awg 3.0: reference documentation

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

Rok Zitko, Jozef Stefan Institute

2026-11-10 (commit 6178370)

Contents

awg: a two-channel function and arbitrary waveform generator **2**
- Before connecting equipment . . . 3
- Quick start . . . 3
- Presets . . . 4
- Specifications . . . 4
- Hardware setup . . . 5
- Documentation . . . 6
- Changing the design . . . 6
- Source layout . . . 6
- Verification . . . 7
- Author . . . 7

awg recipes **7**
- A pulse with slow edges . . . 7
- Three-cycle bursts on an external trigger, with a delay . . . 7
- A gated tone . . . 7
- Triggering on an analog level . . . 8
- A swept sine with a marker . . . 8
- Noise added to a signal . . . 8
- AM by an input, FM by generator 2 . . . 8
- FSK keyed by a pin . . . 8
- An I/Q pair . . . 8
- Your own table . . . 8
- Protecting a sensitive input . . . 8
- An arbitrary waveform . . . 9
- Two waveforms at once, and I/Q . . . 9
- A triggered sequence with a jump . . . 9
- A clock and a frame on the digital outputs . . . 9
- A trigger pattern with every burst . . . 9
- Patterns that follow a sequence . . . 10

awg: theory of operation **10**
- Datapath . . . 10
- Shapes . . . 10
- Noise . . . 11
- Modulation . . . 11
- Sweeps . . . 11
- Bursts, gates and trigger timing . . . 12
- Arbitrary waveforms . . . 12
- Apply . . . 13
- Markers . . . 13
- Digital outputs . . . 14
- Limits . . . 14
- Resources . . . 14
- Roadmap . . . 15

awg: settings, Python methods, TCP commands, command line **15**
The configuration . . . 15
Python (driver on the board, `AwgClient` on a PC) . . . 17
TCP commands (port 5025) . . . 18
Command line . . . 18

awg registers **19**
Map . . . 19
Control and status (0x000–0x0FF) . . . 19
Staged words . . . 20
Sweep segment words . . . 22
Sequencer step words . . . 22
Pattern programs . . . 22

awg verification records **23**
2026-11-09: an asynchronous laser with a chopper for boxcar . . . 23
2026-11-06: analog pulses and digital marks into counter . . . 23
2026-11-05 (evening): awg as a source of known events for counter, timetagger and feedback . . . 23
2026-11-05: digital outputs tagged by timetagger on a second board . . . 23
2026-10-24: driver fixes from a code review (board check, `make test-hw`) . . . 24
2026-10-24: loopback measurements (`make test-hw`, 57 checks) . . . 24
2026-10-24: trigger-to-output latency (oscilloscope) . . . 25
2026-10-24: pins delay against the analog output (oscilloscope) . . . 25
2026-10-24: awg 3.0 on the board (build 0x97cda2f3) . . . 25
2026-10-23: awg 3.0, simulation and offline tests . . . 26
2026-10-23: awg 2.0 on the board (build 0x17e4e55a) . . . 26
2026-10-23: awg 2.0, simulation and offline tests . . . 28
2026-10-23: awg 1.0 on the board (build 0xb088a0b4) . . . 28
2026-10-23: awg 1.0, simulation and offline tests . . . 29

awg: a two-channel function and arbitrary waveform generator

awg turns the Red Pitaya into a **two-channel function, pulse and arbitrary waveform generator** in the style of a bench instrument. Each output, OUT1 and OUT2, has

- a **main waveform**: sine, square (duty cycle), pulse (width), ramp/triangle (symmetry), PRBS (7 to 31 bits), DC, or a **table** (built-in Gaussian, Lorentzian, sinc, exponential rise/fall, haversine, or your own array). Square and pulse have **independently adjustable rise and fall times** (10–90 %; PRBS one time for both edges) from one sample (6.4 ns) up, and the edges of square, pulse, ramp and PRBS move in **sub-sample steps**: no 8 ns jitter when the period is not a whole number of samples;
- a **second generator** with any of these shapes, **added** to the output (two-tone and composite signals) and serving as the **internal modulation source**;
- **Gaussian noise** (to about 7σ) added to the waveform, with an adjustable **flat bandwidth** from 1 Hz to 47.7 MHz;
- **modulation**: AM or double-sideband suppressed-carrier AM, together with one of FM, PM, PWM, FSK or BPSK, by the second generator, by **IN1 or IN2** (one input per output or both by one), or by **the other output**;
- **sweeps**: linear, logarithmic, list (up to 512 frequencies) or a profile, with hold and return times and a **marker at a chosen frequency**, continuous or triggered;
- **bursts** of N cycles from a **start phase to a stop phase**, triggered, or **gated**; between bursts the output holds the start value, the last value or a DC level;
- **triggers** from the **TRIG A/B pins with 1 ns timestamps**, EVENT 0–3 pins, IN1/IN2 with an **adjustable threshold** and hysteresis, software, **two timers**, or the other output's start; any edge; a **delay** from 0 to days in 1/256-sample steps (31 ps);
- **hardware voltage limits** per output, lockable until the FPGA is reloaded, and optional **dither**;
- **true arbitrary waveforms** (2.0): segments of up to 16 M points per output in the board's memory, every point played once at a **fractional point rate** up to 125 MSa/s with **smooth (band-limited), linear or step** reconstruction, in **sequences** of steps (repeats, waits for a trigger, goto, and **gapless jumps** on events at the end of a repetition or at once), started with **sub-sample timing**; the points come in OUT1/OUT2 pairs from one sequencer: **I/Q** at baseband (I on OUT1, Q on OUT2) or **upconverted** to a carrier (I cos − Q sin, with gain imbalance, quadrature error and offsets), with two marker bits per point;

- **eight digital outputs** (3.0), D0–D7 on DIO0_N–DIO7_N: saved **patterns** of up to 32768 words, **binary or Gray counts** and **PRBS** (7 to 31 bits, on D0 with delayed copies on D1–D7), at 125 MHz divided by an integer, for a number of passes or forever; started at an apply, at each burst/sweep/sequence start of an output, or by **sequence steps** (start or stop a program on the step's first point), **aligned with the analog outputs** to the sample and delayable in 8 ns steps; each pin is driven only when enabled (otherwise high impedance).

Both outputs share one time base. Phases are relative to **one zero time t0**, and a configuration change of both outputs lands on **one sample**: equal frequencies on OUT1 and OUT2 stay locked with the set phase difference, and returning to a frequency restores its phase. Two **marker outputs** show the waveform's sync, the burst or sweep state, the sweep marker or a start pulse, with an adjustable delay relative to the analog output.

Volts are **real volts at the SMA connectors** through the **board calibration** shared by every design, which awg can also measure (../common/CALIBRATION.md). The inputs record 8192 samples on the same time base for loopback checks.

awg 3.0 is the function and arbitrary waveform generator with eight synchronized digital outputs.

Before connecting equipment

Read the repository's Connecting to lab equipment: read first section. For awg in particular:

- **Loading awg leaves the outputs quiet**: everything off at 0 in converter units, a few mV from 0 V. After a reboot, the service restores the last configuration.
- **Peaks and limits.** By default the driver refuses settings whose worst-case peak (|offset| + the amplitudes, AM peaks, and 4σ of noise) leaves the DAC range (about ±1.13 V on this board). Set **voltage limits** (`limits: {"low": ..., "high": ...}`) to keep a load's input range: the FPGA clamps every sample to them, after all sums and modulation. `protect(n)` locks them until the FPGA is reloaded; then no configuration, preset, SAFE or calibration can widen them.
- **Load.** Amplitudes are set **into 50 Ω** (the outputs are 50 Ω sources); into 1 MΩ you get about twice. The offset is right into any load.
- **Extension-connector pins are 3.3 V LVCMOS**, not 5 V tolerant: DIO0_P (TRIG A), DIO3_P (TRIG B), DIO4_P–DIO7_P (EVENT 0–3) are inputs with pull-downs; DIO1_P and DIO2_P are the marker outputs: never drive them from outside. DIO0_N–DIO7_N are the digital outputs D0–D7: high impedance until enabled (`digital.enable`) and after SAFE; never connect a driving signal to a pin you enable.
- **Safe state:** `SAFE` (Python `awg.safe()`) mutes both outputs at once (to 0 V or the nearest protected limit), then switches everything off; the digital outputs become high impedance.

Quick start

The board is set up (../common/SETUP.md) and reachable at `192.168.0.149`. Connect a scope to OUT1 and OUT2 through 50 Ω feed-throughs.

- **Jupyter** on the board: open `http://192.168.0.149:9090` (password `xilinx`), then `awg/awg.ipynb`, and run the cells. The panel has three tabs: **Outputs** (function, shape, noise, burst, trigger, sweep, modulation and limits per output; software triggers, zero time, SAFE), **Presets** and **Calibration**.
- **Python on the board:**

```
import rpfpga                                   # must come first
from pynq.overlays.awg import DRIVER
awg = DRIVER()                                  # loads awg if the board runs something else
awg.set_output(1, function="square", frequency=1e6, amplitude=0.4, duty=25.0, rise=20e-9)
awg.set_output(2, function="sine", frequency=1e6, amplitude=0.4, phase=-90.0)
awg.set_output(1, burst={"mode": "triggered", "cycles": 3}, trigger={"source": "trig_a",
 ↪  "delay": 1e-6})
print(awg.describe())
```

- **Python on a PC**, through the board's control service (standard library only), from the repository root:

```
import sys; sys.path[:0] = ["common/python", "awg/control"]
from awg import AwgClient
with AwgClient("192.168.0.149") as awg:          # the board must run awg
    awg.function(1, "sine", 1e6, 0.4)
    awg.set_output(1, noise={"rms": 0.02, "bandwidth": 5e6},
                   am={"mode": "am", "source": "in1", "depth": 0.5})
    print(awg.hardware()["outputs"]["1"])
```

```
# an arbitrary waveform: a 1000-point chirp at 10 MSa/s on OUT2 (OUT1 has noise,
# which shares the resampler with the arbitrary points)
import math
awg.save_segment("chirp", [math.sin(2 * math.pi * 30 * (i / 1000) ** 2) for i in
↪ range(1000)])
awg.play(sequence="chirp", rate=10e6)
awg.set_arb(2, on=True, amplitude=0.3)
# the digital outputs: a 1 MHz binary count on D0..D7
awg.set_digital(run="continuous", program="count", enable=0xFF,
                programs={"count": {"mode": "binary", "rate": 1e6}})
```

If the board runs another design, switch with `rpfpga.client.connect("192.168.0.149").load("awg")`. Reloading invalidates existing handles; reconnect explicitly.

- **TCP** from anything: `nc 192.168.0.149 5025`, then for example `FUNC 1 square 1e6 0.4`, `OUT 1 {"duty": 25, "rise": 2e-8}`, `NOISE 2 0.05 1e6`, `TRIG 1`, `HW?`, `PLAY {"sequence": "chirp", "rate": 1e7}`, `ARB 1 {"on": true, "amplitude": 0.3}`, `DIGITAL {"run": "continuous", "program": "c", "enable": 255, "programs": {"c": {"mode": "gray", "rate": 1e6}}}`, `PRESET burst`, `SAFE`.
- **Command line:** `python3 -m rpfpga.cli --host 192.168.0.149 --func 1=sine,1e6,0.4` (with `common/python` and `awg/control` on `PYTHONPATH`).

docs/PROTOCOL.md lists every setting, method, TCP command and command-line option. RECIPES.md has worked examples.

Presets

Name	Outputs
`off`	everything off, 0 V (the default)
`sine`	OUT1: 1 MHz sine, 0.5 V
`square`	OUT1: 1 MHz square, 0.5 V, 50 %, 10 ns edges
`pulse`	OUT1: 100 kHz, 200 ns pulses from 0 V to 0.5 V
`iq`	OUT1 and OUT2: 1 MHz, 0.5 V, OUT2 90° behind
`noise`	OUT1: Gaussian noise, 0.1 V rms, 10 MHz
`am`	OUT1: 1 MHz carrier, 80 % AM by a 10 kHz sine
`sweep`	OUT1: 1 kHz to 1 MHz linear sweep in 10 ms, repeating
`burst`	OUT1: 5-cycle 1 MHz bursts every 1 ms (timer 1)

Specifications

Computed figures come from the bit-exact model; board-wide figures are in the root README.

Quantity	Value
Outputs	2, one 125 MSa/s time base, 14-bit DACs; changes of both outputs on one sample
Frequency	0 … 62.5 MHz in steps of 0.44 µHz (2^-48 of the sample clock); keep below about 50 MHz (images at 125 MHz ± f)
Phase	relative to t0, steps of 2^-32 cycle; bursts start at their start phase
Amplitude	24-bit words: steps of 2^-11 DAC step (about 0.07 µV) per generator; sine amplitudes corrected for the DAC's sinc droop (and the board's measured response if calibrated)
Offset	steps of 2^-16 DAC step; with dither, sub-step levels on average
Shapes	sine (spurs below −100 dBc before the 14-bit rounding, computed); square/pulse/ramp/triangle/PRBS edges from one sample (6.4 ns 10–90 %) up, edge timing in sub-sample steps (computed: the edge period error below 0.01 sample); tables of 4096 points, linearly interpolated
Noise	Gaussian to about 7.1σ (inverse-CDF table, error mostly below 0.4 LSB, at most 2.3, computed); flat to ±0.5 dB up to 0.31 of the point rate, −3 dB at the set bandwidth (0.382 of the point rate), below −60 dB from 1.6 times the bandwidth (computed); bandwidth 1 Hz … 47.7 MHz. Measured in loopback at 10 MHz: −3 dB at 10.1 MHz, 3–7 MHz within 0.24 dB of 0.5–3 MHz, −51 dB at 16–40 MHz (the input noise floor), kurtosis 3.003 and 0.272 % beyond 3σ (Gaussian: 3 and 0.270 %)

