amplitude tolerance, found while checking the new templates against the float Bloch equations).

- **The emulator's T2 ≤ 2 T1** (2.8): the driver refuses a longer T2; `t2` null with a T1 is 2 T1. Tests and the board test had used T2 > 2 T1, where |M| grows beyond 1 and a pulse clamps a component; with physical values the emulator agrees with the float Bloch equations within 0.1 %.
- **Presets** (11: the emulator's experiments, a loopback, an NMR probe, an IQ bridge), the **Jupyter panel** (`gui.py`), `RECIPES.md`, the physics of section 1; the tables of an APPLY read back without trailing zeros (as the registers hold them).

spin commands (TCP)

The board's service (`rp-overlay.service`, port 5025) takes the common commands of every design (`*IDN?`, `STATUS?`, `CONFIG?`, `APPLY`, `PRESET`, `SAFE`, `LOAD`, `CAL ...`; `../../common/README.md`) and, while spin runs, these (version 1.1):

Command	Reply
`RUN`	`OK`: the program starts at instruction 0 (a running one restarts); t0 is this moment, the first instruction 192 ns later
`STOP`	`OK`: the run ends now
`TRIGGER`	`OK`: a software trigger (a WAIT on software continues 192 ns later)
`STATE?`	`OK {json}`: `running`, `waiting`, `instruction` (current, or null), `fifo`, `error` (null or a text), `shots`, `missed_triggers`, `missed_timer_ticks`, `start_s`, `trigger_s` (the last TRIG edge taken, 1 ns steps), `trigger_after_start_s` (that edge from the run's start, exact in ns; negative if this run took none), `averager` (`active`, `frozen`, `swap_pending`, `closing`, `saturation`, `blocks`), `windows_missed`, `records_lost`, `guard` (`tx1_pulse`, `tx1_duty`, `tx2_pulse`, `tx2_duty`: the checks that tripped since the start; `level_s`: each bucket's level in seconds of full duty; it counts also while the duty check is off, draining at the set rate, 0 by default)
`CAPTURE? [N]`	`OK {json}`: the last capture's (or its first N) samples in volts, `in1`, `in2`, `dt` (8 ns), `taken`, `busy`, `delay_clocks`, `residual_s`
`PINS? [CLEAR]`	`OK {json}`: for TRIG, COND_A, COND_B: `{"pin", "level", "seen"}` (the latest 1 ns sample; a rising edge since the last clear); CLEAR clears the edges
`HARDWARE?`	`OK {json}`: `abi`, `locked` (the samplers' PLL), `writer_running`, `writer_tripped`, `written`, `bursts`, `stalls`, `protected` (the outputs locked), `ao_volts` (the slow outputs' present levels)
`AVERAGE? <w> [<o>]`	`OK {json}`: window w's average from the host sums (region offset o): `scans`, `blocks`, `dt`, `t0`, `volts` ([re, im] per point: a tone's amplitude and phase at the LO; raw: the input's volts)
`AVERAGE CLEAR`	`OK`: forget the host sums
`SWAP`	`OK`: a swap of the averager's halves (at the next SHOT; at once without a run)
`COLLECT`	`OK {"collected": bool}`: read a frozen half into the host sums now (the worker does it every 0.1 s)
`RESULTS? [N]`	`OK [json]`: the last N (100) RESULT records: `window`, `rx`, `state`, `overload`, `shot`, `time`, `s_i`, `s_q`, `volts`
`WORKER?`	`OK {json}`: the record worker's counters
`EMULATOR?`	`OK {json}`: the emulator's Bloch vector `m` = [x, y, z] (in TX1's frame; z = +1 the ground state) and `outcome` (the qubit's last readout: 0 ground, 1 excited)
`RECORDS STREAM`	a stream: `OK {header}` and frames of records (hub.py; magic SPR1)
`PROTECT <n>`	`OK {json}` (the configuration): lock OUT n's limits and TX n's duty guard until the FPGA is reloaded

Configuration

`APPLY {json}` / `CONFIG?` (lists and `shapes` are replaced as a whole; other dicts merge):

```
{"mode": "real" | "iq",                       (OUT1 = TX1, OUT2 = TX2; or OUT1 + i OUT2 = TX1 + TX2)
 "tx": {"1": {"frequencies": [8 x Hz], "phases": [8 x degrees], "amplitudes": [8 x volts],
              "carrier": null | Hz,             (the frequency the amplitudes are corrected at;
                                                 null: frequency register 0)
              "chirps": [7 x Hz/s],             (chirp registers 1 .. 7)
```

```
            "q_gain": 0 .. 1.99, "q_phase": degrees},   (the Q path, I/Q mode: Q = g A sin(theta +
             ↪ delta))
       "2": {...}},
 "analog": {"1" | "2": {"dc": volts,            (OUT n's DC level, e.g. an IQ mixer's LO leakage)
                        "limits": {"low": null | volts, "high": null | volts, "protect": bool}}},
 "guard": {"1" | "2": {"max_pulse": null | seconds, "max_duty": null | 0 .. 1,
                       "burst": seconds}},       (the TX's duty guard; burst: a full pulse from an
                        ↪ empty bucket)
 "shapes": {"1" .. "31": {"amplitude": [2 .. 4096 x 0 .. 1], "phase": null | [degrees],
                          "duration": seconds}},  (the time the points span; together <= 4096 points)
 "ao": {"values": [4 x volts], "registers": [8 x volts]},   (AO0..AO3 idle levels; AO registers)
 "rx": {"1" | "2": {"input": "in1" | "in2" | "iq" | "emulator"}},
 "windows": {"0" .. "7": {"rx": 1 | 2, "lo": 1 | 2 (TX), "raw": bool, "average": bool, "weigh": bool,
                          "points": 1 .. 2048, "decimation": 1 .. 16384, "region": 0 .. 2047,
                          "region_table": null | 0 .. 3, "weights": null | [points x (number | [re,
                           ↪ im])],
                          "phase": degrees, "threshold": volts}},
 "average": {"scans": N},                       (swap after N scans; 0: on request)
 "records": bool,                               (the record ring: RESULT, BLOCK, ERROR, RUN, TIME)
 "program": [instruction, ...],                 (below; a final STOP is added if missing)
 "tables": {"counters": [4 x loop level 0..3], "data": [[words], [words], [words], [words]]},
 "counts": [8 x integers],                      (loop count registers)
 "timer": seconds,                              (the WAIT timer's period, >= 256 ns)
 "capture": {"length": 1 .. 4096},
 "outputs": {"1" .. "8": {"mode": "off" | "low" | "high" | "pattern" | "ttl" | "gate1" | "gate2"
                                  | "gate" | "capture" | "shot" | "run" | "state",
                          "pattern": 0..15, "bit": 0..7, "invert": bool, "delay": seconds}},
                                                (bit: the TTL bit, or the window of "state")
 "gates": {"tx1" | "tx2" | "any": {"lead": seconds, "lag": seconds}},
 "emulator": {"mode": "off" | "qubit" | "ensemble",
              "frequency": Hz,                  (the spin's: TX1 within 3 MHz of it)
              "rabi": Hz,                       (the Rabi frequency per volt of TX1's amplitude)
              "t1": null | seconds, "t2": null | seconds,   (t2 <= 2 t1; t2 null: 2 t1, or none
               ↪ without t1)
             "excited": 0 .. 1,              (the thermal excited population: z_eq = 1 - 2 excited)
              "detuning": {"shape": "none" | "gaussian" | "lorentzian", "width": Hz},   (drawn per
               ↪ SHOT)
              "ao": Hz/V,                       (AO0 shifts the spin's frequency: a field sweep)
              "signal": volts,                  (qubit: the readout's amplitude per volt of TX2;
                                                 ensemble: the signal at |M| = 1, at the input)
              "chi": degrees,                   (the readout's phase: +chi ground, -chi excited)
              "ringdown": seconds,              (the readout resonator's time constant, 2^k clocks)
              "noise": volts,                   (Gaussian, rms, per quadrature, at the input)
              "reset": bool,                    (each run starts from (0, 0, z_eq))
              "seed": null | integer}}          (null: a free-running generator; else the draws
               ↪ repeat for
                                                 runs started at the same clock modulo 16 (128 ns))
```

An instruction (`control/spin/sequence.py`; every key optional):

```
{"op": "cont" | "loop" | "end" | "wait" | "branch" | "jump" | "stop",
 "duration": seconds (16 ns .. 34 s) | {"table": j},
 "level": 0..3, "count": n | {"register": r} | {"table": j},     (loop)
 "target": instruction,                                          (jump, branch)
 "if": {"state": w} | {"cond": "a" | "b"} | {"parity": level}, "not": bool,      (branch)
 "wait": "trigger" | "timer" | "software", "reref": bool,         (wait)
 "ttl": 0..255, "shot": bool, "capture": bool, "offset": 0..1023,
 "tx1", "tx2": {"on": bool, "keep": bool, "freq": r, "freq_table": j, "phase": r,
                "phase_table": j, "amp": r, "amp_table": j,
                "shape": 0..31, "chirp": 0..7},                  (0: rectangular, no chirp)
 "ao": {"output": 0..3, "register": r} | {"output": 0..3, "table": j},
 "window": 0..7, "rx_phase": 0..3}                            (a window starts; its receiver phase)
```

A literal loop count takes a count register no instruction names and appears as `{"register": r}` in `CONFIG?`. Table words are raw (THEORY 2.2): phases in 2^−16 turn, amplitudes in the amplitude registers' units, frequencies in steps of 2^16 frequency units (29.1 mHz), durations in clocks, AO values in their 16-bit codes (`ao_code(output, volts)`). An AO register is converted with the calibration of the output its instructions update (one register per

output when their calibrations differ). Limits and guards marked `protect` (or locked by PROTECT) can only be applied unchanged until the FPGA is reloaded; SAFE, presets and restores keep them. A new shape table stops a running program (the transmitters read it). A line is at most 64 KB (the service's limit): about 200 instructions, or four full tables, fit in one APPLY; a longer program needs the driver on the board (a chunked command is planned).

Presets (`PRESET name`; the list in the README): `off` (all outputs in high impedance, both transmitters off, the program a STOP), `pulse-loopback`, `fid-emulator`, `echo-emulator`, `cpmg-emulator`, `nutation-emulator`, `rabi-qubit`, `ramsey-qubit`, `readout-qubit`, `nmr-fid-in1`, `esr-iq-echo`.

Weights are relative (null: equal), normalized by the driver so that a steady tone of amplitude A at the LO reads A (S in volts), and rotated by `phase`; the weighed windows of each RX share its 1024 weights in window order. The receivers need the capture's delay (from the calibration's input delay) within 7 … 127 clocks.

