

Zen PPU Debug — Datasheet

CSR-Sim — Datasheet (DSHT-00-INDEX)

The product DATASHEET for Zen CSR-Sim (zen-csr-sim-kit-v1) — a YAML-driven **CSR programming-model simulator** (dev/verification IP): define a register file in YAML, run init firmware in C on a wedged-in RV32I ISS, and visualize register/field/bit state over the trace. This is the **row-1 hub** of a hierarchical, multi-file datasheet. Product/integrator-facing. Instantiated from shared-agent-infra/DATASHEET-STANDARD.md. **This is a software dev/verification tool, not silicon RTL** — hardware-only chapters (clocking, physical, RAS) are marked **N/A** with the nearest tool-relevant analog. = outline · = authored (pending review) · = content complete + reviewed.

What this datasheet is (and is not)

- **Is:** the product datasheet for the tool — what it does, the YAML CSR contract, the compile/run/step model, interfaces (API + ports), and how to stand it up + integrate as a co-sim backend.
- **Is not:** the internal design doc. Ground truth = the as-built tool (tools/csr-sim/); **the tool wins** on disagreement.
- **Altitude:** *what CSR-Sim does + how to use it*. Uncharacterized numbers are **TBC** with an owner.

How to read it

- Start at [\[\[docs/datasheets/csr-sim/DSHT-01-OVERVIEW|DSHT-01\]\]](#). The high-value chapters for this tool are [\[\[docs/datasheets/csr-sim/DSHT-10-REGISTER-MAP|DSHT-10\]\]](#) (the YAML CSR-def schema), [\[\[docs/datasheets/csr-sim/DSHT-09-PROGRAMMING-MODEL|DSHT-09\]\]](#), and [\[\[docs/datasheets/csr-sim/DSHT-15-INTEGRATION-GUIDE|DSHT-15\]\]](#).
- **Numbers policy:** every quantitative spec is sourced from the tool, or marked **TBC**. **No unsourced numbers.**

Chapter map

DSHT	Chapter	Scope	Source
[[docs/datasheets/csr-sim/DSHT-00-INDEXIDSHT-00]]	Intro & TOC	this hub	—
[[docs/datasheets/csr-sim/DSHT-01-OVERVIEWIDSHT-01]]	Product overview	positioning · what it does	tools/csr-sim/README · [[docs/kits/CSR-SIM-LICENSE-KIT kit]]
[[docs/datasheets/csr-sim/DSHT-02-FEATURES-COMPLIANCEIDSHT-02]]	Key features	feature matrix · standards fit	tools/csr-sim
[[docs/datasheets/csr-sim/DSHT-03-APPLICATIONSIDSHT-03]]	Applications / use models	target workflows	—
[[docs/datasheets/csr-sim/DSHT-04-CONFIGURABLE-OPTIONSIDSHT-04]]	Configurable options	YAML defs · runtime config	tools/csr-sim
[[docs/datasheets/csr-sim/DSHT-05-FUNCTIONAL-OVERVIEWIDSHT-05]]	Functional overview	compile → run → step → visualize	tools/csr-sim/backend
[[docs/datasheets/csr-sim/DSHT-06-INTERFACES-SIGNALSIDSHT-06]]	Interfaces	REST API · YAML contract · ports	backend/server.py
[[docs/datasheets/csr-sim/DSHT-07-CLOCKING-RESETIDSHT-07]]	Clocking & reset	N/A (software) → “clock”=instruction step	tools/csr-sim
[[docs/datasheets/csr-sim/DSHT-08-POWER-ON-RESET-INITIDSHT-08]]	Startup / init	install · launch	tools/csr-sim/README
[[docs/datasheets/csr-sim/DSHT-09-PROGRAMMING-MODELIDSHT-09]]	Programming model	author CSR-def + firmware C	backend
[[docs/datasheets/csr-sim/DSHT-10-REGISTER-MAPIDSHT-10]]	CSR-def schema	YAML regfile/register/field syntax	csrdefs · csr_model.py
[[docs/datasheets/csr-sim/DSHT-11-PERFORMANCEIDSHT-11]]	Performance	ISS throughput envelope	backend/rv32i.py
[[docs/datasheets/csr-sim/DSHT-12-PHYSICALIDSHT-12]]	Physical	N/A (software)	—
[[docs/datasheets/csr-sim/DSHT-13-RASIDSHT-13]]	Validation & errors	YAML/compile validation surfaces	backend
[[docs/datasheets/csr-sim/DSHT-14-VERIFICATION-DELIVERABLESIDSHT-14]]	Verification & deliverables	what was verified · manifest	[[docs/kits/CSR-SIM-LICENSE-KIT kit]]
[[docs/datasheets/csr-sim/DSHT-15-INTEGRATION-GUIDEIDSHT-15]]	Integration guide	standup · VPD co-sim backend	dev-env/vpd/csrsim-adapter
[[docs/datasheets/csr-sim/DSHT-16-ORDERING-LICENSINGIDSHT-16]]	Ordering / licensing	commercial · provenance	[[docs/kits/CSR-SIM-LICENSE-KIT kit]]
[[docs/datasheets/csr-sim/DSHT-A-APPENDICESIDSHT-A]]	Appendices	acronyms · revision history	—

