

Zen PPU Debug — Datasheet

PPU Debug Co-Processor — Datasheet (DSHT-00-INDEX)

The **product DATASHEET** for the **Zen PPU** (zen-ppu-debug-kit-v1) — a J1-style FORTH debug co-processor a client instantiates in-design to **monitor** (@), **intervene on** (!), and **field-patch** live design registers, with APB / AXI4-Lite host frontends. This is the **row-1 hub** of a hierarchical, multi-file datasheet: how to read it, the chapter map, conventions, and provenance. Product/integrator-facing. Instantiated from shared-agent-infra/DATASHEET-STANDARD.md. Product altitude — *what it does + how to integrate*, not *how it's built*. = outline · = authored (pending review) · = content complete + reviewed.

What this datasheet is (and is not)

- **Is:** the product datasheet — headline features, key specs, operating modes, interface/port map, host register map, programming model, integration/deliverables.
- **Is not:** the internal design spec, μ Arch, or DV testplan. This datasheet **distills** those; each chapter cites its source. Ground truth = the as-built RTL (rtl/debug/zen_ppu*.sv, ip/ppu-debug/rtl/); **RTL wins** on structural disagreement.
- **Altitude:** *what the PPU does + how to use it*. Uncharacterized numbers are marked **TBC** with an owner.

How to read it

- Start at [\[\[docs/datasheets/ppu/DSHT-01-OVERVIEW|DSHT-01\]\]](#), then read the chapters relevant to your integration.
- **Numbers policy:** every quantitative spec is (a) sourced from the RTL/register-map, (b) a standards/ISA requirement, or (c) marked **TBC** with an owner. **No unsourced numbers.**

Chapter map

DSHT	Chapter	Scope	Source
[[docs/datasheets/ppu/DSHT-00-INDEX DSHT-00]]	Intro & TOC	this hub — map · conventions	—
[[docs/datasheets/ppu/DSHT-01-OVERVIEW DSHT-01]]	Product overview	positioning · target apps · block diagram	rtl/debug [[docs/kits/PPU-LICENSE PPU-LICENSE]]
[[docs/datasheets/ppu/DSHT-02-FEATURES-COMPLIANCE DSHT-02]]	Key features & standards	feature matrix · ISA basis	ip/ppu-datasheet/ISA-REQUIREMENTS
[[docs/datasheets/ppu/DSHT-03-APPLICATIONS DSHT-03]]	Applications / use models	target deployments	—
[[docs/datasheets/ppu/DSHT-04-CONFIGURABLE-OPTIONS DSHT-04]]	Configurable options	build-time parameters	rtl/debug/PPU-CONFIGURABLE-OPTIONS
[[docs/datasheets/ppu/DSHT-05-FUNCTIONAL-OVERVIEW DSHT-05]]	Functional overview	black-box behavior · fetch/monitor/intervene	rtl/debug/PPU-FUNCTIONAL-OVERVIEW
[[docs/datasheets/ppu/DSHT-06-INTERFACES-SIGNALS DSHT-06]]	Interfaces & signals	top-level interfaces by group	rtl/debug/PPU-INTERFACES-SIGNALS
[[docs/datasheets/ppu/DSHT-07-CLOCKING-RESET DSHT-07]]	Clocking & reset	clock/reset requirements	rtl/debug/PPU-CLOCKING-RESET
[[docs/datasheets/ppu/DSHT-08-POWER-ON-RESET-INIT DSHT-08]]	Power-on / reset / init	bring-up ordering	rtl/debug/PPU-POWER-ON-RESET-INIT
[[docs/datasheets/ppu/DSHT-09-PROGRAMMING-MODEL DSHT-09]]	Programming model	word-load · run · monitor/intervene	ip/ppu-datasheet/PPU-PROGRAMMING-MODEL
[[docs/datasheets/ppu/DSHT-10-REGISTER-MAP DSHT-10]]	Register map	host CSR offsets/fields/access	rtl/debug/PPU-REGISTER-MAP · zen_ppu
[[docs/datasheets/ppu/DSHT-11-PERFORMANCE DSHT-11]]	Performance	throughput/latency envelope	rtl/debug/PPU-PERFORMANCE
[[docs/datasheets/ppu/DSHT-12-PHYSICAL DSHT-12]]	Physical / implementation	area/fmax · constraints	TBC
[[docs/datasheets/ppu/DSHT-13-RAS DSHT-13]]	RAS & data integrity	error reporting summary	rtl/debug/PPU-RAS
[[docs/datasheets/ppu/DSHT-14-VERIFICATION-DELIVERABLES DSHT-14]]	Verification & deliverables	TB/coverage · manifest	ip/ppu-datasheet/PPU-VERIFICATION-DELIVERABLES [[docs/kits/PPU-LICENSE PPU-LICENSE]]
[[docs/datasheets/ppu/DSHT-15-INTEGRATION-GUIDE DSHT-15]]	Integration guide	instantiation · tie-offs · exposure	ip/ppu-datasheet/PPU-INTEGRATION-GUIDE
[[docs/datasheets/ppu/DSHT-16-ORDERING-LICENSING DSHT-16]]	Ordering / licensing / support	commercial · provenance	[[docs/kits/PPU-LICENSE PPU-LICENSE]]
[[docs/datasheets/ppu/DSHT-A-APPENDICES DSHT-A]]	Appendices	signals · acronyms · revision history	—