Decisions (0.5). A window with weights decides a state at its end: 1 when S_I (the weighed sum rotated by its `phase`) exceeds its `threshold` in volts. The core keeps each window's last state (0 after a RUN); an output in mode `state` shows window `bit`'s (D + 5.5 clocks + the board's `pins_delay_s` after the window's last sample at the IN connector: 332 ns nominal, 281 ns measured on the test board, THEORY 2.1; its own delay only, no alignment). A `branch` plays like a CONT (its duration, TTL bits and transmitters; it starts no window) and continues at `target` if its condition holds (`not` inverts it), else at the next instruction: window w's state (once every window w started before it has ended), COND A or B's level (DIO1_P, DIO2_P) at the branch's start, or loop counter i_level's parity (odd: true). The decision takes time, and a branch that ends before it stops the run with an underflow, so the driver refuses: a branch on COND shorter than 256 ns; a branch on a window without weights; a branch on a window started in the straight-line instructions before it unless those from the window's start to the branch's end last (K + 1) R + D + 27 clocks (K points, R the decimation, D the capture's delay: 89 with the emulator; THEORY 2.2). Durations from tables are not checked. `experiments.decision(ground, excited)` gives a window's phase and threshold from the two references; `active_reset` and `repeat_until_success` are templates on the emulated qubit (or a real one).

Experiments (`control/spin/experiments.py`, on a PC or the board): each template returns a configuration to APPLY (`config_for(device, exp)` fills a field sweep's slow-output table with codes through the board calibration) and `acquire(device, exp)` runs it through these commands (APPLY, SWAP, COLLECT, AVERAGE CLEAR, RUN, STATE?, AVERAGE?). Templates: `fid`, `echo`, `t2`, `cpmg`, `slse` (NQR spin-locked echo trains), `inversion_recovery`, `saturation_recovery`, `nutation`, `field_sweep` (ESR's echo-detected field sweep), `rabi`, `ramsey`, `t1_qubit`, `readout`, `active_reset`, `repeat_until_success`. The Jupyter panel (`spin.make_gui(dev)`) uses the same commands.

The emulator (THEORY 2.8) answers the transmitters at the receivers whose input is `emulator`: TX1 drives the spin in TX1's frame (pulses, phases, shapes, chirps), TX2 reads the qubit out (a readout pulse returns at ±chi with the noise; read it with TX2's LO), and in ensemble mode the transverse magnetization returns at TX1's LO. While it is on, the capture's delay is 89 clocks (windows on IN1/IN2 start later than calibrated). It updates every 128 ns: keep TX1 within 3 MHz of the spin's frequency and the Rabi frequency below 1.95 MHz at full amplitude.

spin registers

`sp_core` is a 256 KiB AXI4-Lite window at **0x43C00000** (32-bit words, byte offsets). Access it only through the driver (`rpfpga.Overlay`): a read or write while the FPGA is not configured hangs the board (`../../common/README.md`, "Safe loading"). The common sysid block is at 0x40000000 (design ID `SPIN`, version 1.1).

Layout version (ABI) 6 (since 0.5), spin 1.1 (the layout of 1.0). A read takes four clocks. Unused offsets read 0 and ignore writes. The definitions are in THEORY.md section 2.

Block RAMs

Offset	Contents
0x08000 + 32 i + 4 w	instruction i (0 … 1023), word w (0 … 4; THEORY 2.2: BRANCH (0.5) op 4 in W1, its condition in W4 [31:29]; words 5 … 7 read 0). Power-up: STOP
0x10000 + 4096 j + 4 i	table j (0 … 3), entry i (0 … 1023)
0x14000 + 4 p	shape point p (0 … 4095): [15:0] amplitude (2^−16), [31:16] phase (2^−16 turn) (THEORY 2.3). Written only while no run uses it (writes are ignored then, reads undefined)
0x28000 + 4 n	the capture's word n (0 … 4095): [15:0] IN1, [31:16] IN2 (ADC codes; read only)

Offset	Contents
0x18000 + 0x2000 k + 8 j	RX k+1's weight j (0 … 1023): + 0 w_I, + 4 w_Q ([17:0], signed; reads sign-extended)
0x30000 + 0x8000 b + 16 p	the averager's frozen half, bank b (RX b+1), point p (0 … 2047): + 0 I [31:0], + 4 Q [31:0], + 8 {Q [34:32] at [10:8], I [34:32] at [2:0]}; 0 if no half is frozen (read only; two clocks)
0x1C000 + 4 i	the emulator's NOISE[i] (i 0 … 1023): [17:0] a Gaussian quantile in ADC codes (signed; reads sign-extended; THEORY 2.8)
0x1D000 + 4 i	the emulator's DET[i] (i 0 … 1023): [17:0] a per-shot detuning quantile in units of w per update (2^−18 turn per 128 ns; signed)
0x20000	reserved

Control registers

Offset	Name	Access	Meaning
0x000	ABI	R	the register layout's version (6)
0x004	STATUS	R	[0] the 1 ns samplers' PLL locked; [3] the ring writer runs; [4] its watchdog tripped
0x008	LEVELS	R	each input's latest 1 ns sample: [0] TRIG (DIO0_P), [1] COND A (DIO1_P), [2] COND B (DIO2_P)
0x00C	SEEN	R / W1C	a rising edge since the last clear, same bits; writing 1 clears a bit
0x010	OUT_EN	RW	[7:0] outputs D1..D8 (DIO0..7_N) driven; 0 = high impedance (power-up, SAFE)
0x014	OUT_PAT	RW	[4k+3:4k] output k+1's bits of each sample period (source pattern), bit 0 first, 2 ns each
0x018	TIME_LO	R	sample periods since the core's reset, bits 31..0 (reading it latches TIME_HI)
0x01C	TIME_HI	R	bits 63..32, latched by the last TIME_LO read
0x040 … 0x064	W_CTRL … TEST_DROPS		the ring writer and its test words (as in 0.0: W_CTRL, W_BASE, W_SIZE, W_RD (heartbeat), W_WD, W_WR, W_BURSTS, W_STALLS, TEST, TEST_DROPS)
0x100	S_CTRL	W	[0] START (a run from instruction 0; t0 = this clock), [1] STOP, [2] software trigger (pulses)
0x104	S_STATUS	R	[0] running, [1] waiting (a WAIT is current), [2] an instruction is current, [7:4] error (1 underflow, 2 TX1's duty guard, 3 TX2's), [17:8] the current instruction, [22:18] the FIFO's entries
0x108	S_SHOTS	R	instructions with SHOT started in this run
0x10C	S_MISSED	R	TRIG edges not taken by a WAIT (while running)
0x110	S_TMISSED	R	timer ticks not taken by a WAIT (while running)
0x114, 0x118	S_START_LO, S_START_HI	R	the clock of the last START (t0 of the run)
0x11C	S_TIMER	RW	the WAIT timer's period in clocks (≥ 32; power-up 1000)
0x120	S_TBL_CTR	RW	[2j+1:2j] the loop counter indexing table j
0x124	C_CFG	RW	the capture: [11:0] length − 1 (K = 1 … 4096), [23:16] delay D in clocks (power-up 0x0012_03FF: K 1024, delay 18); the receivers use D = 7 … 127 (89 while they read the emulator)
0x128	C_STATUS	R	[12:0] samples taken by the last capture, [16] taking samples
0x12C, 0x130	S_TRIG_LO, S_TRIG_HI	R	the last TRIG edge a WAIT took, in ns since the core's reset (8 × its word + its slot)
0x140 + 4 r	S_COUNT[r]	RW	loop count register r (0 … 7)
0x170, 0x174, 0x178	PROG_ID, TBL_ID, SHAPE_ID	RW	the driver's CRC-32 of the program, the tables and the shape table (no function in the FPGA)
0x200 + 8 r, + 4	TX1 FREQ[r]	RW	frequency register r: [31:0], then [15:0] the bits 47..32 (`f = F · 125 MHz / 2^48`)
0x240 + 4 r	TX1 PHASE[r]	RW	[15:0] phase register r (2^−16 turn)
0x260 + 4 r	TX1 AMP[r]	RW	[15:0] amplitude register r (THEORY 2.3)
0x280 + 8 r, + 4	TX1 CHIRP[r]	RW	chirp register r (1 … 7): [31:0], then [15:0] the bits 47..32 (two's complement, 2^−4 frequency steps per clock); r = 0 reads 0
0x2C0	TX1 QGAIN	RW	[16:0] the Q path's gain (65536 = 1; power-up 65536)

Offset	Name	Access	Meaning
0x2C4	TX1 QPHASE	RW	[15:0] the Q path's phase (2^−16 turn)
0x2D0	TX1 G_CFG	RW	the duty guard: [0] the pulse length checked, [1] the bucket checked (locked by PROTECT[0])
0x2D4	TX1 G_LMAX	RW	the longest pulse (clocks; locked)
0x2D8	TX1 G_RATE	RW	[16:0] the bucket's drain per clock (65536 = full duty; locked)
0x2DC, 0x2E0	TX1 G_BMAX	RW	the bucket's limit: [31:0], then [15:0] the bits 47..32 (locked). A change of G_CFG, G_RATE or G_BMAX empties the bucket
0x2E4	TX1 G_LEVEL	R	the bucket's level, bits 47..16 (clocks of full duty)
0x300 … 0x3E4	TX2	RW	as TX1 (locked by PROTECT[1])
0x400, 0x404	OFFSET1, OFFSET2	RW	[13:0] the output's offset (14-bit code, signed): the calibrated 0 V plus the DC level
0x408, 0x40C	LIMIT1, LIMIT2	RW	[13:0] the lowest, [29:16] the highest 14-bit code (signed; power-up 0x1FFF_2000: the full range; locked by PROTECT[n − 1])
0x410 + 4 p	PIN_CFG[p]	RW	output p+1: [2:0] source (0 pattern, 1 TTL bit, 2 gate TX1, 3 gate TX2, 4 gate either, 5 capture window, 6 SHOT, 7 an instruction current; with [17] (0.5): 8 window [5:3]'s state), [5:3] TTL bit (or the window), [6] invert, [16:7] delay in 2 ns steps, [17] source bit 3
0x430 + 4 g	GATE_CFG[g]	RW	gate g (TX1, TX2, either): [7:0] lead, [23:8] lag (clocks)
0x440	OUT_MODE	RW	[0] 1: OUT1 + i OUT2 = TX1 + TX2 (I/Q); 0: OUT1 = TX1, OUT2 = TX2
0x444	PROTECT	R / W1S	[0] OUT1's limits and TX1's guard locked, [1] OUT2's and TX2's: writing 1 sets a bit, which clears only when the FPGA is reloaded
0x448	G_STATUS	R	the guard's trips since the last START: [0] TX1 pulse length, [1] TX1 bucket, [2] TX2 pulse length, [3] TX2 bucket
0x480 + 4 k	AO_SET[k]	RW	[15:0] AO k's idle level (outside runs and at a run's start; 2^−16 of full scale)
0x4A0 + 4 r	AO_REG[r]	RW	[15:0] AO register r (0 … 7), an instruction's value
0x4C0 + 4 k	AO_NOW[k]	R	AO k's present value (the run's, else the idle level)
0x700 + 16 w	WIN_CFG[w]	RW	window w: [0] RX2, [1] LO TX2, [2] raw, [3] average, [4] weigh, [5] (TRACE: not built), [6] its region from a table, [8:7] the table, [22:12] K − 1
0x704 + 16 w	WIN_R[w]	RW	[13:0] R − 1, [19:16] ⌈log2 R⌉
0x708 + 16 w	WIN_BASE[w]	RW	[10:0] the region's base, [25:16] the weights' base
0x70C + 16 w	WIN_THR[w]	RW	the threshold (signed; the state is floor(S_I / 2^16) > it)
0x780, 0x784	RX_CFG1, RX_CFG2	RW	[1:0] the input: 0 IN1, 1 IN2, 2 IN1 + i IN2, 3 the emulator (its LO, no offset or overload)
0x788, 0x78C	IN_OFF1, IN_OFF2	RW	[15:0] the input's offset in ADC codes (subtracted)
0x790	AVG_CTRL	W	[0] request a swap (at the next SHOT's sample; without a run at once, or once the other half is free), [1] release the frozen half (it clears in 2048 clocks without points, about 16.4 µs, then is free)
0x794	AVG_STATUS	R	[0] the active half, [1] a half frozen, [2] which, [3] a swap requested, [4] a half closing, [5] the frozen half saturated; [31:16] halves frozen
0x798	AVG_N	RW	swap after N scans (0: on request only)
0x79C	AVG_SCANS	R	the frozen half's scans
0x7A0	WIN_MISSED	R	windows not taken (their RX busy) since the START
0x7A4	REC_CTRL	RW	[0] records made
0x7A8	REC_DROPS	R	records lost
0x500	EMU_CTRL	RW	the emulator (THEORY 2.8): [1:0] mode (0 off, 1 qubit, 2 ensemble), [2] reset at each START, [3] reseed at each START
0x504, 0x508	EMU_SEED_LO, EMU_SEED_HI	RW	the generator's 64-bit seed
0x50C, 0x510	EMU_FQ_LO, EMU_FQ_HI	RW	the spin's frequency f_q: [31:0], then [15:0] the bits 47..32 (as FREQ)
0x514	EMU_KR	RW	[16:0] k_R: w = ⌊(k_R D + r) / 2^22⌋
0x518, 0x51C	EMU_G1, EMU_G2	RW	[30:0] 1 − e^(−T/T1), 1 − e^(−T/T2) per update (2^31 = 1; T = 128 ns)