Moniker & provenance

- **Moniker DSHT** per shared-agent-infra/DATASHEET-STANDARD.md. Path docs/datasheets/csr-sim/DSHT-NN-NAME.md.
- **Provenance:** distilled from tools/csr-sim/ (README, backend, csrdefs) + the CSR-Sim license kit. **Origin + credit: zen-visualizer (Isabel Fontaine);** transferred into zen-peripheral (zen-ip) 2026-08-21. See [[docs/kits/CSR-SIM-

Dat
D:
De

title:
Proce
crea
28 s
auth
revi
shai
infra
STA

DSHT-01 · Product Overview

Scope. What Zen CSR-Sim is and where it fits. *Source:* [tools/csr-sim/README.md](#), [\[\[docs/kits/CSR-SIM-LICENSE-KIT\[\[kit\]\(#\)\]\]](#) — *SOURCED*.

What it is

- **Zen CSR-Sim** is a **CSR programming-model simulator**: define a block's control/status registers in **YAML**, write init/bring-up firmware in **C**, **compile it for real** (`riscv64-unknown-elf-gcc -march=rv32i`), and **single-step** it on a wedged-in RV32I instruction-set simulator — watching register/field/bit state change live, with a routable LED bank for output visualization.
- A **dev/verification tool** (not synthesizable RTL): a programming-model + firmware-bring-up + visualization aid.

Why it exists

- Almost every block has a CSR bank. A durable, structured CSR definition lets init/bring-up firmware start **in parallel with TLM/RTL development**, and (roadmap) generate sim vectors from the same model for TLM cross-check.

Shape

- **Backend (Python/FastAPI)**: RV32I ISS ([rv32i.py](#)), field-accurate CSR model ([csr_model.py](#)), real gcc compile ([compiler.py](#)), REST server ([server.py](#)). *Source:* [README.md architecture](#) — *SOURCED*.
- **Frontend (React/Monaco SPA)**: register/field/bit view, C editor, sim controls (play/step/scrub), routable LED bank.
- **Ports**: backend **8044**, frontend **3044**. *Source:* [README.md:98–100](#) — *SOURCED*.
- **Provenance**: origin zen-visualizer (Isabel Fontaine); transferred into zen-peripheral (zen-ip) 2026-08-21. See [\[\[docs/datasheets/csr-sim/DSHT-16-ORDERING-LICENSINGIDSHT-16\]\]](#).