Moniker & provenance

- **Moniker DSHT** (DataSheeT) per shared-agent-infra/DATASHEET-STANDARD.md.
- **Path:** docs/datasheets/ppu/DSHT-NN-NAME.md; hub = DSHT-00-INDEX.md; appendices = DSHT-A-APPENDICES.md.
- **Provenance:** distilled from the as-built RTL (rtl/debug/zen_ppu.sv, zen_ppu_apb.sv, zen_ppu_axil.sv, zen_ppu_pkg.sv; ip/ppu-debug/rtl/ppu.sv, ppu_pram.sv), the PPU ISA reference / integration guide, and the PPU license kit. **IP authorship/lineage retained by zen-debug** (Sur → axon.micro → zen-debug); see [[docs/kits/PPU-DEBUG-LICENSE-KIT|PPU-DEBUG-LICENSE-KIT]] + ip/ppu-debug/TRANSFER-PROVENANCE.md.

DSHT-01 · Product Overview

Scope. What the Zen PPU is, where it fits, and its headline shape. *Source:* [rtl/debug/zen_ppu.sv](#), [ip/ppu-debug/](#) + [\[\[docs/kits/PPU-DEBUG-LICENSE-KIT|kit\]\]](#) — *SOURCED.*

What it is

- The **Zen PPU** (Patch Processing Unit) is a small, always-on **J1-style FORTH stack processor** a client instantiates **inside their design** as an in-silicon debug/diagnostic co-processor.
- It runs loadable FORTH diagnostic words from a patch RAM (**PRAM**) and reaches the surrounding design over a 16-bit I/O bus to **MONITOR** (read, @) and **INTERVENE** (write, !) on registers and signals — and its words can be **field-patched live**.
- Ships as **synthesizable RTL** (core + PRAM + APB/AXI4-Lite host frontends) plus a full software/verification stack: FORTH assembler/disassembler/ISA-sim, TLM + visualizer, and a client SDK/CLI.

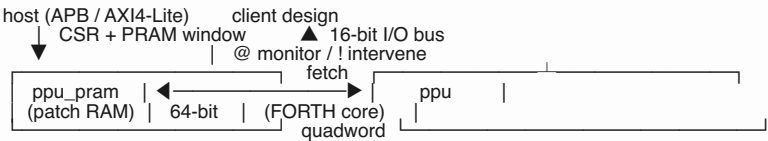
Positioning

- Complements the fixed JTAG-TAP / RISC-V Debug-Module catalog with a **programmable, in-design** debug agent: it keeps running (and observing) even when the main cores are wedged, and its behavior is reprogrammed by loading a new FORTH word — no respin.
- **Resetting the PPU does not reset the design it observes** — it is a bystander instrument, not in the functional datapath.

Headline shape

- **Core:** 16-bit data + 16-bit instruction J1 FORTH machine; dual 32-entry stacks; 13-bit PC (8K-word space). *Source:* [zen_ppu_pkg.sv:37–39](#), [PPU-ISA-REFERENCE.md](#) — *SOURCED.*
- **Host attach:** APB or AXI4-Lite frontend ([zen_ppu_apb.sv](#) / [zen_ppu_axil.sv](#)) exposing a PRAM write window + control/status/PC/TOS/reset-PC registers.
- **Design tap:** one 16-bit MONITOR/INTERVENTION I/O bus into the client's address decode.
- **Provenance:** authored by zen-debug (lineage Sur → axon.micro → zen-debug); transferred into zen-peripheral (zen-ip) 2026-08-21. See [\[\[docs/datasheets/ppu/DSHT-16-ORDERING-LICENSING|DSHT-16\]\]](#).

Block diagram



Source cross-refs

- Kit: [\[\[docs/kits/PPU-DEBUG-LICENSE-KIT|PPU-DEBUG-LICENSE-KIT\]\]](#) · ISA: [ip/ppu-debug/docs/PPU-ISA-REFERENCE.md](#) · Integration: [ip/ppu-debug/docs/PPU-INTEGRATION-GUIDE.md](#).

DSHT-01 · PPU · product overview.

title: “PPU — Key Features & Standards” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-02 · Key Features & Standards

Scope. Headline feature matrix + the ISA/bus bases the PPU conforms to.
Source: ip/ppu-debug/docs/PPU-ISA-REFERENCE.md, rtl/debug/* — **SOURCED.**

Feature matrix

Feature	Detail	Source
FORTH stack core	J1-style; 16-bit data + instruction; TOS+NOS caching	PPU-ISA-REFERENCE.md
Dual stacks	data + return stacks, 32 entries each	zen_ppu_pkg.sv:37–38
Address space	13-bit PC → 8K-word program space	zen_ppu_pkg.sv:39
MONITOR tap	ALU op 0xC (@): read design I/O address = TOS	PPU-ISA-REFERENCE.md
INTERVENTION tap	ALU N→[T] (bit 5, I): write NOS to I/O address TOS	PPU-ISA-REFERENCE.md
Patch RAM	loadable/field-patchable FORTH words; 1 word/cycle load port	PPU-INTEGRATION-GUIDE.md §3.4
Live field patch	a single PRAM word can be patched in place, PPU running	PPU-INTEGRATION-GUIDE.md §5
Host frontends	APB and AXI4-Lite wrappers	zen_ppu_apb.sv, zen_ppu_axil.sv
Quiesce / step	ext_stall_i freezes the pipe; single-step via pulsed stall	PPU-ISA-REFERENCE.md (reset/boot)
Software stack	assembler, disassembler, ISA-sim, TLM+visualizer, SDK/CLI	[[docs/kits/PPU-DEBUG-LICENSE-KIT kit]]