Offset	Name	Access	Meaning
0x520	EMU_DZ	RW	(1 − e^(−T/T1)) z_eq (2^30 = 1; signed)
0x524	EMU_Z0	RW	the reset state's z (2^30 = 1, signed, within ±2^30; power-up 2^30)
0x528	EMU_KAO	RW	[17:0] AO0's coupling (signed): w_z += round(k_AO (AO0 − 32768) / 2^16)
0x52C	EMU_P1	RW	[23:0] the readout's decay probability per slot (2^24 = 1)
0x530	EMU_CHI	RW	[15:0] the readout's phase ±χ (2^−16 turn)
0x534	EMU_GAIN	RW	[16:0] G: qubit, codes per TX2 amplitude (2^16 = 1); ensemble, codes at |M| = 1
0x538	EMU_KAPPA	RW	[3:0] the readout resonator's pole κ (ringdown 2^κ clocks)
0x560	EMU_STATUS	R	[0] the qubit's outcome after the last update (1 excited)
0x580 + 4 k	EMU_V[k]	R	the Bloch vector's component k (0 x, 1 y, 2 z; 2^30 = 1, signed; z0 for z before the first update after a reset)
0x600 + 8 s	SHAPE_DESC[s]	RW	shape s (1 … 31): [11:0] its first point, [24:12] its length (2 … 4096); + 4: [19:0] its step (points per clock, 16 fraction bits)

Configuration registers (restored and checked by `rpfpga.Overlay`): OUT_EN, OUT_PAT, S_TIMER, S_TBL_CTR, C_CFG, S_COUNT, PROG_ID, TBL_ID, SHAPE_ID, the TX registers (but G_LEVEL), the offsets, the limits, OUT_MODE, PIN_CFG, GATE_CFG, AO_SET, AO_REG, SHAPE_DESC 1 … 31, WIN, RX_CFG, IN_OFF, AVG_N, REC_CTRL, EMU_CTRL … EMU_KAPPA. The weights and the emulator's tables are written when they differ from the driver's copy. The driver writes the program, the tables and the shape table only when PROG_ID, TBL_ID or SHAPE_ID changes (a new shape table stops a running program first), and reads them back when it derives a configuration from the registers (the shape table only while no run uses it). The ring registers belong to the service's worker (worker.py: the records, the averager's frozen halves).

Verification record

What was tested on spin, when, and with what result, newest first. Only results that were actually obtained are listed; anything not yet done is said so. Every step is tested on the board before the next one starts (the user's decision, 2026-11-05).

How to repeat the checks: `PATH=~/miniforge3/envs/base-tools/bin:$PATH make test`, `make sim`, `make`, `python3 ../common/tests/verify_build.py --design spin`; on the board `make upload`, `make install-service`, `make test-hw` (twice).

2026-11-08: 1.1, the averager in common (build 0x33bbab94)

The averager moved into `common/` for boxcar (the user's decision): `sp_avg.v` became `../common/rtl/optional/rp_avg.v` (parameterized; spin's parameters AW 11, IN_W 20, SUM_W 35, BWAIT 31, per-half scans, direct host reads give spin's logic), the host words' mux moved into it, the model became `rpfpga.avg.Averager` (spin's `model.Averager` a subclass with its parameters) and the driver's decoding `rpfpga.avg.unpack`. Nothing else changed.

- **`make test`:** 72 passed, 1 skipped; `test_model`'s check that the vectors are current passed, so the model computes the same expectations as before.
- **`make sim`:** all six benches passed unchanged (`tb_avg` instantiates `rp_avg` with spin's parameters; `tb_seq` compares the frozen half's words against the model's vectors).
- **Build 0x33bbab94:** with 1.0's placement (ExtraTimingOpt) routed WNS +0.095 ns, below the +0.10 ns margin; eight implementation variants from its post-opt checkpoint (`.build/impl_explore11`): AltSpreadLogic_high with post-place physical optimization gave +0.263 ns, the best; `design.tcl` now uses it. Final: WNS +0.263 ns, TNS 0, WHS +0.033 ns; 4,141 slices (94.1 %), 13,559 LUTs (77.0 %), 47.5 BRAM36, 71 DSP48; DRC only RTSTAT-10 and the DSP pipelining notes. `verify_build.py`: 8 of 8 checks passed; a fresh copy from the recipe rebuilt 0x33bbab94 with the same WNS (+0.263 ns).
- **Board, `make test-hw` twice on 0x33bbab94** (OUT1 → T → IN1, OUT2 → T → IN2 at 50 Ω, the jumper D8 → COND A): **62 of 62 passed both times** (receipts `data/board-1.1-test_hw-{1,2}.json`), among them the ensemble's FID bit-exact against the model (32 points of the averager) and the frozen half's block read (12,288 words in 127 ms).

2026-11-08: safety rounding and the gates at a guard trip (build 0x5e4b8972)

Two changes after the bench session (below), the user's decision.

- **Rounding to the safe side (driver only):**
  - A gate's lead and lag are minimums. They are now rounded up to whole clocks; they were rounded half to even, which turned 100 ns into 96 ns. 100 ns is now 104 ns; exact multiples stay.
  - The guard's limits are maximums and are now all rounded down. `max_pulse` already was; `max_duty` and `burst` were rounded to nearest.
  - A new test, `test_safety_times_round_to_the_safe_side`, covers both.
  - The presets `nmr-fid-in1` and the I/Q loopback preset now give a 104 ns lead.
- **The gates at a guard trip (RTL):**
  - Until now a trip forced the gate pins closed from clock t + 5. On the test board that was 145 ns before the cut RF's end at the connectors: a receiver-protection gate opened during the pulse's last 145 ns and the probe's ring-down.
  - Removed from `sp_pins` and the model. The trip's stop of the run ends the TXs' `on` 4 clocks after the blanking, so each gate now closes its lag + 4 clocks after the cut RF's end, as after any pulse.
  - The model's guard test pins the gate's end. Scenario C's expected pin words changed in 82 clocks, so the bench checks the new behaviour (the old RTL produced the old words).
  - `make sim`: all six benches passed; spin's `make test`: 72 passed.
- **Build:**
  - The first build placed with WNS +0.039 ns, below the +0.10 ns margin.
  - Six implementation variants from its post-opt checkpoint (`.build/impl_explore`): post-place `phys_opt_design -directive AggressiveExplore` after the ExtraTimingOpt placement gave +0.352 ns, the best; `design.tcl` now enables it.
  - Final: WNS +0.352 ns, TNS 0, WHS +0.034 ns; 4,226 slices (96.05 %), 13,867 LUTs (78.8 %), 47.5 BRAM36, 71 DSP48; DRC only RTSTAT-10 and the DSP pipelining notes.
  - The build ID is a hash of the sources, so both placements are 0x5e4b8972; the board tests below are on the final bitstream.
  - `verify_build.py`: 8 of 8 checks passed; the fresh copy from the recipe rebuilt 0x5e4b8972 with the same WNS (+0.352 ns) and slices.
- **Board** (the bench's wiring, the jumper D8 → COND A, the scope's inputs at 50 Ω):
  - `make test-hw` twice: **62 of 62 both times** (receipts `data/board-1.0-test_hw-gates-{1,2}.json`).
  - The oscilloscope (receipt `data/board-1.0-bench-guard-2026-11-08.json`; as the scope shows the times, see the bench session for the conventions):
    - A gate with lead 100 ns (applied 104 ns) and lag 200 ns opened 103.54 ns before TX1's 1 µs step and closed 199.06 ns after it.
    - With max_pulse 0.5 µs (applied 496 ns) the RF was cut after 497.56 ns at 50 % (error 2). The gate opened 103.52 ns before it and closed **231.09 ns after the cut RF's end**: the lag plus 32.03 ns (4 clocks: the run's stop). Before the change it closed 144.9 ns *before* the end.

2026-11-08: the bench session (latencies, gates, the duty guard, AO0, COND)

Deferred measurements of the release, on build 0xc9ca505b (receipt `data/board-1.0-bench-2026-11-08.json`).