Source cross-refs

- [tools/csr-sim/README.md](#) · Kit: [\[\[docs/kits/CSR-SIM-LICENSE-KIT\[\[kit\]\(#\)\]\]](#)CSR-SIM-LICENSE-KIT]],

DSHT-01 · CSR-Sim · product overview.

title: “CSR-Sim — Key Features” created: 2026-09-28 status: authored (pending review) standard: [shared-agent-infra/DATASHEET-STANDARD.md](#) —

DSHT-02 · Key Features

Scope. Feature matrix + standards fit. *Source:* [tools/csr-sim/README.md](#), [backend/csr_model.py](#) — *SOURCED*.

Feature matrix

Feature	Detail	Source
YAML CSR contract	regfile → registers → fields; per- field access	README.md:24–49
Field access classes	rw/wo (control), ro (status), w1c (write-1- clear)	README.md:43–48
Mixed registers	one register freely mixes control + status bits	README.md:43
Real firmware compile	riscv64-unknown-elf-gcc - march=rv32i + crt0, linked 0x8000_0000	README.md:64–73
RV32I ISS	steps the image; MMIO loads/stores intercepted into the CSR model	README.md:64–73
Instruction-step trace	play / step / scrub; PC + disasm + CSR read/write per step	README.md:52–59
Live register view	control cyan / status teal / w1c amber; 32-bit strip per register	README.md:52–56
Routable LED bank	route any field/bit to LEDs (walking-one demos)	README.md:60–62
Runtime YAML load	upload a def at runtime (server-side validated); no restart	README.md:48–49
Co-sim backend	serves a block's CSR model over the VPD socket wire	[[docs/kits/CSR-SIM- LICENSE-KIT kit]]

Standards / fit

- **Protocol-agnostic:** models any memory-mapped register file; pairs with the Zen peripheral-bus + APB/AXI4-Lite frontends.
- **Toolchain:** standard RISC-V rv32i multilib gcc; firmware is the real lw/sw MMIO init idiom.
- **Scope:** functional/programming-model level — **not** cycle-accurate, **not** a signoff simulator (see [[docs/datasheets/csr-sim/DSHT-13-RASIDSHT-13]] boundaries).

Source cross-refs

- tools/csr-sim/README.md · backend/csr_model.py · Kit: [[docs/kits/CSR-SIM-LICENSE-KIT|CSR-SIM-LICENSE-KIT]].

DSHT-02 · CSR-Sim · key features.

title: “CSR-Sim — Applications / Use Models” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-03 · Applications / Use Models

| **Scope.** Target workflows for Zen CSR-Sim.

Target applications

- **Early firmware bring-up** — write and validate a block’s init/config sequence against its programming model **before** RTL or silicon exists.
- **CSR programming-model validation** — prove a register map’s field semantics (rw/ro/w1c) behave as intended under real firmware.
- **Register-map documentation** — an interactive, executable register-map view (fields/bits) for diagnostic and safety register spaces.
- **VPD co-sim backend** — drive a block’s CSR model live through the VPD socket, A/B against TLM or Verilated RTL on the same wire.

Use models

- **Author + run** — pick/upload a CSR-def YAML, edit firmware C in the browser, compile, run, and scrub the trace.
- **Visual debug** — route fields/bits to the LED bank; watch bits update live as firmware writes them.
- **Regression seed (roadmap)** — record a CSR run to generate TLM cross-check vectors.

Recommended companions

- The block’s own [\[\[docs/datasheets/ppu/DSHT-00-INDEXIPPU\]\]](#) / peripheral datasheets for the register map under test; the VPD co-sim bridge for live RTL/TLM A/B.

Source cross-refs

- Kit: [\[\[docs/kits/CSR-SIM-LICENSE-KITCSR-SIM-LICENSE-KIT\]\]](#) · [tools/csr-sim/README.md](#).

DSHT-03 · CSR-Sim · applications.

title: “CSR-Sim — Configurable Options” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-04 · Configurable Options

Scope. What the user configures (this is a tool, not a parameterized RTL block). *Source:* [tools/csr-sim/README.md](#) — *SOURCED*.