Standards / conformance

- **Instruction set:** J1-style FORTH; the shipped ppu.sv **is** the ISA ground truth — the ISA reference is verified encoding-by-encoding against RTL, and the Python ISA-sim is cross-checked == RTL (zen-debug SPRINT-05). *Source:* PPU-ISA-REFERENCE.md, ip/ppu-debug/scripts/parity_check.py — **SOURCED.**
- **Host buses:** AMBA **APB** and **AXI4-Lite** (register/PRAM access) via the struct-typed peripheral bus frontends used across the Zen portfolio.
- **Synthesis discipline:** Verilator -Wall clean; Yosys §11 design-style (no inferred latches). *Numbers/PPA:* see [[docs/datasheets/ppu/DSHT-12-PHYSICAL|DSHT-12]] (TBC).

Source cross-refs

- ISA: ip/ppu-debug/docs/PPU-ISA-REFERENCE.md · Kit: [[docs/kits/PPU-DEBUG-LICENSE-KIT|PPU-DEBUG-LICENSE-KIT]].

DSHT-02 · PPU · features & standards.

title: “PPU — 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 deployments and typical use models for the Zen PPU.

Target applications

- **Post-silicon bring-up** — a resident agent that reads status registers and drives control signals to unstick a wedged block during first-power-on debug.

- **In-field diagnostics** — ship a diagnostic word library in PRAM; run monitor/intervene sequences on deployed silicon without a debugger attached.
- **Field patch / workaround** — patch a diagnostic word live (or re-point the PPU at a freshly staged library) to work around a bug without a mask respin.
- **NoC / fabric probing** — monitor NoC or fabric status registers and field-patch diagnostic behavior without a rebuild.

Use models

- **Staged-and-run** — hold reset, stream a word image into PRAM, release reset at the entry word; the PPU runs its diagnostic autonomously.
- **Host-driven** — a host (JTAG/APB/AXI4-Lite/mailbox) loads words, starts/stops via the CTRL register, and reads PC/TOS/STATUS for observability.
- **Live intervene** — a running word forces a stuck register to a known value (!) then reads it back (@) to confirm — the canonical detect-stuck-reg → field-patch story.
- **Co-sim / pre-silicon** — driven through the VPD co-sim bridge or the TLM/visualizer against the ISA model or Verilated RTL, same client SDK.

Recommended companions

- [[docs/datasheets/uart/DSHT-00-INDEX|UART]] or a mailbox for a host command channel · the JTAG-TAP / Debug-Module catalog for a standard debug-access path into the PRAM/CSR window.

Source cross-refs

- Kit: [[docs/kits/PPU-DEBUG-LICENSE-KIT|PPU-DEBUG-LICENSE-KIT]] · Cookbook: ip/ppu-debug/docs/PPU-DEBUG-PATTERNS-COOKBOOK.md.

DSHT-03 · PPU · applications.

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

DSHT-04 · Configurable Options

Scope. Build-time parameters. *Source:* rtl/debug/zen_ppu_pkg.sv, ip/ppu-debug/docs/PPU-INTEGRATION-GUIDE.md §1 — *SOURCED.* All defaults RTL-sourced.

Core (ppu)

Parameter	Default	Meaning
STACK_DEPTH	32	Data-stack depth (entries). 5-bit dsp.
RSTACK_DEPTH	32	Return-stack depth (entries). 5-bit rsp.
PC_WIDTH	13	Program-counter width → 2 ¹³ = 8K-word address space.

Patch RAM (ppu_pram)

Parameter	Default	Meaning
PC_WIDTH	13	Must match the core.
WORDS	256	PRAM depth in 16-bit words. Size to your word library.
FETCH_LATENCY	2	Modelled read latency (cycles); the fetch handshake tolerates any value ≥ 1.

Host wrapper (zen_ppu_apb / zen_ppu_axil)

Parameter	Default	Meaning	Source
ADDR_WIDTH	12	Host address width into the wrapper window	zen_ppu_axil.sv:30
IMEM_DEPTH_P	1024	Instruction-memory (PRAM) words behind the write window	zen_ppu_pkg.sv:42 (MEM_DEPTH)

Notes

- Data and instruction widths are **fixed at 16-bit** (J1 architecture) — not parameterized.
- Fetch data path is a fixed **64-bit** aligned quadword (fetch_data [63:0]).

Source cross-refs

- rtl/debug/zen_ppu_pkg.sv:37–42 · zen_ppu_axil.sv:30 · PPU-INTEGRATION-GUIDE.md §1.

DSHT-04 · PPU · configurable options.