- **Setup:**
  - OUT1 → BNC T → IN1, the T's arm by coax to CH3; OUT2 → T → IN2, the arm to CH4; the scope's inputs (Siglent SDS2504X Plus) at 50 Ω.
  - CH2's 10× probe on DIO0_N (D1); CH1's 10× probe on AO0 (E2 pin 17); a jumper DIO7_N → DIO1_P (D8 → COND A).
  - Rectangular steps from a transmitter at 0 Hz (phase 0). Times are 50 % points, interpolated, 20 acquisitions each (sd ≤ 0.06 ns).
  - Pin times are given as the scope shows them, the way the board's `pins_delay_s` was measured. Referred to the connectors means: less CH2's probe delay (6.63 ns, measured 2026-11-07) and plus the RF's two coax runs to the scope (about 2.5 ns each, not measured).
- **`make test-hw` twice: 62 of 62 both times** (receipts `data/board-1.0-test_hw-jumper-{1,2}.json`).
  - The 62 are the 58 of the release plus the four jumper checks: D8 → COND A high, low and released; a 125 MHz pattern's edges; a BRANCH following D8's level; a window's state carried by D8 to a BRANCH.

  - A first run, with the scope's inputs at 1 MΩ, failed 9 checks:
    * Eight loopback checks: the unterminated T arms changed the loop's response (an amplitude read 1.061 V where the clean loop reads 0.430 V).
    * The run-epoch check: the previous run's last window ended exactly 64 clocks before a START. That is a boundary of the check's own criterion (< 64). Windows end every 64 clocks, so a window ending on the START's clock was in flight. The criterion is now ≤ 64; no stale RESULT was seen in any run.
  - At 50 Ω both runs passed. The loop read 0.4280 V for 0.4 V (0.4303 V without the T's).

- **A branch on a window's state:**
  - Window 0 on IN1: K = 4, R = 8, D = 33 (the board's 120 ns converter loop), threshold 0.3 V. The BRANCH was one clock above its least duration.
  - A 0.4 V step gave S = 0.817 V. That is twice IN1's level (0.407 V by a capture): at 0 Hz the receiver's gain, set so that a tone of amplitude A reads A, doubles a DC level.
  - The branch was taken in 20 of 20 shots. OUT2's chosen step came **801.98 ns** (sd 0.04) after IN1's step; the program sets 800 ns.
  - So from the window's last sample at IN to the chosen pulse at OUT, the least branch gives D + 28 clocks = 488 ns (computed), plus 2.0 ns between OUT2's and OUT1's paths at the scope. The board's calibrated OUT2 skew is −0.17 ns, so the 2.0 ns are the cables and T's.
  - Below the threshold (a 0.05 V step, S = 0.103 V): state 0 in 5 of 5 shots, no OUT2 pulse (7 mV swing), D1 low (≤ 67 mV of crosstalk).
  - A first version of this check, with the threshold at 0.1 V, branched: S = 0.103 V was above it. The test was at fault, not the design.

- **The state output on D1:**
  - D1 rose 584.93 ns (sd 0.04) after IN1's step as the scope shows it: **280.9 ns after the window's last sample** (the step + 38 clocks).
  - Computed (THEORY 2.1): D + 5.5 clocks + `pins_delay_s` (−28.6 ns) = 279.4 ns.
  - Referred to the connectors: 276.8 ns, 2.6 ns early.

- **A gate on D1:**
  - Setup: TX1's 1 µs step, lead 96 ns, lag 200 ns. Lead and lag are whole clocks; 100 ns rounds to 96 ns (12.5 clocks, rounded half to even).
  - As the scope shows them: the gate opened 95.52 ns before the RF's 50 % point and closed 199.09 ns after it. The gate was 1296.17 ns wide (96 + 1000 + 200), the RF 1001.56 ns at 50 %.
  - Referred to the connectors: opened 1.2 ns early, closed 2.5 ns early.
  - D7 at 0.1 was 1.5 ns early by the same method. That is within this setup's errors (cable lengths, the probe's ground, pin skews).

- **The duty guard** (TX1's max_pulse 0.5 µs, the same 1 µs pulse):
  - The RF was blanked after 497.56 ns at 50 %: 62 clocks plus the RF's 1.56 ns.
  - Error 2; the pulse check tripped.
  - The gate closed **144.9 ns before the blanked RF's end** as the scope shows it (THEORY 2.3: from player clock t + 5, i.e. 116 ns − `pins_delay_s` = 144.6 ns ahead on this board).

- **AO0** (CH1's mean and peak-to-peak at 20 MHz bandwidth, 5 readings per level):

Set (V)	Mean (V)	Ripple (V p-p)
0.1	0.126	0.23
0.5	0.513	0.75
0.9	0.914	0.91
1.3	1.316	0.77
1.7	1.728	0.25

  - The means are within 28 mV of set: a line 1.0017 × set + 18 mV, residuals ≤ 8 mV. That is within the scope's DC accuracy (about 2 % of full scale). The calibration's AO fields are nominal.
  - The ripple is the 488 kHz PWM through the board's first-order filter, seen at the pin. A field supply or a bias needs an external low-pass.

2026-11-08: the manuals

- The user's manual and the theory manual (`docs/manual/`). Their measured figures come from board recordings on build 0xc9ca505b (`tools/record_manual_data.py`: the emulator's experiments, a loopback, I/Q mode, 2000 + 2000 single shots, active reset and repeat-until-success); each was checked against an independent computation before it went in: the echo trains against the float Bloch equations averaged over the line (with the line's 20 kHz offset; the model equals them over the actual draws to four digits), the FID's line width against the spread of 256-shot runs, the averaging noise against the CIC's noise gain (1.042 against 1.037 mV), the readout's errors against the model's pattern.
- Two presets corrected while recording: `pulse-loopback` now steps the receiver's phase with TX1's (CYCLOPS; before, the four phases cancelled in window 0's average), and `readout-qubit` names its weighed window correctly (0). `make test-hw` on the same build: **58 of 58** (every preset loads and reads back).
- `tests/test_manual.py`: the user's manual's examples run on the driver, the source excerpts match, every figure and value exists, the user's figures regenerate from the recordings, both manuals build; the theory figures regenerate identically (`RPFPGA_SLOW=1`).

2026-11-08: 1.0, the release (plan step 1.0)

- **`make test`** (the repository's): 1,361 tests ran, 22 skipped, none failed; spin's 61 passed. New: every preset loads and reads back from the registers; the Jupyter panel drives the driver (emulator, experiment, outputs) and analyses simulated FID and Rabi runs; the NQR spin-locked echo train (every echo refocuses; the train decays no faster than CPMG's) and the ESR echo-detected field sweep (a 50 kHz line mapped through AO0, its centroid within 0.03 V of the expected level) through the driver and the bit-exact model; the emulator's T2 ≤ 2 T1 rule and `t2` null read back; the FID's amplitude against the float Bloch equations through the same receiver and fit (1 %).
- **Two defects found while checking the new templates against the float Bloch equations:**
  - the templates' first pulse of a run was mis-rotated: after a START TX1's step is 0 until its first coherent start, the emulator's frame follows it, and the slot holding the first pulse's start also held a 10 MHz detuning (clamped to 3.125 MHz): a 90° pulse came out near 76° (|M⊥| 0.97 instead of 1.00, inside 0.4's 3 % amplitude tolerance; diluted over the scans of an average). Each acquisition's prelude now starts TX1's NCO without output (256 ns before the first pulse); THEORY 2.8 says so for hand-written programs;
  - unphysical relaxation was accepted: tests (and the board test's echo and bit-exact FID) used T2 > 2 T1, where |M| grows beyond 1 and a later pulse clamps a component (the echo read up to 30 % low against the float equations). The driver refuses T2 > 2 T1; `t2` null with a T1 is 2 T1. With physical values the emulator agrees with the float Bloch equations within 0.1 % (hard and soft pulses, offsets ±100 kHz). The board test now uses T1 4 µs (bit-exact FID) and 10 µs (the echo).
- **`make sim`:** the vectors regenerated (scenario H follows the driver's templates); all six benches passed.
- **Build 0xc9ca505b** (no RTL change from 0.5 but the sysid version): with the default placement WNS was +0.033 ns (TX1's phase-file read-add-write); with `ExtraTimingOpt` placement 4,164 slices (94.6 %), 13,754 LUTs (78.1 %), 47.5 BRAM36, 71 DSP48, routed WNS +0.203 ns, TNS 0, DRC clean. `verify_build.py`: 8 of 8 checks passed; a fresh copy from the recipe rebuilt 0xc9ca505b with the same WNS and slices.
- **Board, `make test-hw` twice on 0xc9ca505b** (OUT1 → IN1, OUT2 → IN2; no jumpers on the pins): **58 of 58 passed both times** (receipts `data/board-1.0-test_hw-{1,2}.json`). New: every one of the 11 presets loads and reads back from the registers; the `fid-emulator` preset runs its 256 shots. The emulator against the physics (runs 1, 2): the FID's T2* 8.000 µs (8), offset 50.00 kHz, 0.1008 V (0.1); the echo of a 20 kHz line 0.03602 and 0.03614 V against 0.03621 V from the float equations over the line (T1 now 10 µs, T2 20 µs, tr 80 µs); CPMG's T2 40.03, 40.04 µs (40); inversion recovery's T1 10.00 µs (10); the nutation's Rabi frequency 0.9991, 0.9990 MHz (1: 0.4 read 1.0034–1.0054 MHz, the first pulse's error); the qubit's Rabi 1.0027, 0.9996 MHz, Ramsey 500.9, 504.9 kHz (500), T1 20.69, 20.00 µs (20); single-shot fidelity 0.9850, 0.9868 (SNR 7.6, 8.2); seeded active reset and repeat-until-success equal to the model; active reset 0.0332, 0.0319 left (the model 0.023), repeat-until-success 0.9267, 0.9389 (0.952); ten restarts, no stale window. A development run before the echo's tr was lengthened also passed 58 of 58 (its echo 1.9 % low: the shots, 30 µs apart with T1 = 10 µs, started 1.3 % short of equilibrium).
- **Not done:** the COND checks through a jumper and the latencies with an oscilloscope (as at 0.5); the SD image and the project page are not rebuilt or uploaded (`site/README.md`).
- The board was unreachable on the network for 1.5 hours on 2026-11-08 (no ARP reply; it had not rebooted: 8 days' uptime when it answered again).