CSR definitions (the primary “config”)

- `csrdefs/*.yaml` — drop-folder of register-file definitions; a dropdown selects among them. `default.yaml` = the `soc_gpio` worked example. Upload a new def at runtime via the **YAML** button (server-side validated; no restart).

Firmware

- `examples/*.c` — init firmware shown on load; user-editable in the Monaco editor. CSR32(addr) accessor provided; compiled with real rv32i gcc.

Runtime environment

Setting	Value	Source
Backend port	8044	README.md:98–100
Frontend port	3044	README.md:98–100
Python	3.10	README.md:21
Toolchain	riscv64-unknown-elf-gcc (rv32i multilib)	README.md:21,65
Link address	0x8000_0000 (flat image)	README.md:66–67

Source cross-refs

- [tools/csr-sim/README.md](#) · [tools/csr-sim/csrdefs/](#), [examples/](#).

DSHT-04 · CSR-Sim · configurable options.

title: “CSR-Sim — Functional Overview” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-05 · Functional Overview

Scope. The compile → run → step → visualize pipeline. *Source:* [tools/csr-sim/README.md:52–73](#), [backend/](#) — *SOURCED*.

Pipeline

1. **Define** — load a CSR-def YAML: regfile → registers → fields, each field with its own access (rw/wo/ro/w1c) and reset. Registers render on the left (control cyan / status teal / w1c amber).
2. **Author** — edit init firmware C (Monaco). CSR32(addr) accessor provided.
3. **Compile** — real `riscv64-unknown-elf-gcc -march=rv32i (+ crt0 that sets sp and calls main)`, linked at `0x8000_0000`, objcopied to a flat image. Compile errors show inline.

- 4. **Run / step** — the RV32I ISS executes the image one instruction at a time (the “clock” = one instruction). Loads/stores to the CSR address range are **intercepted** and driven into the field-accurate CSR model, respecting each field’s access.
- 5. **Visualize** — every instruction is recorded; the UI scrubs the trace showing PC, disasm, CSR reads/writes, live register bits, and the routable LED bank.

Field semantics (the model)

- On an MMIO write: control fields take the written bits; ro fields are untouched (hardware-owned); w1c fields clear bits written as 1. A read reassembles the register from its fields. *Source: [[docs/kits/CSR-SIM-LICENSE-KITikit]] §The tool — SOURCED.*

Source cross-refs

- tools/csr-sim/README.md:52–73 · backend/rv32i.py, csr_model.py, compiler.py.

DSHT-05 · CSR-Sim · functional overview.

title: “CSR-Sim — Interfaces” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-06 · Interfaces

Scope. For a software tool, “interfaces” = the REST API, the YAML contract, and the network ports. *Source: tools/csr-sim/README.md, backend/server.py — SOURCED.*

REST API (FastAPI backend)

Source: README.md:80–83 — SOURCED.

Endpoint	Purpose
/api/csrdefs	list available CSR definitions
/api/csrdef	fetch/select a definition
/api/parse	validate/parse an uploaded YAML def
/api/compile	compile firmware C → flat rv32i image
/api/run	execute the image; return the instruction trace + CSR state
/ (SPA)	serves the built frontend

YAML CSR contract

- regfile: {name, base, registers[]}; register: {name, offset, fields[]}; field: {name, bits, access, reset}. Full schema: [[docs/datasheets/csr-sim/DSHT-10-REGISTER-MAPIDSHT-10]].

Ports

- Frontend **3044** / backend **8044**. Single-port mode: build the frontend and serve the SPA from the backend at **8044**. *Source: README.md:19,98–100 — SOURCED.*

Co-sim wire (VPD backend)

- Speaks the VPD protocol (16-byte LE records; READ_REQ/WRITE_REQ → READ_RSP/WRITE_RSP over a Unix socket) — same wire as the RTL/TLM servers. Adapter: dev-env/vpd/csrsim-adapter/.

Source cross-refs

- backend/server.py · README.md · [[docs/kits/CSR-SIM-LICENSE-KITikit]] (VPD backend).

