# Generated Specification Document

## INTRODUCTION

-- Purpose and Scope --

Purpose
- Provide a synthesizable USB 2.0 Full-/High■Speed device core that bridges an external 8■bit UTMI PHY to a system-on-chip via a Wishbone slave and a shared 32■bit synchronous SRAM used for endpoint buffers. The core implements link/transaction control, DMA-backed payload moves, and a software-visible register/interrupt model for configurable endpoints.

Scope
- Protocol/link: USB 2.0 device operation over UTMI (FS/HS); PID/token handling, CRC5/CRC16 check/generation, IN/OUT/SETUP control, HS features (PING/NYET), SOF/frame capture, suspend/resume, bus reset detection, HS negotiation (chirps), and remote wakeup.
- Data buffering and memory: External single-port 32■bit SRAM as packet store; fixed■priority arbiter favoring protocol side; byte↔word DMA bridge handles alignment, word packing, and ring■buffer wrap; software can access SRAM opportunistically via Wishbone.
- Software interface: Wishbone B3 classic, single■beat, non■pipelined. Global registers (device address/state, interrupts, SOF timestamp, UTMI vendor sideband) and per■endpoint windows (CSR/INT/BUF0/BUF1). Two interrupt outputs and per■endpoint DMA req/ack.
- Endpoints: EP0 mandatory; EP1..EP15 compile■time selectable (example build EP0..EP3). Per■EP direction/type, max packet size, DMA enable, and double-buffer descriptors; live matched■EP view feeds the protocol engine.
- Clocks/CDC: Two domains—phy_clk (UTMI/protocol/memory) and wb_clk (Wishbone/RF view). Localized CDC in UTMI IF, protocol, memory arbiter, RF, and WB front■end. Reset synchronous by default (optional async build).
- Integration boundaries: Requires external UTMI PHY (8■bit) and 32■bit synchronous SRAM. Remote■wakeup supported; UTMI vendor sideband bridged. Best when wb_clk equals or is phase■related to phy_clk.
- Internal module roles: utmi_ls (line■state, power/speed, attach/reset/suspend/resume, HS chirp, UTMI control pins); crc5/crc16 (combinational CRC blocks); idma (byte↔word DMA, alignment, wrap, ZLP); pe (per■EP transaction/handshake, PID/sequence, DMA orchestration, descriptor write■back, interrupts); mem_arb (fixed■priority M vs W on SRAM); ep_rf (per■EP CSR/INT/descriptors, DMA flow, CDC); ep_rf_dummy (stub for unimplemented EPs); wb (Wishbone front■end and RF/SRAM region bridge).
- Limits/exclusions: Device■only (no host/hub/OTG), FS/HS only (no LS), single outstanding Wishbone transfer, no WB bursts/pipelining, no fairness for SRAM under heavy USB load, no HS test modes; overall USB 2.0 compliance: partial.

-- Features and Capabilities --

- USB 2.0 device function core with Full-Speed and High-Speed operation via 8-bit UTMI PHY
- Autonomous link-state/power control: attach debounce, bus-reset detect/handle, suspend entry/exit, host/device-initiated resume, HS chirp/negotiation with automatic FS fallback; remote-wakeup

generation
- Protocol engine for token/data/handshake sequencing: IN, OUT, SETUP, SOF, and HS PING; supports ACK/NACK/STALL/NYET, data PID selection (DATA0/1/2/MDATA), retries, zero-length packets, and endpoint policy enforcement
- Standards-aligned timing: micro/millisecond windows for reset, suspend, resume, settle, and HS chirp; HS/FS-specific transaction timeouts
- Robust CRC checking/generation: combinational CRC5 (tokens) and CRC16 (data) with PID complement verification
- Flexible endpoint subsystem: mandatory EP0 (control) plus build-time selectable additional endpoints; per-endpoint CSR/INT/BUF0/BUF1 descriptors; double-buffer management with software-visible toggles and automatic write-back
- Per-endpoint DMA enable, max-packet-size enforcement, and convenience flow-control flags (dma_in_buf_sz1, dma_out_buf_avail) for the currently active endpoint
- High-throughput byte↔word DMA bridge: unaligned start/end handling with read-modify-write, automatic last-word flush, two-word prefetch for sustained TX streaming, ring/wrap addressing, and safe aborts
- External memory integration: 32-bit synchronous single-port SSRAM as a shared packet buffer store
- Low-latency memory arbitration: fixed priority (core/DMA over software), combinational request path for M-side, one-cycle paced W-side acks, read-data broadcast, continuous sram_re
- System/bus interface: Wishbone B.3 classic slave with clean one-cycle ack pulses; address decode between Register File and Memory regions; single outstanding transfer FSM; integrated CDC for ack back to wb_clk
- Rich software-visible register map: main control/status, USB function address, interrupt masks/sources (sticky read-to-clear), SOF/microframe timestamping (frm_nat), UTMI vendor sideband status/control
- Interrupt architecture: two top-level outputs with independent masks and aggregation of RF- and per-endpoint events; per-endpoint sticky events for buffer updates, CRC/timeout/sequence errors, and short OUT packets; DMA request/ack vectors per endpoint
- Timing/framing and diagnostics: SOF reception, frame number capture, microframe counting, sub-frame timing, and comprehensive sticky RF events (attach/detach, reset, suspend start/end, protocol errors)
- Link control pins driven per state: XcvSelect (FS/HS), TermSel, OpMode[1:0] (Normal/Non-driving), SuspendM; active K-line drive for resume and HS chirp with safe TxValid/DataOut bounding
- Two-clock-domain design with localized CDC (UTMI/protocol/memory on phy_clk; Wishbone/RF on clk_i); synchronous resets by default; suspend indicator (susp_o) and resume request latching/clearing
- Build-time configurability: endpoint presence (USBF_HAVE_EPx), SRAM/UF controller address widths and region select, HS/FS timing and PID behavior parameters
- Designed for interoperability in many FS/HS scenarios with firmware cooperation; notes: HS Test Modes not implemented; strict Chapter 9 corner cases may require firmware policy

-- Standards Compliance --

Declared standards and conformance levels
- USB 2.0 device (FS/HS over UTMI/UTMI+): Partial compliance. Functionally interoperable for IN/OUT/SETUP, CRC, PIDs, handshakes, suspend/resume, reset, HS negotiation; not certification-complete.
- Wishbone B.3 (classic) slave: Compliant (single-beat, 32-bit, no bursts/tags/STALL/ERR/RTY). Not B.4 (no pipelining/bursts/byte-enables).
- Other on-chip interfaces (SRAM arbiter, DMA, endpoint RF): Proprietary/standards-neutral; no AXI/AHB/Avalon/JEDEC claim.

USB 2.0 — implemented behaviors (device-side)
- Link/PHY via UTMI: attach debounce; bus reset detect and HS negotiation (chirp K/J counting, HS

select after ≥3 pairs; design uses 6 alternations); suspend after idle; resume and remote-wakeup signaling (K for ~1 ms, gated by ≥5 ms suspend); UTMI pin drive (XcvrSelect, TermSel, OpMode, SuspendM) per state.
- Transactions: PID format/complement check; token address/endpoint filtering; CRC5/CRC16 check/generate; IN/OUT/SETUP engines with ACK/NACK/STALL and HS NYET/PING; data PID sequencing incl. HS ISO DATA0/1/2/MDATA; ZLP support; FS/HS timeouts; SOF and HS microframe capture.
- Endpoints: EP0 plus others (e.g., EP1–EP3 in this build). Max-packet-size and direction via CSRs; external-SRAM DMA buffering.

USB 2.0 — gaps and caveats (blockers for full certification)
- Chapter 9 control pipe not fully enforced in hardware: SETUP not guaranteed to always be accepted/override halt; fixed 8-byte SETUP not strictly enforced; no guaranteed auto-clear of halt on SETUP or DATA0 re-init; device address forced to 0 on reset not hard-guaranteed; status-stage/STALL recovery not fully constrained.
- High-Speed test modes missing: No SetFeature(TEST_MODE) or Test_J/Test_K/Test_SE0_NAK/Test_Packet.
- Optional traffic management left to firmware: No hardware enforcement of bInterval or HS multi-transactions per microframe; some ISO corner behaviors may differ from certification expectations.
- No Low-Speed support.

UTMI/UTMI+ integration notes affecting compliance
- SuspendM polarity and XcvrSelect width/mapping are PHY-specific; must be verified against the target PHY to ensure standards-conformant suspend/HS/FS selection.
- Timing windows (ms/µs) are parameterized from phy_clk; integrators must configure for spec limits across PVT.

CRC and PID compliance
- CRC5 (token): G(x)=x^5+x^2+1, LSB-first, init 0x1F; ones' complement compare — compliant.
- CRC16 (data): G(x)=x^16+x^15+x^2+1, LSB-first, init 0xFFFF; receiver residue check 0x800D — compliant.
- PID integrity: 4-bit complement rule enforced; PRE/ERR decode alias requires system-level gating by speed/context.

Wishbone compliance (system control interface)
- B.3 classic subset: CLK_I/RST_I/ADR_I/DAT_I/O/CYC_I/STB_I/WE_I, single ACK_O pulse; word-aligned 32-bit; variable latency allowed. CDC assumption: requests sampled in phy_clk; robust when wb_clk==phy_clk or timing/hold meet classic rules.

Non-claims and proprietary interfaces
- SRAM arbiter, internal memory handshakes, endpoint RF/DMA ports are not standardized buses; no AXI/AHB/Avalon/JEDEC conformance claimed.

Integration and certification expectations
- Use a standards-compliant UTMI/UTMI+ PHY; validate pin polarity/mapping. Configure timing parameters for FS/HS windows. Firmware must implement Chapter 9 requests/descriptors and mitigate noted EP0/control gaps (e.g., force address=0 on reset, SETUP handling, halt/toggle rules). Add HS Test Modes and stricter control-pipe enforcement in RTL for USB-IF certification.

-- External Dependencies and Assumptions --

- UTMI/UTMI+ PHY

- External 8-bit UTMI PHY required: DataIn[7:0], DataOut[7:0], TxValid/TxReady, RxValid/RxActive/RxError, XcvSelect, TermSel, OpMode[1:0], SuspendM, LineState[1:0], and a PHY-supplied clock phy_clk.
- PHY must supply a continuous UTMI clock (typically 60 MHz). All UTMI/protocol timing constants assume standard UTMI HS/FS rates.
- Control pin semantics must match the PHY: XcvSelect (FS/HS select), TermSel (FS termination), OpMode (00=Normal, 10=Disable NRZI/bit-stuff for reset/resume/chirp), SuspendM (active high not-suspended). Verify polarities against the target PHY.
- LineState coding must be UTMI-compliant (00=SE0, 01=J, 10=K, 11=SE1), synchronous, and glitch-free enough for 2-cycle qualification.
- During resume and HS chirp, the PHY must honor OpMode/TermSel/SuspendM to drive K and ignore DataOut. The core may drive zeros on DataOut in these modes.
- VBUS/attach input usb_vbus is required; polarity must match the implementation (high forces POR in the provided code). Invert externally if needed.
- Optional vendor sideband (VControl[3:0]/VControl_Load, VStatus[7:0]) may be left unconnected/tied off if the PHY does not implement it.

- USB link/host behavior
- Host must provide standards-compliant reset (SE0), suspend (idle), resume (K), and HS chirp timing. Device supports FS/HS only; no LS or HS Certification Test Modes implemented.
- Remote wake must follow USB policy (system-controlled); hardware enforces minimum timing only.

- External SRAM-like memory (packet buffer)
- Single 32-bit, single-port interface without byte enables; full-word writes only. sram_re is tied high (continuous read-enable).
- Core assumes zero-wait single-cycle read visibility: arbiter returns mack=mreq and mdin must be valid with that ack. If the memory has latency or requires CE/OE/turnaround, a wrapper must: (a) assert ack only when data is valid, (b) generate device-specific enables, and (c) preserve the presented single-cycle semantics to the core.
- Must tolerate continuous reads and back-to-back selections. Throughput must sustain near-continuous reads to feed 1 byte/cycle bursts on TX.
- Address width parameter SSRAM_HADR must match the actual memory depth and be used consistently system-wide.

- Memory arbitration (internal contract visible to integrator)
- Fixed priority: M (protocol/DMA) always preempts W (Wishbone/software). W can starve under sustained M traffic.
- W side acceptance pulses wack at most every other phy_clk when M is idle; software paths must tolerate variable latency.

- Wishbone system bus
- Classic WB B3 32-bit single-beat slave; no SEL/STALL/ERR/RTY. Software must perform aligned 32-bit accesses only.
- Address split by USBF_UFC_HADR (e.g., bit 17): 0=register file (RF), 1=memory window (MA). Read data must be valid when ACK pulses.
- CDC assumption: wb_clk is expected equal/tightly aligned to phy_clk. If clocks differ, provide a CDC bridge that synchronizes requests into phy_clk and re-times data/ACK back into wb_clk.

- DMA sideband (optional)
- Per-endpoint dma_req_o/dma_ack_i handshakes: exactly one ack pulse per 32-bit word; bursting/hold is allowed. If unused, tie dma_ack_i low and ignore dma_req_o, or supply SoC DMA that honors this contract.

- Clocks and reset
- Two domains: phy_clk (UTMI/protocol/memory) and wb_clk (Wishbone/interrupts). Safest integration runs wb_clk=phy_clk.
- Resets are synchronous unless USBF_ASYNC_RESET is defined. Avoid issuing WB transfers during reset and for a few wb_clk cycles after deassertion.
- Do not introduce unsynchronized fan-in across domains; any cross-domain paths must be synchronized at the boundary modules.

- Software/firmware responsibilities
- Program the USB function address, per-endpoint CSR, and buffer descriptors; manage ring mode and descriptor NA semantics (0xFFFF_FFFF or all-ones address).
- Implement USB Chapter 9 control handling (descriptors, enumeration, address setting timing). Some EP0 edge behaviors are policy/firmware-driven.
- Service interrupts via inta_o/intb_o by reading/clearing sources; treat them as level signals in wb_clk domain.
- Respect OUT size policies (sml_ok/lrg_ok) and IN/OUT buffer provisioning to avoid persistent NAK/NYET.

- Addressing/size conventions
- WB accesses are word-aligned; no byte strobes anywhere on the memory path.
- IDMA ring mode wraps to address 0; use buf_size in bytes (prefer multiples of 4). If a non-zero base is required, add an address-remap shim.

- Build-time configuration dependencies
- Macros/parameters must be consistent system-wide: USBF_UFC_HADR (RF/MEM split bit), SSRAM_HADR (memory address width), USBF_HAVE_EPx (enabled endpoints contiguous from EP0), USBF_ASYNC_RESET (reset flavor), USBF_TEST_IMPL (addressing/test build). If phy_clk differs from standard UTMI rates, audit all protocol timers.

-- Supported Modes and Speeds --

- Supported link speeds: USB 2.0 Full-Speed (12 Mb/s) and High-Speed (480 Mb/s); Low-Speed (1.5 Mb/s) is not supported.
- Speed selection/negotiation: automatic HS chirp after each bus reset. On detecting the required alternating K/J chirps, the core enters HS; otherwise it remains in FS. Speed follows the attached UTMI/UTMI+ PHY capability; there is no software override to force FS. The negotiated speed is reported via main_csr.mode_hs.
- UTMI PHY interface control: XcvSelect (1=FS, 0=HS), TermSel (enable FS termination), OpMode[1:0] (00=Normal, 10=special during reset/resume/chirp), and SuspendM (PHY suspend control). Idles: FS=J, HS=SE0. An FS-only PHY results in FS-only operation.
- HS-specific features when mode_hs=1: accepts/ACKs PING and returns NYET on OUT when buffers are not ready; microframe (125 µs) awareness with a timing snapshot exposed in frm_nat.
- LS-related PIDs (e.g., PRE) may be decoded for classification, but LS operation is not implemented.
- Link power states (independent of speed): attach/detach with debounce, bus reset detection, suspend on long idle, resume by host or device-initiated remote wake (with required USB timing enforced). Status signals (e.g., attached, suspend, suspend_clr) are provided.
- Exclusions: USB 2.0 High-Speed electrical Test Modes (Test_J, Test_K, Test_SE0_NAK, Test_Packet) are not implemented.

-- Use Cases and System Context --

What it is and where it sits
- USB 2.0 device core between a UTMI/UTMI+ PHY and an SoC. Control/status via a Wishbone slave; payloads buffered in an external 32■bit synchronous SRAM shared with firmware. Two clocks: phy_clk (protocol/memory) and wb_clk (bus); best when aligned.

Primary use cases
- General■purpose USB device: EP0 plus up to 15 data endpoints (build■time selectable) for CDC/ACM, HID, vendor■specific control, bulk and interrupt transfers.
- High■throughput bulk streaming: HS/FS IN/OUT using the external SRAM as a ring/pool; platform DMA reacts to per■EP dma_req and acks via dma_ack to sustain throughput.
- Control■plane only devices: EP0 enumeration, configuration, firmware update; optional interrupt endpoints for status/telemetry.
- Isochronous/low■latency streams (with caveats): Time■aligned ISO with microframe scheduling; system must provision SRAM bandwidth and avoid starving the protocol path.
- Prototyping/lab tools: Clear register map, frame time access (frm_nat), UTMI vendor sideband, rich interrupts for bring■up/diagnostics.

System context and data/control flow
- External components: UTMI PHY (FS/HS), single■port 32■bit synchronous SRAM, Wishbone interconnect/CPU, optional WB■master DMA, interrupt controller.
- On■wire/link: usbf_utmi_if + usbf_utmi_ls manage attach debounce, bus reset, suspend/resume, HS chirp/negotiation; drive XcvSelect/TermSel/OpMode/SuspendM; support remote wake via resume_req_i.
- Protocol: usbf_pd/usbf_pa decode/assemble tokens/data; usbf_crc5/16 ensure header/payload integrity; HS PING/NYET supported.
- Memory path: usbf_idma bridges 8■bit USB streams to 32■bit words, handling alignment, partial words, ZLPs, and ring wrapping; usbf_mem_arb gives hard priority to the protocol (M) over Wishbone (W) on the single SRAM port.
- Bus/registers: usbf_wb decodes RF vs memory regions, sequences one transaction at a time, and returns a single■cycle ACK in wb_clk. usbf_rf exposes global regs and per■endpoint windows (CSR/INT/BUF0/BUF1), aggregates interrupts (inta_o/intb_o), and exports per■EP dma_req.
- Endpoints: usbf_pe enforces endpoint policy (CTRL/BULK/INT/ISO), manages handshakes (ACK/NAK/STALL/NYET), data PID, timeouts, and descriptor write■back; integrates with usbf_idma for payload moves. usbf_ep_rf holds per■EP CSRs/descriptors, status/interrupts, and DMA flow counters; dummy instances keep unused EP slots inert but addressable.

Clocks, resets, and CDC
- phy_clk domain: UTMI/link, protocol, IDMA, memory arbiter. wb_clk domain: Wishbone front■end and RF bus view. Synchronous resets; CDC bridges for WB request/ACK and per■EP DMA handshakes. Prefer wb_clk == phy_clk.

Interrupts and DMA
- Two aggregated IRQs (inta_o/intb_o) covering RF■level events (attach/reset/suspend, PID/CRC errors) and per■EP events (buffer■done, CRC16, timeout, seqerr, unexpected PID). Per■EP dma_req signals IN prefill/OUT drain needs; platform returns dma_ack.

Performance/limitations
- HS throughput depends on SRAM bandwidth and minimal WB contention; M■priority arbitration can starve W. Wishbone MEM path is one■op, non■burst; best used in bursts between USB activity or via a system DMA. EP0 Chapter 9 strictness and HS Test Modes are not fully covered; extend in firmware/RTL if certification is required.

Software responsibilities
- Provide descriptors, program endpoint CSRs and buffer descriptors, service interrupts, manage ring buffers (if dma_en), respect remote■wake timing rules, and schedule memory window/DMA traffic around USB bursts.

-- Limitations and Out-of-Scope --

- USB specification and features
- Partial USB 2.0 device compliance: EP0 control-pipe requirements not guaranteed (unconditional SETUP acceptance/override of halt, exact 8-byte SETUP enforcement, STALL/Status semantics, DATA0 re-init, address reset to 0 on bus reset).
- USB 2.0 High-Speed electrical test modes (Test_J/K/SE0_NAK/Test_Packet) not implemented.
- BOS/LPM (USB 2.1), USB 3.x, OTG/HNP/SRP, hubs/TT, and SPLIT transactions are out of scope.
- High-bandwidth ISO/INT scheduling and bInterval enforcement not implemented; isochronous corner cases left to firmware.