2026-11-07: 0.5, decisions and branching (plan step 0.5)

- **`make test`** (the repository's): 1,357 tests ran, 22 skipped, none failed; spin's 57 passed. New: the branch encoding and compilation (`if`, `not`, the state, COND and parity sources), the driver's refusals (COND under 256 ns, a state branch on a window without weights or too early), state outputs (PIN_CFG [17] written and read back), and active reset and repeat-until-success through the driver and the bit-exact model (the decision's phase and threshold from the references).
- **The least durations, from the model.** Sweeping the model before the board showed that a branch on a window's state needs (K + 1)·R + D + 26 clocks from the window's instruction to its end, not THEORY's first K·R + D + 27 (exact only for R = 1: a window's last point is at s + (K + 1)R − 2, 2.4). The driver asked K·R + D + 28 (too short for R ≥ 3) and the qubit templates' branch wait would have underflowed at t_read = 2 µs (R = 16). THEORY 2.2, the driver ((K + 1)·R + D + 27) and the templates were corrected; a branch on COND needs 29 clocks as derived (the compiler asks 32).
- **`make sim`:** `tb_seq` with scenarios I (new: branches on a parity, on window 0 while it runs, inverted, on window 1 after a missed start, on window 2 without weights, on COND A and inverted COND B driven by the bench, out of a loop mid-iteration with the loop's table index read anew; state outputs; the three branches on window 0, window 1 and COND A at their least durations), J (a branch on COND one clock short, after a long parity branch) and K (a branch on a state one clock short): J and K underflow, as the model; all six benches passed (`tb_sysid`, `tb_core` with ABI 6 and PIN_CFG's 18 bits, `tb_seq`, `tb_avg`, `tb_rec`, `tb_emu`).
- **Mutation tests:** 18 one-token mutants of the branch, state and pin logic (the state's validity and window, the parity bit, the inversion, COND's read clock, its arming, its input, the pending count's sources, the engine's wait, the fall-through address, a fresh loop's table index, the states' update and reset, the state pins' window and their pattern gating, window ends without weights): 17 fail `tb_seq` (four survived a first round and were killed by J at 28 clocks, the parity branch before it, A ≠ B at I's reads, and a window 2 that would decide 1). The run epoch's gate survives: no scenario restarts while window starts are in flight; the board checks it (below).
- **Build 0xc2958503:** 4,236 slices (96.3 %), 13,751 LUTs (78.1 %), 47.5 BRAM36, 71 DSP48; routed WNS +0.247 ns, TNS 0, DRC clean (the caps: 98 % slices, 14,300 LUTs, 56 BRAM, 76 DSP). The routed netlist has the branch's registers (BW, COND's arming and count), the eight window states and PIN_CFG's 18 bits per pin. `verify_build.py`: 8 of 8 checks passed; a fresh copy from the recipe rebuilt 0xc2958503.
- **Board, `make test-hw` twice on 0xc2958503** (OUT1 → IN1, OUT2 → IN2; no jumpers on the pins): **56 of 56 passed both times** (receipts `data/board-0.5-test_hw-{1,2}.json`). New, all without wiring:
  - a loop counter's parity: 5 SHOTs in 10 iterations;
  - a branch on a window's state (K 4, R 2, the emulator: D = 89) underflows at 83 and 84 clocks and not at 85 and 86: the least duration (K + 1)·R + D + 26 − 40 = 85 to the clock; on COND: underflows at 27 and 28, not at 29 and 30;
  - seeded active-reset and repeat-until-success runs (4 shots each: noise, outcomes, branches) equal the bit-exact model started at the board's START clock;
  - active reset, 2000 shots on the emulated qubit (T1 20 µs, a π/2 preparation): 0.0295 and 0.0371 excitation left (without the reset 0.439, 0.442; the model 0.023 ± 0.008): mostly T1 decay between the readout's projection and the π pulse (about 1.5 µs);
  - repeat-until-success, 8 attempts: 0.9355 and 0.9435 excited (the model 0.952 ± 0.011);
  - ten restarts during windows every 64 clocks: no window from the previous run's starts after a START (the previous run's last window ended 54 and 61 clocks before it: starts were in flight); no records lost. During development the same build passed 54 of 54 (the first script, 400 shots: 0.042 left, 0.920 after RUS) and 55 of 55 (2000 shots, before the bit-exact check: 0.0295, 0.9223).
- **Not done:** the COND checks through a jumper (an output to DIO1_P or DIO2_P: a branch following its level, and a state output carried to a branch): no such jumper was fitted; the latencies at the connectors with an oscilloscope (deferred to the end, the user's rule): THEORY 2.1 gives D + 28 clocks from a window's last sample at IN to a chosen pulse at OUT (512 ns nominal) and D + 5.5 clocks to a state output (332 ns nominal).

2026-11-07: 0.4, the emulator, experiment templates and fits (plan step 0.4)

- **`make test`** (the repository's): 1,355 tests passed, 22 skipped. spin's 55 tests passed. New:
  - the emulator's model against the float Bloch equations (`bloch.py`): Rabi and Ramsey, relaxation, the norm over 50,000 updates, an ensemble's FID and output; readout statistics; the generated files current (the rotation table's ROM, tb_emu's vectors);
  - `model.settings_from_writes` inverts the register writes of every bench scenario;
  - the driver's emulator settings (compile, derive, tables, the 3 MHz span check, the capture delay 89), `EMULATOR?` through the protocol and the client;
  - `test_experiments.py`: the fits on synthetic data (T2*, T1 by inversion and saturation, oscillations,

spectra with the CIC's droop, populations, readout fidelity against Gaussian blobs), every template compiled, and templates run through the driver and the bit-exact model with the emulator against the physics: an FID (T2*, offset and signal within 2 %, 1 %, 3 %), an echo refocusing a 20 kHz line (its top against the float Bloch equations within 2 %), inversion recovery (T1 within 8 %), nutation (2 %), a qubit's Rabi populations.

- **`make sim`:** `tb_core`, `tb_seq`, `tb_avg`, `tb_rec` and the new `tb_emu` passed.
  - `tb_emu` (new): the emulator alone against `emulator.py`, three scenarios (a qubit with its START at a slot's last clock, an ensemble with its START inside a slot and the drive across it, a qubit with a readout across its START), scripted pre-rolls (a pulse to an excited or transverse state and a readout before the START), structured segments (rotations then readouts from a slot's last clock, restarts, keeps, decays; zero, small, large (w_z clamped) and chirped detunings; small drives on resonance for the largest exponent; power-of-two amplitudes; g2 = 2^26; z0 = 4096 mod 8192). Compared: the output of every clock, and the vector and the readout's outcome after every update.
  - `tb_seq`: scenarios F (an ensemble FID and echo) and G (qubit readouts) from 0.4's start, and H (new): the board test's qubit run as the driver compiles it, with a RESULT per acquisition. Scenario E was stretched for the averager's clearing (below).
  - `tb_avg`: the clearing (below) replaces the epochs: a half reused a third time reads 0 at its old points, a request waits for a clearing half, the clearing waits beside points (a dirty half released while points arrive every clock reads 0 at all 2048 points).
- **The emulator's cut and its bound.** The ensemble's eight isochromats were cut to one Bloch vector (the plan's pre-approved cut), after build 0xd544c60b: 98.07 % slices, WNS −2.723 ns. The processor was rescheduled (one-hot phases, four updates in flight, write-back per component). A first structured vector set found the clamp case $\theta^2 >$ TH2_MAX overflowing the 25-bit q terms (w_z at its limit with a drive: q_zz $\approx 1.68 \cdot 10^7$). w_z is now clamped to ±0.4 turn per update (±3.125 MHz), which bounds θ^2 and leaves M, N and the q terms without clamps; the driver refuses TX1 frequencies more than 3 MHz from the spin's.
- **Mutation tests:** `sp_emu`, 55 one-token mutants (the generator, tables and draws, the drive's rounding and sums, the frame sums and remainders, the reset and its slot, the shot's detuning, AO0, w_z's clamp, the exponent's rules, the interpolation, a w, b w and the q terms' rounding, the M entries' signs and terms, the relaxation's rows, rounding and diagonal, the loop's v_t and rounding, d_z, v's clamp, the readout's threshold, decay, collapse, ring, phase sign, resonator, the epoch, the valid bit, the ensemble's output and its reset, the LO, the noise): 55 of 55 fail `tb_emu`. Equivalent mutants found on the way were removed from the RTL instead: the outcome ring's second bit, its reset, and the clamps the bound makes dead. The averager's clearing: 9 of 9 fail `tb_avg`; the 0.3 averager mutants that still apply fail `tb_seq` or `tb_avg`.
- **A defect from 0.3 found on the board: stale averager points.** Each half carried a 1-bit epoch, so a point written two activations earlier and not since counted again. The 0.3 tests reused one region; the templates' regions moved between runs, and the board's 0.3 receiver check read a tone 40 % high. Now a released half is cleared (zeros, at least 2048 clocks, waiting in clocks with points) before a swap can make it active, the epochs are gone, and a request without a run swaps once the other half is free. `tb_avg` fails on 0.3's averager at "the first time's point reads 0: (21, 27)".
- **A synthesis pitfall (Vivado 2025.1):** the first clearing wrote through the read port (a true dual port): synthesis kept its write enable but dropped the clearing's address from the port's address multiplexer (checked in the synthesized and routed netlists), so the board never cleared while the simulation did. The clearing now writes through the write port (address `clr_a` or the point's, data from the sum registers held at 0 without a point); the routed netlist was checked (`clr_a` in every averager BRAM's address cone).
- **Builds:**
  - 0x127f058b (one vector): WNS +0.125 ns, 13,623 LUTs, 4,210 slices (95.7 %), 47.5 BRAM36, 71 DSP48; `verify_build.py`'s fresh copy rebuilt the same ID.
  - 0x854b5fee (the clearing through the read port, not synthesized as written): WNS +0.144 ns, 4,310 slices (97.95 %).
  - **0x3be7f83d** (final):
    - Timing: WNS +0.181 ns after post-route optimization, TNS 0, WHS +0.011 ns.
    - Resources: 13,642 LUTs (77.5 %), 4,144 slices (94.2 %), 47.5 BRAM36, 71 DSP48.
    - Routed DRC: warnings only (DSP pipelining, one net without routable loads).
  - `verify_build.py`: 8 of 8 checks passed; a fresh copy from the recipe rebuilt 0x3be7f83d (and the saved project's regeneration the same ID).
- **Board** (0x3be7f83d; OUT1 → IN1, OUT2 → IN2, no pin jumpers; receipts `docs/data/board-0.4-test_hw-1.json`, `-2.json`): **49 of 49 checks passed twice.** The 0.1–0.3 checks passed unchanged. New (the emulator through the receivers, no wiring):
  - Bit-exact: a seeded ensemble FID (noise, a 30 kHz detuning drawn per shot, the 2-step cycle, 2 scans) and a seeded qubit Rabi (3 lengths and the two references, 2 scans): the averager's raw sums equal the model run on the board from the run's START clock. (The emulator's 128 ns slots follow the core's clock,

so the model must start where the board started; the bench always starts at the same phase.)

- Free-running: an FID's T2* 7.999–8.000 µs (8), offset 50.00 kHz, 0.1006–0.1008 V (0.1); an echo over 256 shots of a 20 kHz Gaussian line, its top 0.03576–0.03606 V against 0.03601 V from the float Bloch equations averaged over the line (with T1 = 6 µs the dephased spins relax while the 180° pulse carries them through z); CPMG's T2 40.04 µs (40); inversion recovery's T1 10.00–10.08 µs (10); the nutation's Rabi frequency 1.0034–1.0054 MHz (1); the qubit's Rabi frequency 1.002–1.006 MHz, Ramsey fringes 498.6–499.1 kHz (500), T1 19.58–20.73 µs (20, 600 shots per point); single-shot readout over 2000 + 2000 RESULT records: fidelity 0.9858–0.9870, separation SNR 7.8 (T1 = 20 µs acts during the 0.5 µs readout).
- The way there: the first run (0x127f058b) failed 9 checks: the stale averager (above), an echo reference with the 180° pulse 16 clocks early, a seeded T1 realization 3 σ off and a blob threshold set without a reason. With the clearing built (0x3be7f83d) the bit-exact checks failed until the model started at the board's START clock, and a long T1 run read 39.75 µs: the board's service, running spin too, collected the averager's halves and restarted the ring writer during the test. `test_hw.py` now parks the service on analog_echo while it runs and loads spin again afterwards; the service had once crashed (SIGSEGV) after both processes drove the one ring writer.