DSHT-06 · CSR-Sim · interfaces.

DSHT-07 · Clocking & Reset

Scope. **N/A for a software tool** — there is no hardware clock or reset. The nearest analogs are recorded here for completeness.

Analogs

- **“Clock” = one instruction.** The simulation advance unit is a single RV32I instruction; play/step/scrub move along the recorded instruction trace. There is no timing, pipeline, or wall-clock semantics (functional/programming-model level only). *Source: README.md:56–59 — SOURCED.*
- **“Reset” = field reset values + re-run.** Each field’s reset value initializes the register model; re-running the firmware from the start re-applies reset then replays. *Source: README.md:34–40 (per-field reset) — SOURCED.*

Not applicable

- Hardware clock domains, CDC, reset trees, fmax — **N/A** (software). See [[docs/datasheets/csr-sim/DSHT-12-PHYSICALIDSHT-12]].

Source cross-refs

- tools/csr-sim/README.md.

DSHT-07 · CSR-Sim · clocking & reset (N/A — analogs).

DSHT-08 · Startup / Init

Scope. Install and launch (the software analog of power-on/init). *Source: tools/csr-sim/README.md:14–23 — SOURCED.*

Prerequisites

- python3.10 and riscv64-unknown-elf-gcc (rv32i multilib) on PATH. *Source: README.md:21 — SOURCED.*

Launch (two-port dev mode)

```
cd tools/csr-sim
./start.sh      # backend :8044 + frontend :3044 → http://localhost:3044
```

Launch (single-port)

```
cd tools/csr-sim/frontend && npm run build # build the SPA
# run the backend, open http://localhost:8044
```

First run

- Loads the default soc_gpio CSR-def + examples/default.c. Pick/upload a def, edit firmware, Compile, Run, and scrub the trace.

Source cross-refs

- tools/csr-sim/README.md:14–23,98–100 · start.sh.

DSHT-08 · CSR-Sim · startup / init.

DSHT-09 · Programming Model

Scope. How a user drives CSR-Sim: define registers, write firmware, run.
Source: [tools/csr-sim/README.md](#) — *SOURCED.*

1 · Define the register file (YAML)

- Author regfile → registers → fields with per-field access (rw/wo/ro/w1c) and reset. Full schema: [\[\[docs/datasheets/csr-sim/DSHT-10-REGISTER-MAP|DSHT-10\]\]](#).

2 · Write init firmware (C)

- Standard main(); touch CSRs with the provided CSR32(addr) accessor (memory-mapped lw/sw — the real init-firmware idiom).

```
void main(void) {  
    CSR32(0x10000000) = 0x1;    // CTRL.ENABLE  
    while (CSR32(0x1000000C) & 1) ; // poll STATUS.BUSY  
}
```

3 · Compile + run + step

- Compile (real rv32i gcc) → Run (RV32I ISS) → Step/scrub the recorded trace. CSR loads/stores are intercepted into the field-accurate model; register bits update live.

Field-access behavior (what firmware sees)

- rw/wo: writes take effect. ro: writes ignored (hardware-owned; holds reset/last value). w1c: writing 1 clears the bit. A read reassembles the register from its fields.

Source cross-refs

- [tools/csr-sim/README.md:24–73](#) · [backend/csr_model.py](#).

DSHT-09 · CSR-Sim · programming model.

DSHT-10 · CSR-Def Schema (the “register map” the tool consumes)

Scope. The YAML CSR-definition contract — CSR-Sim’s central artifact.
Source: [tools/csr-sim/README.md:24–49](#), [backend/csr_model.py](#) — *SOURCED.*

Schema

```
regfile:  
  name: <block name>  
  base: <base address, e.g. 0x10000000>  
  registers:  
    - name: <REG>  
      offset: <byte offset, e.g. 0x00>  
      fields:  
        - {name: <FIELD>, bits: "<msb:lsb or bit>", access: <rw|wo|ro|w1c>, reset: <value>}
```