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

DSHT-05 · Functional Overview

Scope. Black-box behavior: fetch, execute, and the monitor/intervene taps.
Source: ip/ppu-debug/docs/PPU-ISA-REFERENCE.md, PPU-INTEGRATION-GUIDE.md §2 — *SOURCED.*

Execution model

- A J1-style FORTH machine: each 16-bit instruction is one of **LIT** (push immediate), **JMP**, **JZ** (pop + branch-if-zero), **CALL**, or **ALU** (stack/ALU op with side-effect bits). *Source:* PPU-ISA-REFERENCE.md (encoding) — *SOURCED.*
- State: top-of-stack (tos) + next-on-stack (nos) cached; data + return stacks in RAM; PC in 8K-word space.

Fetch

- The core asserts fetch_req_o with a word address; the PRAM returns the aligned **64-bit quadword** on fetch_data_i after FETCH_LATENCY cycles. The handshake tolerates any latency ≥ 1.

MONITOR (read / @)

- An ALU word with op 0xC drives io_rd_req_o with address = TOS; the design returns data on io_rd_data_i with io_rd_valid_i. The pipe **stalls** until valid. Result becomes the new TOS — this is how a word *observes* live state.

INTERVENTION (write / !)

- An ALU word with the N→[T] bit drives io_wr_req_o with address = TOS, data = NOS; the design pulses io_wr_done_i on commit. The pipe **stalls** until done — this is how a word *drives/patches* live state.

Load & patch

- The host streams FORTH words into PRAM one 16-bit word/cycle via the load port — while the PPU is held in reset/stalled (staging) or **live** (hot-patching a running library).

Quiesce / step / reset

- ext_stall_i freezes the pipe (state held). Single-step = hold stall, pulse it low for one instruction. Reset (rst_ni low) reboots the PC at reset_pc_i and clears the stacks; the PPU issues no fetches while held in reset.

Source cross-refs

- PPU-ISA-REFERENCE.md (ALU table, taps, reset/boot) · PPU-INTEGRATION-GUIDE.md §2,§5.

DSHT-05 · PPU · functional overview.

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

DSHT-06 · Interfaces & Signals

Scope. Top-level interfaces by group. *Source:* rtl/debug/zen_ppu.sv ports + ip/ppu-debug/docs/PPU-INTEGRATION-GUIDE.md §3 — *SOURCED.*

Clock / reset / control (ppu)

Signal	Dir	Width	Function
clk_i	in	1	PPU clock
rst_ni	in	1	Active-low synchronous reset; boots at reset_pc_i
reset_pc_i	in	PC_WIDTH (13)	Boot PC (word address)
ext_stall_i	in	1	1 = quiesce/STALL the pipe (tie 0 if unused)

Fetch bus (ppu ↔ ppu_pram)

Signal	Dir	Width	Function
fetch_req_o	out	1	Fetch request
fetch_addr_o	out	PC_WIDTH+1	Word address
fetch_valid_i	in	1	Fetch data valid
fetch_data_i	in	64	Aligned quadword

Design I/O bus — MONITOR + INTERVENTION

Signal	Dir	Width	Function
io_addr_o	out	16	I/O address (= TOS); shared read/write
io_rd_req_o	out	1	MONITOR read request
io_rd_valid_i	in	1	Read-data valid (completes the stall)
io_rd_data_i	in	16	Read data
io_wr_req_o	out	1	INTERVENTION write request
io_wr_data_o	out	16	Write data (= NOS)
io_wr_done_i	in	1	Write-commit pulse (completes the stall)

PRAM word-load port (ppu_pram)

Signal	Dir	Width	Function
load_en_i	in	1	Assert to write one word
load_addr_i	in	PC_WIDTH	PRAM word address
load_data_i	in	16	16-bit FORTH word

Host frontend

- One of **APB** (zen_ppu_apb) or **AXI4-Lite** (zen_ppu_axil) presents the standard bus signals plus the register/PRAM window — see [[docs/datasheets/ppu/DSHT-10-REGISTER-MAPIDSHT-10]].

Source cross-refs

- Ports: rtl/debug/zen_ppu.sv · PPU-INTEGRATION-GUIDE.md §3 · PPU-INTERVENTION-INTERFACE.md (handshake timing).

DSHT-06 · PPU · interfaces & signals.

title: “PPU — Clocking & Reset” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-07 · Clocking & Reset

Scope. Clock and reset requirements. *Source:* rtl/debug/zen_ppu.sv, ip/ppu-debug/docs/PPU-ISA-REFERENCE.md (reset/boot) — **SOURCED.**

Clocking

- **Single clock domain** clk_i for the core, PRAM, and host wrapper (when the wrapper shares the domain). All internal state is synchronous to clk_i.
- The design I/O bus and fetch bus are **synchronous handshakes** in the clk_i domain; a client crossing to another clock must add its own CDC on the I/O/PRAM taps (client responsibility).
- Target fmax: **TBC** (owner: physical) — see [[docs/datasheets/ppu/DSHT-12-PHYSICALDSHT-12]].