- PHY/link assumptions
- UTMI/UTMI+ only; no ULPI bridge/wrapper. Vendor-specific sideband/test/calibration pins are out of scope.
- Fixed signal mappings/polarities (e.g., XcvSelect 1=FS, TermSel for FS only, OpMode uses 00/10) may not fit all PHYs; SuspendM polarity may be mismatched.
- Line-state handling is simplified (two-sample qualifiers, HS idle treated as SE0); rx_active/tx_ready not used in LS logic.
- HS chirp timing derived from fixed counters; thresholds for 1.0/1.2 ms are effectively the same in RTL; no runtime timing reconfiguration.
- Remote-wakeup timing exists but policy correctness (e.g., minimum suspend time) is left to system/firmware.

- Memory, DMA, and buffering
- Requires a 32-bit single-port SRAM-like interface: writes assume same-cycle ack; reads may be variable-latency. No AXI/AHB/Wishbone master, tags, bursts, or multiple outstanding requests.
- No byte enables; little-endian only. Ring wrap anchored at address 0; buf_size must be 4-byte aligned.
- Limited buffering: TX prefetch is 2 words; RX cannot backpressure. Underflow/overflow risks if memory latency exceeds assumptions.
- No scatter-gather or descriptor chaining; ping-pong only. With dma_en, BUF0 only; BUF1 excluded from DMA capacity accounting.
- Counters/size limits cap per-transfer to ~16 KB; no hardware bounds checking. Abort leaves partial state; no automatic recovery.
- No coherency or isolation with software; concurrent SW/engine memory access can corrupt data. Fixed-priority arbiter can starve software accesses.

- Wishbone and arbitration
- WB slave is classic single-beat: no bursts, pipelining/STALL, TAGs, byte-selects, or ERR/RTY/timeout signaling. One outstanding transfer only.
- Build-specific RF/MEM region select bit; not software-discoverable.
- CDC hazards when wb_clk != phy_clk: requests sampled directly into phy_clk; read data not safely synchronized.
- Memory arbiter gives strict priority to protocol/DMA (M) over Wishbone/software (W); sram_re tied high; no fairness/QoS; W-side capped to one transfer every two phy_clk cycles when served; wclk unused (no CDC).

- Protocol engine and endpoints
- Simplified PID/data toggle policy; no automatic DATA0 re-init on halt clear or after reset. Zero-length IN at multiple-of-MPS is not auto-generated.
- HS PING/NYET handling is coarse; precise readiness prediction not implemented. Unsupported PIDs (e.g., SPLIT/reserved) flagged as sequence errors.
- Timeouts are fixed-cycle (implementation-clock dependent), not calibrated to absolute USB timing. Errors/timeouts abort to IDLE; no retry logic.
- Endpoints are compile-time fixed (contiguous EP0..EP3 in this build); one direction per EP instance; no dynamic add/remove.

- Endpoint register file
- MPS and counters are word-aligned (non-4-byte MPS rounded down). DMA capacity uses BUF0-origin only; BUF1 not included.
- Interrupts are clear-on-read (clear-all); enable/index quirk present; inta/intb generated across domains without formal CDC.
- No protection against concurrent CSR/BUF updates during DMA; event strobes gated by endpoint match only.

- CRC blocks
- usbf_crc5/usbf_crc16 are fixed, combinational (no clock/pipelining/parameterization). Assume LSB-first data and specific seeds.
- CRC16 residue check requires PHY to deliver CRC bytes; no masking of CRC errors on PHY errors; timing closure on XOR depth left to integrator.

- Reset, power, safety
- Synchronous reset by default; optional async reset unvalidated. VBUS sourcing/switching, pull-up/down management, electrical compliance/ESD are out of scope.
- No ECC/parity, access control, range checks, or security features beyond USB CRCs; no BIST or diagnostics.

- Interrupts/diagnostics
- Some interrupt sources are sticky only at RF; per-EP pulses require timely service or events may be missed. No performance counters, traffic monitors, or certification tooling.

- Dummy endpoint
- ep_rf_dummy is a non-functional stub (RAZ/WI, no events/DMA); cannot be used for EP0; present only to tie off unused EP slots.

# ARCHITECTURE

-- Top-Level Block Diagram and Composition --

Overview
- usbf_top is a USB 2.0 device function core built around a UTMI PHY front-end and a shared external 32-bit synchronous SRAM. It exposes a Wishbone B3 classic slave for software control.

Major blocks (top-level composition)

- usbf_utmi_if (phy_clk): Terminates UTMI RX/TX byte streams and hosts the line-state/speed controller usbf_utmi_ls. Bridges PHY pins to the core stream and UTMI control pins.
- usbf_pl (phy_clk): Protocol layer and data path containing usbf_pd (RX decoder + CRCs), usbf_pe (transaction policy/handshakes/descriptor updates), usbf_idma (byte↔word DMA to memory), and usbf_pa (TX packet assembly + CRC16).
- usbf_mem_arb (phy_clk): Single-port SRAM arbiter/multiplexer with fixed priority for the protocol/DMA master (M) over the Wishbone/software master (W).
- usbf_rf (clk_i for bus view, phy_clk for core view): Register file and aggregation. Exposes global regs and 16 endpoint windows, instantiates per-EP blocks (usbf_ep_rf; unused slots can be usbf_ep_rf_dummy), aggregates interrupts, and exports per-EP DMA vectors. Provides a live matched-EP view to the protocol layer.
- usbf_wb (wb_clk=clk_i, control FSM in phy_clk): Wishbone front-end. Decodes RF vs memory regions, sequences single-beat accesses, and returns a clean wb_ack_o in wb_clk.

External interfaces
- UTMI PHY (via utmi_if/utmi_ls): DataIn/Out[7:0], RxValid/Active/Error, TxValid/Ready, XcvSelect, TermSel, SuspendM, OpMode[1:0], LineState[1:0], usb_vbus; vendor sideband VStatus[7:0], VControl[3:0], Load.
- Memory (shared SSRAM): sram_adr_o[SSRAM_HADR:0], sram_data_i[31:0], sram_data_o[31:0], sram_we_o, sram_re_o(=1).
- Wishbone B3 slave: wb_clk=clk_i, wb_addr_i[17:0], wb_data_i/o[31:0], wb_we_i, wb_stb_i, wb_cyc_i, wb_ack_o.
- IRQs: inta_o, intb_o. DMA: dma_req_o[15:0], dma_ack_i[15:0]. Power/link: phy_clk_pad_i, rst_i (drives phy_rst_pad_o), susp_o, resume_req_i.

Clocks and reset
- Domains: phy_clk (utmi_if/utmi_ls, usbf_pl, usbf_mem_arb, rf core view); clk_i (usbf_wb, rf bus view, susp_o register).
- Reset: rst_i is synchronous by default; fans out internally and drives phy_rst_pad_o.
- CDC highlights: utmi_if contains line-state FSM; rf bridges clk_i↔phy_clk for controls, status, sticky events, and one-shots (rf_resume_req, utmi_vend_wr); wb_ack_d (phy_clk) edge-synchronized to wb_ack_o (clk_i).

Data plane (draw as primary buses)
- RX: UTMI PHY -> usbf_utmi_if -> usbf_pl.usbf_pd (PID/CRC5/CRC16, align) -> usbf_pl.usbf_idma -> usbf_mem_arb (M) -> external SSRAM.
- TX: external SSRAM -> usbf_mem_arb (M) -> usbf_pl.usbf_idma -> usbf_pl.usbf_pa (DATA PID, append CRC16) -> usbf_utmi_if -> UTMI PHY.
- Software memory path: Wishbone -> usbf_wb (memory window) -> usbf_mem_arb (W) -> external SSRAM (only when M idle).

Control/status plane
- RF window (via usbf_wb): global regs (device address, main_csr, RF masks/sources, frame counter, UTMI vendor status/control, remote-wake request) and 16 per-EP windows (CSR/INT/BUF0/BUF1).
- usbf_pl <-> usbf_rf (phy_clk): ep_sel and live CSR/BDs into pl; descriptor write-backs (idin with buf0_set/buf1_set/uc_bsel_set/uc_dpd_set/buf0_rl) and event pulses (int_*_set) back to rf; frm_nat into rf.
- Link state/speed from usbf_utmi_ls: mode_hs, usb_reset, usb_suspend, usb_attached, suspend_clr. Remote wake: rf_resume_req -> top -> utmi_ls; utmi_if shapes drive_k during resume.
- UTMI vendor sideband: VStatus latched in rf for SW; VControl+Load written by SW and pulsed across to phy_clk and out to PHY.

Arbitration and memory behavior (usbf_mem_arb)
- Fixed priority master selection: M (protocol/DMA) > W (Wishbone/software).
- Handshakes: mack=mreq (combinational); wack pulses one phy_clk when served and paces continuous W traffic to every other cycle; preempted immediately by M.
- Read data broadcast to both masters; sram_re tied high; masters must sample din only when their ack is asserted.

Wishbone front-end semantics (usbf_wb)
- Address decode uses wb_addr_i[17]: 0=RF, 1=Memory.
- RF: single-cycle rf_re/rf_we pulse (phy_clk) and immediate completion.
- Memory: ma_req (+ma_we for writes) held until ma_ack from mem_arb; one outstanding transfer; no bursts or pipelining.
- wb_ack_o: generated by edge-syncing wb_ack_d (phy_clk) back to wb_clk.

Endpoint model and aggregation (usbf_rf/usbf_ep_rf)
- Up to 16 endpoints; build enables EP0..EP3 (contiguous). Unused slots may be usbf_ep_rf_dummy (tie-offs).
- Per-EP: CSR (dir/type/enable/stall/max packet plus user fields), INT (sticky, read-to-clear with masks), BUF0/BUF1 descriptors; buf0_orig mirrors last SW write.
- Live matched-EP view (phy_clk): priority mux exports csr/buf0/buf1 and convenience flags (dma_in_buf_sz1, dma_out_buf_avail) for the currently selected ep_sel; match is OR of epN_match.
- Interrupts: OR of per-EP inta/intb plus masked RF sticky events -> inta_o/intb_o; int_srca mirrors per-EP live (A|B), int_srcb are RF-level sticky events (read clears).
- DMA: dma_req_o[15:0] vector exported; dma_ack_i fanned back one-to-one to EPs.

UTMI link-state/control details (usbf_utmi_ls via utmi_if)
- Drives: XcvSelect (HS/FS select), TermSel (FS termination), SuspendM (not suspended), OpMode[1:0].
- Reports: mode_hs, usb_reset, usb_suspend, usb_attached, suspend_clr. Asserts drive_k during resume and HS chirp; utmi_if shapes DataOut/TxValid accordingly.

Typical sequences
- OUT/SETUP: Token received and validated -> usbf_pe qualifies match -> usbf_idma writes payload to memory -> usbf_pe selects ACK/NAK/NYET -> descriptor write-backs and interrupts via rf.
- IN: usbf_pe selects buffer and DATA PID -> usbf_idma prefetches -> usbf_pa emits DATA+CRC16 -> ACK/timeout handling -> descriptor write-backs and interrupts via rf.

Build and sizing notes
- SSRAM_HADR sets memory word-address width (typical 14). Effective buffer space is word-addressed. Wishbone RF/memory region select at wb_addr_i[17].

Top-level glue
- rst_i drives internal reset and phy_rst_pad_o. susp_o is usb_suspend registered into clk_i. LineState and UTMI VStatus sampled in phy_clk and forwarded to rf. resume_req_r latched from SW and cleared by suspend_clr to gate remote wake.

Operational cautions
- W-side memory accesses can be starved under heavy protocol traffic; fairness not guaranteed. Best used when wb_clk == phy_clk or phase-related due to simple CDC on request/ack.

-- Data Path Overview --

The core bridges an 8-bit UTMI PHY to a 32-bit, single-port external SSRAM. A protocol engine (PE) orchestrates packet decoding/assembly and DMA, while software configures endpoints and buffers via a Wishbone slave. When the USB side is idle, software can access the same SSRAM through an arbiter.

Receive (Host→Device): UTMI bytes enter usbf_utmi_if (registered on phy_clk) and are observed by the RX decoder (usbf_pd), which classifies PIDs and validates CRC5/CRC16. For accepted OUT/SETUP data, usbf_idma packs bytes into 32-bit words with alignment-aware read-modify-write on the first/last partial words, then writes to SSRAM via the M-side of usbf_mem_arb. On completion, PE updates endpoint descriptors and raises per-EP event pulses via the register file (usbf_rf).

Transmit (Device→Host): PE selects the active endpoint buffer based on CSR/descriptors from usbf_rf. usbf_idma prefetches 32-bit words from SSRAM and streams bytes to the TX assembler (usbf_pa), which emits DATA PID, payload, and CRC16. usbf_utmi_if stages DataOut/TxValid with TxReady-driven holds; tx_first ensures SOP preload and tx_valid_last guarantees the last beat handshake. After TX, PE detects ACK (non-iso) or times out, then writes back descriptors and events.

Memory and arbitration: usbf_mem_arb multiplexes the single SSRAM port between the USB M-side (protocol/DMA on phy_clk, fixed high priority) and the Wishbone W-side (software). Read data is broadcast to both masters. M-side sees mack=mreq (no added wait-states); W-side acks are throttled (one ack every other phy_clk when served) and are immediately preempted by M traffic.

Wishbone data path: usbf_wb decodes address space into RF vs Memory. RF accesses generate direct rf_re/rf_we pulses to usbf_rf and ack immediately. Memory accesses drive the arbiter W-side and ack when served. A small CDC returns a single-cycle wb_ack_o in wb_clk and registers read data.

Control/status coupling: usbf_rf maintains per-EP CSR and BUF0/BUF1 descriptors, exposes a live, match-gated view of the currently addressed endpoint to PE (including convenience flags like dma_in_buf_sz1 and dma_out_buf_avail), accepts descriptor write-backs/events from PE, aggregates interrupts, and latches/forwards UTMI vendor status/control.

PHY sideband and gating: usbf_utmi_ls controls XcvSelect, TermSel, OpMode, and SuspendM to sequence attach, reset, suspend/resume, and HS negotiation. Special drive_k cycles (resume/HS chirp) are integrated with the TX pipe to suppress normal byte traffic during signaling; normal RX/TX resumes when OpMode returns to Normal.

Widths, alignment, and flow: UTMI datapath is 8-bit; SSRAM datapath is 32-bit with word-oriented addressing. usbf_idma bridges byte↔word packing, maintains byte indices, handles partial-word flush on RX, and supports two-word prefetch on TX. Optional ring/wrap is supported via dma_en and buf_size. Zero-length INs send PID+CRC without memory reads; handshake-only packets bypass memory entirely.

Clocks and CDC: UTMI, PE, IDMA, and mem_arb run in phy_clk. Wishbone/RF bus view runs in wb_clk/clk_i. CDC is localized in usbf_utmi_if (PHY staging), usbf_wb (req/ack), usbf_mem_arb (W ack flop), and usbf_rf/usbf_ep_rf (DMA req/ack and interrupt masking).

Throughput/priority: The USB M-side has absolute priority to meet timing at FS/HS rates. Software memory transactions are opportunistic and throttled; sustained M traffic can starve W-side access. Handshake-only paths avoid memory and have minimal latency.

-- Control and Status Path --

Overview
- The design splits control and status across two planes: a Wishbone software plane (clk_i) and a PHY/protocol plane (phy_clk). Software programs control via a compact register-file (RF) with per-endpoint windows; status is collected from the UTMI link, protocol engine, endpoints, and memory/DMA, then exposed as readable registers and interrupts.

Addressing and regions
- Wishbone decode: wb_addr_i[17]==0 selects the RF region; wb_addr_i[17]==1 selects external memory. The Wishbone front end (usbf_wb) serializes one request at a time.
- RF global registers at 0x00..0x05:
• 0x00 main_csr: R returns line_state, usb_attached, mode_hs, suspend; W bit5 asserts rf_resume_req (remote wake one-shot).
• 0x01 funct_adr: R/W 7-bit device address for token matching.
• 0x02 interrupt masks: inta_msk[8:0], intb_msk[8:0] for RF-level stickies and aggregation.
• 0x03 interrupt sources: R returns {int_srcb[8:0] stickies, int_srca live per-EP OR}; read clears only int_srcb.
• 0x04 frm_nat: R snapshot of microframe/frame/time from protocol.
• 0x05 UTMI vendor: R VStatus_r[7:0]; W VControl[3:0] with a write strobe into the PHY.
- Per-endpoint windows at base = 0x10 + 4*EP: offsets 0=CSR, 1=INT (read-to-clear), 2=BUF0, 3=BUF1. CSR holds direction, endpoint number, DMA enable, max packet size, and uc_* bits. BUF descriptors carry address/size and SW reference size.

Status sources and propagation
- UTMI/link layer (usbf_utmi_if/ls, phy_clk): generates mode_hs, usb_reset, usb_suspend, usb_attached, and a suspend_clr one-shot. LineState and VStatus are latched in phy_clk. These feed RF global registers and RF-level interrupt stickies.
- Protocol engine (usbf_pl, usbf_pd, usbf_pa, phy_clk): decodes PIDs/tokens, performs CRC checks, sequences handshakes, and drives per-EP event strobes: int_buf0_set, int_buf1_set, int_crc16_set, int_to_set, int_seqerr_set, int_upid_set, plus descriptor update strobes (buf*_set, uc_*_set, buf0_rl). A global nse_err is raised on addressed-token mismatches.
- Per-endpoint RF (usbf_ep_rf): captures per-EP event strobes into sticky INT bits (read-to-clear), updates descriptors on buffer completion, toggles buffer select, and generates per-EP DMA request/ack and convenience readiness flags (dma_in_buf_sz1, dma_out_buf_avail).
- DMA/Memory (usbf_idma, usbf_mem_arb, phy_clk): idma moves RX/TX bytes to/from memory per descriptors; completion (idma_done) and actual RX size (sizu_c) flow to usbf_pe for descriptor write-back and per-EP interrupts. mem_arb grants core M side priority over the Wishbone W side; read data is broadcast; acks gate sampling.

Interrupt model
- RF-level sticky sources int_srcb[8:0]: [8] usb_reset, [7] rx_err, [6] detach, [5] attach, [4] suspend_end, [3] suspend_start, [2] nse_err, [1] pid_cs_err, [0] crc5_err. Cleared by reading 0x03.
- Per-EP sticky INT bits (read-to-clear via EP INT): timeout, CRC16 error, unexpected PID, buffer-done (buf0/1), sequence error, out_to_small. Per-EP A/B enables live in the EP INT register.
- Aggregation: inta_o/intb_o are ORs of all enabled per-EP interrupts plus enabled RF-level stickies, masked by 0x02. int_srca in 0x03 reflects live per-EP ORs; int_srcb is sticky.

Control paths
- Software control: writes to RF program function address, endpoint CSRs, buffer descriptors, interrupt masks, and vendor control. main_csr bit5 triggers remote wake; writes to EP INT update per-EP A/B enable masks.
- Protocol control: live-view mux in usbf_rf presents the currently matched endpoint's csr/buf0/buf1 and DMA readiness directly to usbf_pl without bus reads. usbf_pe selects transactions

(ACK/NACK/STALL/NYET, IN/OUT/SETUP), starts idma, and computes next descriptors and uc_* state for write-back.
- UTMI control: usbf_utmi_ls drives XcvSelect, TermSel, OpMode, SuspendM and K-drive during reset/resume/chirp based on LineState, timers, and resume_req.

Clock-domain crossings and handshakes
- Wishbone to phy: wb_req is sampled in phy_clk by usbf_wb; wb_ack is generated in phy_clk then turned into a single-cycle wb_clk pulse via a small edge detector. For robust operation, align wb_clk and phy_clk.
- RF one-shots/stickies: RF captures and synchronizes status from phy_clk for software readback; read-to-clear semantics are implemented in RF and per-EP blocks.
- Remote wake bridge: SW writes rf_resume_req in clk_i; usbf_top latches resume_req_r in clk_i and clears it on suspend_clr; usbf_utmi_ls re-synchronizes it into phy_clk.
- Vendor sideband: writes to 0x05 generate a one-shot VControl_Load into phy_clk; VStatus is latched in phy_clk and read via RF.
- DMA req/ack: per-EP dma_req is generated in wclk and dma_ack returned there; a synchronized dma_ack_i into phy_clk advances counters and qualifies set events.

Reset, suspend/resume, and top-level status
- rst_i feeds both domains; phy_rst_pad_o follows rst_i. usb_suspend is mirrored as susp_o in clk_i. resume_req_r is bounded by suspend_clr to ensure a single remote-wake attempt per suspend.