Quantity	Value
Modulation	AM/DSB to 100 % at the source's full scale; FM deviation up to the Nyquist frequency; PM to ±360°; PWM; FSK and BPSK keyed by generator 2 or a pin/threshold; sources: generator 2, IN1, IN2 (input sample n − 7 modulates output sample n), the other output (its sample n − 26). Measured in loopback (5 MHz carrier): FM 499.5 kHz for 500 kHz set, PM 90.0° for 90°, FSK 7.00001/4.99998 MHz for 7/5 MHz, BPSK 179.2° for 180°, FSK keyed by a pin
Sweeps	linear and logarithmic (exp2 table, relative error below 1e-7), list, profile; up to 512 segments; rates up to 0.95 kHz per sample (119 GHz/s) linear, 1/2048 octave per sample logarithmic; hold, return, marker. Measured (instantaneous frequency in loopback, 40 µs sweeps): linear rate within 0.015 % of the set rate (4.5 kHz rms from a line), logarithmic within 0.001 % (0.005 octave rms)
Bursts	1 … 2^32 − 1 cycles or endless; start and stop phase; gated (finish the cycle or stop); idle: start value, last value or DC
Arbitrary waveforms	one sequencer of OUT1/OUT2 point pairs (14 bits + 2 marker bits each), 64 … 2^25 − 1 pairs per segment, 16 M pairs in the default 64 MB buffer; pair rate 0.03 Sa/s … 125 MSa/s (2^-32 of the sample rate); per output: smooth (−3 dB at 0.382 of the rate, delay 6 points), linear (1 point) or step reconstruction, amplitude, offset; sequences of up to 512 steps (1024 in the table) with repeats (1 … 2^32 − 1 or until an event), waits for a trigger, goto/end, jumps on events (at a repetition end or at once, gapless, at least 16 samples after a step starts); I/Q upconversion to 0 … 50 MHz; DDR reads 500 MB/s at 125 MSa/s (computed), streamed without underflow, also while the CPUs copy memory (measured); uploads over Ethernet 8–12 MB/s (measured)
Triggers	TRIG A/B (DIO0_P, DIO3_P): 1 ns timestamps; EVENT 0–3 (DIO4_P–DIO7_P) and IN1/IN2 thresholds: 8 ns (an IN2 threshold on a 20 kHz square: every edge triggers, the burst follows it by a constant 58.56 samples in loopback, spread below 0.01 sample, measured); software; timers (period 16 ns … years, aligned to t0); the other output. Trigger to output: 377.9 ns at the connectors with delay 0 (TRIG A pin at 1.65 V to OUT1's edge, measured with an oscilloscope): 20 samples to the start, 20 through the core, about 58 ns of I/O, converter and analog path
Trigger jitter	1 ns quantization of TRIG A/B (0.29 ns rms if uniform, computed); a marker looped to TRIG A gave the same start in 5 of 5 reads (measured); jitter against an external trigger **not measured**
Markers	DIO1_P, DIO2_P, 3.3 V: sync, run, sweep marker, start pulse (width from 8 ns), the arbitrary points' two marker bits, each with a delay of −136 … +3944 ns plus the board's `pins_delay_s` (−165 … +3915 ns on this board) relative to the analog output in 8 ns steps (with this board's `pins_delay_s` the delay 0 lands 3.3 ns after OUT1's edge, measured)
Digital outputs	D0–D7 (DIO0_N–DIO7_N), 3.3 V, driven per pin or high impedance; one 8-bit word per sample from up to 8 programs: patterns (32768 bytes shared), binary and Gray counts, PRBS 7…31; 125 MHz / N words per second (N = 1 … 2^32), 1 … 2^32 words per pass, 1 … 2^32 − 1 passes or forever; ends on the last word or an idle word; starts at an apply, at each start of OUT1 or OUT2, or by sequence steps; delay −112 … +3968 ns plus `pins_delay_s` (−141 … +3939 ns on this board) relative to the analog output in 8 ns steps (D0 aligned with the markers to 1 ns; tagged by timetagger on a second board: a period's jitter about 21 ps rms, D1 20 ps after D0, an 8 ns word 8.79 ns wide, 2026-11-05; markers and D0–D7 3.3 ns after the analog edge with this board's `pins_delay_s`, measured)
Limits	per output, hardware clamp of every sample to [low, high] (one DAC step), clipped samples counted; lockable
Inputs	capture of 8192 samples of IN1 and IN2; means for the board calibration
Settings after loading	everything off, outputs at 0 in converter units

Hardware setup

- **Outputs.** Use 50 Ω loads for the set amplitudes; add a low-pass filter for the band you use to remove the DAC images near 125 MHz ± f.
- **Triggers, markers and digital outputs** are on the extension connector E1 (ground on the same connector). 3.3 V logic only. The digital outputs D0–D7 are the N pins of DIO0–DIO7 (the second row of E1).
- **Digital self-test.** `make test-hw` also needs DIO1_P (marker 1) wired to DIO0_P (TRIG A), and DIO0_N (D0) wired to DIO3_P (TRIG B).
- **Loopback self-test.** Cable OUT1 → IN1 and OUT2 → IN2 (LV jumpers), 50 Ω terminators at the inputs; `make test-hw` and `awg.capture()` use it.

- **Calibration.** Without one, nominal converter constants are used (about 14 % off in amplitude on this board). awg can measure it in loopback (`CAL LOOPBACK`).

Documentation

Document	Contents
docs/manual/user_manual.pdf	user's manual (38 pages, measured and computed figures): what awg is, safety and setup, time and applies, waveforms and edges, noise, modulation, sweeps, bursts, triggers, thresholds, timers, markers, limits, arbitrary waveforms and sequences, I/Q, digital outputs, calibration, measuring with the inputs, presets, the panel, use from experiment software, application notes, specifications, troubleshooting, quick reference, command reference
docs/PROTOCOL.md	every setting, Python methods, TCP commands, command line
RECIPES.md	worked examples: pulses with edges, bursts on a trigger, gated bursts, sweeps with a marker, noise, AM/FM by an input, FSK by a pin, I/Q pair, limits, arbitrary waveforms and sequences, digital patterns
docs/THEORY.md	how it works: datapath, shapes and sub-sample edges, noise, modulation, sweeps, bursts and trigger timing, arbitrary waveforms, apply, markers, digital outputs, resources, roadmap
docs/REGISTERS.md	the hardware registers and word formats
docs/VERIFICATION.md	what was tested, when; resource use and timing
../common/CALIBRATION.md	the board calibration, shared by every design
../common/README.md	the shared framework: TCP service, common API, build workflow
NOTICE.md	sources
AGENTS.md	notes for maintainers and coding agents

Changing the design

Rebuilding needs Vivado 2025.1 on a build machine (../common/SETUP.md). From this directory:

```
make                        # create the project (first time), build, export the overlay
make test                   # offline tests: model, driver, protocol, client, command line, GUI
make sim                    # simulation: system ID, shapes, core/DAC system against the model
make test-remote            # upload, program the board, check identity and power-up state
make install-service        # the control service, with awg active
python3 ../common/tests/verify_build.py --design awg
```

`make test-hw` runs `scripts/test_hw.py` on the board (loopback OUT1 → IN1, OUT2 → IN2; DIO1_P → DIO0_P and DIO0_N → DIO3_P for the markers and the digital outputs).

Source layout

Path	Contents
`design.tcl`	design description for the common recipe: ID "AWGE", RTL, one `aw_core` cell, DIO pins
`src/aw_core.v`	top: sample counter, registers, triggers, two channels, the sequencer, shared ROMs, capture, markers, the digital pins
`src/aw_arb.v`	the arbitrary-waveform sequencer: DDR reads (HP0), FIFO, step table, hot cache, jumps
`src/aw_dig.v`	the pattern engine of the digital outputs: programs, pattern memory, delay line
`src/aw_chan.v`	one output: generators, modulation, events, burst/gate control, sweep, idle level, AM, noise, sum, limits, dither
`src/aw_shape.v`	phase to shape: sine, trapezoid with sub-sample edges, table, PRBS
`src/aw_noise.v`	Gaussian points and the 12-tap polyphase resampler (noise or arbitrary points)
`src/aw_sweep.v`	sweep segments, linear and logarithmic (exp2)
`src/aw_trig.v`	trigger pins (MMCM, ISERDES), event pins, thresholds, timers
`src/aw_regs.v`	AXI4-Lite registers, staged words and mirror, apply sequence, table/segment/capture access
`src/aw_table.v`, `src/aw_capture.v`, `src/aw_xoro.v`	table RAM, input capture and means, random generator

Path	Contents
`src/aw_*_rom.v`	generated ROMs: sine, exp2, inverse CDF, resampler kernel
`constraints/dio.xdc`	the trigger, event, marker and digital-output pins
`control/awg/`	Python package: bit-exact model (`model.py`), built-in tables (`shapes.py`), segments and sequences (`sequence.py`), driver, TCP commands, client, command-line options, Jupyter GUI
`control/awg.ipynb`	the notebook
`scripts/test_hw.py`	the loopback hardware test (`make test-hw`)
`tools/`	host-only: `record_manual_data.py` (the manual's board recordings, through the service) and `manual_figures.py` (its figures and numbers; numpy, matplotlib)
`tests/`	`tb_shape.sv`, `tb_system.sv` (simulation), `vectors.py` → `aw_vectors.sv`, `aw_*.mem` (the model's vectors), Python tests, `fake_core.py` (a register model of the core)

Verification

Version 3.0 passes the offline tests, both simulations (bit-exact against the model on the core streams, the DAC pins, the markers and the eight digital pins, with the points read from a DDR model) and all 57 loopback checks on the board, including D0 through TRIG B, aligned with the markers to 1 ns, and the measured noise spectrum, sweep rates, demodulated FM/PM/FSK/BPSK and input-threshold triggers (2026-10-24). Build, timing and board results: docs/VERIFICATION.md.

Author

Rok Zitko, Jozef Stefan Institute (Ljubljana, Slovenia), rok.zitko@ijs.si.

awg recipes

Worked examples in Python on the board (`awg = DRIVER()`, after `import rpfpga`) or on a PC (`awg = AwgClient("192.168.0.149")`): the methods are the same. Volts are into 50 Ω.

A pulse with slow edges

```
awg.set_output(1, function="pulse", frequency=200e3, amplitude=0.25, offset=0.25,   # 0 V to 0.5 V
               width=1e-6, rise=50e-9, fall=200e-9)
```

Width is measured at the 50 % points, rise and fall from 10 % to 90 %. The edges are straight lines in time; their position moves in sub-sample steps, so a period that is not a whole number of samples (e.g. at 3.3 MHz; here it is 625) shows no 8 ns jitter.

Three-cycle bursts on an external trigger, with a delay

```
awg.set_output(1, function="sine", frequency=5e6, amplitude=0.3, phase=0.0,
               burst={"mode": "triggered", "cycles": 3, "idle": "start"},
               trigger={"source": "trig_a", "edge": "rising", "delay": 2.5e-6})
```

A rising edge on DIO0_P starts three cycles 2.5 µs (+ 160 ns) later, from phase 0. Triggers during a burst are missed (`awg.hardware()["outputs"]["1"]["missed"]`). Use `stop_phase` to end on another phase, and `idle: "dc"` with `idle_level` for a level between bursts.

A gated tone

```
awg.set_output(2, function="sine", frequency=1e6, amplitude=0.3,
               burst={"mode": "gated", "gate": "event0", "polarity": "high", "finish": True})
```

The output runs while DIO4_P is high and finishes the cycle after it falls.

Triggering on an analog level

```
awg.apply({"inputs": {"in1": {"high": 0.20, "low": 0.15}},
           "outputs": {"1": {"function": "square", "frequency": 1e6, "amplitude": 0.2,
                             "burst": {"mode": "triggered", "cycles": 10},
                             "trigger": {"source": "in1", "edge": "rising"}}}})
```

IN1 crossing 0.20 V upward (after having been below 0.15 V) starts the burst.

A swept sine with a marker

```
awg.apply({"outputs": {"1": {"function": "sine", "amplitude": 0.3,
                             "sweep": {"mode": "continuous", "type": "log", "start": 1e3, "stop": 10e6,
                                       "time": 0.1, "hold": 0.01, "return": 0.01, "marker": 1e5}}},
           "markers": {"1": {"source": "sweep", "channel": 1}}})
```

DIO1_P is high from the start of each sweep until 100 kHz. With `"mode": "triggered"` and a `trigger`, one sweep runs per trigger.

Noise added to a signal

```
awg.set_output(1, function="sine", frequency=1e6, amplitude=0.2,
               noise={"rms": 0.02, "bandwidth": 5e6})
```

The noise is Gaussian and flat up to about 0.82 of the bandwidth (−0.5 dB), −3 dB at the bandwidth. For noise alone use `function="off"`.

AM by an input, FM by generator 2

```
awg.set_output(1, function="sine", frequency=10e6, amplitude=0.25,
               am={"mode": "am", "source": "in1", "depth": 0.5})          # 50 % per volt on IN1
awg.set_output(2, function="sine", frequency=10e6, amplitude=0.25,
               aux={"function": "sine", "frequency": 1e3},
               angle={"mode": "fm", "source": "aux", "deviation": 50e3})  # ±50 kHz at 1 kHz
```

FSK keyed by a pin

```
awg.set_output(1, function="sine", frequency=1e6, amplitude=0.3,
               angle={"mode": "fsk", "hop": 1.2e6, "key": "trig_b"})
```

While DIO3_P is high the output runs at 1.2 MHz, phase-continuously.

An I/Q pair

```
awg.load_preset("iq")                    # 1 MHz on both, OUT2 90 degrees behind OUT1
awg.set_output(2, phase=-89.5)           # adjust the quadrature
```

Both outputs change on the same sample; phases are relative to t0 (`awg.zero_time()`).

Your own table

```
import math
awg.save_table("chirp", [math.sin(2 * math.pi * 20 * (i / 1000) ** 2) for i in range(1000)])
awg.set_output(1, function="arb", table="chirp", frequency=10e3, amplitude=0.3)
```

Tables are resampled to 4096 points and played at the set repetition frequency.

Protecting a sensitive input

```
awg.set_output(1, limits={"low": -0.1, "high": 0.3})
awg.protect(1)                           # locked until the FPGA is reloaded
```

Every sample of OUT1 stays within −0.1 … 0.3 V (the calibration's accuracy), whatever is configured later; SAFE goes to the nearest limit if 0 V is outside.

An arbitrary waveform

```
import math
pts = [math.exp(-((i - 500) / 120) ** 2) * math.sin(2 * math.pi * i / 25) for i in range(1000)]
awg.save_segment("pulse", pts)                        # 1000 points in [-1, 1]
awg.play(sequence="pulse", rate=50e6)                  # 20 µs per repetition, forever
awg.set_output(1, arb={"on": True, "amplitude": 0.4, "filter": "smooth"})
```

Every point is played once at 50 MSa/s. `smooth` reconstructs a band-limited signal (flat to 0.31 of the rate, −3 dB at 0.38); `linear` draws straight lines between the points; `step` holds each point (a rectangular staircase). On the board, `awg.load_segment_file ("pulse", "/home/xilinx/pulse.npy")` reads a `.npy` or CSV file.