Reset

- **rst_ni** — active-low, **synchronous**. On assertion: pc ← reset_pc_i, dsp/rsp/tos ← 0; no fetches issue while held.
- **Independence:** resetting the PPU does **not** reset the design it observes — the PPU is a bystander instrument.
- **Boot PC:** reset_pc_i selects the entry word; a typical image places a branch-to-self “hlt” loop at the entry so an unbooted PPU spins harmlessly.

Quiesce vs reset

- ext_stall_i (STALL) freezes the pipe with state intact — the safe point to hot-patch PRAM or inspect. Reset (KILL) aborts and reboots at reset_pc_i.

Source cross-refs

- PPU-ISA-REFERENCE.md (reset/boot behavior) · PPU-INTEGRATION-GUIDE.md §3.1,§5.

DSHT-07 · PPU · clocking & reset.

title: “PPU — Power-On / Reset / Init” created: 2026-09-28 status: authored (pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-08 · Power-On / Reset / Init

Scope. Bring-up ordering from reset to a running diagnostic. *Source:* ip/ppu-debug/docs/PPU-INTEGRATION-GUIDE.md §5, PPU-ISA-REFERENCE.md (reset/boot) — **SOURCED.**

Bring-up sequence

1. **Hold reset.** Assert rst_ni low (or ext_stall_i high). Stacks and PC are cleared; no fetches issue.
2. **Stage words.** Host streams the FORTH word image into PRAM one 16-bit word/cycle (load_en_i / load_addr_i / load_data_i), placing the entry word at reset_pc_i. Via the host wrapper this is the PRAM write window (offsets 0x000–0x7FF).
3. **Release reset.** Deassert rst_ni (or write CTRL.run=1). The PC loads reset_pc_i; the PPU fetches the entry word and runs.
4. **Run.** The PPU executes continuously, issuing MONITOR reads / INTERVENTION writes as its words dictate, stalling on each I/O op until the design returns valid/done.
5. **Pause.** Assert ext_stall_i to freeze mid-stream (safe to hot-patch/inspect); deassert to resume.
6. **Re-point (KILL).** Reassert rst_ni to abort a runaway word and reboot at reset_pc_i (e.g. after staging a fresh library).

Reset state (RTL-sourced)

- $pc \leftarrow reset_pc_i$, $dsp = rsp = tos = 0$. STATUS reports ready once reset completes. *Source: zen_ppu_pkg.sv:109–117, PPU-ISA-REFERENCE.md — SOURCED.*

Init defaults

- PRAM contents are **undefined** at power-on until loaded — stage a word image (or at minimum a hlt-loop at $reset_pc_i$) before releasing reset.

Source cross-refs

- PPU-INTEGRATION-GUIDE.md §5 · PPU-ISA-REFERENCE.md (reset/boot).

DSHT-08 · PPU · power-on / reset / init.

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

DSHT-09 · Programming Model

Scope. How software drives the PPU: word authoring, host control, and the monitor/intervene primitives. *Source: ip/ppu-debug/docs/PPU-ISA-REFERENCE.md, PPU-TOOLCHAIN-GUIDE.md — SOURCED.*

Two audiences

- **Diagnostic author (FORTH):** writes words assembled by `ppu_asm.py` into a PRAM image; the words do the monitor/intervene work.
- **Host integrator (CSR):** loads PRAM, starts/stops, and observes via the APB/AXI4-Lite register window ([[docs/datasheets/ppu/DSHT-10-REGISTER-MAPIDSHT-10]]).

FORTH primitives (the debug verbs)

- **@ (MONITOR read):** push an I/O address, execute @ (ALU op 0xC) → TOS ← $design[addr]$. *Source: PPU-ISA-REFERENCE.md ALU table — SOURCED.*
- **! (INTERVENTION write):** push value then address, execute ! (N→[T], 0x6122) → $design[addr] \leftarrow value$. *Source: PPU-ISA-REFERENCE.md common words — SOURCED.*
- Stack/ALU words (DUP 0x6081, DROP 0x6103, ADD 0x6203, CALL n 0x4000In, RET 0x700C, ...) per the ISA reference.

Host control sequence

1. Load PRAM image through the write window (0x000–0x7FF).
2. Set RESET_PC (0x810) to the entry word if not default.
3. Write CTRL.run (0x800[0]) = 1 to start; CTRL.reset (0x800[1]) to soft-reset.
4. Poll STATUS (0x804) for running/halted/err_*/io_wait; read PC (0x808) and TOS (0x80C) for observability.

Toolchain

- `ppu_asm.py` (assemble FORTH → PRAM image), `ppu_disasm.py` (disassemble), `ppu_sim.py` (ISA model, == RTL). Diagnostic word library in `tools/forth/"`.`ppu`.

Source cross-refs

- PPU-ISA-REFERENCE.md · PPU-TOOLCHAIN-GUIDE.md · register map: [[docs/datasheets/ppu/DSHT-10-REGISTER-MAPIDSHT-10]].

DSHT-09 · PPU · programming model.

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

DSHT-10 · Register Map

Scope. The host-facing CSR + PRAM window presented by the APB / AXI4-Lite wrapper. *Source:* rtl/debug/zen_ppu_axil.sv:8–13,58–63 (offsets) + zen_ppu_pkg.sv:99–117 (fields) — *SOURCED*.