2026-11-07: 0.3, receivers, averaging, weighted results, records (plan step 0.3)

- **`make test`** (the repository's): 1,337 tests passed, 22 skipped. spin's 37 tests passed. New model tests:
  - the CIC's points equal the triangle filter over 2R − 1 samples;
  - a receiver reads a tone at F_CLK/16 in volts by THEORY 2.4's formula;
  - receiver phases step exactly, and the halves swap at SHOTs;
  - a window is not taken while its receiver is busy;
  - instructions with windows and receiver phases compile and decode back.

  New driver tests:
  - windows, weights and thresholds compile (weights normalised, rotated by the phase, packed per receiver) and derive back;
  - the host sums and `window_average` in volts from a fake core;
  - invalid windows write nothing.
- **`make sim`:** `tb_core`, `tb_seq` and the new `tb_avg` and `tb_rec` passed.
  - `tb_seq`'s new scenario E covers:
    - windows on both receivers (IN1; IN1 + i IN2), both LOs, raw mode, R of 1, 2, 3, 16 and 100;
    - a region across the half's end from a table, and weights across their end;
    - receiver phases from instructions and a table;
    - seven scans with swaps after two:
      - the first half freezes at once;
      - the bench releases it during the run, and the next swap goes back into it while it still holds scans 1–2 of the old epoch (they count as 0);
      - a window of the old half is still running at that swap, so the old half freezes at its last sample;
      - a third swap is refused (the half is not released);
    - a window not taken (receiver busy) and one refused at e + 3 (in the gap);
    - a RESULT exactly at its threshold (state 0);
    - the converters' limit codes, one on a window's last sample (overload);
    - every 5th/7th sample an input of exactly 8192 (the mixer's rounding ties).

    `vectors.py` asserts these boundary cases. Besides the per-clock DAC, pin and slow-output words, the bench compares the records per source (times relative to the run's start) and both frozen halves word by word (with a release and a swap between). The bench now reads its ADC words from `sp_adc.mem`, written by `vectors.py`: it had its own copy of the formula, which a new limit code left behind once.
  - `tb_avg` (the averager alone, directed):
    - 35-bit saturation of I and Q, both signs and banks, after 32,769 full-scale adds;
    - each BLOCK record's saturation flag, cleared when a half becomes active again, and including the last point, which reaches the averager 16 clocks after the freeze;
    - points of an older epoch counting and reading as 0.
  - `tb_rec` (the records alone, directed):

    - records pending together leave as RUN, ERROR, BLOCK, RESULT of RX1, of RX2;
    - a replaced RESULT and a receiver's lost result are counted, also two losses in one clock;
    - the last ERROR record 4 carries REC_DROPS's total.
  - `tb_core` reads back every word of the register files (FREQ, PHASE, AMP and CHIRP of both TX, COUNT, AO_REG).

- **A defect found by `tb_rec`** (in sp_rec): a loss in the same clock that an ERROR record 4 took the count was counted in REC_DROPS but not reported by any later ERROR record, and two losses in one clock counted as one. Now every loss counts and a new loss keeps the next ERROR record pending (THEORY 2.9). Under the old logic the bench's last ERROR record reports 1 lost against REC_DROPS 2.

- **Mutation tests:** 38 one-token mutants of 0.3's RTL, against `tb_seq` (and `tb_avg` or `tb_rec` for the averager's and the records' mutants).
  - Targets: the mixer (rounding, sign, the LO's cosine offset); the CIC (scale, comb start and delay, first capture); the receiver phase, threshold, overload level and its accumulation; the window gap; the MAC (clear, complex sum); the freeze at the last sample; the averager (epochs, saturation of I, of Q and its flag, the flag's clearing, N scans, the free-half rule, a too short BLOCK delay, the BLOCK's saturation, the scan count at a swap, stale points reading 0); the records (RESULT and ERROR lengths, RESULT time, priority, the missed count, a lost RESULT counted, losses added); the event and LO delays; the register files (rebase address, TX1's chirps, TX2's phases, the AO registers); WIN_MISSED.
  - An early partial pass survived the mixer's rounding (no exact ties): the inputs of 8192 closed it. A first full pass (36 mutants, `tb_seq` alone, the earlier scenario E) then left 13 survivors:
    - the threshold (no result at it), the overload level and accumulation (limit codes only in raw windows), the gap (the refused window was at e + 2);
    - the freeze at the last sample and the epochs (no half reused, no window across a swap), the stale reads;
    - the averager's saturation of I, the BLOCK's saturation, the flag's clearing and the BLOCK delay (35-bit sums need 32,768 full-scale adds: out of reach of `tb_seq`; the delay 31 → 30 is equivalent, so the mutant is now 31 → 15, shorter than the points' pipeline);
    - the records' priority and loss counting (per-source comparison; no losses).

    The vector changes above and the two unit benches answer them.
  - **Final pass on the final vectors and benches:** 37 of 38 failed a bench.
    - The survivor is equivalent: the BLOCK record's `| sat_set` term. That term would matter only for a point reaching the averager 32 clocks after the freeze; the last point arrives about 19 clocks after it.
    - The priority, loss-counting and losses-added mutants pass `tb_seq` and fail `tb_rec`.
    - The saturation mutants (I, Q flag, too short BLOCK delay) fail `tb_avg`.

- **Builds:**
  - 0x56187d29: WNS −0.983 ns. Fixes: the window event's delay tap registered; the averager's forwarding removed (instead a receiver keeps 4 samples between windows: the gap rule, THEORY 2.4); the shape descriptors in LUT RAM.
  - 0x479def90: WNS +0.002 ns (below the +0.10 cap), 11,652 LUTs (66.2 %), 12,795 registers, 4,142 slices (94.1 %). Fixes:
    - the averager's N-scans comparison a clock early (exact: SHOTs are at least 2 clocks apart);
    - its freeze from a receiver's last sample without the window-start logic;
    - the named registers' read multiplexer in two halves;
    - the indexed register files (FREQ, CHIRP, PHASE, AMP, COUNT, AO_REG: about 2,300 registers and their multiplexers) in LUT RAM.
  - Vivado 2025.1's synthesis then crashed twice (a segfault in hierarchical graph building). It was deterministic and reproduced out of context. The trigger: `sp_seq`'s outputs addressed register files in the core, whose data went back into `sp_seq`. Moving PHASE, AMP, COUNT and AO_REG into `sp_seq` avoided it.
  - 0x038f5100: WNS +0.204 ns, 10,091 LUTs (57.3 %), 3,421 slices (77.8 %); `verify_build.py` 8 of 8 (a fresh copy rebuilt the same ID); 38 of 38 board checks twice. Then `tb_rec` found the loss-counting defect above.
  - **0x40977a76** (the fix):
    - Timing: WNS +0.279 ns after post-route optimization, TNS 0, WHS +0.009 ns.

    - Resources: 10,069 LUTs (57.2 %, 1,749 as memory), 10,288 registers, 3,347 slices (76.1 %), 43 BRAM36, 60 DSP48.
    - Routed DRC: warnings only (DSP pipelining, one net without routable loads).
  - `verify_build.py`: 8 of 8 checks passed; a fresh copy from the recipe rebuilt 0x40977a76.

- **Board** (0x40977a76; OUT1 → IN1, OUT2 → IN2, no pin jumpers; receipts `docs/data/board-0.3-test_hw-1.json`, `-2.json`): **38 of 38 checks passed twice.** The 0.1 and 0.2 checks passed unchanged. New:

  - A receiver (R = 8, 200 points of a 0.2 V tone at 7.8125 MHz, 9 scans averaged) reads 0.20941 and 0.20939 V; a fit of the capture of the same samples, 0.20940 and 0.20937 V. The points' phase spread is 0.15°.
  - CYCLOPS in I/Q mode: the phase-cycled TX1 adds up to 0.9993–0.9994 of its expected amplitude; an uncycled TX2 alone reads 6–14 µV ($\leq 7 \cdot 10^{-5}$ of TX1's).
  - Weighted results over the records: 0.2094 V (state 1, no overload); a 1.1 V pulse reads 1.123 V with the overload flag (the input clips).
  - 20,000 RESULT records at 100,000 per second (4-point windows every 10 µs): none lost.
  - Reading a frozen half (2 × 2048 points, 12,288 words) and adding it to the host sums takes 141–156 ms.
    - Of that, about 128 ms is the register read. A repeated read of one register costs 2.2 µs per word; `read_burst` adds about 8 µs per word for validating and converting the offset list in Python (common `native_read.py`).
    - First 307–317 ms; the compiled reads in the worker and numpy sums brought it down.
    - The halves swap only into a free half, so a slow host lengthens the averaging but loses nothing.
  - On 0x038f5100 (before the fix) the same 38 checks passed twice. Its first round had 37 of 38: the weighted check's own test was wrong in two ways.
    - RESULT records of the earlier checks came out of the ring writer's buffer when the test's worker restarted it. The check now keeps only the records from its run's RUN start, and the earlier checks run without records.
    - The 1.1 V pulse selected frequency register 1 but amplitude register 0.

- **Not tested yet** (deferred to the end of the design, the user's decision): the slow outputs' levels on E2, the gate pins against the RF blanking on a scope, the trigger checks with a jumper on this build, the outputs' AC response tables.

2026-11-07: 0.2, I/Q outputs, shapes, chirps, limits, the duty guard, slow outputs (plan step 0.2)

- **Common:** `rp_ao` (the slow outputs AO0..AO3: feedback's `fb_ao` with the value as its input) in `common/rtl/optional`, its bit-exact reference `rpfpga.ao`; documented in `common/README.md`.

- **`make test`:** 30 tests passed. New model tests:

  - an envelope equals the exact linear interpolation of its points (50 random descriptors, 400 clocks each);
  - in I/Q mode the outputs are the complex sum of both TX within 1.3 LSB of the floating-point cosines and sines (Q gains and phases);
  - a chirp steps the NCO's increment by exactly the slope per clock;
  - the guard's pulse length (the RF ends after L_max samples, the run 3 clocks after the trip, the gate pins from clock t + 5) and its bucket (the trip clock from the formula);
  - the limits clamp the outputs and the idle level;
  - the slow outputs' mean duty is v / 65536 within 2 periods' worth over 300 periods.