Elements

Level	Key	Meaning
regfile	name	block/regfile identifier
regfile	base	base address of the register file
register	name / offset	register name + byte offset from base
field	name	field identifier
field	bits	bit range "msb:lsb" or single "n"
field	access	rw/wo (control), ro (status), w1c (write-1-clear)
field	reset	reset value

Rules

- **No register-wide access assumption** — every *field* carries its own access; a register freely mixes control and status bits. *Source: README.md:43 — SOURCED.*
- **Runtime-loadable** — a def can be uploaded at runtime; it is validated server-side (/api/parse) before use.
- **Multiple blocks** — csrdefs/*.yaml holds independent CSR spaces; a dropdown selects among them.

Worked example

- csrdefs/default.yaml = soc_gpio (CTRL.ENABLE/MODE control, STATUS.BUSY ro, STATUS.ERR w1c). *Source: README.md:26–41 — SOURCED.*

Source cross-refs

- tools/csr-sim/README.md:24–49 · tools/csr-sim/csrdefs/default.yaml · backend/csr_model.py.

DSHT-10 · CSR-Sim · CSR-def schema.

title: “CSR-Sim — Performance” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-11 · Performance

Scope. Simulation-throughput envelope. *Source: tools/csr-sim/backend/rv32i.py, README.md:94–96 — architectural; absolute rates TBC.*

Character

- **Interactive-scale ISS.** The RV32I simulator executes and records the firmware image instruction-by-instruction for scrubbing; it targets **bring-up-class firmware** (tens to thousands of instructions), not long production workloads. *Source: README.md:94–96 (worked example = 30 instructions) — SOURCED.*
- **Per-instruction recording** trades memory/throughput for full scrub-back — appropriate to the interactive debug use model.

Absolute numbers

- Instructions/sec and max trace length: **TBC** (owner: tools). Not characterized; performance is interactive-tool-class, not a signoff simulator.

Not applicable

- Silicon throughput/latency/fmax — **N/A** (software tool; models programming behavior, no timing).

Source cross-refs

- tools/csr-sim/backend/rv32i.py · README.md:94–96.

DSHT-11 · CSR-Sim · performance.

title: "CSR-Sim — Physical / Implementation" created: 2026-09-28 status: N/A
(software tool) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-12 · Physical / Implementation

Scope. N/A — CSR-Sim is a software dev/verification tool, not silicon.
There is no area, fmax, power, or floorplan.

Deployment footprint (the analog)

- Runs as a Python/FastAPI backend + a built React SPA on a developer host or CI runner. Requires python3.10 + riscv64-unknown-elf-gcc. No silicon footprint.

Source cross-refs

- Runtime env: `[[docs/datasheets/csr-sim/DSHT-04-CONFIGURABLE-OPTIONS/DSHT-04]]` · startup: `[[docs/datasheets/csr-sim/DSHT-08-POWER-ON-RESET-INIT/DSHT-08]]`.

DSHT-12 · CSR-Sim · physical (N/A — software).

title: "CSR-Sim — Validation & Errors" created: 2026-09-28 status: authored
(pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-13 · Validation & Errors

Scope. The tool analog of RAS: how CSR-Sim surfaces bad input and enforces the model. *Source:* `tools/csr-sim/README.md`, `[[docs/kits/CSR-SIM-LICENSE-KIT/kit]]` — SOURCED.

Error / validation surfaces

- **YAML validation** — an uploaded CSR-def is parsed and **validated server-side** (`/api/parse`) before use; malformed defs are rejected, not silently accepted. *Source:* `README.md:48–49` — SOURCED.
- **Compile errors** — firmware C is compiled with real gcc; errors are surfaced **inline** in the editor. *Source:* `README.md:55–56` — SOURCED.
- **Field-access enforcement** — the model enforces `ro` (writes ignored), `w1c` (write-1-clear), `rw/wo` per field on every MMIO access — so a firmware bug that writes a read-only field is *visibly* a no-op, not a silent success. *Source:* `[[docs/kits/CSR-SIM-LICENSE-KIT/kit]] §The tool` — SOURCED.