Address window

Source: zen_ppu_axil.sv:8–13,88–89 — *SOURCED*. ADDR_WIDTH=12.

Range	Region	Access	Function
0x000–0x7FF	Instruction memory (PRAM)	W	Word-write window, 1K x 16-bit — stage/patch FORTH words
0x800	PPU_CTRL	R/W	Run / reset control
0x804	PPU_STATUS	R	Running / halted / error / io-wait
0x808	PPU_PC	R	Current program counter
0x80C	PPU_TOS	R	Top of stack (debug)
0x810	PPU_RESET_PC	R/W	Boot PC value

Field detail

Source: zen_ppu_pkg.sv:99–117 — *SOURCED*. - **PPU_CTRL** — [0] run · [1] reset (soft) · [15:2] reserved. - **PPU_STATUS** — [0] ready · [1] running · [2] halted · [3] err_dstack · [4] err_rstack · [5] io_wait · [15:6] reserved. - **PPU_PC** — 13-bit current PC (zero-extended to the bus width). - **PPU_TOS** — 16-bit top-of-stack snapshot. - **PPU_RESET_PC** — 13-bit boot PC (zero-extended).

Bit-exact reset values per field: **TBC** (owner: verif) where not asserted in zen_ppu_axil.sv. The IMEM window is **write-only** (readback of staged words is via the PPU, not the CSR read path).

PPU-internal I/O space (for reference)

- Separate from the host CSR map above, the PPU’s own 16-bit I/O bus uses a {space[15:12], offset[11:0]} layout (GPIO/CTRL/STATUS/JTAG spaces) defined in zen_ppu_pkg.sv:52–82 — this is the *design-tap* address space FORTH words reference, mapped by the client’s decode (see [[docs/datasheets/ppu/DSHT-15-INTEGRATION-GUIDE|DSHT-15]]).

Source cross-refs

- Offsets: rtl/debug/zen_ppu_axil.sv:58–63 (mirror in zen_ppu_apb.sv:59–63) · Fields: rtl/debug/zen_ppu_pkg.sv:99–117.

DSHT-10 · PPU · register map.

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

DSHT-11 · Performance

Scope. Throughput/latency envelope at the architectural level. *Source:* rtl/debug/zen_ppu.sv, ip/ppu-debug/docs/PPU-ISA-REFERENCE.md — *architectural; absolute rates TBC*.

Architectural throughput

- One FORTH instruction per cycle** for non-I/O, non-stalled words (LIT/JMP/JZ/CALL/ALU) — single-cycle J1 execution. *Source:* PPU-ISA-REFERENCE.md (per-instruction effects) — *SOURCED (architectural)*.
- Fetch:** one aligned 64-bit quadword per fetch after FETCH_LATENCY cycles (default 2); handshake tolerates ≥ 1.

Stall behavior (dominant cost)

- **MONITOR read** and **INTERVENTION write** each stall the pipe until the design returns `io_rd_valid_i` / `io_wr_done_i`. Effective throughput of monitor/intervene sequences is therefore set by the **client's I/O decode latency**, not the PPU.
- Minimal single-cycle regfile decode → each @/! completes in ~1–2 cycles; slow/off-chip taps extend proportionally.

Absolute numbers

- **fmax / MIPS at a given node: TBC** (owner: physical) — this IP ships with functional (Verilator/\$11) sign-off; timing/PPA is a downstream implementation activity. See [\[\[docs/datasheets/ppu/DSHT-12-PHYSICAL/DSHT-12\]\]](#).

Source cross-refs

- PPU-ISA-REFERENCE.md (execution + stalls) · PPU-INTERVENTION-INTERFACE.md (handshake).

DSHT-11 · PPU · performance.

title: “PPU — Physical / Implementation” created: 2026-09-28 status: outline
(numbers TBC) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-12 · Physical / Implementation

Scope. Area / fmax / power envelope and implementation constraints.
Ground truth = as-built RTL; absolute PPA is uncharacterized.

Status

- **Area, fmax, power: TBC** (owner: physical). This IP ships with **functional** sign-off only — Verilator -Wall clean and Yosys \$11 design-style (no inferred latches). Timing/PPA closure is a downstream, node-specific activity outside datasheet scope.

Implementation notes (structural, RTL-sourced)

- Small core (~400 lines) with two stack RAMs (32×16-bit each), PC/TOS/NOS registers, and a single-cycle ALU — no multi-cycle arithmetic, no caches. *Source: ip/ppu-debug/rtl/ppu.sv — SOURCED (structure only).*
- PRAM sized by WORDS (default 256 × 16-bit); host IMEM window IMEM_DEPTH_P (default 1024). Memory is the dominant area term and scales with these parameters.
- Single clock domain; synchronous active-low reset (see [\[\[docs/datasheets/ppu/DSHT-07-CLOCKING-RESET/DSHT-07\]\]](#)).

Source cross-refs

- ip/ppu-debug/rtl/ppu.sv, ppu_pram.sv · synth discipline: [\[\[docs/datasheets/ppu/DSHT-02-FEATURES-COMPLIANCE/DSHT-02\]\]](#).