Two waveforms at once, and I/Q

A segment holds pairs: OUT1's points and OUT2's (both the same if you give one list).

```
n = 4096
i_pts = [math.cos(2 * math.pi * 7 * k / n) for k in range(n)]
q_pts = [math.sin(2 * math.pi * 7 * k / n) for k in range(n)]
awg.save_segment("iq", i_pts, values2=q_pts)
awg.play(sequence="iq", rate=10e6)
awg.set_output(1, arb={"on": True, "amplitude": 0.3})       # I
awg.set_output(2, arb={"on": True, "amplitude": 0.3})       # Q, the same time base
```

To put I/Q on a carrier (single sideband at 10 MHz + 17.1 kHz) on OUT1, keep OUT2 playing Q (at amplitude 0 if you don't want it on OUT2):

```
awg.set_output(1, arb={"upconvert": {"on": True, "frequency": 10e6, "q_gain": 1.0, "q_phase": 0.0}})
awg.set_output(2, arb={"amplitude": 0.0})
```

`q_gain` and `q_phase` (degrees) set an I/Q gain imbalance and quadrature error; `point_offset` adds a DC offset to I (OUT1's) or Q (OUT2's) points: carrier leakage.

A triggered sequence with a jump

```
awg.save_segment("idle", [0.0] * 512)
awg.save_segment("tone", [math.sin(2 * math.pi * i / 64) for i in range(6400)])
awg.save_sequence("demo", [
    {"segment": "idle", "repeat": 0, "on_event": "end", "event_target": 1},   # until EVENT0
    {"segment": "tone", "repeat": 3, "marker": True, "next": "end"}])
awg.play(sequence="demo", rate=125e6, run="triggered", trigger={"source": "trig_a", "delay": 1e-6},
         jump={"source": "event0", "edge": "rising"})
awg.set_output(1, arb={"on": True, "amplitude": 0.3, "filter": "step"})
awg.apply({"markers": {"1": {"source": "start", "channel": 1, "width": 1e-6}}})
```

A rising edge on DIO0_P starts the sequence 1 µs (+160 ns) later, with the sub-sample timing of the edge. It plays silence until DIO4_P rises, then (at the end of the silence's repetition, without a gap) three repetitions of the tone, with a 1 µs pulse on DIO1_P at the tone's start; then it waits for the next trigger.

A clock and a frame on the digital outputs

```
awg.save_pattern("frame", [0x01] + [0x00] * 9)              # D0: one word high in ten
awg.set_digital(run="continuous", program="tick", enable=0x03, programs={
    "tick": {"mode": "memory", "pattern": "frame", "rate": 10e6}})
```

D0 (DIO0_N) is high for 100 ns every 1 µs; D1 (enabled, bit 1 of every word 0) stays low; D2–D7 are high impedance. A binary count (`{"mode": "binary", "rate": 2e6}`) gives divided clocks instead: Dk is a square wave of 2e6 / 2^(k+1) Hz.

A trigger pattern with every burst

```
awg.set_output(2, function="sine", frequency=1e6, amplitude=0.3,
               burst={"mode": "triggered", "cycles": 5}, trigger={"source": "timer1"})
awg.apply({"timers": {"1": {"period": 1e-3, "on": True}},
           "digital": {"run": "out2", "program": "gate", "enable": 0xFF, "programs": {
```

```
            "gate": {"mode": "binary", "start": 0, "length": 5, "rate": 1e6, "repeats": 1,
                     "end": "idle"}}}})
```

With each burst of OUT2, D0–D7 count 0 … 4, one value per microsecond, aligned with the burst's first sample, then return to the idle word 0. `delay` (in `digital`) shifts them in 8 ns steps, e.g. `-40e-9` for a pre-trigger.

Patterns that follow a sequence

```
awg.save_pattern("addr", [0x10, 0x11, 0x12, 0x13])
awg.save_sequence("scan", [
    {"segment": "tone", "digital": "addr_out"},                # start the address pattern
    {"segment": "idle", "digital": "stop", "next": 0}])         # then the idle word again
awg.apply({"arb": {"sequence": "scan", "rate": 125e6},
           "outputs": {"1": {"arb": {"on": True, "amplitude": 0.3}}},
           "digital": {"run": "sequencer", "idle": 0x00, "enable": 0xFF, "programs": {
               "addr_out": {"mode": "memory", "pattern": "addr", "rate": 125e6 / 1600, "repeats": 1,
                ↪  "end": "hold"}}}})
```

Each time the tone step starts, D0–D7 step through the four addresses once (1600 samples each: one per 25 periods of the tone) and hold the last, starting on the sample where the tone's first point reaches OUT1; the idle step returns them to 0. A PRBS (`{"mode": "prbs", "order": 15}`) puts the sequence on D0 and its delayed copies on D1–D7, for testing a receiver's bit alignment.

awg: theory of operation

This document explains how awg works inside the FPGA and why it is built this way. The user-level description is in the README; registers are in REGISTERS.md. `control/awg/model.py` is the bit-exact reference of everything below: the testbenches compare the RTL with it sample by sample.

Datapath

Both outputs run on one 125 MHz sample clock; one sample per clock. Per output:

```
                        sweep segments ─┐ (F from a list of linear/log pieces)
modulation m (gen2, IN1, IN2, other) ──► angle: ΔF / ΔP / ΔH = (DEV m) >> 10, FSK/BPSK key
                                        ▼
burst/gate ── phase accumulator 1 (48 bit) ──► shape 1 ──► idle level ──► × k (AM) ──► c
              phase accumulator 2 (48 bit) ──► shape 2 ──────────────────────────────► g2
 Gaussian points, or the sequencer's points ──► 12-tap polyphase resampler ──────────► nz
S = A1 c + A2 g2 + AN nz + offset   (2^-27 DAC steps)
  (I/Q upconversion: S = 64 nz c + 64 nz_other s + AN nz + offset, see "I/Q")
code = clamp(floor((S + 2^26 + dither) / 2^27), low, high)  ──► DAC
```

- **Phase.** A 48-bit accumulator advances by the frequency word F every sample (2^-48 cycle per unit: 0.44 µHz). The shapes see p = acc + POW·2^16 (+ the PM or BPSK offset), so phases are in 2^-32 cycle steps relative to the accumulator.
- **Coherence.** An apply loads each accumulator with P = F (n_a − t0) at the apply sample n_a, so the phase at sample n is F (n − t0) + POW: phases refer to t0 whatever happened before, both outputs alike. In continuous mode the accumulators run on (phase-continuous changes).
- **Generator 2** runs 13 samples ahead of generator 1 (its accumulator holds sample n when the counter reads n − 13), so its value is ready as a modulation source.
- **Single rounding.** Every term is summed at full precision (2^-27 DAC step) and rounded once to the 14-bit DAC, optionally with triangular dither of ±1 DAC step (xoroshiro128+).

Shapes

All shapes are 18-bit values with full scale VMAX = 131071, computed in six clocks with one DSP48 per generator (`aw_shape.v`):

- **Sine**: synth's quarter-wave table of 1024 entries {s, d} at the entry centres with a first-order Taylor correction, y = (s 2^22 + d (2f − 4095) + 2^17) >> 18 for the 12-bit fraction f, negated in quadrants 2 and 3; g = (y + 8) >> 4. Error below 1.1 LSB of 2^-17 (rounding included), spurs below −100 dBc before the 14-bit rounding (computed).

- **Trapezoid** (square, pulse, ramp, triangle): rising over [0, R) of the phase, high until HE, falling over [HE, FE), low after. On an edge the value is floor((VMIN 2^21 + (p >> sr) a) / 2^21) (rising) or floor((VMAX 2^21 − ((p − HE) >> sf) a) / 2^21) (falling): the offset into the edge, shifted to 17 bits, times a slope word a = round(2 VMAX 2^21 / (width >> s)). **Sub-sample edges**: when every edge is at least one sample wide (R ≥ F), each edge has at least one sample on it, whose value is linear in the edge's exact position. A band-limited reconstruction then moves the edge in continuous time: there are no 8 ns steps in the edge position when the period is not a whole number of samples (the model's test: the 50 % crossings of a 3.3 MHz square have the exact period within 0.01 sample). Edges shorter than one sample would jump; the driver refuses them. The driver sets the shapes' phase so that phase 0 is the 50 % point of the rising edge.
  - square: duty d, R and FE − HE from the 10–90 % rise and fall times (full edge = time / 0.8), HE = R/2 + d − (FE − HE)/2;
  - pulse: the same with d = width · f;
  - ramp: R = symmetry, falling for the rest (at least one sample); triangle: 50 %.
  - **PWM** moves HE and FE together by ΔH = (DEV m) >> 10 (2^-48 cycle).
- **Table**: 4096 points of 18 bits, T[i] + ((T[i+1] − T[i]) f) >> 12 with i = p[47:36], f = p[35:24] (linear interpolation, wrapping). Two reads per sample from one block RAM (both ports); the host's writes take the second port for one clock: write tables while they are not played.
- **PRBS**: a Fibonacci LFSR x^n + x^m + 1 (ITU-T O.150: n = 7, 9, 11, 15, 20, 23, 31) steps when the phase wraps forward (by less than half a cycle); the bit rate is the generator's frequency. A bit change makes an edge from the previous bit's level over [0, R) of the bit, as the trapezoid's rising edge (sub-sample, as above). A coherent apply loads the register with the state of the bit that sample n_a falls in, counted from t0 (`model.lfsr_jump`: x^k modulo the characteristic polynomial), so the sequence is coherent with t0 as the phase is; a burst restarts it from all ones.

Noise

- **Points.** A Gaussian value with σ = 2^14 from each 64-bit word of a xoroshiro128+ generator: the sign from bit 63, a 40-bit magnitude u from bits 62..23. The inverse normal CDF y = $-\Phi^{-1}$(u/2^41) is piecewise linear: the octave k (leading zeros of u, up to 39) and 32 sub-segments per octave index a table of {c0, c1}; y = c0 + ((c1 x) >> 18). The table is the minimax line through each segment's representable points (error below 2.3 LSB of 2^-14 σ, mostly below 0.4). The largest value is $-\Phi^{-1}$(1.5 · 2^-41) ≈ 7.1σ (the deepest octave).
- **Resampler.** Points enter a 12-point history at a rate set by a 32-bit phase step (the point rate r = rate/2^32 · 125 MSa/s); each output sample is Σ_k C[μ][k] h[k] with the position μ of the output after the newest point (512 kernel phases). The kernel is a Kaiser-windowed sinc (cutoff 0.42 of the point rate, β = 7); every phase sums to 2^16. The noise spectrum is flat to ±0.5 dB up to 0.31 r, −3 dB at **0.382 r** (the bandwidth), below −60 dB from 0.60 r (computed with the model). Since the output is a weighted sum of independent Gaussian points, it is Gaussian; its rms is 0.874 σ (the kernel's rms gain, averaged over phases; ripple 0.44 % over a point period, computed). The same resampler plays the arbitrary points. (1.0 had 16 taps, β = 8, −3 dB at 0.389 r; 2.0 has 12 to fit the sequencer.)
- **Bandwidth** 1 Hz (r ≈ 2.6 Sa/s) to 47.7 MHz (r = 125 MSa/s).

Modulation

The sources are 18-bit values m: generator 2's shape value (±VMAX), IN1/IN2 as 2 (code − input offset) (±1 V on LV is about ±2^17; input sample n − 7 modulates output sample n), or 16 × the other output's code (its sample n − 26). Each output has one source for AM and one for the angle modulation.

- **AM**: k = 2^15 + (DEPTH m) >> 17, c = sat((g1' k) >> 16): depth d at the source's full scale has DEPTH = d 2^32/m_fs. **DSB-SC**: k = (DEPTH m) >> 17. Without AM, c = g1'/2.
- **FM**: F_eff = F + (DEV m) >> 10 (DEV 42-bit signed; deviation up to the Nyquist frequency). **PM**: the phase offset (DEV m) >> 10. **PWM**: see Shapes.
- **FSK**: F_eff = key ? F_hop : F; **BPSK**: phase + (key ? φ : 0). The key is generator 2's first half cycle (its frequency is the keying rate) or the level of a pin or threshold (edges seen up to the previous sample).
- The deviation is computed with two DSP48s from m in clock n − 5 and ready in clock n − 1 for sample n; FM, FSK and the sweep enter the accumulator, PM and BPSK the shape phase, PWM the trapezoid.

Sweeps

A sweep is a list of up to 512 segments {start word, rate, count, flags}. A 72-bit accumulator (a 48-bit word and 24 fraction bits) advances by the rate each sample; after `count` samples the next segment follows, JUMP loads its

start word. The word is the frequency word, or for LOG segments log2 of it (6 integer and 42 fraction bits), with F = 2^L from a 1024-point exp2 table, linearly interpolated (relative error below 1e-7). The driver builds linear and logarithmic sweeps (split at the marker frequency, the first part with the MARK flag), hold (rate 0) and return segments, list sweeps (JUMP with rate 0, dwell = count) and profiles (linear pieces), with rates rounded so that each piece lands exactly on its end word (`model.ramp_rate`). The rate (56 bits, 24 fraction) limits a linear piece to 2^31 words per sample (0.95 kHz per sample) and a logarithmic one to 2^-11 octave per sample. A continuous sweep repeats the list; a triggered one plays it once per trigger and holds the last value.

Bursts, gates and trigger timing

- **Event times.** Every event has a time τ in ticks of 1/256 sample (31.25 ps). TRIG A and TRIG B are sampled by ISERDESE2 in oversample mode at 1 ns (250 MHz clocks at 0 and 90° from an MMCM): an edge x ns after the clock edge where the counter becomes n gets τ = 256 n − 128 + 32 ⌈x⌉ (in simulation; on the board the input delay adds a constant). EVENT pins and the IN thresholds have τ = 256 n (8 ns), software triggers the clock of their register write, timers START + k PERIOD (exact ticks), the other output its start time.
- **Seeing an event.** An event is presented in clock floor(τ/256) + 6, registered by the channel and decided in the next clock for sample s = floor(τ/256) + 8; the bus carries only τ mod 256. A start at T = τ + 20·256 + delay (delay in ticks) has its first sample n0 = ⌈T/256⌉ ≥ s + 12 and starts generator 1 at phase start + F μ/256 with μ = 256 n0 − T: the output's time origin is T to 1/256 sample, so a TRIG A/B start has only the 1 ns quantization of its timestamp (0.29 ns rms if the edge is uniformly distributed, computed).
- **Counters.** The FPGA never compares with the 64-bit sample counter on this path: with delay = 256 dhi + dlo and sum = (τ mod 256) + dlo, n0 = s + 12 + dhi + ⌈sum/256⌉, and a counter of dhi + 11 + ⌈sum/256⌉ clocks reaches zero in the clock before n0.
- **Busy.** A trigger is accepted when the burst is idle (or the triggered sweep neither runs nor waits for a start); others are counted as missed.
- **Burst length.** With rem = span − (phase since the start), span = N cycles + the stop phase offset (N = 0: endless), the burst runs while rem > 0; rem decreases by F_eff each sample, so FM and FSK bursts end on their phase too.
- **Gate.** An opening edge (per polarity) starts like a trigger; a closing edge schedules a stop at ⌈(τ_close + 20·256 + delay)/256⌉: there the burst finishes its cycle (span := the next cycle boundary plus the stop offset) or stops at once. A gate open at an apply opens the burst as if its edge came at the apply sample.
- **Idle level.** The start value (the accumulator holds the start phase), the last value (held), or a DC level (in shape units, so within ± the amplitude); AM continues.