  New driver tests:

  - I/Q mode, shapes and chirps compile and derive back; an unchanged shape table is not rewritten, and a new one stops a running program first;
  - invalid shapes, chirps, modes, guards and AO levels write nothing;
  - DC levels and limits in volts; PROTECT locks OUT1's limits and TX1's guard, and SAFE keeps them;
  - the guard's registers and its state;
  - the slow outputs' levels, registers and instruction fields.

- **`make sim`:** `tb_core` and `tb_seq` passed. `tb_seq` now runs four scenarios from one command file and one expectation file. Each clock it compares the DAC words, the pin words and the slow outputs' PWM pins with the model. The START is aligned to W mod 256 = 232, because the PWM periods are absolute.

  - A and B: as at 0.1.
  - C (I/Q mode):

    - shapes: interpolated, continued with `keep`, across the table's end, ending inside their instruction, the slowest and fastest steps;
    - chirps: rising, falling, continued, while the TX is off;
    - Q gains and phases, offsets, saturation and a clamping limit;
    - slow outputs from registers and tables (also a value 0);
    - the guard's pulse length on TX1: error 2, the RF blanked, the gate pins closed.
  - D: the guard's bucket on TX2, tripping on a chosen clock (error 3).

- **A 0.1 bug found by scenario C:**
  - The symptom: two coherent frequency starts 2 clocks apart gave the first one the second's phase. Its phase-file reading was kept in a hold register that the second start overwrote before use.
  - Not found at 0.1: scenario A never had two coherent starts within 3 clocks.
  - Fix: a three-stage delay line. A mutant that shortens it fails the bench.

- **Mutation tests** of `tb_seq`: 35 one-token mutants of 0.1's and 0.2's RTL.
  - New mutants: the coherent read's delay, the Q rounding, the chirp's sign and register, the envelope's end, parity, phase scale, rectangular value, `keep` and start, the Q phase and gain rounding, the envelope's amplitude rounding, the guard's blanking, compare, charge and latency, the I/Q routing, the upper limit, the slow outputs' run values, the gate blanking, the fault's stop, TX2's shape bank.
  - First run: all but 3 failed the bench (as they should). The survivors were the Q rounding, the envelope's amplitude rounding and the bucket's charge per clock (65536 against 65535).
  - Vector changes that closed them: an amplitude giving A_Q = 2^15; a constant odd-amplitude shape played at A = 2^15; D's B_max one below the bucket at a chosen clock. All three then failed too.
  - **Final run on the final vectors:** all 35 failed the bench.

- **Builds:**
  - 0xf8d1c25c: WNS +0.022 ns after post-route optimization (−0.107 routed). The read multiplexer of 0.2's registers was 9–11 LUTs deep.
  - Fix: the reads take one clock more (four), with three partial multiplexers registered first.
  - **0xade0ed52** (WNS +0.064 ns) **was built from a stale Vivado project.**
    - The project had not been regenerated after `design_bd` gained the port `ao_o`, so the slow outputs' pins were unconnected. The constraints matched no ports, which gives only critical warnings and lets the build succeed.
    - A first round of board tests ran on it (34 of 34 passed; the AO check reads the hardware's value, not the pins).
    - `verify_build.py` caught it: its fresh copy built a different ID.
    - The project was regenerated (common/README.md: remove `spin.xpr`, `spin.srcs` and the generated directories, `make`).

- **Build 0xd6c2126b** (from the recipe; the same ID as `verify_build.py`'s fresh copy):
  - Timing: routed WNS **+0.198 ns**, TNS 0, WHS +0.033 ns.
  - Pins: AO0..AO3 on T10, T11, P15, U13.
  - Utilization: 8,293 LUTs (47.1 %), 9,735 flip-flops, 3,114 slices (70.8 %), 23 BRAM36, 20 DSP.
  - DRC: only RTSTAT-10 (and the DSP pipelining notes).
  - The slices are the scarce resource: 70.8 % at 0.2, with the receivers, the averager, the weighted integrators and the emulator still to come (THEORY, the plan's budget).
  - In 0xade0ed52's placement the common `rp_ring_writer`'s `avail` path was the worst (+0.064 ns); it is the path to watch.

- **`verify_build.py --design spin`** (an isolated source-only copy, under the memory cap): all 8 stages passed; the regenerated build has the ID 0xd6c2126b.

- **Board test** (`make test-hw` twice, 2026-11-07, board 192.168.0.102; cables OUT1 → IN1 and OUT2 → IN2, no jumper; build 0xd6c2126b): **PASS, 34 of 34 checks both times**. Receipts are in `data/board-0.2-test_hw-{1,2}.json`; the round on 0xade0ed52 gave the same numbers.
  - **The 0.1 checks again:**
    - identification, the timer, the underflow, the ring;
    - the pulse's start, phase cycles;
    - a register's restored phase, a table frequency, t0 at a software trigger.
  - **A shape:**

    - a raised cosine of 65 points over 4 µs: the envelope at IN1 within **0.8–0.9 %** of its peak over 891 samples;
    - phase points of a 720° ramp: **500.0–500.1 kHz** for 0.5 MHz.
  - **A chirp** of 5 MHz in 10 µs: **4.999·10^11 Hz/s** for 5·10^11.
  - **The duty guard:**
    - a 5 µs pulse with max_pulse 2 µs: cut after **250** samples of RF, error TX1, the length check flagged;
    - a 50 % duty with max_duty 0.25 and a 4 µs burst: cut after **500** samples, the bucket flagged.
  - **OUT1's limits** (−0.2 … +0.1 V) clamp a 0.3 V tone: −0.206 … +0.105 V at IN1 (through the loop's gain).
  - **The DC level** 0.1 V: 0.0969 V at IN1 (the DC loop gain, 0.965).
  - **AO0:** 1.2000 V (its register) during a run, 0.2000 V (idle) after it. This is the hardware's value; the PWM's analog level needs a meter on E2 and is unmeasured.
  - **I/Q:**
    - OUT2 is OUT1's quadrature: −89.82°, Q/I 1.003 through both loops;
    - q_gain 0.7 and q_phase 20° change the ratio by ×0.7002–0.7003 and the phase by 19.97–19.98°;
    - TX1 at +3.906 MHz and TX2 at −6.25 MHz multiplexed: 0.206 V and 0.156 V (0.2 and 0.15 set, through the loops' gains), images at **−54 dB** and **−46.5 dB** with no correction.
  - **The first round's one failure was the test's:** the multiplexing check kept the previous check's Q gain and phase (partial applies merge). It now sets them.

- **Not done at 0.2:**
  - the slow outputs' analog levels (a meter or scope on E2);
  - the gate pins' closing against the blanked RF on the oscilloscope;
  - the 0.1 trigger checks with a jumper on this build (no jumper fitted; the trigger path is unchanged).

2026-11-07: 0.1 on the wired bench (the loopback and trigger checks)

Wiring: a coax OUT1 → IN1 (no terminator, IN1 on LV) and a jumper DIO7_N → DIO0_P (TRIG). The test found both by itself. Same build, 0xd799d563.

- **First two runs failed in the test script, not in the design:**
  - Run 1 stopped at a Python format error in a detail text (`"50 % reached"`).
  - Run 2 failed one check, "the edge's time steps 2 ns per 2 ns of delay". The steps read about 10^8 ns because the check subtracted TRIG times counted from the core's reset, across separate runs.
  - Fix: `state()` now also gives `trigger_after_start_s`, the edge's time from its run's start, exact in nanoseconds (`docs/PROTOCOL.md`). The test compares those whole nanoseconds. `make test` still passes its 19 tests, with the new field checked.
- **Board test** (`make test-hw` twice, after `make upload`, 2026-11-07): PASS, 27 of 27 checks both times. Receipts are in `data/board-0.1-wired-test_hw-{1,2}.json`. The 17 checks of 2026-11-06 passed again, plus these:
  - **Pulse start at the connector:** the IN1 signal reaches 50 % at capture sample 51, for an instruction at sample 50. The calibration's input delay is 119.9 ns; the residual not taken in whole clocks is −0.09 ns.
  - **Amplitude:** 0.4305 V read for 0.4 V set at 12.5 MHz (check: within 10 %). This is the board's loop response, not spin; see the sweep below.
  - **Coherence:** frequency register 0's phase was the same (134.23°, 134.26°) before and after 512 ns at another frequency, within 0.01°.
  - **Phase cycle 0/90/180/270° from a table:** worst error 0.06° and 0.05°.
  - **A table frequency continues the phase:** 132.98° at the switch against 134.28° expected. That is 1.3° at a 1 MHz step; the check allows 2°.
  - **t0 at a software trigger:** the RF phase repeated within 0.25° and 0.28° over six runs. Without the new reference the phases spread over 147–173°.
  - **IN1 noise** with OUT1 idle on the cable: 1.38–1.52 mV rms (a note, not a check).
  - **D8 (DIO7_N) → TRIG:**
    - high, low and released low, as expected;
    - a 125 MHz pattern gives edges;
    - a WAIT on TRIG that raises the TTL bit itself sees the edge at 389, 391, 393, 395 and 397 ns from the run's start for output delays 0, 2, 4, 6 and 8 ns. These are the expected 2 ns steps, in both runs;
    - with t0 at the edge, the RF phase follows the edge's slot within 0.02° and 0.05° (slots 5, 7, 1, 3, 5).
- **Loop response** (an extra sweep through spin's capture, receipt `data/board-0.1-loop-response.json`): IN1's amplitude over OUT1's set amplitude.
  - Settled DC (phase 0° and 180° at frequency 0): **0.9646**, steady from 100 µs to 1 s.
  - Higher frequencies:

Frequency	Ratio
20 kHz	0.996
100 kHz	1.025
0.25–2 MHz	1.026–1.030
5 MHz	1.034
10 MHz	1.062
12.5 MHz	1.077
20 MHz	1.106
30 MHz	1.051
40 MHz	1.025–1.044

  - The ratios are the same at 0.1 V and at 0.4 V.
  - This matches what lockin and awg measured on this board with the same calibration:
    * lockin: 0.960 at 1 kHz, 1.022 at 1 MHz, 1.056 at 10 MHz, 1.102 at 20 MHz;
    * awg: DC 3.6 % low.