DSHT-12 · PPU · physical / implementation.

title: “PPU — RAS & Data Integrity” created: 2026-09-28 status: authored
(pending review) standard: shared-agent-infra/DATASHEET-STANDARD.md —

DSHT-13 · RAS & Data Integrity

Scope. Error detection/reporting the PPU surfaces. *Source: rtl/debug/zen_ppu_pkg.sv:109–117 (STATUS), ppu_state_e — SOURCED.*

Error reporting

- **STATUS register** (0x804) exposes error/health bits: `err_dstack` (data-stack over/underflow), `err_rstack` (return-stack over/underflow), `halted`, `io_wait`, `running`, `ready`. *Source: zen_ppu_pkg.sv:109–117 — SOURCED.*
- **State machine** `ppu_state_e`: `HALT` / `RUN` / `ERROR` (stack fault) / `WAIT_IO`. An

ERROR state is a latched stack-integrity fault. *Source: zen_ppu_pkg.sv:88–93 — SOURCED.*

Containment properties

- **Bystander isolation:** the PPU cannot corrupt functional traffic it is not driving — it acts only through the explicit MONITOR/INTERVENTION taps a client decodes. Resetting the PPU leaves the observed design untouched.
- **Bounded intervention:** a write only reaches signals the client’s I/O decode chooses to make writable; read-only taps simply omit a write decode (client controls the exposure surface).
- **Runaway control:** ext_stall_i (freeze) and rst_ni (KILL/reboot) give the host two independent ways to stop a misbehaving word.

Not covered (scope)

- No ECC/parity on the stacks/PRAM in v1 (data-integrity of the PRAM image is the loader’s responsibility). PRAM/stack hardening: **TBC** (owner: arch) if a licensee requires it.

Source cross-refs

- rtl/debug/zen_ppu_pkg.sv:88–117 · behavior: [\[\[docs/datasheets/ppu/DSHT-05-FUNCTIONAL-OVERVIEW|DSHT-05\]\]](#).

DSHT-13 · PPU · RAS & data integrity.

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

DSHT-14 · Verification & Deliverables

Scope. How the PPU is verified and what a licensee receives. *Source: [\[\[docs/kits/PPU-DEBUG-LICENSE-KIT|kit\]\]](#), ip/ppu-debug/docs/PPU-DELIVERABLE-BUNDLE.md, ip/ppu-debug/scripts/ — SOURCED.*

Verification

- **ISA ⇄ RTL parity:** the Python ISA simulator is cross-checked instruction-for-instruction against ppu.sv (zen-debug SPRINT-05 RTL cross-check); scripts/parity_check.py guards against drift.
- **Push-button harness:** make demo (end-to-end on real RTL), make regress (ISA==RTL parity), make audit (bundle manifest), make check (CI gate). *Source: ip/ppu-debug/Makefile, scripts/* — SOURCED.*
- **Co-sim proofs:** integrated + passing in the VPD flow — PPU-VPD, BAREMETAL-PPU, UNIFIED-VPD: PASS. *Source: [\[\[docs/kits/PPU-DEBUG-LICENSE-KIT|kit\]\]](#) — SOURCED.*
- **Lint/synth:** Verilator -Wall clean; Yosys \$11 design-style.

Deliverables (what a licensee receives)

Layer	Path (ip/ppu-debug/)
PPU RTL (+ RTL-TLM)	rtl/, rtl-tlm/
Host frontends	rtl/debug/zen_ppu_apb.sv, zen_ppu_axil.sv (zen-peripheral tree)
FORTH toolchain	tools/forth/ppu_asm.py, ppu_disasm.py, ppu_sim.py
Diagnostic words	tools/forth/*.ppu
TLM + visualizer	viz/ (ISA or Verilated-RTL backend)
Client SDK / CLI	sdk/ppu_debug.py, sdk/examples/
Docs	User Guide, Integration Guide, ISA Reference, Toolchain Guide, TLM-API, Cookbook, this datasheet
Manifests	docs/PPU-DELIVERABLE-BUNDLE.md, [[docs/kits/PPU-DEBUG-LICENSE-KIT kit]] , TRANSFER-PROVENANCE.md

Source cross-refs

- [\[\[docs/kits/PPU-DEBUG-LICENSE-KIT|PPU-DEBUG-LICENSE-KIT\]\]](#) · ip/ppu-

DSHT-14 · PPU · verification & deliverables.

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

DSHT-15 · Integration Guide

Scope. Instantiation, tie-offs, and design-signal exposure. Distilled from the full guide. *Source:* [ip/ppu-debug/docs/PPU-INTEGRATION-GUIDE.md](#) — *SOURCED.*

Instantiate

- Instantiate **ppu** (core) + **ppu_pram** (patch RAM); wire the fetch bus between them (widths per PC_WIDTH). Attach one host frontend — **zen_ppu_apb** or **zen_ppu_axil** — for PRAM load + control/status.
- Tie-offs: `ext_stall_i` = 0 if unused; `reset_pc_i` = entry word of your staged library.