Arbitrary waveforms

One sequencer (`aw_arb`, the engine) plays **point pairs** for both outputs: each output's resampler, the one that filters its noise, takes its half of the pairs instead of noise when asked (per output; the noise is then off), at the sequencer's carries and kernel phase. The generators are still summed (generator 1 runs continuously). The sequencer uses OUT1's run control (trigger, gate, delay, starts with sub-sample timing): OUT1 has no bursts or sweeps while it plays. (Two sequencers, one per output, did not fit the 7010's LUTs next to the function generator: 2.0's choice.)

- **Pairs** are 32-bit words: OUT1's point word in bits 15–0, OUT2's in 31–16, a point word being {value[13:0], marker 2, marker 1}. They are in a DDR buffer of the processor (64 MB of contiguous memory by default: 16 M pairs), in segments of 64 ... 2^25 − 1 pairs starting at 64-byte boundaries. An output's resampler takes (value << 4) + POFF, saturated: value 8191 is 131056 of the shapes' full scale 131071. POFF (each output's point offset, taken at a restart) adds to every point, the idle ones too.
- **Rate.** A carry of the sequencer's 32-bit position (ARATE per sample, or every sample at 2^32) takes the next pair: every pair is played once, at the pair rate r = ARATE/2^32 × 125 MSa/s (125 MSa/s at most).
- **Filters** (per output). Smooth: the noise kernel (flat to 0.31 r, −3 dB at 0.382 r); the output shows the point that entered the history 6 points earlier (a delay of 6/r). Linear: taps 1 and 0 with coefficients 2^16 − c and c = 128 ph + 64 (ph = the position's 9 top bits): straight lines from point to point, a delay of 1/r. Step: tap 0, each point held for 1/r, no delay. The point markers follow the same tap (6, 1, 0).
- **Sequencer:** a table of 1024 steps. A step plays its segment `repeat` times (0: until an event jumps away), then goes to the next step, a goto target, or ends (idle before the start step). It may wait for the trigger (both outputs holding their last points or a level) before it starts, mark its start (marker source "start"), and jump on an event of the jump source: at the end of a repetition or at once. Rules (sample n; events seen at sample s, model.J_EV = 8):
  - a jump event takes part from sample s + 8; it counts for the step if it arrives after the step's first pair, and a jump happens at least J_MIN = 16 samples after it;

  - an end jump happens at the carry after a repetition's last pair, an immediate jump at the next carry; the target's first pair is taken then (or the target waits);
  - a start (trigger, gate, or the apply of a continuous sequence) at T = tau + 20·256 + delay ticks takes the waiting step's first pair at sample n0 = ⌈T/256⌉ with the position (r μ)/256, μ = 256 n0 − T: pair k of the step is at time T + k/r exactly, to 1/256 sample (31 ps). A continuous sequence starts 20 samples after its apply.
  - a trigger starts the sequence when it waits: triggered: before the start step or in a waiting step; continuous: in a waiting step; gated: the gate's opening, before the start step (gated sequences ignore the wait flags); a closing gate stops at once or at the end of the repetition (finish);
  - an apply restarts the sequencer only when asked (OUT1's LOAD bit 1): the driver does so when the sequence or its settings change. Otherwise the sequence goes on.
- **Hot cache and gapless jumps.** A start and a jump play the first 512 pairs of their step from a block-RAM cache (8 slots of 512 pairs), while DDR reads restart behind them: a step of at most 512 pairs plays from the cache in all its repetitions. Each step's jump target has its first 16 pairs queued in advance (two alternating buffers), so a jump costs no gap at any rate. The driver puts the start step and the targets in slots; DDR must deliver the rest within the cached pairs' time (4.1 μs at 125 MSa/s for 512 pairs; keep targets that are followed quickly by DDR reads long).
- **DDR reads.** A fetcher walks the same steps as the sequencer and reads AXI3 bursts of up to 16 beats (128 bytes, never across a 4 KB boundary, only within ARB_BASE … ARB_BASE + ARB_SIZE) into a FIFO of 2048 beats (4096 pairs, 33 μs at 125 MSa/s), at most 128 beats in flight. A reposition (start, jump, stop) flushes the FIFO; beats still in flight are dropped as they arrive. At 125 MSa/s the sequencer reads 500 MB/s of the HP0 port's 1 GB/s (computed). A pair that is due with none ready stops the sequence (UNDERFLOW).
- **I/Q.** Baseband I/Q: I in the pairs' OUT1 halves, Q in their OUT2 halves, each output with its own amplitude (gain imbalance), offsets and filter. Upconversion on output n: S = 64 r_n c + 64 r_other s, with c = (g1 DEPTH) >> 16 (generator 1: the cos carrier) and s = (DEV g2 >> 10) >> 16 (generator 2: the sin carrier, DEV < 0 for −Q sin), so OUT = A (I cos − g Q sin(ωt + ε)): gain imbalance g and quadrature error ε in the carriers, I/Q offsets in POFF. The AM and angle modulation (their multipliers make c and s) are off then. (There is no sub-sample skew between the halves: they share the sequencer's position.)

Apply

Configuration words are written into pending registers (with a block-RAM mirror for reading back) and take effect at an apply: at once (APPLY_NOW: sample now + 21) or at a scheduled sample (APPLY_AT with APPLY_TIME). The apply sequence `apl` runs 40 clocks from n_a − 20; every live register loads in the clock that makes it apply from sample n_a on, at its own pipeline stage (generator 2 at n_a − 13, the resampler's rate at n_a − 9, the sequencer at n_a − 8, the sweep at n_a − 7, the run control at n_a − 1, the amplitudes and offset at n_a + 13 … n_a + 18). Pending writes wait while the sequence runs. So **every setting of both outputs changes on sample n_a**, and a coherent apply loads the accumulators for phases relative to t0. The driver's coherent apply is 10 ms (apply_delay) after the call; continuous mode applies at once.

Markers

Each marker shows a source of one channel for sample n: sync (the first half of generator 1's cycle), run (the burst runs, or always in continuous mode), the sweep segment's MARK flag, or a start pulse (width in samples, from each burst or sweep start); inverted on request. A delay line of 1 … 511 samples follows; the pin shows sample n's level when the DAC shows sample n − (delay − 18), so the default delay 18 aligns it with the analog output (simulation, DAC model). The markers are on the 8 ns grid. In arbitrary-waveform mode the sources follow the resampler's output tap: run = a played point (not an idle one), start = a step with its marker flag or a start, and the two point markers.

The delay line puts a marker at a chosen time relative to the analog output inside the FPGA. The connector adds its own delays: the DAC, its output filter and the analog front end on one side, the FPGA's output buffers and the board traces on the other. The board calibration's `pins_delay_s` (measured once with an oscilloscope, see `common/CALIBRATION.md`) is the remaining offset of the extension-connector pins relative to OUT1; the driver subtracts it from the requested delay in whole samples, so the pins land within ±4 ns of the requested time. On this board the pins change 28.6 ns before OUT1 (`pins_delay_s` = −28.6 ns, measured 2026-10-24): with delay 0 they are shifted 4 samples and land 3.3 ns after OUT1's edge.

Digital outputs

A pattern engine drives D0–D7 (DIO0_N–DIO7_N) with one 8-bit word per sample, through a delay line like the markers' (1 … 511 samples, 15 aligns word n with analog sample n in simulation; the board calibration's `pins_delay_s` applies as for the markers) and the pins' output registers. Each pin is driven only when enabled (DIG_OE); otherwise it is high impedance, also before the first apply and after SAFE.

```
apply (restart), OUT1/OUT2 start pulses, step actions ⟶ start / stop
program (16 entries, two halves) ⟶ engine: samples per word, words per pass, passes
     memory: 32768 bytes from START │ binary/Gray count from START │ PRBS (LFSR)
⟶ word (or the idle word, or the held last word) ⟶ delay line ⟶ D0–D7
```

- **Programs.** Up to 8 at a time (the live half of a 16-entry table): a mode, START, words per pass, samples per word (125 MHz / an integer: the word rate is exact and every word lasts the same number of samples), passes (0: forever) and the end: the last word held or the idle word. The driver writes new programs into the other half and switches halves at an apply, so a running program is never changed under it.
- **Modes.** Memory plays bytes START … START + length − 1 (each pass from START). The binary and Gray counts begin at START each pass (Gray: one bit changes per word, also at the wrap of 255 → 0). The PRBS is the LFSR of the PRBS shape (x^n + x^m + 1, all ones at the start), one step per word, running on across passes: D0 is the sequence, Dk the same sequence k words earlier (bits at and above the order stay 0).
- **Starts.** Continuous: at an apply that restarts the engine (the driver restarts it when the run mode, the program or the programs change); at OUT1's or OUT2's start pulses (the start marker's source: a burst or sweep start, or a sequence start or marked step); or by sequence steps: a step's action (start a program, stop) takes effect on the sample whose output shows the step's first point, at the output tap of the output it follows (OUT1 if it plays points, else OUT2: the smooth filter's 6-point delay is included). A start while running starts again from the first word.
- **Timing.** The engine works on sample n's events in clock n + 2, reads the memory in n + 3 and registers the word in n + 4; a word changes on the sample grid (8 ns), aligned with the analog output by the delay line. The engine keeps the counters' "last word" flags in registers so that its 32-bit counts do not limit the clock.

Limits

Every code is clamped to [low, high] (14-bit DAC codes from the volts and the board calibration), after all sums, modulation, noise and dither; clamped samples are counted. PROTECT (write-once until the FPGA is reloaded) blocks writes to an output's limit word, so neither the driver nor raw register writes can widen it. MUTE (SAFE) forces 0 V or the nearest limit until the next apply.

Resources

On the xc7z010 (build 0x97cda2f3 of 2026-10-24, see VERIFICATION.md): 13,869 LUTs (79 %), 22,054 flip-flops (63 %), 48 DSP48 (of 80), 58 block-RAM tiles (of 60), both MMCMs; timing met with 0.003 ns of slack at 125 MHz, after a post-route physical optimization (`design(impl_properties)`). At this fill the placement is close to the routing limit: one 3.0 build failed to route with a congestion-spreading placer directive, another missed timing by 2.7 ns from congestion alone. (2.0: 13,099 LUTs, 21,114 flip-flops, 48 DSP48, 48 block-RAM tiles; 1.0: 11,306 LUTs, 20,213 flip-flops, 54 DSP48, 38 block-RAM tiles.)

2.0 first had a sequencer per output and 16 resampler taps, as 1.0's noise filter did. That needed more LUTs than the chip has (17,600). Two choices made it fit and meet timing:

- **One sequencer for both outputs.** It reads OUT1/OUT2 point pairs from one DDR stream and keeps one FIFO, one step table and one hot cache. Each output keeps its own resampler, filter, amplitude, offset and upconversion. Lost: independent sequences per output, and a sub-sample skew between I and Q (the points of a pair are played at the same position).
- **12 resampler taps instead of 16** (the agreed first fallback). The noise filter and the smooth filter are the same resampler, so both changed: the noise bandwidth is 0.382 of the point rate, the band is flat within ±0.5 dB to 0.31 of the rate, and more than 60 dB down from 0.60 of it (computed). This also freed 8 DSP48; I/Q upconversion uses 2 of them.

The block RAM holds the sine and table ROMs, the kernel coefficients, the sweep segments, for 2.0 the sequencer's FIFO (2048 × 64 bits), step table (1024 × 128 bits) and hot cache (4096 pairs), and for 3.0 the pattern memory (32768 bytes, 8 tiles), the program table and the digital outputs' delay line. The markers' delay lines are in LUT RAM.

Roadmap

- **2.0 true arbitrary waveforms** and **3.0 digital outputs** are done (see above). The chip has no room left for additions: 2 block-RAM tiles are free, and the routing is congested at 79 % of the LUTs. Further features would have to replace existing ones; the largest block is the staged-word register file (about 5,400 flip-flops), which a block-RAM staging scheme could shrink.

awg: settings, Python methods, TCP commands, command line

All values are physical: **volts at the SMA connectors** (amplitudes are peak values into 50 Ω), **Hz**, **degrees** relative to t0, **seconds**. The common commands and methods (CONFIG, SNAPSHOT?, PRESET, SAFE, CAL, RANGE, ...) are described in ../../common/README.md.