Boundaries (honest)

- **Not cycle-accurate**; **not** interrupt-driven (no autonomous hardware event/pending engine — status/pending bits are firmware-visible but not self-driven); ISS is **RV32I only** (bring-up-class firmware, not full privileged spec). *Source:* `[[docs/kits/CSR-SIM-LICENSE-KIT/kit]] §Boundaries` — SOURCED.
- Known enhancement (zen-peripheral-owned): a first-class hardware-set/backdoor path for `ro/w1c` fields so a co-sim backend can drive status/pending cleanly.

Source cross-refs

- `tools/csr-sim/README.md` · `[[docs/kits/CSR-SIM-LICENSE-KIT/CSR-SIM-LICENSE-KIT]]` (boundaries + coordination).

DSHT-13 · CSR-Sim · validation & errors.

title: "CSR-Sim — Verification & Deliverables" created: 2026-09-28 status: authored
(pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-14 · Verification & Deliverables

Scope. What was verified and what a licensee receives. *Source:* [\[\[docs/kits/CSR-SIM-LICENSE-KIT|kit\]\]](#), [tools/csr-sim/README.md:93–96](#) — *SOURCED*.

What was verified

- **End-to-end pipeline:** real C→RV32I compile, ISS execution, and CSR field decode against the default soc_gpio block — 30 instructions, LED walking 0x1→0x80→0xAA, CTRL.ENABLE/MODE set. *Source:* [README.md:94–95](#) — *SOURCED*.
- **Frontend:** compiles under strict TypeScript and builds; SPA served by the backend. *Source:* [README.md:96](#) — *SOURCED*.
- **VPD co-sim integration:** enforced rw/ro/w1c end-to-end through the socket on zen_gpio — CSRSIM-VPD: PASS. *Source:* [\[\[docs/kits/CSR-SIM-LICENSE-KIT|kit\]\]](#) — *SOURCED*.

Deliverables (zen-csr-sim-kit-v1)

Component	Path (tools/csr-sim/)
CSR-def YAML syntax	csrdefs/*.yaml
Backend model / ISS / compile	backend/csr_model.py, rv32i.py, compiler.py
Backend server (FastAPI)	backend/server.py
Frontend visualizer (SPA)	frontend/
Examples	examples/*.c, csrdefs/default.yaml
VPD backend adapter	dev-env/vpd/csrsim-adapter/
Manifests	[[docs/kits/CSR-SIM-LICENSE-KIT kit]] , TRANSFER-PROVENANCE.md, this datasheet

Source cross-refs

- [\[\[docs/kits/CSR-SIM-LICENSE-KIT|CSR-SIM-LICENSE-KIT\]\]](#) · [tools/csr-sim/README.md](#).

DSHT-14 · CSR-Sim · verification & deliverables.

title: “CSR-Sim — Integration Guide” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-15 · Integration Guide

Scope. Standing CSR-Sim up on a project and wiring it as a VPD co-sim backend. *Source:* [tools/csr-sim/README.md](#), [\[\[docs/kits/CSR-SIM-LICENSE-KIT|kit\]\]](#) — *SOURCED*.

Stand it up on your block

1. Write your block’s register file as a CSR-def YAML (schema: [\[\[docs/datasheets/csr-sim/DSHT-10-REGISTER-MAP|DSHT-10\]\]](#)); drop it in csrdefs/ (or upload at runtime).
2. Write init firmware C against CSR32(base+offset).
3. ./start.sh, open the SPA, Compile → Run → scrub. Route fields/bits to the LED bank for a quick visual.

Wire as a VPD co-sim backend

- The dev-env/vpd/csrsim-adapter/ presents the block’s CSR model over the **VPD protocol** (16-byte LE records; READ/WRITE_REQ → READ/WRITE_RSP over a Unix socket) — the same wire the RTL and TLM servers speak, so a guest/host drives csr-sim, TLM, or Verilated RTL through one interface.
- Demonstrated on rtl/gpio/zen_gpio.sv: rw (DIR/OUT), ro (IN), w1c (INT_PEND) enforced end-to-end → CSRSIM-VPD: PASS.