Software guidance
- Keep wb_clk and phy_clk aligned where possible. Program funct_adr after SET_ADDRESS. Use EP CSR and BUF descriptors to manage buffer ownership; with DMA enabled, ownership returns via buf0_rl instead of descriptor rewrite. Clear RF stickies by reading 0x03; clear per-EP stickies by reading EP INT. Remote wake should be requested only while suspended.

-- Clock and Reset Domains --

- Clocks and domains
- phy_clk_pad_i (PHY/core domain): Hosts usbf_utmi_if (UTMI/link-state), entire protocol layer usbf_pl (usbf_pd, usbf_pa, usbf_pe, usbf_idma), usbf_mem_arb, the core-side of the register file (usbf_rf.clk), and the target/FSM side of usbf_wb. External SRAM pins are driven here; UTMI status is captured here.
- clk_i (Wishbone/bus domain): Hosts the Wishbone read-data/ACK side of usbf_wb (wb_clk), the bus-side of the register file (usbf_rf.wclk) for software-visible registers/interrupts/DMA request control, and top-level suspend output (susp_o).

- Reset scheme
- Single active-low reset rst_i is fanned out as rst to all blocks and mirrored to the PHY as phy_rst_pad_o.
- Resets are synchronous to the local clock in most blocks (used as if (!rst) in posedge processes). Optional asynchronous clear via USBF_ASYNC_RESET exists in some modules (e.g., usbf_utmi_if, usbf_mem_arb, usbf_wb).
- Some pipelines are intentionally unreset (e.g., usbf_wb ACK pipeline) and initialize functionally.

- CDC strategy and known crossings
- Recommended: run clk_i == phy_clk_pad_i (same frequency/phase) to honor assumed-synchronous paths.
- Wishbone req (clk_i→phy_clk, usbf_wb): wb_stb_i/wb_cyc_i are sampled in phy_clk without a 2■FF synchronizer; add a synchronizer/handshake if clocks differ.
- Wishbone ack (phy_clk→clk_i, usbf_wb): hardened via a 3■FF pipeline and edge detector

(wb_ack_d→wb_ack_o).
- Register file (usbf_rf) split-plane: explicit handshakes for one-shots (wclk→clk: rf_resume_req, utmi_vend_wr; clk→wclk: link-state edges). Some error strobes are only single■flop sampled into wclk; strengthen if independent clocks. Bus accesses affecting clk-domain state assume synchronous clocks; add CDC if not.
- Top-level status: usb_suspend and suspend_clr (phy_clk→clk_i) are single■flop sampled; resume_req (clk_i→phy_clk) is consumed by usbf_utmi_if, with a 2■FF sync inside utmi/link-state.
- Memory hub: usbf_mem_arb and both masters (usbf_idma M side and usbf_wb W side) are in phy_clk; wclk is unused in mem_arb.

- Module notes
- usbf_utmi_ls: single phy_clk domain; resume_req is 2■FF synchronized; usb_vbus is treated as synchronous—add sync if asynchronous in integration.
- usbf_crc5/usbf_crc16: purely combinational; reside in the consumer's domain (phy_clk here). Register/synchronize if crossing domains.

- Integration guidance
- Prefer clk_i equal to phy_clk_pad_i. If not, add CDC on: WB request/data/address toward phy_clk, RF bus accesses and status reflection between wclk and clk, and top-level suspend/resume paths. Ensure rst_i deasserts synchronously in each domain or add domain-local reset synchronizers.

-- Clock-Domain Crossing Strategy --

Clock domains
- phy_clk: UTMI PHY interface, link-state logic, protocol layer (PD/PA/PE/IDMA), external SRAM arbiter, and the core-side of RF.
- wb_clk (aka wclk): Wishbone slave front-end, software-visible RF (bus view), selected top-level status/controls.

CDC primitives and where they are used
- 2-FF level synchronizers (clk→wclk): For slow/long-lived core status (e.g., usb_attached, usb_suspend) in usbf_rf; edges are derived in wclk for SW events.
- Sticky level capture (clk→wclk): Error/event flags (usb_reset, rx_err, nse_err, pid_cs_err, crc5_err, crc16_err) are sampled in wclk and made sticky. Safe only if source holds multi-cycle or clocks are related; otherwise use stretching/toggle.
- Write-to-pulse one-shots (wclk→clk): usbf_rf converts bus writes into single-cycle pulses in phy_clk (e.g., rf_resume_req for remote wake, UTMI vendor writes). Auto-clear on wclk side prevents re-issuance.
- Edge-synchronized ACK (phy_clk→wb_clk): usbf_wb returns a single wb_clk-cycle wb_ack_o via a 3-FF edge detector from wb_ack_d generated in phy_clk. Guarantees one ACK per transaction.
- Local 2-FF sync on external control: usbf_utmi_ls re-synchronizes resume_req into phy_clk even though rf already one-shots it; adds safety margin.

Key crossings by subsystem
- Wishbone bridge (usbf_wb):
- Requests (wb_stb_i & wb_cyc_i) and address/data/WE are directly sampled in phy_clk (no multi-FF); assumes wb_clk≈phy_clk. If asynchronous, add a request/ack or toggle handshake that captures address/data/WE coherently in phy_clk.
- Read data (rf_din/ma_din from phy_clk) is registered in wb_clk; validity is aligned to the synchronized wb_ack_o pulse. Producers must hold data stable through the sampling window.
- Register-file bridge (usbf_rf and per-endpoint usbf_ep_rf):
- clk→wclk: level sync + edge detect for attach/suspend; sticky capture for error/events; inta/intb are computed in wclk from sticky bits.

- wclk→clk: bus writes (adr/we/din) decoded in clk to update CSR/BUF and clear interrupts; use one-shots for side-effect strobes (e.g., resume). If clocks are async, synchronize write strobes/data or move the read mux and clear logic to wclk with a handshake.
- DMA handshake: wclk posts dma_req and holds it; dma_ack is latched in wclk and 2-FF synchronized into clk to form dma_ack_i. Hold logic sustains bursts. Keep all multi-bit counters/state in clk; if wclk consumes them, add 2-FF or relocate logic to clk and export only 1-bit results.
- UTMI/link-state (usbf_utmi_if/utmi_ls): Entirely phy_clk. LineState is registered and deglitched; outputs are registered. If usb_vbus is asynchronous, add sync/debounce at its entry.
- Memory path (usbf_mem_arb): Entirely phy_clk. W-port signals reaching mem_arb come from usbf_wb's phy_clk FSM; the only CDC is the ACK edge back to wb_clk handled in usbf_wb.

Top-level miscellaneous crossings
- suspend_clr (phy one-shot) sampled in wb_clk to clear resume_req_r; resume_req_r is held as a level and observed in phy_clk. usb_suspend is reported to SW as susp_o via a wb_clk register. These rely on long hold times; add 2-FF if clocks are asynchronous.

Guidance for asynchronous wb_clk and phy_clk
- Add synchronizers/handshakes at: Wishbone request and associated address/data/WE into phy_clk; RF write strobes/data; INT clear/read ordering; single-cycle event pulses (crc5/16_err, seqerr, timeouts) via pulse-stretch or pulse-to-toggle; top-level suspend_clr and resume_req; any multi-bit bus (use valid/ack or small CDC FIFOs) rather than raw sampling.
- Ensure read-data producers (rf_din/ma_din) hold data stable until one wb_clk after wb_ack_o; if not possible, add a read-data valid handshake or register data in wb_clk.

Reset policy
- Resets are treated synchronous within each domain; if an asynchronous global reset is used, synchronize its deassertion per clock domain or enable the project's async-reset option consistently.

Design intent
- Best practice is to run wb_clk == phy_clk (or tightly phase-locked) to satisfy direct samplings and minimize CDC risk. If clocks are truly asynchronous, apply the above hardening to guarantee metastability-safe operation.

-- Memory/Buffer Architecture and Arbitration --

Central packet store and capacity: A single shared external synchronous SRAM (32■bit wide, word■oriented) holds all endpoint payloads; there are no large internal FIFOs. Read■enable is tied high and the interface has no byte■enables. Address width is parameterized by SSRAM_HADR (test build: 14). Effective capacity is (1 << (SSRAM_HADR+1)) * 4 bytes. The DMA uses byte addresses for descriptors but converts to SRAM word address + byte offset; the Wishbone path passes system addresses into the memory window and should use word alignment.

Endpoint buffers and descriptors: Each endpoint exposes two software■visible descriptors (BUF0/BUF1) carrying base byte address and remaining size plus control/status in the EP CSR. The protocol engine consumes a live, muxed view of the currently matched endpoint's CSR/BUF0/BUF1 and performs in■band descriptor updates via a write■back bus (idin) with strobes: buf0_set/buf1_set (commit address/size), uc_bsel_set (active buffer select), uc_dpd_set (PID base/toggle), and buf0_rl (release in DMA ring mode).

DMA bridge (USB bytes ↔ 32■bit words): usbf_idma issues all core■side memory traffic. RX packs incoming bytes into words, handles unaligned starts with a seed read (read–modify–write), and flushes partial last words on short/odd packets. TX maintains a two■word ping■pong prefetch to sustain continuous streaming and hide SRAM latency. Descriptor■driven addressing supports wrap/ring and size accounting. Convenience readiness flags exported to the protocol engine: dma_in_buf_sz1 (enough prefetched IN data) and dma_out_buf_avail (enough OUT space).

Arbitration (usbf_mem_arb): One SRAM port is multiplexed between two masters in phy_clk with fixed priority, no fairness/round■robin. M (USB core/DMA/protocol) signals: madr/mdin/mdout/mwe/mreq/mack. W (Wishbone/software via usbf_wb, served in phy_clk) signals: wadr/wdin/wdout/wwe/wreq/wack. Priority: M always wins. Selection: wsel = (wreq | wack) & !mreq; this holds the W grant through its acknowledge. Handshakes: M side mack = mreq (no arbiter■inserted waits; upstream DMA hides device latency). W side wack is a clean one■cycle pulse produced every other phy_clk while granted and is suppressed immediately if M asserts mreq (immediate preemption). Data/control muxing: sram_adr/sram_dout/sram_we are selected by wsel; read data is broadcast to both masters, each sampling only on its own ack.

Wishbone path and CDC (usbf_wb): Address decode separates RF vs memory (test build: wb_addr_i[17] = 1 selects memory). Exactly one outstanding transfer: memory ops assert ma_req (+ma_we) and wait for ma_ack from the arbiter; RF ops pulse rf_re/rf_we and ack in the fast path. wb_ack_o is re■timed back into wb_clk via a small synchronizer. Best integration when wb_clk == phy_clk or timing is otherwise closed across domains.

Performance and policy: Strict M priority guarantees USB timing. Software memory accesses proceed only when M is idle and are paced to at most one accepted transfer every two phy_clk cycles when uncontested. Unaligned RX/TX is handled entirely inside the DMA; software should adhere to word alignment on the memory window. Schedule large software moves during idle/suspend intervals to avoid starving the M port.

-- Address Map Partitioning --

- Top-level split (USBF_UFC_HADR=17 in this build): wb_addr_i[17]=0 → Register File (RF); wb_addr_i[17]=1 → External SRAM buffer memory.
- Addressing rules: 32-bit word addressing only; wb_addr_i[1:0] ignored; no byte-lane (SEL) support. In RF, only adr[6:0] is decoded → RF page is mirrored across the entire wb_addr_i[17]=0 region (use the lowest mirror).

RF region (wb_addr_i[17]=0; byte offsets shown, multiples of 4)
- Global page (mirrored): adr[6:2]==0x00 and 0x01 map the same block; 0x02..0x03 read as 0.
- 0x00: main_csr (R). W with bit5=1 issues rf_resume_req (remote wake) one-shot.
- 0x01: function address (R/W; 7-bit).
- 0x02: interrupt masks A/B (R/W).
- 0x03: interrupt sources (R; read-to-clear RF stickies int_srcb; int_srca is live per-EP OR).
- 0x04: frm_nat timestamp (R).
- 0x05: UTMI vendor: VStatus (R); VControl + write strobe (W).
- 0x08..0x0F: reserved (reads 0).
- Per-endpoint windows (EPn base = 0x10 + 4*n, n=0..15):
- +0 CSR (R/W), +1 INT (R/W; read-to-clear EP-local stickies), +2 BUF0 (R/W descriptor), +3 BUF1 (R/W descriptor).
- Build example: EP0..EP3 present at 0x10..0x1F; others may be dummy (read 0, ignore writes) but remain decoded at their addresses.

External SRAM region (wb_addr_i[17]=1)
- Direct pass-through to synchronous SRAM via usbf_mem_arb; Wishbone word address is forwarded, read data returned on completion.
- Size set by SSRAM_HADR (example SSRAM_HADR=14 → effective accessible size =

2^(SSRAM_HADR+1) words × 4 bytes = 128 KiB).
- Accesses beyond physical depth alias/wrap per external SRAM decoding.
- Arbitration: usbf_mem_arb gives priority to the protocol engine (M) over Wishbone (W); this affects bandwidth/latency only, not addressing.

Notes
- UTMI/link state, CRC, PD, PE, and IDMA blocks introduce no additional bus-visible address space; all software-visible registers are in RF, and all data buffers reside in the external SRAM region.
- All addresses are 32-bit word aligned; byte addresses shown are multiples of 4.

-- DMA Architecture --

Two-tier DMA over shared 32-bit SRAM
- Tier 1 (internal, always present): usbf_pl/usbf_idma streams 8-bit UTMI payloads to/from a shared 32-bit word-addressed SSRAM to meet USB on-wire timing.
- Tier 2 (system-side, optional per endpoint): usbf_rf exposes per-endpoint dma_req_o/dma_ack_i so an external DMA engine can move complete buffers via a Wishbone-accessible memory window (usbf_wb) when the USB side is idle.

Internal packet DMA (Tier 1)
- RX (USB→SRAM): packs bytes into 32-bit words; handles unaligned starts with a seed read/modify/write; writes full words as they form; flushes a partial tail on packet end (rx_data_done).
- TX (SRAM→USB): two-word ping-pong prefetch sustains streaming; bytes are released in lockstep with UTMI TxReady (rd_next pacing); supports zero-length IN without memory fetch.
- Memory interface (single outstanding op): madr[SSRAM_HADR:0], mdout/mdin[31:0], mwe, mreq/mack; request/ack style with one op in flight.
- Addressing and wrap: byte-based internal addressing; word address = adr[SSRAM_HADR+2:2], lane = adr[2:0]; optional ring/wrap by programmed buffer size.
- Descriptor integration: usbf_pe selects BUF0/BUF1, provides base/size, and performs in-band descriptor write-backs/ownership updates after transfers.

Shared memory and arbitration
- usbf_mem_arb multiplexes the single SRAM port between M-side (protocol/DMA) and W-side (Wishbone/system).
- Policy: fixed priority; M-side always wins (mack=mreq). W-side is granted only when M-side idle; wack pulses at most every other phy_clk when granted.
- Data path: sram_re held high; read data broadcast to both masters; no SRAM byte-enables (RX relies on RMW for unaligned/tail writes).

System-side DMA (Tier 2) and Wishbone path
- Per-endpoint handshake: EP asserts dma_req_o when buffers need draining/filling (policy in usbf_ep_rf); external DMA performs WB reads/writes of the SRAM window; a pulse on dma_ack_i[ep] clears/advances the EP mini-FSM and buffer ownership.
- Convenience flow-control to protocol: dma_in_buf_sz1 (≥1 MPS of IN data), dma_out_buf_avail (≥1 MPS of OUT space).
- usbf_wb: decodes RF vs memory; issues one-beat W-side requests to the arbiter; ACK returned to wb_clk via a small synchronizer (clean single-cycle wb_ack_o). Best-case one WB beat every two phy_clk cycles when M-side idle.

Clocking and domains
- High-rate path (utmi_if/utmi_ls, usbf_pl/usbf_idma, usbf_mem_arb, usbf_pe) runs in phy_clk (no CDC on the M-side DMA path).
- Control/bus bridges: usbf_rf bridges clk_i↔phy_clk; usbf_wb bridges wb_clk↔phy_clk.

Constraints and guarantees
- On-wire timing is protected by M-side fixed priority plus TX prefetch and RX packing; W-side/system DMA may be delayed or starved under heavy USB load (no fairness).
- No SRAM byte-enables; software/system DMA should operate on word granularity; internal RX uses RMW for unaligned and tail writes.
- Capacity and addressing: external SRAM is 32-bit, word-addressed with width SSRAM_HADR; effective capacity ≈ (1 << (SSRAM_HADR+1)) × 4 bytes.
- Error/status gating: TOKEN CRC5 and DATA CRC16 checks (in usbf_pd/usbf_crc*) qualify DMA commit and inform descriptor/status write-backs without stalling the internal pack/stream pipeline.

-- Interrupt Architecture --

Overview
- Two top-level interrupt outputs: inta_o and intb_o (clk_i domain). They are driven by usbf_rf as the OR of: (a) all per-endpoint A/B interrupt lines and (b) RF-level (global) sticky events after independent A/B masking.

Sources
- Per-endpoint causes (generated in the PHY/protocol domain, latched in each EP's read-to-clear INT register, then routed to A/B by per-EP masks): buffer completion (int_buf0_set, int_buf1_set), CRC16 error on RX data (int_crc16_set), timeout (int_to_set), sequence error (ISO OUT only, int_seqerr_set), unexpected PID (int_upid_set). Additional endpoint status bits may exist in the EP INT register as defined by the EP submodule; only enabled causes contribute to A/B lines.
- RF-level (global) sticky causes (latched in clk_i, read-to-clear at 0x03): int_srcb[8:0] = {usb_reset[8], rx_err[7], deattach[6], attach[5], suspend_end[4], suspend_start[3], nse_err[2], pid_cs_err[1], crc5_err[0]}.

Aggregation and polarity
- inta_o = OR(all ep_inta) | OR(int_srcb & inta_msk).
- intb_o = OR(all ep_intb) | OR(int_srcb & intb_msk).
- Active-high level signaling; lines remain asserted while any contributing sticky source (EP or RF-level) is set.

Software-visible registers (Wishbone, clk_i)
- 0x02 Interrupt masks (R/W): {7'b0, intb_msk[8:0], 7'b0, inta_msk[8:0]} — independent masks to route each RF-level cause to A or B.
- 0x03 Interrupt sources (R, read-to-clear RF stickies): {3'b0, int_srcb[8:0], 4'b0, int_srca[15:0]}; int_srcb are sticky RF-level bits (cleared by this read). int_srca[15:0] is a live per-EP summary where bit N = epN_inta | epN_intb (not sticky, not mask-affected).
- Per-EP windows: base 0x10 + 4*N; EPx INT at offset +1 is read-to-clear and holds the endpoint's latched causes; per-EP A/B enable fields determine routing to ep_inta/ep_intb.

Clearing and masking rules
- RF-level stickies: cleared only by reading 0x03; masks at 0x02 affect routing to A/B but do not clear sources.
- Per-EP stickies: cleared only by reading the EPx INT register; EP-local A/B masks control routing to ep_inta/ep_intb.
- Top-level lines deassert once all contributing stickies (EP and RF-level) are cleared.

Clock domains and CDC
- Endpoint interrupt pulses originate in the PHY/protocol clock; usbf_ep_rf latches them into EP INT and

synchronizes ep_inta/ep_intb into clk_i for aggregation.
- RF-level sources are formed in clk_i from synchronized core/link status: usb_reset, attach/deattach and suspend edges are derived from usbf_utmi_ls levels/edges; rx_err from UTMI RxError; crc5_err, pid_cs_err, nse_err from protocol/token checks.

Driver flow
- On inta_o/intb_o: read 0x03 to view/clear RF-level stickies and to obtain int_srca (which endpoints need service); then read each asserted EPx INT to retrieve and clear per-EP causes; configure 0x02 and EP-local A/B masks as desired.

Notes
- DMA request/ack and memory arbitration are independent of interrupts and do not gate assertion/clearing.
- Remote wake/resume does not directly interrupt; suspend_end is observed when usb_suspend deasserts.

-- Power Management and Link State Control --

Ownership and scope
- The UTMI link-state controller (usbf_utmi_if/usbf_utmi_ls) owns all link and PHY power sequencing: attach/detach, bus reset detect/handling, suspend entry/exit, host resume, device remote wake, and HS/FS speed selection. It drives UTMI control pins and exports link-power status to software and the SoC. No other module gates clocks or powers down logic internally.