The configuration

`awg.config()` returns, and `awg.apply(patch)` merges, a JSON-compatible dict:

```
{"outputs": {"1": OUTPUT, "2": OUTPUT},
 "inputs": {"in1": {"high": 0.1, "low": -0.1}, "in2": {...}},     # threshold triggers (V),
  ↪  hysteresis
 "timers": {"1": {"period": 1e-3, "on": False}, "2": {...}},      # trigger timers, aligned to t0
 "markers": {"1": {"source": "low", "channel": 1, "invert": False, "delay": 0.0, "width": 1e-6},
  ↪  "2": {...}},
 "arb": SEQUENCER,                # the arbitrary-waveform sequencer (below)
 "digital": DIGITAL,              # the digital outputs D0..D7 (below)
 "phase_mode": "coherent",        # or "continuous": changes at once, phase-continuous
 "t0": 0,                         # zero time (a sample index): phases refer to it
 "droop_correction": True,        # sine amplitudes corrected for sinc(f/fs) (and the calibrated
  ↪  response)
 "allow_clip": False,             # True: allow peaks beyond the output range (they saturate)
 "apply_delay": 0.01}             # coherent changes happen this long after the call (1 ms ... 1 s)
```

One output (`OUTPUT`):

Key	Default	Meaning
`function`	`"off"`	`off`, `sine`, `square`, `pulse`, `ramp`, `triangle`, `prbs`, `arb` (a table), `dc` (the offset only)
`frequency`	1e6	Hz (PRBS: the bit rate); 0 ... 62.5 MHz
`amplitude`	0.0	V, peak (square: ±amplitude around the offset)
`offset`	0.0	V
`phase`	0.0	degrees relative to t0; in bursts the start phase
`duty`	50.0	square: %, at the 50 % points
`width`	100e-9	pulse: s, at the 50 % points
`rise`, `fall`	0.0	10–90 % edge times (s); 0: the shortest (one sample)
`symmetry`	50.0	ramp: % of the period rising
`order`	7	PRBS: 7, 9, 11, 15, 20, 23, 31
`table`	`"gaussian"`	arb: `gaussian`, `lorentzian`, `sinc`, `exp_rise`, `exp_fall`, `haversine`, or a saved table
`noise`	`{"rms": 0.0, "bandwidth": 10e6}`	added Gaussian noise, V rms; bandwidth (−3 dB) 1 Hz ... 47.7 MHz
`aux`	generator settings	generator 2 (`function`, `frequency`, `amplitude`, `phase`, shape keys): added with its amplitude, and the `aux` modulation source (at full scale)
`am`	`{"mode": "off", "source": "aux", "depth": 0.0}`	`am` or `dsb`; `source` `aux`, `in1`, `in2`, `other`; `depth` at the aux full scale, or per volt
`angle`	`{"mode": "off", "source": "aux", "deviation": 0.0, "hop": 2e6, "bpsk_phase": 180.0, "key": "aux"}`	`fm` (deviation Hz), `pm` (degrees), `pwm` (s of width), at the aux full scale or per volt; `fsk` (to `hop` Hz) and `bpsk` (by `bpsk_phase`) keyed by `key`: `aux` (its first half cycle; its frequency is the keying rate) or a pin/threshold level

Key	Default	Meaning
`sweep`	`{"mode": "off", "type": "linear", "start": 1e3, "stop": 1e6, "time": 1e-3, "hold": 0.0, "return": 0.0, "marker": None, "points": [], "profile": []}`	`mode` `continuous` or `triggered`; `type` `linear`, `log`, `list` (`points` [[Hz, s], ...]) or `profile` ([[s, Hz], ...], linear between points); `marker`: the sweep marker's frequency
`burst`	`{"mode": "off", "cycles": 1, "stop_phase": None, "idle": "start", "idle_level": 0.0, "polarity": "high", "finish": True, "gate": "trig_b"}`	`triggered` (N cycles, 0 endless) or `gated` (by `gate` while it is `polarity`; `finish` the cycle or stop at once); `stop_phase` (None: the start phase); `idle`: `start` value, `last` value, or `dc` (`idle_level` V, within ± the amplitude)
`trigger`	`{"source": "software", "edge": "rising", "delay": 0.0}`	`trig_a`, `trig_b`, `event0` ... `event3`, `in1`, `in2` (thresholds), `software`, `timer1`, `timer2`, `other` (the other output's start); `edge` `rising`, `falling`, `both`; `delay` s (31 ps steps)
`dither`	False	triangular dither of ±1 DAC step
`limits`	`{"low": None, "high": None, "protect": False}`	V; the FPGA clamps every sample; `protect` locks them until the FPGA is reloaded
`arb`	see below	this output's half of the sequencer's point pairs

A sweep excludes bursts and angle modulation. The applied configuration has the realized values (e.g. frequencies on the 0.44 µHz grid).

Arbitrary waveforms

One sequencer plays **point pairs** for both outputs; each output plays its half of them (instead of its noise) when `outputs.n.arb.on`. The sequencer (`SEQUENCER`, the top-level `arb`):

```
{"sequence": "",                  # a saved sequence, or a saved segment (played forever)
 "rate": 125e6,                   # pairs per second (every pair is played once), up to 125e6
 "run": "continuous",             # "continuous" (from the apply), "triggered", "gated"
 "trigger": {"source": "software", "edge": "rising", "delay": 0.0},   # starts; waiting steps
 "gate": {"source": "trig_b", "polarity": "high", "finish": True},    # gated: finish the repetition
 "jump": {"source": "event0", "edge": "rising"}}                       # the steps' jump events
```

`run` `continuous` starts the sequence 20 samples (160 ns) + `trigger.delay` after the apply; `triggered` at each trigger (the trigger also starts steps that wait); `gated` while the gate is open (`finish`: to the end of the repetition). `trigger.delay` has 31 ps steps. The sequencer uses OUT1's run control: OUT1 has no bursts or sweeps while it plays (its noise and generators are free).

An output's `arb`:

```
{"on": False,
 "filter": "smooth",              # "smooth" (band-limited, a delay of 6 pairs), "linear" (1),
  ↪  "step" (none)
 "amplitude": 0.0,                # V at the point value 1.0 (peak)
 "point_offset": 0.0,             # added to every point (-1 ... 1): with upconversion, the carrier
  ↪  leakage
 "idle": "start",                 # before the start and after the end: "start" (the first point),
 "idle_level": 0.0,               #   "last" (hold), or "dc" (idle_level, -1 ... 1)
 "upconvert": {"on": False,       # OUT = amplitude (I cos(2 pi f t + phase) - q_gain Q sin(... +
  ↪  q_phase)):
              "frequency": 10e6, "phase": 0.0,      #   I: this output's half, Q: the other output's
              "q_gain": 1.0, "q_phase": 0.0}}       #   (which needs on, with amplitude 0 if unused)
```

Noise and sweeps are off on an output with `arb.on`; upconversion also needs `am` and `angle` off. Generator 1 and `aux` stay summed (without upconversion).

Segments are lists of at least 64 pairs: OUT1's and OUT2's points in [−1, 1] (14 bits), each with two marker bits (marker sources `arb1`, `arb2`). **Sequences** are lists of steps:

Key	Default	Meaning
`segment`		the segment's name
`repeat`	1	times; 0: until an event jumps away (or forever)
`wait`	False	wait for the trigger before the step
`wait_level`	0.0	both outputs while waiting (−1 ... 1), or `"hold"` (the last points)

Key	Default	Meaning
`marker`	False	a start pulse on markers with source `start`
`next`	`"follow"`	the following step, `"end"`, or a step index (goto)
`on_event`	`"none"`	`"end"`: at the end of the repetition, `"now"`: at once, jump to `event_target` on a `jump` event
`event_target`	0	a step index
`digital`	`"none"`	on the step's first point: start this program of `digital.programs` (by name), or `"stop"`

At most 512 steps; at most 3 different event targets besides step 0. A sequence plays its first step and its event targets from a cache of their first 512 pairs, the rest from DDR (see THEORY.md for the timing).

Digital outputs

Eight 3.3 V outputs D0–D7 on DIO0_N–DIO7_N play one 8-bit word per sample (bit k drives Dk). `DIGITAL`:

```
{"run": "off",                    # "off" (the idle word), "continuous" (program from the apply on),
                                  #   "sequencer" (started/stopped by steps), "out1"/"out2" (program
                                  #   from each start of that output: burst, sweep or sequence start)
 "program": "",                   # the program of continuous/out1/out2
 "programs": {},                  # up to 8 programs by name (a patch replaces the whole set)
 "idle": 0,                       # the word when nothing plays (0 ... 255)
 "enable": 0,                     # pins driven (bit k: Dk); the others are high impedance
 "delay": 0.0}                    # s relative to the analog outputs (8 ns steps; -112 ... +3968 ns
                                  #   plus the board calibration's pins_delay_s)
```

A program:

```
{"mode": "memory",                # "memory" (a saved pattern), "binary", "gray" (counts), "prbs"
 "pattern": "",                   # memory: a saved pattern's name
 "start": 0,                      # memory: the first word's index; counts: the first value (0 ... 255)
 "length": 0,                     # words per pass; 0: memory to the pattern's end, counts 256,
                                  #   prbs one period (2^order - 1)
 "rate": 125e6,                   # words per second: 125 MHz / an integer (the applied value is exact)
 "repeats": 0,                    # passes; 0: forever
 "end": "idle",                   # after the last pass: "idle" (the idle word) or "hold" (the last
  ↪  word)
 "order": 7}                      # prbs: 7, 9, 11, 15, 20, 23, 31
```

Patterns are saved lists of 1 ... 32768 words (0 ... 255); the programs in use share the 32768-byte pattern memory. A PRBS puts its sequence on D0 and the same sequence k words earlier on Dk. With `delay` 0 a word changes with the analog sample it belongs to; the board calibration's `pins_delay_s` (common/CALIBRATION.md) corrects the pins' own delay (the same for the markers' `delay`). Starts and timing: THEORY.md, "Digital outputs".

Python (driver on the board, `AwgClient` on a PC)

Method	Meaning
`set_output(n, **settings)`	patch output n; returns its applied settings
`config()`, `apply(patch, replace=False)`, `describe()`, `status()`, `safe()`, presets	common API
`function(n, name, Hz, V, deg)` (client)	the function shortcut (TCP FUNC)
`trigger(n=None)`	software trigger of output n, or both
`zero_time(delay=None)`	t0 := the next apply sample; re-apply
`protect(n)`	lock output n's limits until the FPGA is reloaded
`table_names()`, `save_table(name, values)`	built-in and saved tables; save one (values in [-1, 1])
`hardware()`	counters, states, levels, last start times, protection
`clear_counts()`	trigger, missed and clip counts to 0
`capture(delay=0.001)`	8192 samples of IN1 and IN2: `{"start", "in1", "in2", "volts"}`
`save_segment(name, values, markers=None, values2=None, markers2=None)`	save a segment of pairs (OUT1's values in [−1, 1], OUT2's `values2`, or the same; markers per point: bit 0 marker 1, bit 1 marker 2); lists or numpy arrays: with numpy, 4 M pairs take about 5 s on the board, without it 34 µs per pair (measured)

Method	Meaning
`segment_names()`, `segment_values(name, start=0, count=None)`, `delete_segment(name)`	saved segments; a segment's pairs `{"1": {"values", "markers"}, "2": ...}`
`load_segment_file(name, path)` (driver)	a segment from a `.npy` or CSV file on the board (columns out1 [, out2 [, markers1 [, markers2]]])
`save_sequence(name, steps)`, `sequence(name)`, `sequence_names()`, `delete_sequence(name)`	sequences
`play(**settings)`	patch the sequencer's settings (TCP PLAY)
`set_arb(n, **settings)` (client)	patch output n's `arb` settings (TCP ARB)
`save_pattern(name, values)`	save a pattern: 1 … 32768 words (integers 0 … 255, or bytes)
`pattern_names()`, `pattern_values(name)`, `delete_pattern(name)`	saved patterns
`set_digital(**settings)`	patch the digital outputs' settings (TCP DIGITAL); returns them as applied
`set_pins_delay(seconds)`	store the board calibration's pins delay (common; TCP `CAL PINS`)

TCP commands (port 5025)

Command	Reply
`OUT <o> <json>`	patch output o; its applied settings (JSON)
`OUT <o>?`	its applied settings
`FUNC <o> <function> [<Hz> [<V> [<deg>]]]`	function, frequency, amplitude, phase
`FREQ <o> <Hz>`, `VOLT <o> <V>`, `OFFS <o> <V>`, `PHAS <o> <deg>`	one setting
`NOISE <o> <V rms> [<Hz>]`	added noise and its bandwidth
`TRIG [<o>]`	software trigger
`ZERO [<s>]`	t0 := the next apply sample (`OK <t0>`)
`PROTECT <o>`	lock output o's limits
`TABLES?`	`{"builtin": [...], "saved": [...]}`
`TABLE <name> <json values>`, `TABLEZ <name> <base64 zlib>`	save a table
`SEGMENT <name>` + binary data	save a segment: a `BINARY` request (see common/README.md) of 32-bit little-endian pair words (OUT2's point word << 16, OUT1's; a point word {value << 2, m2 << 1, m1})
`SEGMENT? <name> [<start> <count>]`	its pairs `{"1": {"values", "markers"}, "2": ...}` (at most 20000 with a range; replies are limited to 1 MB)
`SEGMENTS?`, `SEGDEL <name>`	saved segments; delete one (not one the applied configuration plays)
`SEQUENCE <name> <json steps>`, `SEQUENCE? <name>`, `SEQUENCES?`, `SEQDEL <name>`	sequences; delete one (not the one played)
`PLAY <json>`, `PLAY?`	patch, read the sequencer's settings
`ARB <o> <json>`	patch output o's `arb` settings
`PATTERN <name>` + binary data	save a pattern: a `BINARY` request of one byte per word
`PATTERN? <name>`, `PATTERNS?`, `PATDEL <name>`	a pattern's words; saved patterns; delete one (not one a program of the applied configuration plays)
`DIGITAL <json>`, `DIGITAL?`	patch, read the digital outputs' settings
`HW?`	hardware state (JSON; `arb`: the sequencer's state, step, underflow, FIFO; `digital`: the pattern engine's state, program index, starts)
`CLEAR`	counts to 0
`CAPTURE? [<s>]`	`{"start", "in1", "in2"}` (codes)

Errors reply `ERR <message>`; a refused setting changes nothing.

Command line

```
python3 -m rpfpga.cli [--host H] [common options] [awg options]
```

Option	Meaning
`--func O=F[,HZ[,V[,DEG]]]`	output O's function, frequency, amplitude, phase (repeatable)
`--offset1 V`, `--offset2 V`	offsets
`--noise1 RMS[,HZ]`, `--noise2 ...`	added noise
`--play NAME[,RATE]`	the sequencer plays a sequence or segment at RATE pairs/s
`--arb O=V[,FILTER]`	output O plays its half of the pairs, V at value 1 (`O=off`: stop)
`--segment NAME FILE`	save a segment from a `.npy` or CSV file on this machine (columns out1 [, out2 [, m1 [, m2]]])
`--pattern NAME FILE`	save a pattern from a text file on this machine (words 0 … 255: decimal, `0x` hex or `0b` binary)
`--digital JSON`	patch the digital outputs' settings
`--trigger [O]`	software trigger (both without O)
`--zero`	t0 := the next apply sample

awg registers

AXI4-Lite at 0x43C00000, 256 KiB (18-bit byte addresses), 32-bit words. Reading the low word of a 64-bit value latches its high word (read the high word next). The system-ID block is at 0x40000000 (common). Never access these registers outside `rpfpga.Overlay` (see `../../common/README.md`, "Safe loading").