Notes

- Firmware-visible only: ro/pending fields hold reset/last value unless a backend pokes them (see the hardware-set enhancement in [\[\[docs/datasheets/csr-sim/DSHT-13-RASIDSH-13\]\]](#)).

Source cross-refs

- [tools/csr-sim/README.md](#) · [adapter: dev-env/vpd/csrsim-adapter/](#) · [\[\[docs/kits/CSR-SIM-LICENSE-KIT|kit\]\]](#).

DSHT-15 · CSR-Sim · integration guide.

title: “CSR-Sim — Ordering / Licensing / Support” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-16 · Ordering / Licensing / Support

Scope. Commercial packaging, provenance, support. *Source:* [\[\[docs/kits/CSR-SIM-LICENSE-KIT|kit\]\]](#), [tools/csr-sim/TRANSFER-PROVENANCE.md](#) — *SOURCED*.

Bundle

- **zen-csr-sim-kit-v1** — dev/verification IP (not silicon RTL): CSR-def YAML syntax + backend model/ISS/compile + FastAPI server + React SPA + examples. Contents: [\[\[docs/datasheets/csr-sim/DSHT-14-VERIFICATION-DELIVERABLES|DSHT-14\]\]](#).
- **In-repo home:** [tools/csr-sim/](#).

Licensing

- **Internal soft-license** by default; kit is license-ready-structured for external.
- **Provenance / lineage (retained):** origin + credit **zen-visualizer (Isabel Fontaine)**. Ownership transferred into zen-peripheral (zen-ip) **2026-08-21** (Rod-authorized, IP-ASSIM PHASE-07 §5-A); origin repo retires its copy on zen-visualizer’s timeline. *Source:* [\[\[docs/kits/CSR-SIM-LICENSE-KIT|kit\]\]](#) header, [TRANSFER-PROVENANCE.md](#) — *SOURCED*.

Support / roadmap

- **Roadmap:** route bits/fields to other panes/visualizers; generate TLM cross-check vectors from a recorded run; a hardware/status model to drive ro/status fields; more access types (rc/w1s/rw1c variants). *Source:* [README.md:86–91](#) — *SOURCED*.

Source cross-refs

- Kit: [\[\[docs/kits/CSR-SIM-LICENSE-KIT|CSR-SIM-LICENSE-KIT\]\]](#) · provenance: [tools/csr-sim/TRANSFER-PROVENANCE.md](#).

DSHT-16 · CSR-Sim · ordering / licensing / support.

title: “CSR-Sim — Appendices” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-A · Appendices

Scope. Acronyms, field-access glossary, revision history.

A.1 · Acronyms

- **CSR** — control/status register.
- **ISS** — instruction-set simulator (here, RV32I).
- **regfile** — register file (top of the YAML CSR-def).
- **SPA** — single-page application (the React frontend).
- **TLM** — transaction-level model (a fast functional model; a CSR-Sim A/B target).
- **VPD** — Virtual Platform Demonstrator (the co-sim bridge CSR-Sim plugs into as a backend).

A.2 · Field-access glossary

- **rw / wo** — control fields (firmware-owned; writes take effect).
- **ro** — status fields (hardware-owned; firmware writes ignored, hold reset/last value).
- **w1c** — write-1-to-clear (writing 1 clears the bit).

A.3 · Revision history

Date	Change
2026-09-28	Initial datasheet authored (all chapters) — pending review.

Source cross-refs

- Schema: [\[\[docs/datasheets/csr-sim/DSHT-10-REGISTER-MAPIDSHT-10\]\]](#) · [hub: \[\[docs/datasheets/csr-sim/DSHT-00-INDEXIDSHT-00\]\]](#).

DSHT-A · CSR-Sim · appendices.