UTMI controls and line management
- Controls driven per link state: XcvSelect (0=HS, 1=FS), TermSel (1=FS termination on), OpMode (2'b00 normal, 2'b10 special for reset/resume/chirp), SuspendM (asserted when not suspended; deasserted in suspend), and a drive_k signal used for special K signaling.
- During special signaling (reset chirp and resume), the interface neutralizes DataOut and bounds TxValid to remain UTMI-compliant even if TxReady is held low by the PHY in special modes.

Link-state detection and transitions
- Attach/detach: VBUS/LineState based detection with ~100 ms debounce; status exported as usb_attached and reflected to software.
- Bus reset: Qualified SE0 assertion triggers RESET handling. The controller sets OpMode special, enables FS termination as required, and exposes a usb_reset event (sticky via RF).
- HS negotiation after reset: Device drives chirp K (~1.2 ms), counts host K/J toggles, and selects HS on 6 alternations; otherwise falls back to FS. Speed status is mode_hs.
- Suspend entry: Long idle (FS=J, HS=SE0; ~≥3 ms typical) puts the link into SUSPEND. SuspendM is deasserted and usb_suspend is asserted.
- Host resume: A long K during suspend exits SUSPEND. The controller restores normal OpMode/receivers, reselects HS if previously HS, waits ~100 µs to settle, and signals a one-shot suspend_clr on exit.

Device-initiated remote wake
- Software writes main_csr bit (0x00) to request remote wake. The request is latched at top level until suspend exit (auto-cleared by suspend_clr).
- When suspended for ≥~5 ms minimum, the controller sequences: enable FS termination, set OpMode special, drive K for ~1 ms, then return to normal operation via the ~100 µs settle path. A suspend_clr pulse marks suspend exit.

Software visibility and control
- Live status (main_csr 0x00 read): {line_state[1:0], usb_attached, mode_hs, usb_suspend}.

Remote-wake request (0x00 write) generates rf_resume_req.
- Sticky events (0x03 read-to-clear): attach, detach, suspend_start, suspend_end, usb_reset, and error flags (e.g., PID complement, CRC5, rx_err). Maskable into inta_o/intb_o.
- UTMI vendor sideband (0x05): VStatus[7:0] readback; VControl[3:0] write plus strobe for vendor PHY features.

Clocking, reset, and CDC
- Domains: Link FSM and UTMI interface run in phy_clk. RF/software and top-level control run in clk_i/wclk. usb_suspend is mirrored to susp_o in clk_i. LineState and vendor status are captured in phy_clk and exposed to RF. Resume request writes cross into the core and PHY domains via one-shot/synchronizers and are auto-cleared on suspend exit.
- Chip reset rst_i resets logic; USB bus reset is detected and handled by the link FSM without requiring a chip reset.

Power-management posture and guidance
- Internal clock/power gating: None. Only PHY low-power is managed via SuspendM and link timing. Protocol/DMA/memory blocks naturally idle when no valid traffic occurs.
- System-level gating: Use susp_o (mirrored usb_suspend) and RF sticky events to coordinate deeper platform savings (e.g., gate phy_clk, SRAM clocks, or SoC domains). Quiescent indicators: usb_suspend=1, no DMA requests (mreq=0), no Wishbone requests (wreq=0), mem arbiter idle (sram_we=0, stable address), and no pending RF interrupts (or masked).
- Gating safety: Do not gate phy_clk while Wishbone or memory transactions are active; ensure wb_stb_i=0, wb_cyc_i=0, wb_ack_o=0, and no MA requests before gating. Resume signaling and remote-wake sequences require clocks to run until suspend_clr pulses.

Limitations and assumptions
- HS test modes (SetFeature TEST_MODE) are not implemented. The design assumes a UTMI PHY honors SuspendM for electrical power savings; deeper vendor power controls, if any, can be exercised via VControl. Timing constants are derived from phy_clk and RTL macros; verify against your clock rate to meet USB 2.0 margins.

# OPERATION

-- Reset and Initialization Sequence --

- Reset sources, polarity, and clocking
- rst_i is active-low. Unless USBF_ASYNC_RESET is defined, resets are applied synchronously in each clock domain: phy_clk_pad_i (UTMI/protocol/memory arbiter) and clk_i/wb_clk (Wishbone/top-level status). Some flops use async reset only when USBF_ASYNC_RESET is enabled.
- phy_rst_pad_o mirrors rst_i (active-low reset output to the UTMI PHY).

- Effects while rst_i is asserted (low)
- UTMI/link
- usbf_utmi_ls: FSM forces POR; selects FS view (XcvSelect=FS, TermSel=1), mode_hs=0, clears usb_suspend and usb_attached, clears internal timers; next state ATTACH.
- usbf_utmi_if: rx_valid/rx_active/rx_err=0; TxValid=0; no handshakes driven; data buses may be don’t-care but are gated.
- Protocol and DMA

- usbf_pl/usbf_pe: main FSMs to IDLE; token/dir latches cleared; no send_token, no rx/tx DMA enables.
- usbf_idma: FSM IDLE; mreq/mwe=0; counters reset/parked; no memory traffic.
- Memory arbiter and SRAM
- usbf_mem_arb: wack_r=0 (W-side acks suppressed). mack mirrors mreq combinationally; parents should hold requests low. sram_re=1; sram_we=0 (no writes).
- Register file and endpoints
- usbf_rf: funct_adr=0; global masks and sticky RF events clear.
- usbf_ep_rf (all instances): CSR=0; iena/ienb=0; int_stat=0; buf0/buf1=0xffffffff (sentinel); dma_en=0; no per-EP DMA requests; interrupt outputs low. Dummy EPs drive fixed zeros and never match.
- Wishbone interface
- usbf_wb: PHY-domain FSM to IDLE; no rf_re/rf_we/ma_req. wb_ack history flops are not reset, but acks require a new wb_ack_d edge post-reset, preventing stray ACKs.
- Top-level/status
- susp_o is cleared (tracks usb_suspend=0). resume_req_r (remote-wake request latch) is cleared. LineState_r/VStatus_r registers will reinitialize on phy_clk after release.

- Reset release and power-on initialization sequence (rst_i goes high)
1) PHY/link bring-up
- phy_rst_pad_o deasserts. usbf_utmi_ls transitions POR→ATTACH; runs ~100 ms debounce on VBUS/LineState; when stable, usb_attached=1 and FSM enters NORMAL. Initial operating point is FS (mode_hs=0); OpMode is driven to normal during first RESET/RESUME sequence.
2) Speed negotiation on first host reset
- On SE0-long detection, usb_reset pulses; HS chirp negotiation runs. Enter HS after 6 K/J chirps; otherwise remain in FS. Firmware note: funct_adr is not auto-cleared by usb_reset; software must write address 0 per USB 2.0.
3) Status propagation
- susp_o begins reflecting usb_suspend in clk_i domain. LineState_r/VStatus_r continuously capture UTMI pins on phy_clk.
4) Datapath readiness
- usbf_pl/usbf_pe remain idle until a valid token match; usbf_idma is idle; no SRAM writes occur until a master requests. usbf_mem_arb grants fixed priority to the protocol M-side.
- First W-side memory ack after reset requires M idle; with wreq held high, wack pulses every other phy_clk once enabled.
5) Bus/register interface
- usbf_wb services RF region immediately (single-ACK). Memory region completes when usbf_mem_arb returns ma_ack. wb_ack_o is generated only on a new rising edge of wb_ack_d after reset; avoid WB accesses until clocks are stable.
6) Endpoint/RF defaults
- All EP windows read back cleared; no DMA requests or interrupts until software programs BUF0/BUF1 and CSR, then sets dma_en (last). Interrupt masks default to 0.
7) Remote wake capability
- resume_req_r=0 after reset. Software may arm remote wake by setting rf_resume_req; it is auto-cleared on suspend exit via suspend_clr.

- Behavior on runtime reset (rst_i asserted while active)
- All FSMs return to IDLE; protocol/DMA transactions are aborted; W-side acks stop immediately; M-side mack follows mreq combinationally (parents should deassert requests). No spurious transmissions or Wishbone ACKs occur because wb_ack_o requires a post-reset wb_ack_d rising edge.

- Software bring-up after attach

- Program RF: interrupt masks, function address (after SET_ADDRESS), optional UTMI vendor control/status.
- For each enabled endpoint: write BUF0/BUF1 (BUF0 also seeds capacity), then CSR (direction, endpoint number, max packet size), then set dma_en. Only after this will the core engage DMA and respond with ACK/STALL/NACK per policy.

- CDC/integration assumptions
- Design assumes wb_clk == phy_clk or tightly related; wb_req sampling and ack edge-detect are minimally synchronized. If unrelated clocks are used, add CDC synchronizers externally and avoid bus activity until rst release and clocks are stable.

- Simulation-only note
- usbf_top includes an initial block that prints configuration and checks endpoint contiguity; it has no hardware effect.

-- Attach/Detach, USB Reset, and Speed Negotiation --

Scope and ownership
- Implemented in the UTMI link layer (usbf_utmi_if → usbf_utmi_ls). usbf_top only forwards status; protocol/DMA/memory blocks consume status and remain speed■agnostic.

Attach and detach
- Qualification: Uses usb_vbus and UTMI LineState[1:0]; attach requires ~100 ms of stable presence before asserting usb_attached. Detach clears usb_attached when qualification is lost. Integrators must confirm usb_vbus polarity to ensure correct gating.
- Software view: main_csr at 0x00 exposes {suspend, mode_hs, usb_attached, line_state[1:0]} (live). Sticky RF events at 0x03 (read■to■clear): attach=bit5 (rising usb_attached), detach=bit6 (falling usb_attached). Masking at 0x02 feeds inta_o/intb_o.

USB reset detection and handling
- Detection: In NORMAL (FS view), a sustained SE0 is qualified using microsecond/millisecond windows (e.g., >2.5 µs short qualifier and >3.0 ms long window) to assert usb_reset and enter RESET handling.
- UTMI control during RESET: Forces non■NRZI/bit■stuffing OpMode (2'b10), enables FS termination (TermSel=1), and prepares for speed negotiation. After ~1.0 ms, the device proceeds to HS chirp.
- Software event: usb_reset is latched as a sticky RF bit at 0x03[8] (read■to■clear) and can interrupt via masks at 0x02. The function address (0x01) is not auto■cleared; firmware must reinitialize Chapter 9 state after a bus reset.

High■speed (HS) speed negotiation
- Device chirp: The device drives K (drive_k=1) for ~1.2 ms following reset.
- Host chirp detection: Counts alternating qualified host K/J chirps; six alternations select HS, otherwise fallback to FS when SE0/idles return.
- Mode selection and UTMI pins:
• HS success: mode_hs=1; XcvSelect=0 (HS path), TermSel=0, OpMode=Normal (2'b00).
• FS fallback: mode_hs=0; XcvSelect=1 (FS path), TermSel=1, OpMode=Normal.
- Persistence: The chosen speed remains until the next qualified reset. Protocol uses mode_hs to enable HS■only features (e.g., PING/NYET) and select HS/FS timeouts; data movers are otherwise unchanged.

Line-state and signaling integration
- LineState is sampled in the PHY clock and exported to software via main_csr for diagnostics during

attach/reset/chirp. While drive_k is asserted, the UTMI IF neutralizes DataOut and shapes TxValid so special signaling does not emit data.

Interrupt aggregation
- RF sticky events (attach, detach, usb_reset, etc.) are masked via 0x02 and ORed with per■endpoint A/B to form inta_o/intb_o. Reading 0x03 returns {int_srcb,int_srca} and clears int_srcb.

Top■level and timing notes
- rst_i drives phy_rst_pad_o; no additional top■level FSMs affect attach/reset/negotiation. suspend/resume logic is orthogonal (drive_k also used for resume).
- Timing uses counters in phy_clk to meet USB 2.0 targets: ~100 ms attach debounce; microsecond■scale SE0 qualifiers; ~1.0–1.2 ms device K■chirp; host K/J chirp counting to six alternations.

Non■ownership reminders
- usbf_pl/usbf_pe/usbf_idma/usbf_mem_arb/usbf_wb do not detect attach/reset/speed; they rely on usb_attached/usb_reset/mode_hs. CRC engines (CRC5/CRC16) are independent of these link events.

-- Suspend/Resume and Remote Wakeup --

- Scope and ownership
- Suspend entry/exit, host-initiated resume detection, and device-initiated remote wakeup are implemented in the UTMI link-state logic (usbf_utmi_ls with integration in usbf_utmi_if). The top-level exposes status/interrupts via usbf_rf and mirrors suspend to the SoC clock domain.

- Suspend entry and reporting
- Condition: After a qualified long idle on the bus (> ~3 ms), the UTMI FSM declares SUSPEND.
- FS view: idle = J; HS view: idle = SE0, then observed in FS view for a short window to decide SUSPEND vs RESET.
- Actions in suspend: assert usb_suspend, deassert SuspendM (per PHY polarity), set OpMode for special/non-driving as required, and manage XcvSelect/TermSel for low power.
- System view: usb_suspend (phy_clk) is registered into susp_o (clk_i). Software can read main_csr.suspend and receives sticky suspend_start/suspend_end events.

- Host-initiated resume
- While suspended, a qualified K on the bus triggers resume.
- Actions on resume: pulse suspend_clr, restore normal UTMI operation (OpMode normal, proper HS/FS select and terminations), wait a short settle (~100 µs), then return to NORMAL. susp_o follows usb_suspend low in clk_i.

- Device-initiated remote wakeup
- Request sources:
- Software: write main_csr bit 5 to raise rf_resume_req (one-shot in clk domain via usbf_rf).
- External pin: resume_req_i at usbf_top.
- Top-level latch: requests are ORed and latched in clk_i as resume_req_r; the latch is cleared by suspend_clr to allow at most one remote wake per suspend session.
- CDC: resume_req_r is two-flop synchronized into phy_clk before use by the UTMI FSM.
- Gating and timing (implemented by counters in phy_clk):
- Honored only while in SUSPEND and after a minimum suspend dwell (≥ ~5 ms).
- Optional wakeup preparation delay (~5 ms) before signaling.
- Drive resume K on the bus for ~1 ms (within USB 2.0 1–15 ms window) with OpMode set for special signaling.
- Post-resume settle of ~100 µs before returning to NORMAL.

- TX-path integration during signaling: usbf_utmi_if asserts TxValid while drive_k is active to ensure a compliant UTMI handshake even if TxReady is low; DataOut is forced to a neutral value during drive_k.

- Reset interaction
- A qualified long SE0 observed during suspend forces exit from suspend and entry to RESET; suspend_clr is pulsed and usb_reset is asserted. Software sees usb_reset as a sticky RF event.

- Software-visible control and status (usbf_rf)
- main_csr (0x00 read): {suspend, mode_hs, usb_attached, line_state[1:0]}.
- Remote wake request (0x00 write bit 5): generates rf_resume_req one-shot in clk domain.
- Sticky RF events at 0x03 (read-to-clear), maskable at 0x02:
- suspend_start -> int_srcb[3]
- suspend_end -> int_srcb[4]
- usb_reset -> int_srcb[8]
- plus attach/deattach and error flags.

- Effects on datapath and DMA
- While suspended, rx_active=0 and no tokens are seen; usbf_pd/usbf_pe/usbf_idma remain in IDLE with no CRC or DMA activity. No special re-initialization is required; CRC and protocol state restart naturally on the first packet after resume.
- usbf_mem_arb and usbf_wb continue to operate; software RF reads/writes are acknowledged. At resume, M-side (protocol/DMA) preempts W-side immediately.

- Clocks and CDC
- UTMI FSM, usb_suspend, suspend_clr, and signaling run in phy_clk.
- susp_o and the resume request latch live in clk_i; suspend_clr is safely sampled in clk_i to clear resume_req_r.
- rf_resume_req crosses from wclk to clk as a one-shot; resume_req_r crosses from clk_i to phy_clk with two-flop sync.

- UTMI control behavior
- OpMode=2'b10 during reset/resume special signaling; OpMode=2'b00 in NORMAL.
- XcvSelect/TermSel updated per state; HS/FS selections and terminations are restored after resume.
- SuspendM polarity must match the PHY (UTMI defines SuspendM high = not suspended).

- Timing summary (implementation constants; dependent on phy_clk)
- Suspend entry after > ~3 ms idle.
- Remote wake eligibility: ≥ ~5 ms in suspend; optional ~5 ms prep delay.
- Device-driven K width: ~1 ms.
- Post-resume settle: ~100 µs.

- Compliance and integration notes
- Ensure rf_resume_req (software) is ORed with resume_req_i (pin) into the UTMI resume path.
- Validate/tune constants to USB 2.0 requirements (e.g., 5 ms minimum suspend before remote wake; 1–15 ms K signaling).
- During UTMI-driven resume signaling, the normal TX/RX datapaths remain idle; no packets or DMA are issued until after settle.

-- Receive Path Operation --

Receive Path Operation
- Preconditions and clocking

• All receive decoding, policy, DMA, and memory arbitration run in phy_clk. Register-file (RF) exposure and Wishbone access run in clk_i; CDC one-shots/stickies are contained in usbf_rf/usbf_wb.

- UTMI ingress and link-state gating
• usbf_utmi_if registers RxValid/RxActive/RxError/DataIn each phy_clk as rx_valid/rx_active/rx_err/rx_data.
• usbf_utmi_ls classifies line state and drives mode_hs, usb_reset, usb_suspend, usb_attached, suspend_clr and UTMI control. Receive processing is meaningful only when attached, not in reset or suspend, and OpMode is normal.

- Packet decode and integrity checks
• First byte is PID with ones'-complement check; PID errors are flagged but do not propagate DATA to DMA.
• For TOKEN PIDs (OUT/IN/SETUP/SOF/PING), the 2-byte header is captured, token_fadr/ep_sel or frame_no extracted, and CRC■5 checked (x^5+x^2+1; compare to ones'-complement in header).
• For DATA PIDs, payload bytes stream on rx_data_st with rx_data_valid; at end-of-packet rx_data_done pulses. CRC■16 running sum (init 16'hffff, reflected 0x8005) is checked by residue (16'h800d) at rx_data_done; crc16_err flags mismatch.
• SOF updates frame number and a naturalized timestamp (frm_nat) for software.

- Address/endpoint match and policy gate
• A token is applicable when match_o = !pid_bad & (token_fadr==fa) & match & token_valid & !crc5_err, where match is the OR of per■EP matches from usbf_rf; pid_bad filters ACK/NACK/STALL/NYET/PRE/ERR/SPLIT and PING when !mode_hs. Non■matching addressed traffic raises nse_err.
• usbf_rf multiplexes the matched endpoint's CSR, BUF0/BUF1, and convenience flags (dma_out_buf_avail, dma_in_buf_sz1) to the protocol engine. Policy checks include enable/stall state, direction/type, MaxPacketSize, small/large OK bits, and buffer/DMA availability.

- RX start and DMA write to external SRAM
• On an acceptable OUT/SETUP, the protocol engine asserts rx_dma_en, selects a buffer (buf0/buf1) and supplies base address and receive size (min(buf_size, MPS)).
• usbf_idma assembles bytes into 32■bit words: performs a seed read for unaligned starts (RMW), writes a word on lane3 completion, and flushes the final partial word at rx_data_done. sizu_c counts received bytes; idma_done mirrors rx_data_done.

- Memory arbitration (protection of RX bandwidth)
• usbf_mem_arb grants strict priority to the protocol/DMA master (M) over Wishbone/software (W). While M asserts mreq, mack mirrors mreq (no added wait states), sram_adr/dout/we are driven by M, and sram_re is tied high. W is served only when M is idle, preventing RX back■pressure.

- Completion and handshakes
• At rx_data_done, CRC■16 and policy outcomes select the handshake: ACK on normal completion; NACK for to_small/to_large per CSR policy; STALL if stalled; HS may send NYET when no next buffer is available; PING is answered per HS rules. Isochronous OUT follows no■retry semantics; unexpected DATA PID in non■iso may elicit ACK with discard.

- Descriptor/state updates
• The protocol engine writes back descriptor updates over idin with buf*_set/buf*_rl and toggles uc_bsel/uc_dpd. new_size/new_adr advance by MPS for DMA OUT, or by sizu_c when not using DMA. buffer_done is true when remaining size < MPS; in DMA mode buffer_done may trigger buf0_rl. For HS OUT with DMA that ends short of MPS, out_to_small is flagged and writeback preserves the original

address while recording the short length.