Map

Address	Contents
0x00000–0x000FF	control, status, counters, apply time, capture, scratch
0x00100–0x00147	global staged words (18)
0x00200–0x002FF	OUT1 staged words (64: generator 1 0–15, generator 2 16–31, output 32–58)
0x00300–0x003FF	OUT2 staged words
0x08000–0x0FFFF	capture buffer: word i = {IN2, IN1} of sample CAP_START + i (8192 words)
0x10000–0x1FFFF	tables: table t (0 OUT1 gen 1, 1 OUT1 gen 2, 2 OUT2 gen 1, 3 OUT2 gen 2) at 0x10000 + t·0x4000, point i at + 4i (18-bit signed; reads are sign-extended)
0x20000–0x27FFF	sweep segments: OUT n at 0x20000 + (n−1)·0x4000, segment j at + 32 j, words 0–5; writing word 5 stores the segment
0x28000–0x2BFFF	the sequencer's steps: step j (0–1023) at 0x28000 + 16 j, words 0–3; writing word 3 stores the step
0x30000–0x33FFF	its hot cache: pair i (slot s: pairs 512 s …) at 0x30000 + 4 i
0x34000–0x340FF	the pattern engine's programs: entry j (0–15; two halves of 8) at 0x34000 + 16 j, words 0–3; writing word 3 stores the entry
0x38000–0x3FFFF	the pattern memory: bytes 4 i … 4 i + 3 (byte k in bits 8k + 7 … 8k) at 0x38000 + 4 i (readable)

The arbitrary points are pairs (32-bit words: OUT2's point word in bits 31–16, OUT1's in 15–0) read from DDR (the PS7's S_AXI_HP0 port, AXI3, 64-bit beats of two pairs, ARID 0) at physical addresses ARB_BASE + 64·(step address) + 4·pair.

Control and status (0x000–0x0FF)

Offset	Name	Access	Contents
0x000	ID	R	0x41574745 "AWGE"
0x004	VERSION	R	0x00030000 (3.0)
0x008	CTRL	W	strobes: 0 APPLY_NOW, 1 APPLY_AT, 2 CANCEL (a scheduled apply), 3 MUTE, 4 CLEAR (counts), 5 TRIG1, 6 TRIG2 (software triggers), 7 CAPTURE, 8 MEAN, 9 CAPTURE_STOP
0x00C	STATUS	R	0 apply pending, 1 applying, 2 last apply late, 3 muted, 4 capture busy, 5 capture done, 6 capture late, 7 mean busy, 8 mean done, 9 trigger clock locked, 10–11 PROTECT
0x010/0x014	COUNT	R	sample counter (64 bits)

Offset	Name	Access	Contents
0x018	LATENCY	R	20: counter value n → output sample n on the DAC stream
0x01C	CAPTURE_SAMPLES	R	8192
0x020/0x024	APPLY_TIME	RW	the sample of an APPLY_AT
0x028/0x02C	LAST_APPLY	R	the sample of the last apply
0x030, 0x034	APPLY_COUNT, LATE_COUNT	R	applies; late ones (APPLY_TIME less than 21 samples ahead)
0x038, 0x03C	STATE1, STATE2	R	{sweeping, running, armed}
0x040, 0x044	TRIGS1, TRIGS2	R	accepted starts
0x048, 0x04C	MISSED1, MISSED2	R	events refused (busy)
0x050, 0x054	CLIPS1, CLIPS2	R	samples clamped to the limits or the DAC range
0x058	LEVELS	R	[8:1] levels of TRIG A, TRIG B, EVENT 0–3, IN1/IN2 thresholds; [16] the trigger clock's MMCM locked
0x060/0x064	CAP_TIME	RW	the first sample of a capture
0x068/0x06C	CAP_START	R	the first sample captured
0x070	MEAN_N	RW	samples of a mean
0x074/0x078, 0x07C/0x080	SUM1, SUM2	R	48-bit sums of IN1, IN2 (sign-extended)
0x084	PROTECT	R, W1S	lock the limit word of output n (bit n−1) until the FPGA is reloaded
0x088	LOADS	R	LOAD flags written since the last apply: {OUT2 gen2, OUT2 gen1, OUT1 gen2, OUT1 gen1}
0x090–0x09C	SCRATCH0–3	RW	driver: flags (1 continuous, 2 no droop correction, 4 allow clip), t0 low, t0 high, apply delay (µs, 0 = 10 ms)
0x0A0/0x0A4, 0x0A8/0x0AC	LAST_START1, 2	R	start time T (ticks of 1/256 sample) of the last accepted start
0x0C0–0x0DC	DATA_ID0–7	RW	driver: table contents IDs (4), sweep IDs (2), function codes, the sequence ([14:0] ID, [15] the step table's half)
0x0E0	ARB_BASE	RW	physical address of the DDR buffer of the arbitrary points (64-byte aligned)
0x0E4	ARB_SIZE	RW	its size in bytes: reads beyond it stop the fetch (WINDOW error)
0x0E8	ARB_STATUS	R	[1:0] the sequencer: 0 idle, 1 waiting, 2 playing; [11:2] the step being turned into pairs; [14] UNDERFLOW (a pair was due with none ready), [15] WINDOW (sticky, cleared by a restart or CLEAR); [27:16] beats in the FIFO
0x0EC	DIG_STATUS	R	the pattern engine: [1:0] 0 idle, 1 running, 2 holding its last word; [4:2] its program (in the live half)
0x0F0	ARB_FIFO_MIN	R	[11:0] the fewest beats left in the FIFO when the sequencer took one, since the last restart or CLEAR (2048: none taken)
0x0F4	DATA_ID8	RW	driver: the programs and patterns ([14:0] ID, [15] the half of the program table)
0x0F8	DIG_STARTS	R	program starts since CLEAR

Staged words

Each output's words at 0x200 + (n−1)·0x100 + 4w; a write to a staged word waits (no BVALID) while an apply sequence runs.

Generator words (generator 1: w = 0–15, generator 2: w = 16 + 0–15)

w	Name	Bits	Contents
0, 1	F	48	frequency word: 2^-48 cycle per sample
2, 3	P	48	phase preload (written at each coherent apply; not configuration)
4	POW	32	phase offset, 2^-32 cycle
5	SHAPE	6	[2:0] 0 off, 1 sine, 2 trapezoid, 3 table, 4 PRBS; [5:3] PRBS order 7, 9, 11, 15, 20, 23, 31
6, 7	R	49	trapezoid: end of the rising edge (2^-48 cycle)
8, 9	HE	49	end of the high part

w	Name	Bits	Contents
10, 11	FE	49	end of the falling edge
12	SLOPE_R	29	[23:0] a, [28:24] s of the rising edge (and PRBS edges)
13	SLOPE_F	29	the same for the falling edge
14	LFSR	31	PRBS register preload (at a LOAD; burst starts: generator 1's seed)
15	LOAD	W	bit 0: load P and LFSR at the next apply; bit 1 (OUT1's generator 1): restart the arbitrary-waveform sequencer at the next apply; bit 2 (OUT1's generator 1): restart the pattern engine at the next apply

Output words (w = 32–58)

w	Name	Bits	Contents
32, 33, 34	A1, A2, AN	24	amplitudes of generator 1 (c), generator 2 and the noise, in 2^-27 DAC steps per unit
35	OFFSET	32	signed, 2^-16 DAC step
36	FLAGS	2	0 dither
37	CLAMP	30	[13:0] low, [29:16] high (14-bit DAC codes); locked by PROTECT
38	MOD	20	[3:0] angle 0 off, 1 FM, 2 PM, 3 PWM, 4 FSK, 5 BPSK; [5:4] AM 0 off, 1 AM, 2 DSB; [9:8] AM source, [13:12] angle source (0 gen 2, 1 IN1, 2 IN2, 3 the other output); [19:16] key (0 gen 2, else an event source with a level)
39	DEPTH	18	AM: k = 2^15 + (DEPTH m) >> 17 (DSB: without 2^15)
40, 41	DEV	42	FM/PM/PWM: (DEV m) >> 10
42, 43	FHOP	48	FSK hop frequency word
44	PBPSK	32	BPSK phase, 2^-32 cycle
45	NRATE	32	point rate of the resampler (noise or arbitrary points), 2^-32 of the sample rate
46	RUN	24	[1:0] 0 continuous, 1 burst, 2 gated; [3:2] idle 0 start, 1 last, 2 DC; [4] gate finishes the cycle; [5] gate open when high; [11:8] trigger source; [13:12] edges (1 rising, 2 falling); [19:16] gate source; [21:20] sweep 0 off, 1 continuous, 2 triggered
47	NCYC	32	burst cycles (0 endless)
48, 49	THS	48	burst start phase (2^-48 cycle)
50, 51	STOP	48	stop phase offset after the last cycle
52, 53	DELAY	56	trigger delay, ticks of 1/256 sample
54	DC	18	idle DC level, shape units
55	ARB	22	[0] the sequencer on (OUT1's word; it then uses OUT1's RUN, DELAY and trigger logic); [2:1] this output's filter 0 smooth, 1 linear, 2 step; [3] I/Q upconversion; [4] the sequencer's rate 2^32 (125 MSa/s); [14:5] its start step; [18:15] its jump event source; [20:19] its edges ([4], [14:5], [20:15]: OUT1's word); [21] this output takes its half of the pairs (instead of noise)
56	POFF	18	this output's point offset (added to its points, taken at the sequencer's restart), signed, 2^-4 of a 14-bit point step
57	AIDLE	15	this output's idle point before the start step and after an end: [13:0] level (a 14-bit point value), [14] the last point instead
58	ARATE	32	the sequencer's pair rate, 2^-32 of the sample rate (OUT1's word)

An output taking points has its AN multiply its resampler's points; with I/Q upconversion generator 1 (scaled by DEPTH: k = DEPTH) multiplies its own points (sum term A1) and generator 2 (scaled by DEV: s = (DEV g2 >> 10) >> 16) the other output's (sum term A2).

Event sources: 1 TRIG A, 2 TRIG B, 3–6 EVENT 0–3, 7 IN1 threshold, 8 IN2 threshold, 9 software (the output's own), 10 timer 1, 11 timer 2, 12 the other output's start.

Global words (0x100 + 4w)

w	Name	Contents
0, 1	T1_START	timer 1's first event, ticks (64 bits; written at each apply, aligned to t0)
2, 3	T1_PER	timer 1's period, ticks (56 bits)
4–7	T2_START, T2_PER	timer 2
8	TIMERS	[0] timer 1 on, [1] timer 2 on

w	Name	Contents
9, 10	CMP1, CMP2	IN1/IN2 threshold: [15:0] high, [31:16] low (ADC stream codes)
11	IN_OFF	[15:0] IN1, [31:16] IN2 offsets (ADC codes) subtracted for modulation
12, 14	MARK1, MARK2	[2:0] 0 low, 1 high, 2 sync, 3 run, 4 sweep, 5 start pulse, 6 point marker 1, 7 point marker 2; [3] channel 2; [4] invert; [24:16] delay (samples, 18 aligned)
13, 15	MARK1_WIDTH, MARK2_WIDTH	start pulse width, samples
16	DIG	[2:0] the pattern engine: 0 off (the idle word), 1 continuous (starts PROGRAM at an apply with LOAD bit 2), 2 sequencer (step actions), 3 at OUT1's starts, 4 at OUT2's starts (PROGRAM); [5:3] PROGRAM; [6] the half of the program table; [7] the actions come with OUT2's points (else OUT1's); [15:8] the idle word; [24:16] delay (samples, 15 aligned)
17	DIG_OE	[7:0] D0–D7 driven (1) or high impedance (0)

Sweep segment words

Word	Contents
0, 1	start word (48 bits: F, or log2 F with 42 fraction bits for LOG)
2, 3	rate (56 bits signed, 24 fraction bits per sample)
4	count (samples, at least 4)
5	flags: 1 LOG, 2 JUMP, 4 MARK, 8 LAST (writing this word stores the segment)

Sequencer step words

Word	Bits	Contents
0	[19:0]	segment start in DDR, units of 16 pairs (64 bytes) from ARB_BASE
	[22:20]	hot-cache slot (when the step is a start or jump target)
	[23], [24]	marker (start pulse), wait for the output's trigger
	[26:25]	next: 0 the following step, 1 goto (word 3), 2 end (idle; a restart begins at the start step)
	[28:27]	event: 0 none, 1 jump at the end of a repetition, 2 jump at once
	[29]	while waiting: the last points instead of the level (both outputs)
1	[24:0]	segment length, pairs (64 … 2^25 − 1)
	[31:25]	waiting level [6:0]
2	32	repeats (0: forever)
3	[9:0], [19:10]	goto target, event target
	[26:20]	waiting level [13:7]
	[28:27]	pattern action on the step's first point: 0 none, 1 start program [31:29] (of the live half), 2 stop
	[31:29]	the program

Pattern programs

Word	Bits	Contents
0	[1:0]	mode: 0 memory (bytes from START), 1 binary count (from START[7:0]), 2 Gray count, 3 PRBS (D0 the newest bit, Dk the bit k words earlier)
	[4:2]	PRBS order 7, 9, 11, 15, 20, 23, 31 (the LFSR of the PRBS shape; all ones at the start)
	[5]	at the end: hold the last word (else the idle word)
	[30:16]	START: a memory address (wrapping at 32768), or a counter's first value
1	32	words per pass − 1
2	32	samples per word − 1
3	32	passes (0: forever); memory and counters begin each pass at START, the PRBS goes on

A point word is {value[13:0], marker 2, marker 1} (value −8191 … 8191 for −1 … 1); an output's resampler takes (value << 4) + POFF, saturated to 18 bits.

awg verification records

Newest first. Only results actually obtained are recorded; unmeasured quantities are marked as such.

2026-11-09: an asynchronous laser with a chopper for boxcar

- awg 3.0 on 192.168.0.104 (nominal calibration) as a pump–probe laser for boxcar 1.0 on a second board: two sequencer steps (pump on, pump off), each a segment of one shot (64 to 125,000 points at 125 MSa/s, a pulse of rise 10 ns and decay 40 ns, smooth filter) on OUT1 and OUT2, with a digital program per step (sync on D0, chopper on D1; with longer segments of 1000 shots, per-shot random factors common to both outputs and NEXT pulses on D2). boxcar took every shot from 1 kHz to 1.95 MHz (4 million shots), the categories equal to awg's chopper; awg's crystal 4.3 ppm fast. $\Delta T/T$ from awg's quantized points (0.019984) was measured within $\pm 4 \cdot 10^{-4}$ for pulses up to 0.68 V at boxcar's IN1 (the two boards' converters together). A triggered sequence on TRIG A from boxcar's P1 started each pulse; its rise reached boxcar's IN1 392 ns after P1's edge. boxcar's receipt: `boxcar/docs/data/board-1.0-awg-session-2026-11-09.json`.