    So spin's volts agree with the framework. The deviation is the board's analog path (a step of about 6 % between DC and 100 kHz, then the cable and the input's load), which the DC calibration does not correct. Response tables would correct it, but they need the output's and the input's shares measured separately.
- **A pin against OUT1 on the oscilloscope** (Siglent SDS2504X Plus; receipt `data/board-0.1-scope-pins.json`).
  - **Setup:**
    * OUT1 by a 0.5 m coax into a BNC T on CH3, at 50 Ω.
    * CH2's 10× probe on DIO6_N (D7, source TTL bit 0), its ground on E1 GND.
    * OUT1 stepped from −0.4 V to +0.4 V on one sample every 512 ns, and the TTL bit rose on the same sample.
  - **Measured:** the pin's 1.65 V crossing minus OUT1's 0 V crossing, 40 acquisitions per delay setting, 0.16–0.34 ns rms.
  - **The probe's delay:** +6.63 ns (sd 0.18 ns, 60 acquisitions), measured with the probe on the T's free arm. The 2026-11-05 session measured 7.04 ns for this probe.
  - **At the connectors** (minus the probe's delay, plus about 2.5 ns for the cable): with the pin's delay at 0 ns, D7 changes **1.5 ns before OUT1**.
    * Expected: 0.6 ns before. spin compensates the board's `pins_delay_s` (−28.6 ns) in 2 ns steps (14 steps), so that much is left over if the simulated `PIN_ALIGN` is right.
    * The 0.9 ns difference is within this setup's errors: the cable's type (unknown), the probe's ground, the skews between pins, and the board calibration's own measurement (awg's, with another scope, probe and pin).
    * `PIN_ALIGN` stays at the simulated value. A board correction would belong in `pins_delay_s`, not in the design.
  - **The output's delay setting:** 0, 2, 4, 6, 8 ns moved the edge by 1.88, 1.95, 2.07 and 2.05 ns per step.
- **An asynchronous trigger from awg on a second board** (awg 3.0 on 192.168.0.104).
  - **Wiring:**
    * awg's marker 1 (its DIO1_P; 96 ns every 1 ms from timer 1) → spin's DIO0_P (TRIG), a Dupont wire, the grounds joined; the jumper DIO7_N → DIO0_P removed;
    * awg's marker 2 (DIO2_P, the same source) on CH2's probe;
    * the boards' crystals run free, about 5 ppm apart, so the edges fall anywhere in spin's clock.
  - **Scope** (OUT1 into the T on CH3; receipt `data/board-0.1-scope-async-trigger.json`): a loop of WAIT on TRIG, then a 3.2 µs pulse of 0.4 V at 30 MHz. The carrier's phase was fitted over 400 ns of each pulse and timed from the edge (CH2 at 1.65 V), in two runs of 100 acquisitions per mode.
    * **t0 moved to the edge:** the carrier holds within **sd 0.28 / 0.27 ns, 0.99 / 1.03 ns peak to peak** of the edge. That is the 1 ns slot: an edge uniform within the slot gives sd 0.29 ns.
    * **t0 not moved:** sd 9.3 / 9.2 ns and 32 / 33 ns peak to peak, the whole 33 ns period.
    * **The pulse's start** against the edge, measured with a DC pulse of 0.4 V (frequency 0, its 0.2 V crossing, 200 acquisitions): 380.7–388.5 ns on the scope, about uniform over 7.8 ns (sd 2.11 ns). The start moves with spin's 8 ns clock, as designed.
    * At the connectors (minus the probe's 6.63 ns, plus the cable's 2.5 ns) that is **377–384 ns from TRIG to OUT1**. awg's DIO1_P/DIO2_P skew and the 20 cm wire are not corrected (≤ about 1 ns).
    * A start found from the carrier (half amplitude, or half the envelope) spread 14–18 ns. It depends on the carrier's phase at the start, so it does not measure the start.
  - **Board test with awg on TRIG** (`make test-hw` twice; receipts `data/board-0.1-awg-trigger-test_hw-{1,2}.json`): PASS, 24 of 24 checks both times. With no jumper, the 3 jumper checks do not run.

    - **The external-trigger check** took 30 runs per mode. Every slot 0–7 appeared. With t0 at the edge, each run's captured phase, corrected by its slot from the TRIG time, agreed within ±0.14° and ±0.18° (limit 1°). Before the correction the phases spread ±39–41°; without the new reference, ±173–175°.
    - **The check as first written** expected ±6.4° (half a 1 ns slot at 30 MHz) without the slot correction, and failed twice at ±38.6° and ±38.2°. The capture starts on the clock, so it sees the edge-locked RF at −360°·f·j ns for the edge's slot j, the relation the jumper check had confirmed. The check now corrects each run by its measured slot (`scripts/test_hw.py`).
    - **What the check proves:** that the phase given to the NCO is the time-stamped slot, over all eight slots. That the slot is the edge's own is shown by the jumper check (2 ns steps) and by the scope above.
  - awg's previous configuration was restored afterwards.

2026-11-06: 0.1, the pulse program, real transmitters, pins, capture (plan step 0.1)

- **Common:** `rp_sine`/`rp_sine_rom` moved to `common/rtl/optional`, generated by `rpfpga.sine` (the bit-exact reference). `common/tests/test_sine.py` passed: the file is current, the error is ≤ 1.04 LSB (0.56 before the final rounding), and it equals lockin's model for 20,000 random phases.
- **`make test`:** 19 tests passed. Model tests:
  - the vectors are current;
  - the phase reference matches exact rational arithmetic (with the 1 ns fraction);
  - the DAC codes equal the floating-point cosine within 0.65 LSB;
  - a coherent return to a frequency restores its phase;
  - the underflow threshold;
  - a WAIT on TRIG with t0 at the edge;
  - 200 random programs round-trip through the compiler.

  Driver, protocol and client tests: outputs and gates, transmitters in volts, program and tables (written only when their CRC changes), invalid settings, safe, run control, the TRIG time, the capture in volts.
- **`make sim`:** `tb_sysid`, `tb_core` and `tb_seq` passed. `tb_seq` compares every clock's DAC words and pin words with the model, in two scenarios:
  - Scenario A: 61 instruction starts: nested loops, tables (one loop's counts from a table indexed by its own counter), coherent and table frequencies, the timer, TRIG edges taken and missed, saturation, every pin source.
  - Scenario B: an underflow.

  It then checks the counters, the status, 300 captured words, the TRIG time, and a software trigger's 24 clocks.
- **Mutation tests** of `tb_seq` (one token each): 12 of 13 failed the bench, as they should:
  - the phase constant, the trigger's latency and its word rule, the rebase factor, the phase file's read, the rounding;
  - a pin slot, the gate span, the capture delay;
  - the loop compare, the skip test, the next address.

  The rounding, the trigger word rule and the skip test failed only after the scenario gained amplitudes of 2^15, an edge on the boundary and a table indexed by its own loop.

  The survivor, a fresh LOOP's table index after a loop is left mid-iteration, is reachable only with BRANCH: it will be covered at 0.5.
- **Builds:** the first three failed timing. The engine's decode straight from the block RAM gave WNS −1.88 and then −1.17 ns, and the AXI write data's fanout and the ring writer's `avail` path also failed. The engine is now F1 (the instruction, registered), F2 (the tables' addresses), F3 (the resolution, registered), P (the push and the next address); it is still 4 clocks per instruction, clock-equivalent to the model. The AXI writes got a replicated register stage, `fresh` is decided in F1, and post-route physical optimization is on.
- **Build 0xd799d563:** routed WNS +0.469 ns, TNS 0, WHS +0.033 ns, THS 0; DRC only RTSTAT-10. Utilization: 5,832 LUTs (33.1 %), 6,393 flip-flops, 2,163 slices (49.2 %), 18 BRAM36, 8 DSP, the common MMCM plus one PLL.
- **Board test** (`make test-hw` twice, 2026-11-06, board 192.168.0.102): PASS, 17 of 17 checks both times. Receipts are in `data/board-0.1-test_hw-{1,2}.json`.

  - Identification and no alias in the window.
  - The program and the tables read back as compiled, and the program is derived from the block RAMs.
  - A WAIT on software waits, and a software trigger continues it; STOP ends a run.
  - 1000 WAITs on a 10 µs timer: 1000 shots, 1 missed tick (the one at the first instruction).
  - An underflow stops the run with error 1.
  - The ring through HP0: about 502,000 test words in order, none dropped.
  - SAFE released every output.

  Not run: the bench had no cable OUT1 → IN1 (IN1 read 0.1 mV; IN2 carried the 10 MHz GPSDO sine) and no jumper to TRIG. So the analog loopback checks and the TRIG and phase checks did not run: pulse timing at the connector, volts, phase cycles, coherence, table frequencies, t0 at a trigger, and the TTL → TRIG 1 ns steps. They ran on 2026-11-07, with the wiring (above).

- **`verify_build.py --design spin`** (an isolated source-only copy, under the memory cap): all 8 stages passed; the regenerated build has the same ID, 0xd799d563.

2026-11-06: 0.0, the skeleton (plan step 0)

- **`make test`**: 8 tests passed (driver round trip, invalid settings write nothing, safe and presets, pins and hardware readings, protocol, client over localhost, loading through the fake PYNQ).
- **`make sim`**: `tb_sysid` and `tb_core` passed:
  - the registers, and the 256 KiB window's decoding (no alias at 0x10010; unused offsets read 0);
  - the behavioural 1 ns sampler: a 3 ns and a 2 ns pulse seen, a held level, clearing;
  - the real OSERDESE2/OBUFT models: 2 ns bits on two enabled outputs, high impedance on the others and after disabling;
  - the ring writer's test words, in order, through an HP0 responder, with the written position matching the bursts.
- **Build 0x66db145b** (2026-11-06, under a 4.5 GB memory cap; peak 2.1 GB).
  - Placement: the 3 ISERDESE2 on DIO0..2_P and the 8 OSERDESE2 with tristate on DIO0..7_N, the P and N pins of the pairs DIO0..2 split between input and output (3 ILOGIC, 26 OLOGIC with the converters').
  - Pull-downs: on all the pins, including the tristate outputs (the I/O report).
  - Routed: WNS +0.452 ns, TNS 0, WHS +0.024 ns, THS 0.
  - DRC: only RTSTAT-10, the SmartConnect nets without loads that every design has.
  - Utilization: 1,343 LUTs (7.6 %), 2,001 flip-flops, 661 slices (15.0 %), 4 BRAM36 (the ring writer's queue), 0 DSP, the common MMCM plus one PLL (the samplers').
- **Board test** (`make test-hw` twice, 2026-11-06, board 192.168.0.102, no wiring): PASS, 12 of 12 checks both times. Receipts are in `data/board-0.0-test_hw-{1,2}.json`.
  - Identification: sysid SPIN 0.0, the build ID matching the overlay, ABI 1, the samplers' PLL locked.
  - No alias in the 256 KiB window: 0x10010 and 0x3FFFC read 0, and a write at 0x10010 leaves OUT_EN unchanged.
  - The inputs are low when open.
  - The ring through S_AXI_HP0 into DDR through the common ring worker's Python path: test words at 1 Mword/s for 0.5 s (501,957 and 502,002 words), every word in order, the second run continuing the first's count exactly. No word was dropped, there was no watchdog trip or restart, and the written position matched (2,007,936 bytes).
  - SAFE released every output.
  - The wired output checks were skipped: no jumper was fitted. They run when a jumper (e.g. DIO7_N → DIO1_P) is present; the high-impedance state was tested only in simulation.
- **`verify_build.py --design spin`** (2026-11-06, an isolated source-only copy, under the memory cap): all 8 stages passed (recipe build, unchanged, missing export, vendor HDL edit, the design's RTL edit, a failed build blocks the upload, clean, regeneration from the saved project); the regenerated build has the same ID, 0x66db145b.