- Errors, aborts, and timeouts
• CRC■5/CRC■16 errors, buffer overflow (sizu_c > buf_size while receiving), a fresh token match while busy, to_large, or an inactivity timeout during OUT cause abort of the reception and selection of a non■ACK handshake as policy dictates. Events are surfaced to software: RF■level stickies (rx_err, pid_cs_err, crc5_err, nse_err, attach/detach, suspend edges, usb_reset) and per■EP stickies (CRC16 error, timeout, sequence/unexpected PID, buffer■done). Reads of the respective registers clear the stickies.

- Speed, reset, and suspend impacts
• mode_hs gates HS■only behavior (PING/NYET) and timeouts; usb_reset reinitializes receive■side state; usb_suspend halts reception until resume, after which suspend_clr assists release of holds.

- Software observation and data access
• usbf_wb arbitrates RF vs memory space (addr[17]: 0=RF, 1=memory) and returns clean single■cycle wb_ack_o pulses in wb_clk. Because mem_arb prioritizes M, Wishbone reads of RX buffers naturally wait until RX DMA is idle, ensuring non■intrusive access. RF exposes live link state and per■EP windows (CSR/INT/BUF0/BUF1) for configuration and status.

- Compliance note
• The RTL may NAK/STALL SETUP based on policy; the USB spec requires always■accept SETUP. This caveat concerns control■pipe compliance rather than data movement.

-- Transmit Path Operation --

Purpose and scope
- Describes how the USB device transmit path (IN/data or handshakes) moves bytes from endpoint buffers in external SRAM to the UTMI PHY, including arbitration, packetization, UTMI handshakes, special link states, and completion.

End-to-end flow (IN/data)
1) Token/Policy: usbf_pd qualifies IN tokens; usbf_pl/usbf_pe check endpoint readiness/policy. If ready, select DATA PID (DATA0/1/2/MDATA) and start TX; otherwise select a handshake (NAK/NYET/STALL).
2) Readiness gate: usbf_ep_rf indicates buffer readiness via dma_in_buf_sz1 (>= one max packet and MPS!=0). ZLP is requested when MPS==0.
3) DMA prefetch: usbf_pe asserts tx_dma_en with adr, size=min(buf_size, MPS), buf_size, and dma_en. usbf_idma issues SRAM reads via usbf_mem_arb, maintaining a two-word ping–pong buffer and producing a paced byte stream (tx_data_st). usbf_pa pulls bytes with rd_next only when tx_ready allows.
4) Packet assembly: usbf_pa emits DATA PID → payload bytes → CRC16, asserting tx_first at SOP, holding tx_valid during the packet, and asserting tx_valid_last on the final CRC byte. For ZLP, payload is skipped.
5) UTMI handoff: usbf_utmi_if preloads DataOut on tx_first so the first byte is stable at TxValid assertion, holds TxValid until UTMI TxReady, and mirrors tx_ready upstream for flow control. usbf_utmi_ls may gate payload during special link states.
6) Completion: usbf_idma asserts idma_done when size is exhausted. For non-iso IN, usbf_pe waits for host ACK (observed on RX) within HS/FS-specific timeouts, then updates descriptors/toggles and raises interrupts; isochronous IN skips ACK wait.

Byte, CRC, and ordering
- Payload and CRC bits are sent LSB-first on UTMI. CRC16 polynomial: $x^{16} + x^{15} + x^2 + 1$; seeded

to 16'hFFFF at SOP; transmitted field is ~crc as two bytes: low then high. ZLP transmits 16'h0000 after the DATA PID. Device never transmits CRC5 (RX-only).

UTMI boundary behavior (usbf_utmi_if + usbf_utmi_ls)
- DataOut: driven with tx_data when (TxReady==1) or at SOP (tx_first==1). Otherwise holds its last value; forced to 00h during drive_k (special signaling).
- TxValid: asserts on tx_valid | tx_valid_last | drive_k; holds until TxReady; a one-cycle delayed copy of drive_k bounds special signaling pulses even if TxReady stays low.
- tx_ready: registered mirror of UTMI TxReady to pace upstream rd_next/tx_valid.
- Special signaling (usbf_utmi_ls): during RESET/RESUME/HS chirp, sets OpMode=2'b10, asserts drive_k (DataOut=00h, bounded TxValid pulse), and suppresses payload; returns to NORMAL (OpMode=2'b00) after settle. XcvSelect/TermSel are switched per speed/state.

Memory and arbitration (usbf_mem_arb)
- Fixed priority: protocol/DMA M-port always wins over Wishbone W-port. mack=mreq (no inserted waits); read data is broadcast. W-port acks pulse only when M is idle and are preempted immediately by any M request. TX uses reads only (mwe=0).

Endpoint readiness and DMA accounting (usbf_ep_rf)
- Tracks per-endpoint DMA prefill count (words) vs capacity and MPS. dma_in_buf_sz1 asserts when at least one max packet is available to avoid underrun. Generates dma_req toward the memory subsystem (wclk domain) and consumes dma_ack to step counters. Latches descriptor write-backs and status/interrupts used by usbf_pe.

Wishbone/software interaction (usbf_wb)
- Software fills endpoint buffers via the memory-array (MA) window and programs endpoint CSRs/descriptors via the register-file (RF) window. MA operations are arbitrated by usbf_mem_arb and may be stalled during active TX; RF accesses ack immediately. M-port priority guarantees software cannot starve transmit prefetch.

Handshake-only path
- For ACK/NAK/STALL/NYET (and HS PING responses), usbf_pa emits a single PID byte. tx_first + tx_valid_last wrap a one-beat transfer that honors UTMI TxReady using the same handshake rules.

Flow control and underrun avoidance
- tx_ready gates rd_next so usbf_idma advances only when the PHY accepts bytes. Two-word prefetch plus fixed-priority arbitration sustain continuous streaming; tx_valid_last guarantees the final byte handshake.

Completion and status updates
- After idma_done (and ACK for non-iso), usbf_pe computes new_size/new_adr, updates/toggles descriptors (buf*_set/rl, uc_*_set), raises interrupts (buffer-done, timeout), and returns to idle. Any new token match preempts and returns to IDLE on the next cycle.

Timing, speed, and availability
- mode_hs indicates HS/FS selection for PID choice and timeout windows. During RESET, SUSPEND, RESUME_SIG, HS chirp, and settle windows, payload TX is unavailable; normal TX resumes only in NORMAL state. All UTMI TX-side registers are synchronous to phy_clk.

Principal interfaces (TX-side)
- pa→utmi_if: tx_data[7:0], tx_valid, tx_valid_last, tx_first; utmi_if→pa/pl: tx_ready.
- idma→mem_arb/SRAM: mreq/mack, madr, mdin (read), mwe=0.
- pl/pe→pa/idma: data_pid_sel, send_zero_length, tx_dma_en, size/adr; pa→idma: rd_next; idma→pa:

tx_data_st, send_data.

-- Endpoint Policy and Buffer Management --

- Scope
- Defines how tokens are admitted to endpoints, how per-endpoint policy is enforced, how packet data is buffered in external SRAM, and how descriptors are committed, with software interaction via the register file and Wishbone memory window.

- Admission and link-state gating
- A transaction engages an endpoint only when token_valid with good PID complement and CRC5; non-matching or malformed tokens have no buffer side effects.
- While unattached, suspended, or in reset/chirp/resume windows, endpoints NAK and internal DMA stays idle; descriptors are preserved. On usb_reset, firmware reinitializes toggles/descriptors and IDMA state.

- Per-endpoint policy (CSR in usbf_rf/usbf_ep_rf)
- Direction/type (IN/OUT/CTRL), endpoint number, transfer type, stall/disable, max packet size (MPS), small/large OK bits, DMA enable (ring mode), and software-visible bases uc_bsel (buffer select) and uc_dpd (DATA PID base).
- HS/FS behavior: HS enables PING/NYET and tighter timeouts; FS filters PING. IN waits for host ACK within speed-specific timeouts; OUT uses activity windows.

- Buffering model and selection
- All payloads live in a single external 32-bit SRAM. Each endpoint has two 32-bit descriptors BUF0/BUF1 (base byte address + size). Reset value 0xffff_ffff denotes not allocated.
- Selection: CTRL maps OUT/SETUP→BUF0 and IN→BUF1. Non-CTRL prefers uc_bsel unless NA, then auto-selects the other. With DMA/ring mode enabled, BUF0 is used exclusively.
- Live availability hints exported for the currently matched endpoint: dma_in_buf_sz1 (IN has ≥1 MPS prefilled) and dma_out_buf_avail (OUT has ≥1 MPS capacity). These directly gate accept/NAK/NYET decisions.

- Data movement (IDMA)
- RX (OUT): Packs bytes to 32-bit, performs read-modify-write on unaligned first word, flushes partial last word. CRC16 residue must be good at rx_data_done or the receive is dropped (no commit).
- TX (IN): Two-word prefetch sustains streaming; CRC16 is appended on the fly and excluded from descriptor sizes. Zero-length packets can be requested without memory reads.
- Ring/wrap: With dma_en, addressing wraps within BUF0 size to support rings; descriptors reflect payload bytes only.

- Handshakes, PID/size policy, and errors
- usbf_pe selects ACK/NAK/STALL/NYET based on policy, availability, and speed. HS PING is ACKed; NYET returned when no buffers are ready.
- Data PID selection honors transfer type, speed scheduling (ISO in HS), and uc_dpd; unexpected PIDs raise seqerr (ISO proceeds; non-ISO does not update descriptors).
- On pid/check errors, CRC failures, overflow/abort, or mid-transaction re-match, DMA is aborted and buffers are not committed.

- Descriptor updates and lifecycle
- Transaction size = min(buf_size, MPS) for IN; OUT updates use received count (sizu_c). new_adr/new_size computed accordingly.
- Commit via buf0_set/buf1_set with idin; uc_bsel_set and uc_dpd_set commit next buffer select and DATA PID base; buf0_rl releases BUF0 after DMA when buffer_done. OUT short in DMA mode is

reported (out_to_small) and may preserve base.
- Buffer_done criteria: IN when size exhausted; OUT when remaining size < MPS (no room for another full packet). Non-DMA toggles uc_bsel on buffer_done; DMA pins to BUF0.

- Interrupts and observability
- Per-EP read-to-clear INT bits: buffer-set, CRC16 error, timeout, sequence/UPID events, short-packet; aggregated into inta/intb. Software re-arms by updating descriptors and acknowledging interrupts.

- Software interaction (usbf_wb/usbf_rf)
- RF region programs CSR and BUF0/BUF1; MEM region gives direct SRAM access. The USB core has priority on the single SRAM port, so MEM accesses can be delayed under traffic; RF accesses are immediate.
- Typical flow: IN—prefill MEM, arm descriptor, wait for interrupt, re-arm; OUT—arm descriptor, on interrupt read size then drain MEM and re-arm. Coordinate to avoid races; do not modify in-use descriptors.

- Known caveats
- EP0 control exceptions (e.g., SETUP always-accept) may require firmware policies; HS Test Modes not implemented. MPS=0 enables ZLP generation but no prefill hint (dma_in_buf_sz1=0).

-- Memory Arbitration Behavior --

Single 32-bit, word-addressed synchronous SSRAM is shared by two masters via usbf_mem_arb on phy_clk: M (core/DMA from usbf_idma) and W (software via usbf_wb). Arbitration is fixed-priority (M > W) with no fairness or parking; W may be starved. Selection uses wsel = (wreq | wack) & !mreq, keeping W selected through its ack and immediately de-selecting W on any M request. Handshakes: M sees mack = mreq (no arbiter-added wait states); W sees a one-phy_clk-cycle wack only when M is idle. With wreq held and M idle, wack pulses every other phy_clk (≤ 0.5 transfer/clk); any mreq instantly suppresses wack (preemption). Address/data/WE mux: if wsel, sram_adr=wadr, sram_dout=wdin, sram_we=(wreq&wwe;); else sram_adr=madr, sram_dout=mdin, sram_we=(mwe&mreq;). sram_re is tied high; read data is broadcast (mdout=wdout=sram_din); each master must sample only on its own ack. Throughput: M is limited only by SSRAM timing and DMA patterns; W is limited by the arbiter's every-other-cycle grants and further by usbf_wb, which inserts two extra phy_clk cycles after each ack (≈ one MEM transfer every three phy_clk in steady idle). Clocking/CDC: arbiter logic is entirely in phy_clk with a single state bit (wack_r); usbf_wb samples WB requests into phy_clk and returns a synchronized one-cycle wb_clk ack. Non-participants (UTMI, PD/PA, CRC, RF/EP RF) do not access SSRAM; they only indirectly shape when usbf_idma issues M requests. Wishbone mapping: wb_addr_i[17]=1 routes to MEM (arbiter W port); 0 routes to RF (bypasses arbiter). Assumptions/risks: continuous read-enable requires compatible SSRAM; W can be delayed or starved; broadcast read data must be latched strictly on the owner's ack.

-- Wishbone Transaction Sequencing and Acknowledge --

Protocol and scope: Classic Wishbone B3 single-beat slave. 32-bit, word-aligned accesses. No SEL/CTI/BTE, no ERR/RTY/STALL; ACK_O is the only termination and is owned by the usbf_wb front-end.

Address regions and decode: wb_addr_i[17]=0 selects RF (register file); wb_addr_i[17]=1 selects MEM (external SRAM window). ma_adr passes wb_addr_i; rf_dout/ma_dout = wb_data_i.

Sequencing (one outstanding transfer): A new request is sampled only in IDLE using wb_stb_i & wb_cyc_i (phy_clk domain FSM). After each completed transfer the FSM inserts two phy_clk spacing cycles (W1, W2) before returning to IDLE, so no pipelined back-to-back acceptance.

- RF region (addr[17]=0): In IDLE, assert rf_re/rf_we for one phy_clk. Immediate acknowledge via W0, then W1→W2→IDLE. Endpoint/global RF side-effects (e.g., read-to-clear) occur after the read and never delay ACK.
- MEM region (addr[17]=1): In IDLE, assert ma_req (and ma_we for writes). Hold until ma_ack. On ma_ack, assert ACK, then W1→W2→IDLE.

ACK generation and timing: wb_ack_d is asserted in phy_clk on RF W0 or on MEM ma_ack. It is retimed to wb_clk with a 3■FF edge detector producing exactly one wb_clk-cycle ACK_O pulse per transfer (never stretched). Read data wb_data_o is registered in wb_clk and muxed by region (rf_din vs ma_din); it must be sampled by the master on the ACK_O cycle.

MEM acknowledge source and arbitration: ma_ack is usbf_mem_arb.wack. The arbiter gives fixed priority to the USB core M-side; W-side (Wishbone) is served only when M is idle. When M is idle, W can be acknowledged at most every other phy_clk. Any M activity suppresses wack, so MEM-region ACK latency is variable and can be arbitrarily extended under DMA/USB traffic.

Throughput and latency: RF region—deterministic immediate service (plus ACK retime). MEM region—peak acceptance ~1 transfer per 2 phy_clk when M is idle; actual latency depends on arbiter availability. FSM enforces a two-phy_clk gap after each ACK before a new request is sampled.

Reset and CDC: FSM resets to IDLE on active-low reset in phy_clk; ACK pulse generation is in wb_clk. Request sampling crosses domains with minimal protection; best operated with wb_clk == phy_clk or otherwise timing-closed. ACK_O path is safely retimed.

Master usage (classic WB): Hold CYC_I and STB_I until ACK_O; change ADR/DAT only after ACK_O. No bursts/pipelining; issue one request per ACK. Expect variable wait for MEM accesses under active USB DMA; RF accesses always use the immediate-ACK path.

-- Error Detection, Reporting, and Recovery --

Multi-layer detection, reporting, and recovery are implemented across the UTMI/link layer, protocol engine, endpoint/register file, and software-visible interrupt surfaces. Recovery uses USB-compliant handshakes (ACK/NACK/STALL/NYET), timers, descriptor updates, and link-state transitions (reset/suspend/resume).

Detection
- UTMI/link: rx_err mirrors UTMI RxError; link FSM detects attach/deattach, bus reset (SE0 window), suspend entry/exit, and HS chirp integrity (6 qualified host chirps to enter HS else FS fallback).
- Header/data integrity: PID complement check (pid_cs_err); TOKEN CRC5 (crc5_err); DATA CRC16 residue (per-EP int_crc16_set on rx_data_done & mismatch).
- Addressing/endpoint: nse_err flags tokens not for this function/endpoint.
- Sequencing/policy: unexpected or illegal PIDs (int_upid_set); data PID/toggle errors (int_seqerr_set); size violations vs MaxPacketSize (to_small/to_large); HS OUT short-size flag (out_to_small); buffer/DMA unavailability (no_bufs, bufX_na); overflow and preemption aborts.
- Timing: speed-dependent timeouts for IN ACK wait and OUT activity (int_to_set; typical HS=22, FS=36 cycles).

Reporting (software surfaces)
- Per-endpoint INT (0x10+4*EP+1, read-to-clear) latches strobes when ep_match: [6] out_to_small, [5] int_seqerr_set, [4] int_buf1_set, [3] int_buf0_set, [2] int_upid_set, [1] int_crc16_set, [0] int_to_set. Enables iena/ienb select which bits feed per-EP inta/intb.
- RF sticky events (0x03, read-to-clear): [8]=usb_reset, [7]=rx_err, [6]=deattach, [5]=attach,

[4]=suspend_end, [3]=suspend_start, [2]=nse_err, [1]=pid_cs_err, [0]=crc5_err.
- Interrupt aggregation: inta_o/intb_o = OR of all endpoint inta/intb OR (|(int_srcb & inta_msk/intb_msk)). Masks at 0x02. int_srca[15:0] is live per-EP OR(A|B); int_srcb[8:0] is RF sticky.
- Diagnostics: main_csr (0x00) exposes suspend, HS/FS, attached, line_state; frm_nat (0x04) gives SOF/microframe timestamp; UTMI vendor access at 0x05 aids PHY diagnostics.

Recovery (automatic + policy-driven)
- Protocol engine (transaction level):
- OUT: If buffers/DMA unavailable, NACK (FS) or NYET (HS). On to_small/to_large, NACK; on HS short-size with DMA, set out_to_small and special write-back preserving address and writing received size. With CRC16 error/timeout/abort, discard and do not update descriptors; host retries.
- Sequencing: Non-iso OUT with toggle error ACKs but discards (no update); ISO OUT records error and proceeds to update. If ep_stall set and token matches, respond STALL.
- IN: Start TX or ZLP as needed. Non-iso waits for host ACK; on timeout, raise int_to_set and return to IDLE (host retries). ISO IN skips ACK-wait.
- Descriptor/state updates occur only on success: buf0/1 write-back or release (buf0_rl), uc_bsel_set and uc_dpd_set commit buffer select and next data PID.
- Link level (automatic):
- Reset: assert usb_reset, force non-driving OpMode, perform HS chirp (~1.2 ms); choose HS only after 6 qualified host chirps else FS; then return to normal.
- Suspend/resume: enter on long idle; on host resume K, clear suspend after SE0 observation and ~100 µs settle. Remote-wakeup via resume_req enforces >=5 ms quiescent, drive K ~1 ms, then settle.
- IDMA containment: RX flushes last partial word on normal end; on abort drops partial (no commit). TX stops cleanly on completion/abort. idma_done pulses for PE state transitions. Address wrap at last_buf_adr is normal; no error signaling.
- Memory/Wishbone: mem_arb grants absolute priority to the USB master; W port may stall but sees no ERR/RTY. usbf_wb returns a single ACK per request; stalls appear as waitstates, not faults.

Limits/cautions
- No ECC/parity/timeout detection on SRAM/Wishbone; no ERR/RTY. Sustained M traffic can starve W.
- CDC assumptions in Wishbone front-end: run wb_clk ~= phy_clk or close timing; violations manifest as stalls, not explicit errors.
- EP0 edge cases: SETUP/length mismatch handling is simplified; firmware should reinit EP0 toggles/descriptors on any SETUP and on usb_reset.
- PHY nuances: verify SuspendM polarity and OpMode behavior with target PHY. HS Test Modes not implemented.

Firmware guidance
- On usb_reset sticky: reprogram function address and reinitialize endpoint state. Use per-EP INT windows to service buffer-done/errors; read 0x03 to log/clear RF sticky events; frm_nat helps correlate timeouts and flow.

-- Timing and Latency Considerations --

- Clocks, reset, and CDC
- Two active clocks: phy_clk (UTMI/protocol/DMA/memory arbiter) and wb_clk/clk_i (Wishbone/register file/interrupts). Resets are synchronous unless USBF_ASYNC_RESET is enabled. Best practice: keep wb_clk phase-related or equal to phy_clk.
- CDC paths are intentionally minimal: WB request sampled in phy_clk (no 2-FF sync, assumes wb_clk≈phy_clk), WB ACK returned to wb_clk via short edge detector (typically 1–2 wb_clk cycles), DMA ack to RF uses 2-FF sync (~2 clk cycles), resume_req has a 2-FF sync in UTMI LS.