2026-11-06: analog pulses and digital marks into counter

- awg 3.0 on 192.168.0.104 (nominal calibration): OUT1 cabled to counter's IN1; a sequence step looping a 256-point segment with a digital program per play (TRIG on D2, a mark on D0 per pulse): 1.98 million marks per second, one play every 2.048 µs (gapless). Counter's C9 (aligned with its pins to 0.5 ns) sees the analog pulses 33.5 ns after their digital marks: awg's analog-to-digital alignment on a board without a measured calibration, plus the cable. Not measured: the same on a calibrated board. Receipt: `../counter/docs/data/board-1.0-awg-tests-2026-11-06.json`.
- A negative pulse smoothed by the "smooth" filter overshoots positive enough to cross +0.05 V once per pulse at the counter (−0.75 V pulses).

2026-11-05 (evening): awg as a source of known events for counter, timetagger and feedback

- **Setup:** awg 3.0 on 192.168.0.104 with the afternoon's wiring to the second board (192.168.0.102), which ran counter, then timetagger, then feedback. Receipt: `../counter/docs/data/board-1.0-awg-tests-2026-11-05.json`; results in counter's, timetagger's and feedback's VERIFICATION and the guide `guides/combining`.
- **Patterns:** 32768-word random patterns played an exact number of passes (`repeats`, end `idle`) gave exact expectations for counter's sums, records and clusters and timetagger's coincidence engine. A finished program restarts only after `run: off`.
- **Sequencer loop** (repeat until success): a step of 4.096 µs starting a TRIG pattern and jumping on EVENT0 rising, then a step answering on D1: 204167 jumps in 2047844 attempts, none missed or spurious (every success answered once); the attempt after a jump began 2608 ns after the jumping attempt's TRIG.
- **Edges on a 20 cm jumper**, measured by counter's sub-ns statistics: an 8 ns pulse's rising edge 95 ps early 8 ns after the previous pulse's end and 94 ps late 16 ns after it (the wire's ringing). Two outputs rising in the same word made false falling edges on wires idling high (crosstalk; the guide's section on it).
- **Clock:** 5.02 ppm (counter), 5.00 ppm (timetagger), 5.05 ppm (feedback, 1 s gate) fast against the other board's crystal.

2026-11-05: digital outputs tagged by timetagger on a second board

- **Setup:** awg 3.0 (0x97cda2f3, nominal calibration) on 192.168.0.104; its D0, D1 jumpered (Dupont, about 20 cm) to timetagger's channels 1, 2 (picoseconds) and D2, D3 to its markers 1, 4 (1 ns) on 192.168.0.102 (timetagger 2.0, 0x9b2ec69e); grounds joined. Receipt: `guides/combining/data/awg-timetagger-2026-11-05.json` (the guide `guides/combining/`).
- **Period and jitter:** D0's 8 ns pulse every 125 words: 2 million periods of 999994.983 ps in timetagger's time, so awg's crystal runs 5.02 ppm fast against the timetagger board's (that board reads a GNSS 10 MHz 5.66 ppm high: awg's crystal is then about 0.6 ppm slow, inferred); one period's rms 30.0 ps, of which about 21 ps are timetagger's two tags: about 21 ps rms of awg's output and the two crystals together.
- **Pins:** D1 20.0 ps after D0 for edges in the same word (rms 25.8 ps), including the two jumpers' lengths; a one-word (8 ns) pulse 8.79 ns wide at timetagger's threshold (rms 30.8 ps).

- **Crosstalk:** D0 and D1 rising in the same word fired timetagger's markers on the D2 and D3 wires (marker 4 on every edge, marker 1 on 75--85 %, hundreds of thousands of glitches per second), with D2, D3 driven low or not driven; not with D2, D3 driven high, not with D0 and D1 a word apart, not with either alone; a second ground jumper did not help: coupling in the jumper bundle, not awg's drivers or the ground path (at awg's own quiet pins at most 0.65 V on a probe).
- **Not measured:** D2--D7's timing against D0 with picoseconds; the outputs into 50 ohms.

2026-10-24: driver fixes from a code review (board check, `make test-hw`)

- **Changes (driver only, same build 0x97cda2f3):** derived configurations keep the marker sources `arb1`/`arb2`; segments too large for the buffer leave the layout in place; an apply that writes step or program tables waits for a pending apply; the exit handler silences outputs that played points and holds no driver; one DDR buffer per process; segments, sequences and patterns the applied configuration plays are not deleted; the pins delay is limited to ±100 ns (common calibration).
- **Board:** five LOADs of awg, each playing an arbitrary waveform: free contiguous memory stayed at 12 MB (one 64 MB buffer). 20 × three sequence changes issued at once (flat, two-step, ramp): no underflow or window error, the ramp played. Deleting the playing segment was refused. `make test-hw`: 57 of 57 checks pass.

2026-10-24: loopback measurements (`make test-hw`, 57 checks)

- **Setup:** awg 3.0 (build 0x97cda2f3), OUT1 → IN1 and OUT2 → IN2 with 50 Ω terminators, DIO1_P → DIO0_P, DIO0_N → DIO3_P; the board calibration with `pins_delay_s` = −28.6 ns. `scripts/test_hw.py` now also demodulates and analyses the captures with numpy on the board. 57 of 57 checks pass (two runs; the numbers below are the second's, the first agreed within the scatter).

Check	Measured	Expected
noise 0.1 V rms, 10 MHz: 3–7 MHz vs 0.5–3 MHz (16 captures, Welch)	+0.24 dB	0 ± 1 dB
its −3 dB point	10.10 MHz	10 ± 1 MHz
16–40 MHz vs the pass band	−51.3 dB	below −35 dB
kurtosis (131,072 samples)	3.003	3 ± 0.1
beyond 3σ	0.272 %	0.27 ± 0.05 %
linear sweep 1 → 3 MHz in 40 µs: rate / set	0.99985	1 ± 0.01
its deviation from a line (rms)	4.5 kHz	below 20 kHz
log sweep 1 → 4 MHz in 40 µs: octaves per second / set	0.99999	1 ± 0.01
its deviation from an exponential (rms)	0.005 octave	below 0.01
FM of 5 MHz by generator 2 (100 kHz), deviation	499.5 kHz	500 ± 10 kHz
PM by generator 2, deviation	89.999°	90 ± 2°
FSK keyed at 50 kHz: hop / base frequency	7.00001 / 4.99998 MHz	7 / 5 ± 0.01 MHz
its time at the hop	0.510	0.5 ± 0.05
BPSK keyed at 50 kHz: phase step / time flipped	179.2° / 0.505	180 ± 3° / 0.5 ± 0.05
IN2 threshold (0.1 V) on a 20 kHz square: OUT1 starts / edges	0.99989	1 ± 0.01
threshold to burst start, spread over 6 bursts	0.005 sample	below 1 sample
sequence: a software event jumps to the next step	step 0 → 1	+1
the jump target's level	0.192 V	0.2 ± 0.02 V
sequence gated by D0 on TRIG B, D0 = 1 / 0 / 1	playing / idle / playing	the same
IN1 rms gate closed / open	0.008	0 ± 0.05
FSK keyed by D0 on TRIG B = 1 / 0	7 / 5 MHz	the same

- **Information:** the noise's largest sample was 4.47σ in 131,072 (Gaussian: about 4.5σ expected); the burst on the IN2 threshold started 58.56 samples after IN2 crossed 0.1 V in the capture (the comparator, the trigger path and the loopback through OUT1 and IN1); the FSK frequencies were measured from the instantaneous frequency (32-sample average), the FM deviation by fitting it, PM and BPSK from the analytic phase.
- **The first run's two failures:** the threshold check used a 20 kHz sine, whose slope of 0.4 mV per sample against about 1 mV of input noise moved the comparator's decision by up to 4.8 samples (the square's edges fixed it); and restoring the user configuration at the end failed: with `pins_delay_s` set, a marker at the shortest delay line setting is reported at −164.6 ns, outside the uncalibrated range the driver's validation still used (fixed: the validation includes the pins delay; a test covers it).
- **Still not measured:** the AM/DSB and PWM shapes beyond the 50 % AM depth check, list and profile sweeps,

EVENT 0–3 pins, timer jitter, jitter against an asynchronous trigger.

2026-10-24: trigger-to-output latency (oscilloscope)

- **Setup:** awg 3.0 (build 0x97cda2f3), the board calibration with `pins_delay_s` = −28.6 ns. OUT2 starts a 1-cycle burst every 100 µs (timer 1); marker 1 (OUT2's start pulse) is wired to TRIG A; OUT1 starts a 1-cycle 1 MHz square burst on TRIG A's rising edge, delay 0, idle at −0.4 V, its first rising edge's 50 % point (0 V) at the start instant. Scope as below (OUT1 by coax on channel 1, the TRIG A node by the probe on channel 2).
- **Measured:** TRIG A at 1.65 V to OUT1's 0 V crossing: 379.6 ns on the scope, **377.9 ns** at the connectors (the probe leads the coax by 1.7 ns). The device's start times put OUT1's start 46.25 samples after OUT2's (42.25 before the pins calibration moved marker 1 by 4 samples).
- **Model:** 160 ns from the trigger to the start (L_TRIG, 20 samples) and 160 ns through the core (LATENCY), the rest (about 58 ns) the input pin and ISERDES, the DAC interface, the converter and the analog output path. The start times and the pins calibration predict 374.3 ns on the scope; the 5.3 ns difference is not explained (the marker's slow edge, the probe position and I/O delays outside the model are candidates).
- **Not measured:** jitter against an asynchronous trigger (the marker loop is synchronous with the sample clock, so the pins' 1 ns quantization does not show); TRIG B, EVENT and threshold triggers (EVENT pins are sampled at 8 ns, so their latency differs by up to a sample).

2026-10-24: pins delay against the analog output (oscilloscope)

- **Setup:** awg 3.0 (build 0x97cda2f3) playing a step-filtered square of point pairs at 125 MSa/s (32 points at −1, 32 at +1; OUT1 0.4 V: a −0.4 → +0.4 V step on one sample boundary every 512 ns), marker 1 on the points' marker bit (set on the first 4 high points), delay 0. Siglent SDS2104X Plus (500 MHz); OUT1 by 1 m RG316 into the scope's 50 Ω (channel 1), DIO1_P with a PMK PML711A probe (channel 2).
- **Before the calibration:** the marker's 1.65 V crossing came 30.3 ns before OUT1's 0 V crossing. The probe on OUT1's SMA (shared with the coax) led the coax by 1.7 ns, so at the connectors the pins lead OUT1 by **28.6 ns**. Stored as the board's `pins_delay_s` = −28.6 ns (`CAL PINS`).
- **After:** the driver delays the markers and D0–D7 by 4 samples (32 ns): the scope read the marker 1.6 ns after OUT1 (expected 1.7 ns), i.e. **3.3 ns after OUT1 at the connectors**, the best the 8 ns grid allows. D0 matches marker 1 to 1 ns (TRIG B vs TRIG A loopback, same day), so the digital outputs land the same way.
- **Not measured:** OUT2 (0.17 ns from OUT1 by the stored skew); the pins' rise times and the delay's dependence on load, temperature or the 3.3 V level threshold.

2026-10-24: awg 3.0 on the board (build 0x97cda2f3)

- **Build** (Vivado 2025.1, xc7z010clg400-1; post-route physical optimization from `design(impl_properties)`): routed timing met, WNS +0.003 ns, WHS +0.022 ns (47,311 endpoints); routed DRC warnings only (DSP pipelining, one unloaded net, and AVAL-250 for the eight digital pins: their output and tristate IOB registers start at 0 and 1, so a pin is high impedance until enabled). Utilization (placed): 13,869 LUTs (78.8 %), 22,054 flip-flops (62.7 %), 58 block-RAM tiles (96.7 %), 48 DSP48 (60.0 %), 2 MMCMs. Earlier 3.0 builds: −0.17 ns (the start product's reset on the trigger path), −0.27 ns (the pattern engine's start path), −2.65 ns from routing congestion alone, and one that failed to route with the placer directive AltSpreadLogic_high; the program table and the digital delay line then moved from LUT RAM into block RAM.
- **`common/tests/verify_build.py --design awg`**: all 8 checks pass (recipe build with `design(impl_properties)`, unchanged reuse, missing export, vendor HDL edit, design RTL edit, a failed build blocks the upload, clean, saved-project regeneration); the recipe build reproduces build ID 0x97cda2f3.
- **`make test-remote`**: loaded through the driver, sysid AWGE 3.0 and the build ID verified, power-up state verified, preset `off`.
- **`make test-hw`**: 26 of 27 checks pass: the 26 of 2.0, with the same results within their scatter (I/Q image −91.6 dB, carrier −91.5 dB this run). The digital check "D0 held high reads high on TRIG B" fails: with D0–D7 driven high, TRIG B (and every other input pin) stayed low, so the jumper DIO0_N → DIO3_P was not in place (or the outputs do not drive; this cannot be told apart without the jumper), and the D0 timing checks did not run. They were run later with the jumper (next item).
- **`make test-hw` with DIO0_N wired to DIO3_P** (later the same day): 33 of 33 checks pass, the digital ones with the same results as the run through the service below (D0 and marker 1 both 42.25 samples after OUT2; +8 ns moves D0's start by one sample; 1614 pattern starts and 1615 OUT2 starts through D0 in about 0.2 s).

- **Digital pins through the service** (DIO0_N wired to DIO3_P; from the PC with `AwgClient`, the checks of `scripts/test_hw.py` plus a −8 ns step, while SSH from the build machine's session failed: see the observation below): 9 of 9 pass.

Check	Measured	Expected
D0 held high / low: TRIG B's level	high / low	high / low
D0 pulse at each OUT2 start → TRIG B → OUT1 start, spread of 5 reads	0	≤ 0.125 sample
digital delay +8 ns / −8 ns: the start moves	+1 / −1 sample	±1 sample
D0 (TRIG B path) vs marker 1 (TRIG A path)	0.00 ns	0 ± 2 ns
D0 disabled: TRIG B starts	0	0
sequence steps: pattern starts per 128 µs	0.99988	1 ± 0.02
pattern starts minus OUT2 starts through D0	0 (1667 each in 0.213 s)	0 ± 1

  D0 and marker 1, both at delay 0, started OUT1 42.25 samples after OUT2 through their trigger pins, the same to the pins' 1 ns resolution: the digital outputs are aligned with the markers. **Not measured:** D1–D7 at the pins (only D0 is wired; the simulation covers all eight), high-frequency patterns at the pins. (The pins' delay relative to the analog outputs was measured later the same day: see the record above.)