Expose design signals to the PPU

- The PPU sees the design as a flat **16-bit I/O address space** (`io_addr_o` = TOS). Assign each observable/controllable signal a 16-bit address and document it — FORTH words reference these as literals.
- **MONITOR path:** on `io_rd_req_o`, decode `io_addr_o`, return the signal on `io_rd_data_i` + pulse `io_rd_valid_i`.
- **INTERVENTION path:** on `io_wr_req_o`, decode `io_addr_o`, write `io_wr_data_o` into the target, then pulse `io_wr_done_i`.
- Signals > 16 bits → split across addresses; < 16 bits → zero-extend; read-only → omit the write decode. Minimal single-cycle regfile: reflect `req`→`valid`/`done` one cycle later (see `sim/design_stub.sv`).

Word-load

- Drive the PRAM load port from any host path (JTAG/APB/AXI4-Lite/mailbox): one 16-bit word/cycle. Load while held in reset/stalled (staging) or live (hot-patch).

Bring-up

- Stage → release reset → run → (pause/patch) → (KILL/re-point). Proven end-to-end in [ip/ppu-debug/docs/evidence/SPRINT-01-smoke-test.md](#).

Source cross-refs

- Full guide: [ip/ppu-debug/docs/PPU-INTEGRATION-GUIDE.md](#) · handshake timing: [PPU-INTERVENTION-INTERFACE.md](#) · signals: [\[\[docs/datasheets/ppu/DSHT-06-INTERFACES-SIGNALS|DSHT-06\]\]](#).

DSHT-15 · PPU · integration guide.

title: "PPU — 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, and support. *Source:* [\[\[docs/kits/PPU-DEBUG-LICENSE-KIT|kit\]\]](#), [ip/ppu-debug/TRANSFER-PROVENANCE.md](#) — *SOURCED.*

Bundle

- **zen-ppu-debug-kit-v1** — silicon RTL + FORTH toolchain + TLM/visualizer + client SDK. Contents enumerated in [\[\[docs/datasheets/ppu/DSHT-14-VERIFICATION-DELIVERABLES|DSHT-14\]\]](#) and the kit manifest.
- **In-repo home:** `ip/ppu-debug/` (core/TLM/tools/viz/sdk) + `rtl/debug/` (host wrappers).

Licensing

- **Internal soft-license** by default (free IP flow within zen-semi); kit is license-ready-structured for external.
- **Provenance / lineage (retained)**: authored by zen-debug — lineage **Sur** → **axon.micro** → **zen-debug** (Dr. Lena Vasquez, Sana Rahimi, Bianca Moreau). Ownership transferred into zen-peripheral (zen-ip) **2026-08-21** (Rod-authorized, IP-ASSIM PHASE-07 §5-A); zen-debug retires its copy on its own timeline. Credit/lineage remain zen-debug's. *Source: [[docs/kits/PPU-DEBUG-LICENSE-KIT|kit]] header, TRANSFER-PROVENANCE.md — SOURCED.*

Support / roadmap

- **Verification status**: functional sign-off (ISA==RTL parity, VPD co-sim PASS); PPA/timing is downstream ([[docs/datasheets/ppu/DSHT-12-PHYSICALIDSHT-12]]).
- **Known future work**: in-band program-load-over-wire primitive; automotive qualification of the PPU RTL (if a licensee requires); PRAM/stack integrity hardening (TBC).

Source cross-refs

- Kit: [[docs/kits/PPU-DEBUG-LICENSE-KIT|PPU-DEBUG-LICENSE-KIT]] · provenance: ip/ppu-debug/TRANSFER-PROVENANCE.md.

DSHT-16 · PPU · ordering / licensing / support.

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

DSHT-A · Appendices

| **Scope.** Signal index, acronyms, revision history.

A.1 · Signal index (quick reference)

- **Control**: clk_i, rst_ni, reset_pc_i[12:0], ext_stall_i.
- **Fetch**: fetch_req_o, fetch_addr_o, fetch_valid_i, fetch_data_i[63:0].
- **MONITOR**: io_rd_req_o, io_addr_o[15:0], io_rd_valid_i, io_rd_data_i[15:0].
- **INTERVENTION**: io_wr_req_o, io_wr_data_o[15:0], io_wr_done_i.
- **PRAM load**: load_en_i, load_addr_i, load_data_i[15:0].
- Full detail: [[docs/datasheets/ppu/DSHT-06-INTERFACES-SIGNALS|DSHT-06]].

A.2 · Acronyms

- **PPU** — Patch Processing Unit (the debug co-processor).
- **PRAM** — Patch RAM (the FORTH program memory + word-load port).
- **TOS / NOS** — top / next-on-stack.
- **MONITOR / INTERVENTION** — the read (@) / write (!) design taps.
- **TLM** — transaction-level model (fast functional model of the IP).
- **J1** — the FORTH stack-CPU architecture family the core follows.

A.3 · Common encoded words (from ISA reference)

- DUP 0x6081 · DROP 0x6103 · SWAP 0x6180 · ADD 0x6203 · ! 0x6122 (INTERVENTION) · @ op 0xC (MONITOR) · CALL n 0x4000In · RET 0x700C. Full table: ip/ppu-debug/docs/PPU-ISA-REFERENCE.md.

A.4 · Revision history

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

Source cross-refs

- ISA: ip/ppu-debug/docs/PPU-ISA-REFERENCE.md · hub:

[[docs/datasheets/ppu/DSHT-00-INDEXIDSHT-00]].

DSHT-A · PPU · appendices.