- UTMI interface and link-state timing

- RX path is fully registered at phy_clk: rx_valid/active/err and rx_data are visible 1 phy_clk after the UTMI pins.
- TX handshaking: TxValid is held until TxReady; tx_first preloads SOP for setup margin; tx_valid_last may extend the final byte by up to 1 phy_clk if TxReady lags.
- Special signaling (resume/chirp): drive_k is registered in UTMI LS and delayed one more cycle in UTMI IF, producing a bounded 1–2 cycle TxValid pulse independent of TxReady.
- UTMI control/status in LS are registered: XcvSelect/TermSel/OpMode/SuspendM and mode/status take effect one phy_clk after assertion.
- Line-state qualifiers (k/j/se0/idle) require two consecutive samples: add 1–2 phy_clk cycles before transitions depending on them.
- Timers: base tick ~250 ns; sub-ticks aggregate to ~100 µs and 0.5 ms steps for ms-scale windows. Key windows (approx.): attach debounce ~100 ms; suspend detect >3 ms; resume K drive ~1 ms; remote-wake requires ≥5 ms in suspend then ~1 ms K; HS reset ~1.0 ms then ~1.2 ms device K and six chirps. VBUS change to POR: 1 phy_clk.

- Protocol engine and packet-level latency
- Token/PID/CRC5 check aligns so crc5_err is valid with token_valid one cycle after token1 capture; including UTMI input reg, ≈2 phy_clk after the second token byte at the UTMI pins.
- Handshakes: decision→registered controls in 1 phy_clk→fixed 1-cycle TOKEN; usbf_pa transmits next.
- IN: data launch latency dominated by IDMA first read ack; ACK-wait timeouts HS=22 cycles, FS=36 cycles. Best-case ACK detect occurs the cycle token_valid hits in IN2.
- OUT/SETUP: ISO commits immediately on rx_data_done; non-ISO pipelines two cycles (OUT2A→OUT2B) before registering the handshake.
- Preemption: any new addressed token forces return to IDLE on the next cycle.

- CRC16 (DATA path)
- Combinational CRC16 engine; seed on rx_active rising; per-byte update each phy_clk while data_valid. Residue check (16'h800d) occurs in the same cycle as rx_data_done; error flag aligns with rx_data_done. From last CRC byte to flag: UTMI reg + end recognition (few cycles total).

- IDMA and buffer/memory latency
- TX (mem→USB): first payload byte becomes available 1 phy_clk after the first memory read ack (with mem_arb mack=mreq). Two-word ping-pong sustains 1 byte/phy_clk when memory can accept a read at least every 4 cycles. send_data deasserts in the cycle the last byte is consumed; idma_done (TX) asserts 1 cycle later and stays high until next start.
- RX (USB→mem): unaligned starts require a seed read before first write (≥1 extra read latency). Writes occur every 4 bytes; each write’s ack is recognized 1 cycle later. Final partial-word flush adds a short tail; idma_done (RX) pulses 1 cycle after rx_data_done and can precede the last write ack by a few cycles.

- External SRAM arbiter (mem_arb)
- Fixed priority: M (core/DMA) always wins. mack=mreq (no inserted wait states). W ack (wack) is a 1-cycle pulse at most every other phy_clk when M is idle; immediate suppression on any M activity. Worst-case W latency is unbounded under sustained M traffic.

- Wishbone front-end (usbf_wb)
- One outstanding request; sampled in phy_clk only in IDLE. RF region ACK: generated in the next phy_clk (W0), then returned to wb_clk as a 1-cycle pulse (typically 1–2 wb_clk cycles later). Memory region ACK: on mem_arb wack (≥ every other phy_clk when M idle) plus the ACK synchronizer. Inter-access gap: fixed two phy_clk cycles (W1/W2) after each ACK before next request acceptance.
- Best-case latency (RF): ≈1 phy_clk (accept/ack) + 1–2 wb_clk (sync). Best-case (MEM): ≥2 phy_clk

(arbiter cadence) + 1–2 wb_clk (sync); worst-case unbounded if M is busy. Read data must be valid at the wb_clk that asserts wb_ack_o.

- Register file and interrupts (usbf_rf)
- Events in phy_clk are synchronized/sticky into clk_i; inta_o/intb_o update in wclk/clk_i with ~1 cycle latency from the register domain. Typical end-to-end interrupt latency: a few clk_i cycles from the originating phy_clk event. Vendor UTMI status sampled in clk_i; control writes produce one-shot pulses into phy_clk on the next clk_i edge.

- Suspend/resume and remote wake
- usb_suspend generated in phy_clk; susp_o is its reflection in clk_i with 1 clk_i latency. Host resume clears suspend after a 2-sample K qualify (~2 cycles) plus one register cycle; remote-wake timing is governed by ms-scale UTMI LS timers (≥5 ms holdoff, ~1 ms K) with negligible extra CDC latency.

- Data alignment nuances
- RX outputs are registered: rx_data_valid_r/rx_data_st_r add 1 cycle; rx_data_st is a two-byte delayed stream relative to immediate rx_data used for CRC16 updates. Token match and status are registered where consumed; live EP view in the RF has ~1 clk selection latency.

- Throughput and system guidance
- UTMI can sustain one byte per phy_clk when TxReady is asserted and IDMA/memory keep pace (e.g., 60 MHz HS, 12 MHz FS). The arbiter's M priority protects protocol timing; Wishbone memory traffic is throttled accordingly. Schedule software memory-window accesses during USB idle to avoid long WB latencies.

- Reset and VBUS behavior
- rst_i drives internal resets and phy_rst_pad_o; outputs update on the next respective clock edge. VBUS assertion forces POR in UTMI LS on the next phy_clk; attach debounce timing restarts thereafter.

-- Corner Cases and Sequencing Rules --

- Line-state, reset, suspend, resume
- Line sampling is double-qualified (two consecutive samples) for J/K/SE0 and idle. Non-idle clears/extends timers.
- Attach declared only after ~100 ms debounce; until then TermSel/OpMode may be non-normal. usb_vbus forces POR; ensure correct polarity or FSM may be stuck in POR.
- Enter suspend after ≈3 ms idle; usb_suspend asserted only in SUSPEND. On HS idle, switch to FS view for ~100 µs before deciding SUSPEND vs RESET.
- Bus reset (SE0 >~2.5 µs and <~3 ms) clears suspend immediately and forces RESET, then HS negotiation: device chirp K for ~1–1.2 ms, require 6 alternating host chirps; else fall back to FS.
- Host resume: detect K-long in SUSPEND, clear usb_suspend, wait for SE0, restore OpMode, then ~100 µs settle to NORMAL.
- Remote wake: honored only if in SUSPEND ≥50 ms; delay ~5 ms, drive K ~1 ms (OpMode=10), then ~100 µs settle. suspend_clr is a one-shot on any suspend exit; use it to clear upper-layer latches.
- resume_req is level-synced and only effective in SUSPEND when eligible. Top-level may latch resume_req until suspend_clr; asserting it while not suspended can trigger later.

- UTMI control and data handshakes
- XcvSelect 1=FS/0=HS; OpMode=2'b10 during resume/reset/chirp; verify PHY expectations for SuspendM and OpMode behavior.
- During special K drive (resume/chirp): DataOut forced to 0x00; TxValid pulsed with hold-until-ready disabled on the next cycle; do not start normal TX concurrently.

- Normal TX: TxValid holds until TxReady; assert tx_valid_last on final beat to avoid tail drop; tx_first can preload first byte.
- RX pins are registered once per phy_clk; downstream must not rely on combinational UTMI timing.

- Protocol engine (transactions, preemption, timeouts)
- Token acceptance requires good PID complement, valid CRC5, address match, and speed-legal PID (e.g., PING only in HS).
- New token match preempts any state and forces next-cycle IDLE; nse_err pulses on valid token to non-matching EP.
- IN: start TX DMA or ZLP; non-iso waits for ACK only (NAK/STALL ignored) within HS/FS timeout, else timeout (int_to_set) with no data-toggle update. Iso skips ACK wait.
- OUT: while receiving, tx_data_to/crc16_err/abort returns to IDLE with no handshake. At rx_data_done, iso updates without handshakes; non-iso selects final handshake (ACK/NACK/STALL/NYET-HS). to_small/to_large branch selects NACK but implementation does not assert send_token in that path (no handshake emitted).
- Data PID: iso OUT accepts with int_seqerr_set; non-iso OUT ACKs but discards on sequence error (no descriptor update).
- HS PING ACKed only in HS; filtered in FS by parent gating.

- CRC5/CRC16 sequencing
- CRC5 computed after token1 capture; token_valid pulses one cycle later. token0/1 not reset; gate crc5_err with “previous PID was TOKEN” to avoid false flags (ACK also drives token_valid).
- CRC16 seeded to 16'hFFFF at rx_active rise; LSB-first 8-bit updates over payload plus the two CRC bytes; residue must be 16'h800d at rx_data_done. If PHY strips CRC bytes, switch to explicit field compare.
- Two-byte RX pipeline suppresses CRC bytes from payload; on rx_err/early EOP, last one/two payload bytes may not emerge.

- DMA (usbf_idma) and memory arbitration (usbf_mem_arb)
- Start on rx_dma_en/tx_dma_en (RX priority if both). rx adr counters reload on start; incidental signals while IDLE are harmless.
- RX path always seeds first word via a read (RMW), even if aligned; flushes partial last word at packet end. idma_done asserts on rx_data_done, not on final write ack; last partial word can be dropped if aborts late.
- TX path uses two-word ping-pong prefetch; decrement logic avoids double count when rd_first&ack; coincides with rd_next. ZLP: no reads; require size=0 or idma_done will not assert.
- Writes require same-cycle ack (mreq pulse); delayed write acks are missed. Reads hold mreq until ack.
- Ring wrap compares next word address to last_buf_adr; require word-aligned buffer size and 0-relative base.
- rd_next must be issued only while send_data is high; upstream must align to tx_ready.
- Arbiter: M side (core/DMA) always wins; W side (Wishbone) acks at most every other phy_clk and is preempted immediately by M. sram_re tied high; masters must qualify data with their ack.

- Register file and endpoint RF
- Sticky event bits in RF clear only on read of 0x03; clear is level-sensitive to the read strobe. EP INT bits are per-EP read-to-clear; clearing one does not clear the other.
- Live endpoint view exposes the highest-priority matched EP (ep0..ep15); duplicated EP numbers cause ambiguous live view and selection.
- ep_match_r gates in-band updates; hold *_set strobes at least one clk after a new match. Bus writes override in-band buf*_set/out_to_small in the same cycle.
- DMA counters in EP RF operate in 32-bit words; P=floor(MPS/4). buf0_orig reset is 0xFFFF_FFFF;

program BUF0 before enabling DMA or IN may prefetch unboundedly.
- Direction encoding must be valid (01 IN, 10 OUT); any other disables requests. MPS==0 forces IN “enough-data” low and disables OUT requests stepping.
- Ack+set same-cycle prioritization: counters apply +/-1 (ack) first, then apply P-step next cycle via set_r.

- Wishbone access (usbf_wb)
- One request at a time; sample only in IDLE; no bursts/pipelining. Hold ADR/DAT/WE/CYC/STB stable until ACK_O.
- Region select wb_addr[17] (0 RF, 1 Memory) must remain stable through ACK; read data valid only on ACK cycle.
- RF acks immediately; Memory acks when ma_ack observed. ACK_O is a one-cycle pulse in wb_clk derived via CDC; wb_clk≈phy_clk strongly recommended. wb_ack pipeline not reset—keep bus idle across reset.

- Top-level glue and compliance
- susp_o is clk_i-registered copy of usb_suspend; software polling must allow for clock-domain latency.
- resume_req_r latches until suspend_clr; asserting resume while not suspended latches a future wake.
- Initial EP enables assumed contiguous; gaps/duplicates can mis-select live view.
- Chapter 9 caveats: HW allows NAK/STALL of SETUP and lacks forced DATA0/clear-halt; firmware must enforce legal EP0 behavior.
- After bus reset, device address must be 0 before responding; ensure firmware complies if not hardwired.

- Timing and eligibility notes
- All thresholds are approximate and clock/macro dependent: attach ~100 ms; suspend ≈3 ms; remote-wake eligibility ≥50 ms; wake delay ~5 ms; resume K ~1 ms; settle ~100 µs; HS chirp windows ~1–1.2 ms.
- Timers run only while idle_long; non-idle resets/divides counters and stretches detection windows.

## REGISTER MAP

-- Addressing Scheme and Regions --

- Bus and width: 32-bit Wishbone slave; word-aligned accesses only (no byte enables). One transaction at a time; data/ACK returned per access.
- Top-level region select (this build): wb_addr_i[17]
- 0 = Register-File (RF) region (software-visible control/status)
- 1 = Memory Array (MA) region (external SRAM for packet buffers)

- RF region (wb_addr_i[17] = 0)
- Decode granularity: only adr[6:0] are used; upper Wishbone bits (except bit17) mirror the RF window.
- RF word index: adr[6:2]; adr[1:0] is forwarded only to endpoint sub-windows.
- Global RF window (mirrored): adr[6:2] == 0x00 (low page) and 0x01 (mirror) read the same values; side effects occur only on the low page.
- 0x00: main_csr (R); W with din[5]=1 issues a remote-wakeup (rf_resume_req) pulse
- 0x01: function address (R/W; 7-bit device address)
- 0x02: interrupt masks A/B (R/W)

- 0x03: interrupt sources (R; read-to-clear sticky RF int_srcb on low page)
- 0x04: frm_nat timestamp (R)
- 0x05: UTMI vendor status (R) / vendor control + write strobe (W on low page)
- Per-endpoint windows (EP0..EP15): RF word addresses 0x10..0x4F; base = 0x10 + 4*EP
- Offsets within each EP (adr[1:0]): 0=CSR (R/W), 1=INT (R/W; read-to-clear status), 2=BUF0 (R/W), 3=BUF1 (R/W)
- All 16 windows are decoded; unused/dummy endpoints read as zero and ignore writes.

- MA region (wb_addr_i[17] = 1)
- Linear 32-bit word addressing into external synchronous SRAM; no sub-decoding inside MA.
- Effective physical address bits: sram_adr[SSRAM_HADR:0] (word address). Higher bits alias within the SRAM depth.
- Example depth: SSRAM_HADR=14 → 2^15 words = 512 KiB. Software must keep descriptors within this depth.
- No byte lanes; sub-word payload updates are done by the DMA via read–modify–write.

- Descriptor/address interaction
- RF BUF0/BUF1 fields carry byte addresses [(SSRAM_HADR+2):0] and 14-bit sizes; the DMA converts to word addresses madr[SSRAM_HADR:0] for MA.

- Handshake/ACK visibility to software
- RF accesses are decoded and ACKed immediately; MA accesses are forwarded to the memory arbiter and ACKed when served; read data is muxed by region (RF vs MA).

- Modules that do not add address space
- utmi_if/utmi_ls, crc5/crc16, packet decoder/encoder, IDMA/protocol engine, memory arbiter introduce no additional bus-visible regions; any software-visible status/control is surfaced via the RF region.

- Aliasing/mirroring summary
- RF: mirrored on upper address bits; only adr[6:0] are effective; global window side effects only on low-page 0x00..0x05.
- MA: addresses above the physical SRAM width alias within its depth.
- Access width everywhere: 32-bit words only.

-- System/Core Registers --

- Bus/region overview
- 32-bit Wishbone slave, word addressed. wb_addr_i[17] selects:
- 0: System/Core Register File (RF) described below
- 1: External SRAM packet-buffer space (not part of the register set)
- RF addressing uses adr[6:0] internally. Each access returns a one-cycle ACK pulse.

- Global RF registers (wb_addr_i[17]=0)
- 0x00 main_csr (R) / resume_req (W)
- Read: {27b0, line_stat[4:3], usb_attached[2], mode_hs[1], suspend[0]}
- suspend: 1 when core is in SUSPEND
- mode_hs: 1 = High-Speed, 0 = Full-Speed
- usb_attached: 1 when VBUS/attach qualified (debounced)
- line_stat: UTMI LineState snapshot (00=SE0, 01=J, 10=K, 11=SE1)
- Write: din[5]=1 issues a remote-wakeup request one-shot; other bits ignored
- 0x01 funct_adr (R/W)
- din[6:0] are stored as the USB device address (others read as 0)

- 0x02 interrupt masks (R/W)
- {7b0, intb_msk[8:0], 7b0, inta_msk[8:0]}; masks apply to RF-level sticky events only
- 0x03 interrupt sources summary (R; read-to-clear for RF stickies only)
- {3b0, int_srcb[8:0], 4b0, int_srca[15:0]}
- int_srca[15:0]: live OR of per-endpoint (inta|intb), one bit per EP; does not clear on read
- int_srcb[8:0] (sticky, clears on read):
- [8] usb_reset (qualified bus reset/SE0 detected)
- [7] rx_err (UTMI RxError)
- [6] deattach (usb_attached falling edge)
- [5] attach (usb_attached rising edge)
- [4] suspend_end (leaving SUSPEND)
- [3] suspend_start (entering SUSPEND)
- [2] nse_err (token not for this function/EP)
- [1] pid_cs_err (PID complement check failed)
- [0] crc5_err (token CRC5 failed)
- 0x04 frm_nat (R)
- SOF-captured snapshot: {mfm_cnt[3:0], 1'b0, frame_no[10:0], 4'h0, sof_time[11:0]}
- 0x05 UTMI vendor port
- Read: {24b0, utmi_vend_stat[7:0]} (latched copy of PHY vendor status)
- Write: utmi_vend_ctrl[3:0] <= din[3:0] and a one-shot write strobe is generated across the clock domain
- 0x08..0x0F reserved (read as 0; writes ignored)

- Per-endpoint windows (n = 0..15), base = 0x10 + 4*n
- +0 CSR (R/W): Endpoint configuration/status (HW may auto-update some fields)
- uc_bsel[31:30]: buffer select/toggle
- uc_dpd[29:28]: data PID base/toggle
- ep_dir[27:26]: 01=IN, 10=OUT, 00=CTRL
- txfr_type[25:24]: transfer type policy (ISO/BULK/INT)
- ep_state[23:22]: enable/disable/STALL
- lrg_ok[17], sml_ok[16]: OUT size policy vs MPS
- dma_en[15]: DMA ring/wrap mode enable
- tr_fr[12:11]: HS microframe scheduling (ISO)
- max_pl_sz[10:0]: MaxPacketSize
- +1 INT (R/W; read-to-clear for status)
- Read returns sticky status; read clears it. Writes update the per-EP INT masks
- Sticky sources include: TIMEOUT, CRC16_ERR, UNEXPECTED_PID, BUF0_DONE, BUF1_DONE, SEQ_ERR; OUT_TO_SMALL may be present per build
- +2 BUF0 (R/W): Buffer descriptor 0 (address/size; 0xFFFF_FFFF denotes not allocated)
- +3 BUF1 (R/W): Buffer descriptor 1 (address/size; 0xFFFF_FFFF denotes not allocated)
- Build note: EP0..EP3 are implemented; EP4..EP15 are inert/dummy (reads return 0; writes ignored)

- Interrupt aggregation
- inta_o/intb_o = OR of all per-EP interrupts combined with RF-level sticky events masked by 0x02. int_srca is a live summary; int_srcb is sticky and clears on read of 0x03.