- **Pattern starts by sequence steps** (an earlier run of the same check, valid without the jumper: it counts the engine's starts): a two-step sequence at 1 MSa/s whose first step starts a program gave 1612 starts in about 0.2 s, 1.03 times one per 128 µs. In that run the D0 timing checks read stale start times from the marker check (the counters were not cleared), and the disabled-D0 check passed only because nothing was wired; the test now clears the counters and checks the wiring first.
- **Through the service** (`AwgClient` on a PC): PATTERN (binary; 256 words in 13 ms, 32768 in 14 ms), PATTERNS?, PATTERN?, DIGITAL (about 90 ms per change: programs, enables, program), DIGITAL?, HW? (`digital`: running, the program index, starts), describe, SAFE (back to high impedance, state idle), PATDEL all work. Afterwards synth was made active again.
- **Observation:** after these checks, SSH logins to the board stalled for at least 10 minutes while the service kept answering on port 5025 (no reboot; load 0.08 when SSH came back); the cause of the stall is not established. A later "Permission denied (publickey)" was not the board's: the build machine's shell had a stale SSH_AUTH_SOCK (the agent of an ended login), and the passphrase-protected key works only through an agent (the desktop keyring's).

2026-10-23: awg 3.0, simulation and offline tests

- `make test` (offline): all suites pass. awg: 69 tests, new for 3.0: the pattern engine in the model (binary counts wrapping at 255, Gray codes changing one bit per word, PRBS bits as delayed copies with a period of 127 words and 64 ones for order 7, passes ending on the last word or the idle word, an apply without a restart, restarts by OUT1's burst starts, sequence-step actions on the step's first point at the smooth filter's output tap); the driver against `tests/fake_core.py` (patterns, programs in alternating halves of the table by sorted name, the pattern memory, restarts only when the run mode, program or programs change, validation without writes, step actions and the output they follow, the board's pins delay moving markers and digital delays by whole samples, derive, SAFE to high impedance; the model run on the words the driver wrote gives the requested words); TCP PATTERN (binary), PATTERN?, PATTERNS?, PATDEL, DIGITAL, DIGITAL?; the client; the GUI's digital box. common: `pins_delay_s` in the calibration (a loopback refit keeps it), `CAL PINS`, describe keys beyond the outputs.
- `make sim`: `tb_shape` (4621 values) and `tb_system` pass: the eight phases of 2.0, now with the pattern engine in each (a memory pattern and a PRBS from applies, a binary count at OUT1's burst starts held after two passes, a Gray count at OUT2's starts after an apply without a restart, sequence steps that start, stop and start again through jumps and a waiting step, the other half of the program table following OUT2's points, an apply that keeps a running pattern, off with partial output enables), 38,011 samples per output bit-exact on the streams, the DAC pins, both markers and the eight digital pins (driven or high impedance), the pattern starts and a read-back of the pattern memory.

2026-10-23: awg 2.0 on the board (build 0x17e4e55a)

- **Build** (Vivado 2025.1, xc7z010clg400-1): routed timing met, WNS +0.200 ns, WHS +0.021 ns (43,663 endpoints); routed DRC warnings only (DSP input/output pipelining, one unloaded net). Utilization (placed):

13,099 LUTs (74.4 %), 21,114 flip-flops (60.0 %), 48 block-RAM tiles (80.0 %), 48 DSP48 (60.0 %), 2 MMCMs (100 %). Getting there took the shared sequencer (one engine for both outputs) and 12 resampler taps instead of 16; see THEORY, "Resources".

- **`common/tests/verify_build.py --design awg`**: all 8 checks pass (recipe build, unchanged reuse, missing export, vendor HDL edit, design RTL edit, a failed build blocks the upload, clean, saved-project regeneration); the recipe build reproduces build ID 0x17e4e55a.
- **`make test-remote`** (192.168.0.149, rp-overlay.service stopped): loaded through the driver, sysid AWGE 2.0 and the build ID verified, power-up state verified, preset `off`.
- **`make test-hw`**, wiring as for 1.0: 26 of 26 checks pass (the 16 checks of 1.0 with the same results within their scatter, plus 10 for arbitrary waveforms).

Check	Measured	Expected
3200 pairs (a 64-pair sine) at 125 MSa/s, step filter, 0.2 V: IN1 vs the 0.2 V sine	0.998	1 ± 0.03
no underflow at 125 MSa/s	none	none
64-pair sine at 10 MSa/s (fractional rate): the strongest of ±1 %, ±2 %	156.25 kHz	156.25 kHz
smooth filter: its amplitude vs the 0.2 V sine	0.994 / 1.008 (two runs)	1 ± 0.03
a step with `wait` holds the sequencer	waiting	waiting
a software trigger starts the waiting step	playing	playing
I/Q upconversion (I = cos, Q = sin of 125 kHz on a 1.95 MHz carrier): upper sideband vs 0.2 V sine	0.998	1 ± 0.05
image (lower sideband)	−85.1 / −89.9 dB	below −40 dB
carrier	−79.0 / −97.0 dB	below −40 dB
both outputs stream 1 M pairs at 125 MSa/s (500 MB/s from DDR) for 1 s	no underflow	no underflow

The image and carrier are at the lock-in's noise level (8000 samples): they vary from run to run. The FIFO's fewest beats left (`fifo_margin_beats`) read 490–492 of 2048 in every 125 MSa/s run (984 pairs, 7.9 µs); this minimum may be set while the FIFO first fills after a restart. Marker 1 → TRIG A started OUT1 42.25 samples after OUT2, 2 more than in 1.0: the marker follows the output, whose pipeline is 2 clocks longer. The first run of the wait check failed: the triggered step was only 6.4 ms long, while `trigger()` and `hardware()` each take 6–9 ms on the board, so the status was read after the step had ended (the sequencer had played it and stopped). The check now uses a 64 ms step.

- **DDR under load** (driver on the board, service stopped): both outputs streamed a 4 M-pair (16 MB) segment at 125 MSa/s for 10 s while two processes copied 32 MB buffers (1.8 GB in total): no underflow, fewest beats left 492.
- **Times on the board** (driver, service stopped): loading the driver 0.9 s; `save_segment` of 4 M pairs from numpy arrays 5 s (1 M pairs from a list 1.3 s; the conversion was 34 µs per pair in pure Python before numpy was used, now kept as the fallback); the first apply with a 16 MB segment 2.5 s (1.7 s of it the resampler's kernel table, once per process; allocating the 64 MB buffer 0.2 s); an apply that keeps the segments 0.10 s; switching to another 4 MB segment 0.17 s. A 16 MB copy into the buffer takes 0.04 s.
- **Through the service** (`make install-service` with awg active; `AwgClient` on a PC on the LAN): SEGMENT (binary, 4 MB in 0.3–1.8 s, 16 MB in 1.4–2.1 s: 8–12 MB/s, varying with the SD card's writes), SEGMENT? ranges, SEGMENTS?, SEQUENCE, SEQUENCE?, SEQUENCES?, PLAY (86 ms), ARB (the first switch-on 0.6 s, then about 0.1–0.3 s), HW? arb status, CAPTURE?, SAFE (sequencer idle), SEGDEL, SEQDEL all work. Playing a new 16 MB segment by PLAY 0.36–0.53 s. Service memory: 66 MB resident after these. In an earlier run the first switch-on took 4.6 s while the driver kept every segment's words in memory; it now keeps only their checksums. Afterwards synth was made active again.
- **Not measured**: event jumps (immediate and at a repetition end) and gated sequences on the board (simulated only); the resampler's frequency response beyond the one smooth-filter amplitude; arb markers on the pins; skew between OUT1 and OUT2 points; underflow behavior (the hardware flag was never set); I/Q at other carriers.

2026-10-23: awg 2.0, simulation and offline tests

- `make test` (offline): all suites pass. awg: 57 tests (model: arbitrary-waveform semantics of the engine, pairs, waits, repeats, jumps, gates, restarts, I/Q; the noise bandwidth of the 12-tap kernel; driver against `tests/fake_core.py`: segments, sequences, the DDR buffer and its placement, step tables and hot caches in alternating halves, restarts, validation, capture of the model's points, upconversion and derive, SAFE, loading through FakePynq, TCP commands with binary uploads, client, command line, GUI with ipywidgets; numpy and standard-library pair words agree bit for bit). common: binary requests through the server and client.
- `make sim`: `tb_shape` (4621 values) and `tb_system` pass: the 1.0 phases plus four of arbitrary waveforms (a triggered sequence of pairs with hot starts, repeats, jumps at a repetition end and at once and a waiting step started by TRIG A, both outputs with their own filters and offsets; I/Q upconversion; an apply without a restart; a gated sequence with OUT2 playing noise), 38,011 samples per output bit-exact against the model on the core streams, the DAC pins and both marker pins, with the points read from a DDR model (AXI3, random acceptance, latencies of 20–90 clocks, gaps, up to 8 bursts in flight).

2026-10-23: awg 1.0 on the board (build 0xb088a0b4)

- **Build** (Vivado 2025.1, xc7z010clg400-1): routed timing met, WNS +0.064 ns, WHS +0.012 ns (40,058 endpoints); routed DRC warnings only (DSP output pipelining, one unloaded net). Utilization (placed): 11,306 LUTs (64.2 %), 20,213 flip-flops (57.4 %), 38 block-RAM tiles (63.3 %), 54 DSP48 (67.5 %), 2 MMCMs (100 %).
- **`common/tests/verify_build.py --design awg`**: all 8 checks pass (recipe build, unchanged reuse, missing export, vendor HDL edit, design RTL edit, a failed build blocks the upload, clean, saved-project regeneration); the recipe build reproduces build ID 0xb088a0b4.
- **`make test-remote`** (192.168.0.149, rp-overlay.service stopped): loaded through the driver, sysid AWGE 1.0 and the build ID verified, power-up state verified, preset `off`.
- **`make test-hw`**, OUT1 → IN1 and OUT2 → IN2 with 50 Ω terminators, DIO1_P wired to DIO0_P: 16 of 16 checks pass.

Check	Measured	Expected
0.2 V sine at 1.95 MHz, IN1 / IN2	0.2050 / 0.2055 V	0.18 … 0.32 V (loopback)
IN1/IN2 amplitude ratio	0.997	1 ± 0.05
OUT2 phase −90°: change of IN1 − IN2 phase	89.80°	90 ± 0.5°
square 1 MHz, 30 % duty (50 % crossings)	29.995 %	30 ± 1 %
pulse width 203 ns (25.375 samples)	203.42 ns	203 ± 2 ns
pulse edge, phase steps of 1/10 sample (mean of 3)	0.109 sample	0.1 ± 0.03
noise 0.05 V rms, 10 MHz (on the sine's scale)	0.0506 V	0.05 ± 0.005 V
timer bursts of 3 cycles: length above 30 % of the amplitude	185 samples	189 ± 6
timer burst period 20 µs	2500 samples	2500 ± 2
software triggers counted	3	3
AM depth 50 % by generator 2	50.08 %	50 ± 3 %
DC +0.2 V with the high limit at +0.1 V, minus DC +0.1 V	−0.3 mV	0 ± 3 mV
DC −0.5 V with the low limit at −0.3 V, minus DC −0.3 V	−0.006 mV	0 ± 3 mV
marker 1 (OUT2's start) → TRIG A → OUT1 start, spread of 5 reads	0	≤ 0.125 sample
SAFE: IN1 rms	1.5 mV	below 10 mV

  The loop of marker 1 to TRIG A started OUT1 40.25 samples (322 ns) after OUT2, the same start to 1/256 sample in all 5 reads. The edge step reads 0.109 sample per 0.1-sample step in three runs, from linear interpolation of the sampled 50 % crossings; the cause of the 10 % excess is not established.

- **Loopback gains** (the board calibration of 2026-10-04, observation): a DC level reads 3.6 % low on IN1 (+0.1 V → 0.0964 V, −0.3 V → −0.2878 V), a 1.95 MHz or 19.5 kHz sine about 2.5–3 % high (0.3 V → 0.3095 V). A sine's top clipped by a limit therefore reads higher than the same DC level (0.111 V for +0.1 V), so the hardware test compares DC levels; the clamp itself holds the DAC code.
- **Through the service** (`make install-service` with awg active; `AwgClient` on a PC on the LAN): FUNC, OUT, OUT?, NOISE, TABLE (1000 points → 4096), TABLES?, ZERO, software TRIG (2 counted), HW?, CAPTURE? (8192 samples), SAFE all work. Round trips: a configuration change 79–81 ms, DESIGN? 11 ms, HW? 13 ms; the first change after the service started 1.7 s. Service CPU at idle: 0 %. Afterwards synth was made

active again.

- **Not measured**: absolute trigger-to-output latency and jitter against an external trigger (needs an oscilloscope), noise spectrum and crest factor on the board, sweep law, FM/PM/FSK/BPSK/PWM on the board, IN threshold triggers, EVENT pins, TRIG B/gates, the sine's spurs.

2026-10-23: awg 1.0, simulation and offline tests

- `make test` (offline): the model tests (generated ROMs and vectors current, sine accuracy, sub-sample edges, table interpolation, PRBS register jumps, exp2, Gaussian statistics, noise spectrum, burst/trigger/sweep semantics) and the driver tests against `tests/fake_core.py` (coherent apply with PRBS states, shapes and edges, calibration, noise, modulation words, bursts, sweeps, tables, limits and protection, derive, presets, timers, a capture of the model in loopback, calibration hooks), loading through FakePynq, protocol, client, command line; the GUI test with ipywidgets: pass.
- `make sim`: `tb_shape` (4621 values of every shape: sine and table phases at the quadrant and entry boundaries, trapezoids around their edges with PWM-moved HE/FE, PRBS of every order with loads and paused advances) and `tb_system` (a four-phase scenario of 12,411 samples per output bit-exact against the model on the core streams, the DAC pins (actual DAC VHDL) and both marker pins, plus trigger, missed and clip counts, the last start times and the apply count): pass. The trigger pins' ISERDES (UNISIM model) timestamps an edge x ns after the clock edge where the counter becomes n as tau = 256 n - 128 + 32 ceil(x), monotonic in 1 ns steps.