- Behavioral notes
- main_csr write only uses bit[5] to request remote wake; request is ignored unless in SUSPEND and minimum suspend time elapsed; pulse self-clears after being consumed
- Per-EP INT sticky bits clear only by reading the EP’s INT register; reading 0x03 does not affect per-EP stickies
- CTRL EP buffer rule: OUT/SETUP use BUF0; IN uses BUF1 regardless of uc_bsel

- Many status fields (line_stat, usb_attached, mode_hs, frm_nat, utmi_vend_stat) originate in the PHY clock domain and are synchronized/latched before bus exposure

- Reset defaults
- funct_adr=0x00; inta_msk=intb_msk=0; RF sticky int_srcb=0
- Per-EP: masks cleared; BUF0/BUF1=0xFFFF_FFFF; CSR fields reset per endpoint block defaults

-- Interrupt Masks and Status --

Interrupt masks and status are split between a global RF (register-file) bank and per-endpoint INT registers. Global RF masks/status (addr[17]=0, RF window): - 0x02 Interrupt masks (R/W): {7'b0, intb_msk[8:0], 7'b0, inta_msk[8:0]} • Reset: all 0 (RF events do not raise interrupts until enabled). • Purpose: gate only RF-level sticky events into the top-level lines; no effect on per-endpoint interrupts. - 0x03 Interrupt sources (R, read-to-clear for RF stickies): {3'b0, int_srcb[8:0], 4'b0, int_srca[15:0]} • int_srca[15:0]: live per-endpoint summary; bit N = (epN_inta | epN_intb). Not sticky; reads have no side effect. • int_srcb[8:0]: sticky RF-level events, all cleared by a read of 0x03; masks at 0x02 do not clear them. · [8] usb_reset (from UTMI line-state FSM entering RESET) · [7] rx_err (UTMI RxError observed) · [6] deattach (usb_attached falling edge) · [5] attach (usb_attached rising edge) · [4] suspend_end (exit from suspend, e.g., resume or remote-wakeup) · [3] suspend_start (entry to suspend) · [2] nse_err (token addressed to function but no matching endpoint) · [1] pid_cs_err (PID complement check error) · [0] crc5_err (token CRC5 error) Per-endpoint INT register (per EP window at base = 0x10 + 4*EP): - Offset +1 (R/W): read returns {2'b0, iena[5:0], 2'b0, ienb[5:0], 9'b0, int_stat[6:0]}; write updates masks only (iena <= din[29:24], ienb <= din[21:16]). - int_stat[6:0] are sticky, read-to-clear, set only when this EP matches: • [6] out_to_small • [5] seqerr (sequence error on ISO OUT) • [4] buf1_done • [3] buf0_done • [2] unexpected PID • [1] CRC16 error • [0] timeout - Masking to outputs (per EP): • inta_ep = OR over i of (int_stat[i] & iena[mapping(i)]) • intb_ep = OR over i of (int_stat[i] & ienb[mapping(i)]) • Mapping quirk: int_stat[4] (buf1_done) uses enable index 3 (same as buf0_done) in both A and B. - Reset: iena=0, ienb=0, int_stat=0 (no per-EP interrupts until enabled). Top-level interrupt formation (registered in clk_i domain): - inta_o = (OR of all epN_inta) | (|(int_srcb & inta_msk)) - intb_o = (OR of all epN_intb) | (|(int_srcb & intb_msk)) Notes and semantics: - Separation of masking: RF masks (0x02) gate only RF stickies (int_srcb); per-EP masks (iena/ienb) gate only per-EP causes. - Clearing behavior: reading 0x03 clears all RF stickies (int_srcb) but never affects per-EP int_stat or the live int_srca; reading EPx INT clears only that endpoint’s int_stat and never affects RF stickies. - Sources and CDC: RF stickies originate in the PHY/protocol domain (usb_reset/attach/detach/suspend via UTMI LS, rx_err via UTMI IF, nse_err via PE, crc5_err & pid_cs_err via PD) and are synchronized and latched sticky in the bus/Wishbone clock domain. Per-EP cause pulses are latched as stickies in the endpoint block and masked there. - Software service model: on inta_o/intb_o assertion, read 0x03 to see RF stickies (clears int_srcb) and which EPs need service (int_srca); then read EPx INT for each indicated EP to obtain and clear detailed per-EP causes; program 0x02 (RF masks) and EP iena/ienb as needed.

-- Per-Endpoint Windows (CSR, INT, BUF0, BUF1) --

Overview
- Each endpoint exposes a 4-word window in the RF (register-file) region: CSR, INT, BUF0, BUF1 (all 32-bit).
- Up to 16 endpoints (EP0..EP15) are decoded; this build enables EP0..EP3. Unused windows are reserved (typically read 0 for CSR/INT and 0xFFFF_FFFF for BUF0/BUF1; writes ignored).

Addressing (inside RF region)
- RF region select: wb_addr_i[17] = 0.
- Word offsets: EPn_base = 0x10 + 4*n

- EPn.CSR = base + 0
- EPn.INT = base + 1
- EPn.BUF0 = base + 2
- EPn.BUF1 = base + 3
- Internal decode (usbf_rf): adr[6:2] selects EP index; adr[1:0] selects CSR/INT/BUF0/BUF1.

CSR (per-endpoint control/status)
- Programs endpoint identity and policy consumed by the protocol engine; read returns the current image including in-band hardware updates to low fields.
- Representative fields (exact bit positions are implementation-defined in usbf_ep_rf):
- uc_bsel[1:0] (buffer-select base/toggle) and uc_dpd[1:0] (DATA PID base/toggle); hardware may commit updates via uc_bsel_set/uc_dpd_set after transfers.
- Direction/type (IN/OUT/CTRL), endpoint number, transfer type (iso/bulk/etc.), enable/disable/halt.
- Policy: small/large OK vs Max Packet Size (MPS), MPS value (bytes).
- dma_en for DMA/ring mode (CTRL excluded).
- Software writes program CSR fields; uc_bsel/uc_dpd are modified only by in-band commits.

INT (per-endpoint sticky status and enables)
- Read returns sticky status and per-line enables; reading EPn.INT clears the sticky status for that endpoint (read-to-clear).
- Status events (set by protocol/endpoint logic after a valid token selection): BUF0-done, BUF1-done, CRC16 error, timeout, sequence error, unexpected PID, and HS OUT short (out_to_small). Exact bit mapping is defined in usbf_ep_rf (int_stat[6:0]).
- Writes to EPn.INT program per-endpoint interrupt enables (A-line iena[5:0] and B-line ienb[5:0]); writes do not set/clear status.
- Per-EP inta/intb = OR(status & enables); usbf_rf aggregates all endpoints and adds RF-level events to drive top-level inta_o/intb_o.

BUF0/BUF1 (buffer descriptors)
- Hold byte address into external SRAM and a size/capacity field; software initializes these prior to arming an endpoint.
- Sentinel/NA: unallocated buffers use 0xFFFF_FFFF; NA may also be inferred by a dedicated NA bit and/or all-ones address (implementation-defined).
- Software write to BUF0 also captures buf0_orig (capacity reference used for DMA accounting); reset default for BUF0/BUF1 is 0xFFFF_FFFF.
- Hardware write-back on transfer completion (only for the matched endpoint):
- usbf_pe asserts buf0_set/buf1_set and writes a packed result (idin) with updated remaining size, next address, and a done flag.
- Special HS short OUT (out_to_small): writes back the received size without advancing address and sets the corresponding INT bit.
- After descriptor update, CSR low fields (uc_bsel/uc_dpd) may be atomically committed via uc_*_set.
- In DMA mode, BUF0 may also see buf0_rl (release) on completion per ownership convention.

Selection, attribution, and flow
- Endpoint windows are consulted/updated only after a valid TOKEN: token_valid && !CRC5 error and address/endpoint match.
- DATA CRC16 failures set the per-EP CRC16 INT bit; by themselves they do not modify BUF descriptors.
- Double buffering: non-DMA uses uc_bsel preference with automatic skip of NA buffers; completion may toggle selection per policy. CTRL uses fixed buffers (OUT/SETUP->BUF0, IN->BUF1). DMA mode uses BUF0.

Access and timing
- RF accesses via usbf_wb ACK on a fast path (single transaction); data is synchronous to wb_clk. RF accesses are independent of the memory region.
- BUF addresses point into external SRAM arbitrated by usbf_mem_arb; core/DMA traffic has priority over Wishbone memory accesses.

Reset/disable
- On reset: per-EP INT status and enables clear; CSR and ots_stop clear; BUF0/BUF1 and buf0_orig initialize to 0xFFFF_FFFF.
- Windows for disabled/unimplemented endpoints are reserved and should not be used.

Software guidance
- Program BUF0/BUF1 with valid byte addresses and sizes; avoid modifying descriptors while a transfer is active. Use per-EP INT BUFx-done to determine ownership handoff.
- For DMA ring mode, note wrap targets absolute address 0 in this design; only enable if the ring starts at address 0.
- Tokens failing CRC5 do not update per-EP windows or set per-EP INT; software may observe such errors only via RF-level status (if exposed).

-- Vendor/PHY Access Registers --

- Purpose
- Software access to UTMI PHY vendor sideband plus link status/control:
- Read UTMI VStatus[7:0]
- Write UTMI VControl[3:0] with an automatic one-shot Load strobe
- Issue a device-initiated resume (remote-wake) request

- Addressing (Wishbone Register File, RF)
- Region select: wb_addr_i[17] = 0 selects RF (wb_addr_i[17] = 1 is MEM and unrelated)
- RF global offsets use adr[6:0]
- 0x05: UTMI vendor status/control (R/W)
- 0x00: main_csr (RO status; WO pulse on bit[5])

- Registers
- 0x05 UTMI vendor status/control
- Read: [7:0] = snapshot of UTMI VStatus[7:0]; [31:8] = 0; no side effects
- Write: [3:0] -> UTMI VControl[3:0]; generates exactly one VControl_Load strobe per write; [31:4] ignored
- 0x00 main_csr (link/PHY state; RO with one WO bit)
- RO fields: [0] usb_suspend, [1] mode_hs, [2] usb_attached, [4:3] line_state[1:0]; [31:5],[2:0] otherwise 0
- WO pulse: writing bit[5]=1 asserts a remote-wake (resume) request; all other bits ignored on write

- UTMI signal mapping (top level)
- Read path: VStatus_pad_i[7:0] -> 0x05[7:0] (snapshot)
- Write path: 0x05[3:0] -> VControl_pad_o[3:0]; write emits VControl_Load_pad_o one-shot
- Remote-wake: 0x00[5] write drives resume_req into the UTMI/link controller

- Access semantics
- 32-bit word accesses; use full-word reads/writes (partial-byte semantics undefined)
- 0x05 reads are side-effect free; each 0x05 write updates VControl and asserts one Load strobe
- 0x00 reads are side-effect free; 0x00[5] acts as a write-one pulse only

- Reset behavior
- VControl defaults to 0; VControl_Load deasserted
- VStatus readback upper bits read 0
- main_csr RO fields reflect live link state after reset completes

- Clock-domain and timing
- VStatus is sampled into the bus clock domain for readback (snapshot semantics)
- VControl_Load and resume_req pulses are synchronized into the PHY/link clock domain; pulse width handled in hardware
- Remote-wake is honored only when USB suspend timing allows; the UTMI/link controller enforces resume/suspend/reset timing

- Ownership and compliance
- UTMI control (XcvSelect, TermSel, OpMode, SuspendM) is owned by the UTMI controller FSM; vendor registers do not override these to avoid USB spec violations
- VControl/VStatus bit meanings are PHY-vendor specific; the core passes them through without interpretation

- Usage notes
- To change PHY vendor control: write 0x05 with desired value in [3:0]; hardware strobes Load automatically
- To request remote wake: write 0x00 with bit[5]=1 when allowed by USB 2.0 rules (hardware may defer until legal)

-- Resume/Wakeup Control --

Resume/Wakeup Control

Ownership and scope
- The UTMI link-state controller (usbf_utmi_if/usbf_utmi_ls) implements suspend detection, host-initiated resume, and device-initiated remote-wakeup. The top/regfile provide request latching and software visibility; other blocks (pd/crc/idma/pe/mem_arb/wb) are agnostic.

Suspend entry/exit
- Detection: Qualified UTMI LineState idle triggers suspend in usbf_utmi_ls; live state exported as usb_suspend.
- Software visibility: main_csr.suspend (RF 0x00) and susp_o (clk_i output) reflect usb_suspend; line_state and mode_hs are also in main_csr.
- Sticky RF events (read-to-clear at 0x03): suspend_start (int_srcb[3]) and suspend_end (int_srcb[4]); maskable to inta_o/intb_o via 0x02.
- A one-shot suspend_clr pulse is generated on suspend exit (phy_clk) and retimed to clk_i to clear any latched resume request.

Host-initiated resume
- Recognition: In suspend, a long K on the bus is detected by usbf_utmi_ls.
- Actions: Exit suspend, sequence UTMI controls back to NORMAL, observe a settle interval (~100 µs), and restore FS/HS operating mode as appropriate.

Device-initiated remote-wakeup
- Request sources:
• External pin: resume_req_i (clk_i domain).

• Software: write RF 0x00 with bit[5]=1 (rf_resume_req one-shot into clk/phy domain).
- Latching/clearing: Top level latches OR(resume_req_i, rf_resume_req) as resume_req_r until serviced; automatically cleared by suspend_clr on suspend exit to guarantee one action per request.
- Eligibility and timing (enforced in usbf_utmi_ls):
• Honored only while suspended and after a minimum suspend dwell >= 5 ms (configurable; some builds may use ~50 ms).
• Pre-drive delay ~5 ms, then drive K for ~1 ms, then settle ~100 µs before returning to NORMAL.
- UTMI signaling during remote wake:
• OpMode set to special signaling during drive (disable NRZI/bit-stuff), then back to normal; XcvSelect/TermSel set per FS/HS state.
• drive_k asserts to present K; TX path forces DataOut=00h and a bounded TxValid pulse even if TxReady is low, ensuring visible signaling.
• If previously HS, HS selection is re-applied per negotiation after resume/settle.

UTMI control pins
- SuspendM, OpMode, XcvSelect, and TermSel are driven by usbf_utmi_ls. SuspendM follows the UTMI definition (active-high "not suspended"); verify PHY polarity/inversion in integration.

Software/programming model
- Issue remote wake: (1) Poll main_csr.suspend==1. (2) Write RF 0x00 bit[5]=1. (3) Observe main_csr.suspend==0 and/or sticky suspend_end at 0x03; clear sticky events by reading 0x03. Multiple writes generate pulses but are ignored unless link is suspended.
- Diagnostics: frm_nat (0x04) provides frame/microframe timing useful for measuring wake latency.

System/data-path interactions
- No dedicated suspend/resume logic in pd/crc/idma/pe/mem_arb/wb. During suspend/resume, no valid tokens/data are presented, so cores idle naturally; in-flight transfers may hit normal protocol timeouts (e.g., ACK/data timeouts) and return to IDLE.
- Recommendation: Higher layers assert DMA abort around suspend entry and keep DMA quiesced until resume settle completes; arbitration and Wishbone front-end behavior are unchanged by power-state transitions.

Compliance
- Timings satisfy USB 2.0: minimum suspend dwell >= 5 ms (meets spec minimum), K drive ~1 ms (within 1–15 ms), settle ~100 µs. Exact constants reside in usbf_utmi_ls.

-- Memory Access Window and Alignment --

Memory window selection and size
- Wishbone address space is split into Register File (RF) and external SRAM. In this build, wb_addr_i[17]=1 selects the external memory window; wb_addr_i[17]=0 selects the RF window.
- External SRAM port is 32-bit and word-addressed by sram_adr[SSRAM_HADR:0]; lower 2 byte address bits are ignored. Keep wb_addr_i[1:0]=2'b00 for memory accesses.
- Total SRAM capacity in bytes = 2^(SSRAM_HADR+1) × 4. Example: SSRAM_HADR=14 → 512 KiB.

Access granularity, alignment, and endianness
- No byte-enable lanes on the SRAM port; all writes are full 32-bit words. Software byte/half-word updates must be done via read–modify–write (RMW).
- Natural ordering within 32-bit words: increasing byte addresses map to [7:0],[15:8],[23:16],[31:24]; no lane swapping.
- RF window accesses are also 32-bit word-aligned; treat reads/writes as full words (e.g., INT is read-to-clear on a 32-bit read).

USB DMA alignment behavior (internal)
- The USB data path (usbf_pl/usbf_idma) accepts byte-granular buffer addresses/sizes and maps them to the word-aligned SRAM:
• RX: supports any start alignment via seed read + RMW of the first word; writes full words as 4 bytes accumulate; flushes a final partial word at packet end.
• TX: streams bytes from a two-word ping–pong prefetch buffer; byte lanes follow natural order.
• Address mapping: adr[SSRAM_HADR+2:2] → word address; adr[2:0] is the byte index. Optional ring mode wraps at adr+buf_size when dma_en=1.
- Functional for any byte alignment; for best performance, align buffer starts to 32-bit boundaries to avoid initial RMW and final partial-word flush.

Arbitration and timing
- Single-port SRAM shared by core/DMA master (M) and Wishbone/software master (W) via usbf_mem_arb. Fixed priority: M always wins; W is served only when M is idle.
- M side: mack is combinational (mack=mreq); arbiter inserts no wait for M.
- W side: when served, wack pulses at most every other phy_clk (best-case one transfer per two phy_clk). W can be delayed or starved during heavy USB traffic.
- Read data is broadcast; each master must sample only when its acknowledge is asserted. sram_re is tied high. No bursts; one outstanding transfer at a time.

Wishbone front-end and CDC
- usbf_wb samples bus requests into phy_clk and returns a one-cycle wb_ack_o pulse when RF completes immediately or when memory ma_ack arrives from the arbiter.
- Best practice is wb_clk=phy_clk or ensure CDC timing closure.

Coherency and software guidance
- The memory window exposes raw endpoint buffers. Software must follow per-endpoint ownership in the RF descriptors: read only from buffers released by the core; write only to buffers not owned by DMA.
- For software accesses: perform 32-bit reads/writes; use RMW for sub-word updates; keep wb_addr_i[1:0]=00; prefer 32-bit aligned buffer bases and schedule memory accesses during idle USB periods to minimize arbiter-induced latency.

-- Reset Values and Access Types --

- Reset style and domains
- Default: synchronous resets in each clock domain. Top-level rst_i is active-high; most submodules use active-low rst internally (if (!rst) ...). If USBF_ASYNC_RESET is defined, selected UTMI datapath flops use async reset. Two main clocks: phy_clk (PHY/core) and clk_i/wclk (Wishbone/RF side).

- Software-visible register map (RF region: wb_addr_i[17]==0)
- 0x00 main_csr: RO live status {line_state[1:0], usb_attached, mode_hs, suspend}. Reset: suspend=0, mode_hs=0, usb_attached=0; line_state undefined until first UTMI sample. Special write: bit5 is W1P rf_resume_req (pulse only; resets idle 0). Other writes ignored.
- 0x01 function address: RW [6:0]. Reset 0x00.
- 0x02 interrupt masks: RW inta_msk[8:0] at [8:0], intb_msk[8:0] at [24:16]. Reset 0x00000000.
- 0x03 interrupt sources: int_srca[15:0] RO live (not sticky), int_srcb[8:0] sticky RC at [11:3]. Reset: int_srcb=0; int_srca reflects live EP A/B OR (0 at reset if idle). Read clears only int_srcb.
- 0x04 frm_nat: RO timing snapshot. Reset 0x00000000; updates on SOF.
- 0x05 UTMI vendor port: Read: RO latched utmi_vend_stat[7:0], no explicit reset (undefined until captured). Write: RW lower 4 bits drive utmi_vend_ctrl[3:0] and assert W1P utmi_vend_wr; Reset:

utmi_vend_ctrl=0, pulse idle 0.

- Per-endpoint windows (base 0x10 + 4*EP, EP=0..15)
- +0 CSR: RW with mixed RO fields. Reset 0x00000000. Notes: uc_bsel[1:0] and uc_dpd[1:0] are RO to SW (HW-updated via uc_*_set); bit14 reads 0, writes ignored.
- +1 INT: RW for masks, RC for status. Read data: {.. iena[5:0] .. ienb[5:0] .. int_stat[6:0]}. Reset: iena=0, ienb=0, int_stat=0. Read clears int_stat only.
- +2 BUF0: RW. Reset 0xFFFF_FFFF (sentinel). SW write also updates buf0_orig (capacity mirror). HW updates when EP matches: buf0_set writes idin; buf0_rl reloads from buf0_orig. SW write has priority if coincident.
- +3 BUF1: RW. Reset 0xFFFF_FFFF. HW updates when EP matches: buf1_set or out_to_small write idin. SW write has priority if coincident.

- External Memory Array window (MA region: wb_addr_i[17]==1)
- 32-bit synchronous SRAM aperture. Access RW. Reset contents undefined (device/board dependent). Arbitration gives core (M) priority over Wishbone (W).

- Interrupt outputs (top-level aggregation)
- inta_o/intb_o: RO outputs aggregated from per-EP A/B and RF sticky events. Reset deasserted (0) because masks/status reset to 0. Per-EP int_src bits: masks RW, status RC as above.

- DMA handshakes (per-EP via usbf_rf)
- dma_req_o[15:0]: RO outputs, reset 0. dma_ack_i[15:0]: inputs. No software-visible registers; behavior driven by EP CSR/BUF and HW.

- UTMI/link interface (hardware-only, not memory-mapped)
- usbf_utmi_if: rx_valid/active/err reset 0; rx_data/tx_ready/drive_k_r have no explicit reset (undefined until first valid sample). TxValid resets 0; DataOut driven only when valid or forced 0x00 during drive_k.
- usbf_utmi_ls: forces FS defaults on POR (on rst==0 or usb_vbus==1): XcvSelect=FS, TermSel=1, mode_hs=0, usb_suspend=0, usb_attached=0; OpMode has no defined reset until first FSM write. Many timers/counters have no explicit reset but are synchronously cleared by POR strobes. All signals are hardware-only; SW reads live reflections via main_csr.

- Wishbone front-end (usbf_wb)
- No software-visible storage. FSM state resets to IDLE (active-low rst). ACK pipeline and read-data mux flops have no explicit reset (self-initialize on first transaction). Regions: wb_addr_i[17]==0→RF, ==1→MA.

- Memory arbiter (usbf_mem_arb)
- No software-visible storage. wack_r resets 0; all other paths combinational. sram_re tied high.

- Protocol/endpoint engines and DMA (hardware-only)
- usbf_pe: main FSM and direction latches reset to IDLE/0; most other flops no explicit reset but default low/idle under FSM gating. Exports pulses for descriptor updates and interrupts; no direct SW registers.
- usbf_idma: key flops reset (state=IDLE, send_data_r=0, sizd_c=14'h3fff, sizu_c=0, adr_cb=0); many data/handshake buffers no explicit reset but only observed under valid enables. No SW-visible registers.
- usbf_pd (packet decode): pid resets to 8'hf0 (pid_RES=1); crc16_sum functionally cleared on rx_active rise (not by rst). Token/data pipelines have no explicit reset but are qualified by valid strobes. No SW-visible registers.
- usbf_crc5/usbf_crc16: purely combinational; no reset or SW access.

- Dummy endpoint (usbf_ep_rf_dummy)

- For any EP implemented as dummy: all reads return 0x00000000; writes ignored. Core-side live constants: csr=0, buf0=buf1=0xFFFFFFFF, ep_match=0, interrupts=0, dma_req=0. Reset has no effect (no state).

- Registers/signals without explicit reset
- UTMI line/vendor status latches, various pipelines (ack sync, token/data stages), and some counters/flags come up undefined and are made valid by first use or FSM gating. Software must rely on documented valid strobes and RO live fields; undefined values are not exposed as sticky state.

## INTERFACE SPECIFICATIONS

-- Clock and Reset Interfaces --

- Primary clock domains
- phy_clk_pad_i (UTMI/data-path domain): Drives usbf_utmi_if (incl. usbf_utmi_ls), usbf_pl (usbf_pd/usbf_pe/usbf_pa/usbf_idma), usbf_mem_arb, and the core-view side of usbf_rf. Module ports named clk in these blocks map to phy_clk_pad_i.
- clk_i (Wishbone/bus domain): Drives usbf_wb.wb_clk, the bus-view side of usbf_rf (wclk), and selected top-level status flops. Ports named wclk map to clk_i. Recommendation: tie clk_i == phy_clk_pad_i to avoid CDC; if not, add synchronizers as noted below.

- Resets and polarity
- rst_i: Single, active-low reset distributed into both domains. Default is synchronous to the receiving clock; a project option (USBF_ASYNC_RESET) may allow asynchronous assertion (deassertion must remain synchronous per domain).
- phy_rst_pad_o: Mirrors rst_i (same polarity) to reset the external UTMI PHY; adapt externally if the PHY requires different polarity/timing.

- Module-level clock/reset behavior
- usbf_utmi_if/usbf_utmi_ls (phy_clk_pad_i): Synchronous, active-low reset. utmi_ls FSM enters POR on rst=0; many control flops initialize via POR-state strobes after reset release. usb_vbus synchronously forces the FSM to POR while asserted.
- usbf_pl complex (pd/pe/pa/idma, phy_clk_pad_i): Synchronous, active-low reset. Main FSMs go IDLE; several data/qualifier registers are not explicitly reset and rely on IDLE defaults with quiescent inputs at release.
- usbf_crc5/usbf_crc16: Purely combinational; no clock/reset.
- usbf_mem_arb (phy_clk_pad_i): Synchronous, active-low reset clears wack_r; all arbitration/muxing is combinational. Port wclk is unused.
- usbf_rf/usbf_ep_rf (clk_i and phy_clk_pad_i): Active-low resets in both domains. clk-domain clears CSRs, buffers, interrupt status, and DMA counters; wclk-domain clears DMA request/ack capture and registers interrupts. A two-flop sync is used for dma_ack back into clk; request and bus strobes assume clk==wclk unless externally synchronized.
- usbf_ep_rf_dummy: No sequential logic; clk/wclk/rst are unused.
- usbf_wb (wb_clk=clk_i, phy_clk=phy_clk_pad_i): FSM in phy_clk has synchronous, active-low reset to IDLE. wb_clk-side ACK pipeline and wb_data_o are not explicitly reset and depend on requests being idle during reset.

- Clock-domain crossings (CDC)
- Wishbone requests: wb_stb_i&wb;_cyc_i are sampled directly in phy_clk without a synchronizer; safe when clk_i == phy_clk_pad_i or phase-related with timing closure. Otherwise add a request pulse synchronizer.
- Wishbone ACK: phy_clk wb_ack_d is returned to wb_clk via a 3-FF edge detector to generate single-cycle wb_ack_o.
- Register file (usbf_rf): Uses sticky/edge-detected bits and one-shots to bridge clk_i↔phy_clk for selected events and vendor UTMI sideband. Bus strobes (adr/re/we) are consumed in the clk/core domain and assume clk==wclk unless synchronized.
- Top-level status/control: usb_suspend and suspend_clr (phy_clk) are sampled into clk_i with simple flops; resume_req (clk_i) is two-flop synchronized inside utmi_ls (phy_clk).

- Integration and sequencing guidance
- Provide stable, glitch-free phy_clk_pad_i and clk_i. Hold rst_i low until both clocks are stable and the PHY clock is present.
- Keep Wishbone CYC/STB low during reset and for a few cycles after; keep protocol/DMA/memory handshakes idle at and just after reset so unreset flops settle deterministically.
- If USBF_ASYNC_RESET is enabled, assertion may be asynchronous; ensure deassertion is synchronized to each active clock.
- If clk_i and phy_clk_pad_i are asynchronous, add explicit synchronizers for: Wishbone request into phy_clk, rf bus strobes into the clk/core domain, and any top-level status/control crossings not internally synchronized. Prefer tying clk_i == phy_clk_pad_i in minimal-risk integrations.
- No derived external clocks are generated; internal timers/dividers (e.g., in utmi_ls) are synchronous to their local clk.

-- Wishbone Slave Interface --

Protocol and signals: Wishbone B.3 classic, single-beat slave. Ports: clk_i (wb_clk), rst_i (active-low inside slave), wb_addr_i[N:0], wb_data_i[31:0], wb_data_o[31:0], wb_we_i, wb_stb_i, wb_cyc_i, wb_ack_o. Not supported: SEL_I, ERR_O, RTY_O, STALL, CTI/BTE, LOCK, tags. Data width and alignment: 32-bit only; word-aligned addressing (ADR[..:2]); no partial/byte writes. Address map (this build): wb_addr_i[17]=0 -> Register File (RF via usbf_rf); wb_addr_i[17]=1 -> external SRAM Memory (MA via usbf_mem_arb W-port). Select bit is parameterizable (USBF_UFC_HADR). Handshake/operation: One outstanding transfer at a time; requests sampled only in IDLE. Reads return data and writes commit on the cycle wb_ack_o is asserted. Master must deassert STB/CYC after ACK to avoid unintended repeats. ACK timing: wb_ack_o is exactly one clk_i cycle per transaction. RF region: immediate fast-path ACK after a short phy_clk pipeline; then two settle states (W1/W2) before a new request is accepted. MA region: ACK when usbf_mem_arb asserts wack; latency varies with arbitration and external SRAM timing. Arbitration/throughput (MA): Single-port SRAM arbitrated between M (core/DMA) and W (Wishbone) with fixed priority to M. When M is idle and W is served, W ACKs can occur every other phy_clk (~1 transfer per 2 phy_clk). While DMA is active, W may be deferred/starved; protocol semantics unchanged. Clocking/CDC: Transaction FSM and ACK generation reside in phy_clk (phy_clk_pad_i). Requests (wb_stb_i & wb_cyc_i) are sampled into phy_clk without a multi-FF synchronizer; ACK is generated in phy_clk and re-timed to wb_clk with a small synchronizer/edge detector to produce a single-cycle wb_ack_o pulse. Best used when clk_i == phy_clk or otherwise phase-related/timing-closed. Data path: wb_data_o is registered in wb_clk; RF reads return rf_din; MA reads return ma_din on the ACK cycle. Reset: internal FSM uses active-low reset; Wishbone-visible outputs are synchronous; some wb_clk-domain flops rely on power-on state but ACK is self-clearing. Software-visible RF highlights: global regs at 0x00..0x05 (e.g., device addr, interrupt masks/sources with read-to-clear where noted, SOF timing, vendor UTMI), per-endpoint windows at 0x10 + 4*EP {CSR, INT (read-to-clear stickies), BUF0, BUF1}. Guidance: perform only 32-bit aligned accesses; rely solely on ACK for completion; expect variable MA latency under USB traffic; deassert STB/CYC after ACK.

-- UTMI PHY Data and Control Interface --

- Purpose and scope
- Terminates a UTMI+ Level 3 HS/FS 8-bit PHY and provides clean, clocked data, control, and status to the USB core. Implemented by usbf_utmi_if (data path) with usbf_utmi_ls (line state/link FSM), entirely in the phy_clk_pad_i domain.

- Clocks and reset
- phy_clk_pad_i: UTMI data/control clock for utmi_if, utmi_ls, and core-side consumers.
- phy_rst_pad_o: driven from rst_i; synchronous reset unless USBF_ASYNC_RESET is enabled.
- Software/RF domain is separate (clk_i or wclk); status is re-timed across domains.

- UTMI pins and sideband
- Data: DataIn[7:0], RxValid, RxActive, RxError, DataOut[7:0], TxValid, TxReady.
- Line control: XcvSelect, TermSel, OpMode[1:0], SuspendM, LineState[1:0], usb_vbus.
- Vendor sideband: VStatus[7:0] readback; VControl[3:0] with VControl_Load strobe on write.

- Receive path (PHY to core)
- RxValid, RxActive, RxError, DataIn are registered every phy_clk and forwarded as single-cycle, reset-clean signals.
- Protocol decode (usbf_pd) uses these to latch PID, extract TOKEN fields (ADDR, ENDP, FrameNo), and stream DATA payload via a fixed two-byte pipeline: rx_data_st[7:0], rx_data_valid, rx_data_done.
- CRC checks: TOKEN CRC5 over {ADDR[6:0], ENDP[3:0]} with seed 0x1F, LSB-first; compare to ones' complement. DATA CRC16 seed 0xFFFF, LSB-first; residue must equal 0x800D at end of packet. seq_err is raised only for protocol anomalies when the PHY did not assert RxError.

- Transmit path (core to PHY)
- TxReady is registered back to the core as tx_ready.
- DataOut SOP preload when (TxReady || tx_first) ensures first-byte timing. TxValid is held until TxReady; tx_valid_last guarantees final byte acceptance.
- Special signaling windows (resume and HS chirp) assert drive_k: OpMode set to 2'b10, DataOut forced to 8'h00, TxValid emits a bounded pulse independent of TxReady. Representative update: TxValid_next = tx_valid | drive_k | tx_valid_last | (TxValid & !(TxReady | drive_k_r)).

- Line state and link management (usbf_utmi_ls)
- LineState decoded with two-cycle qualifiers into J, K, SE0, SE1; idle_long gates timers.
- Link FSM handles: attach debounce (~100 ms, gated by usb_vbus), bus reset detection in SE0, HS negotiation via FS chirps (enter HS on 6 alternating host K/J), suspend entry on long idle, and resume sequencing (host or device-initiated).
- Drives UTMI control pins: XcvSelect (HS/FS select), TermSel (FS termination), OpMode (00 normal, 10 special signaling), SuspendM (active high = not suspended).
- Status outputs: mode_hs, usb_reset, usb_suspend, usb_attached, suspend_clr (one-shot on suspend exit).

- Remote wake (device-initiated resume)
- Software sets rf_resume_req via RF; a pulse is latched in the core and consumed by utmi_ls.
- Enforces timing: minimum ~5 ms in suspend before driving K for ~1 ms; settle ~100 µs before returning to normal operation. Request auto-clears on suspend exit.

- Software-visible status and control
- main_csr exposes {suspend, mode_hs, usb_attached, line_stat[1:0]}.

- frm_nat provides SOF or microframe timing snapshot.
- Vendor sideband register at RF addr 0x05: read VStatus, write VControl[3:0] with a one-shot strobe into the UTMI domain.
- RF sticky events aggregate UTMI/link conditions (e.g., usb_reset, attach/deattach, suspend start/end, rx_err, pid_cs_err, crc5_err) with maskable interrupt routing.

- Domain crossings
- All UTMI data/control remain in phy_clk. LineState and vendor status are captured in phy_clk and re-timed to the software domain. Software-originated pulses (rf_resume_req, utmi_vend_wr) cross into the core domain via one-shots.

- Integration with core datapath
- Transmit byte pacing is aligned to TxReady via rd_next from the packetizer; receive payloads are presented as rx_data_valid/done. Memory movement and arbitration are outside this block but are designed to never stall UTMI flow.

- Compliance and notes
- UTMI conventions are LSB-first; PHY handles bit-stuffing, NRZI, SYNC/EOP.
- SuspendM polarity is active high (not suspended); verify with target PHY.
- Timing constants derive from ~250 ns ticks to meet USB 2.0 µs/ms windows.
- HS Test Modes (SetFeature TEST_MODE) are not implemented.

-- External SSRAM Interface --

External synchronous single-port SRAM provides the shared packet-buffer storage for all endpoints. It is accessed exclusively through usbf_mem_arb, which multiplexes two internal masters onto one 32-bit, word-addressed port in the phy_clk domain.

Pins and width (top level)
- sram_adr_o[SSRAM_HADR:0]: word address bus (SSRAM_HADR+1 bits)
- sram_data_o[31:0]: 32-bit write data
- sram_data_i[31:0]: 32-bit read data
- sram_we_o: write enable, active-high
- sram_re_o: read enable, held high continuously
- No byte-enable strobes; all external writes are full 32-bit words

Addressing and capacity
- Port uses word addressing; internal byte addresses are converted in the DMA bridge
- Effective capacity (bytes) = 4 × 2^(SSRAM_HADR+1); e.g., SSRAM_HADR=14 → 128 KiB

Arbitration and masters (inside usbf_mem_arb)
- M (protocol/DMA via usbf_pl/usbf_idma): absolute priority
- W (Wishbone/software via usbf_wb): best-effort when M is idle
- Read data from sram_data_i is broadcast; each master samples only when owning the bus and receiving its own acknowledge (internal to arbiter)

Handshakes and timing (internal to arbiter)
- M side: mack = mreq (no inserted wait states). DMA must align its sampling of sram_data_i to the memory’s address-to-data latency; the arbiter does not qualify data-valid beyond bus ownership
- W side: wack is a one-cycle pulse generated at most every other phy_clk when granted and while M is idle (max sustained W throughput = 1 transfer per 2 phy_clk cycles)

Read/write behavior
- sram_re_o is tied high; the memory device/wrapper must tolerate continuous read-enable or provide external CE/RE gating
- Unaligned/partial-byte RX writes use DMA read-modify-write on the first and last words; TX uses two-word prefetch buffering in the DMA

Clocking and CDC
- All SRAM-facing logic (arbiter, DMA, protocol) runs in phy_clk (typically 60 MHz HS, 12 MHz FS)
- usbf_wb captures WB requests into phy_clk and returns ACK in wb_clk; ensure wb_clk ≈ phy_clk or close CDC appropriately

Integration constraints
- Use synchronous SRAM/BRAM or a wrapper that meets phy_clk timing and presents read data with a fixed, known latency
- If the memory requires explicit CE/RE gating (for power or device protocol), add a small wrapper; keep sram_we_o synchronous and single-cycle for writes
- No byte strobes: software partial-word updates should be avoided; hardware partials are handled by the DMA RMW path
- Under heavy USB traffic, the M master can starve W accesses; plan software SSRAM accesses during idle/suspend/reset windows

-- DMA Handshake Interface --

Scope
- Defines the DMA handshakes used to service endpoint buffers: (1) the external per-endpoint system-facing handshake (dma_req_o/dma_ack_i) and (2) the internal memory-side handshakes (IDMA mreq/mack and Wishbone pacing) that influence service throughput.

External per-endpoint DMA handshake (system-facing)
- Ports and domain: dma_req_o[15:0] (output, clk_i), dma_ack_i[15:0] (input, clk_i). Active-high; one bit per endpoint (EP0..EP15). In this build, EP0..EP3 are enabled; other bits may be tied off by dummy endpoints.
- Purpose: Coordinate system-side movement of data to/from the external SSRAM backing endpoint buffers. No address/size information is encoded in the handshake.
- Semantics per endpoint:
- Request assertion (req): Generated by endpoint RF logic when service is needed.
• OUT (host→device): drain data present in the buffer.
• IN (device→host): prefill buffer until capacity policy is met.
• Level-sensitive; at most one outstanding stream of work per endpoint.
- Acknowledge (ack): System asserts one pulse per completed 32-bit word of service while req is high.
• Minimum width: ≥1 clk_i cycle.
• Spurious acks when req=0 are ignored.
- Clear/hold: Endpoint logic holds req high until internal counters indicate no further service is needed. Policy may hold req across acks to enable efficient bursts; if more work remains after an ack, req stays high and further acks are expected.
- CDC: At the top level both signals are in clk_i. Internally, dma_ack_i[n] is resynchronized into the PHY clock domain and converted to a single-cycle pulse that updates per-EP counters and commits descriptor/ownership changes; req originates in the endpoint/RF logic and is held until that internal pulse is seen.
- Software/system DMA contract:
- Discover work by reading dma_req_o (or RF status/interrupts).
- Read endpoint CSR/BUF0/BUF1 via the RF window to obtain buffer addresses/sizes and policy; the

handshake carries no metadata.
- Perform 32-bit word moves through the MEM window over Wishbone, proceeding only on wb_ack_o from usbf_wb.
- After each serviced 32-bit word, pulse dma_ack_i[n] for one clk_i cycle. Continue until req deasserts or the intended service is complete.
- Do not assert ack when req=0; do not compress multiple words into one ack.
- Advisory flags (not part of the handshake): RF exports dma_in_buf_sz1 (enough IN data for ≥1 max packet) and dma_out_buf_avail (enough OUT space for ≥1 max packet) to the protocol engine; these reflect the effects of servicing but are not driven by the system.
- Reset/suspend and edges:
- On rst_i, requests are cleared; acks with req=0 are ignored.
- Handshake is independent of USB suspend/resume; requests reflect buffer state.
- No fairness across endpoints is provided; software must arbitrate among asserted req bits.
- Granularity is 32-bit words; align system transfers and ack pulses accordingly.

Internal memory-side handshakes (context, not the external req/ack)
- Protocol IDMA (core→SSRAM via mem_arb M port): mreq/mack (mack≈mreq combinational), mwe, madr, mdin/mdout; PHY clock domain. One request per memory beat; near-zero added latency from arbiter.
- System/Wishbone (W port via usbf_wb): wreq/wack at mem_arb with fixed M priority. When M is idle, wack pulses at most every other phy_clk; when M asserts mreq, W is immediately throttled. usbf_wb converts wack into a clean wb_ack_o in wb_clk. External DMA must tolerate stalls and proceed strictly on wb_ack_o.

Disabled endpoints
- Dummy endpoints drive their dma_req bit low permanently and ignore dma_ack. System DMA should ignore those bits.

Recommended servicing flow
- Poll dma_req_o to find active endpoints.
- For each asserted bit: (1) read RF CSR/BUF descriptors; (2) move 32-bit words via the MEM window, honoring wb_ack_o; (3) after each word, pulse dma_ack_i[n] for one clk_i cycle; (4) write back descriptor updates if required by policy.

-- Interrupt Outputs --

Top-level interrupt outputs are inta_o and intb_o.
- Type/polarity/domain: Active-high, level; registered and synchronous to clk_i (Wishbone). Assertion/deassertion has 1–2 clk_i cycles latency due to synchronization.
- Aggregation model (per line):
• Endpoint group: OR of all endpoint usbf_ep_rf inta (for A) or intb (for B). Each endpoint holds sticky status bits and routes them to A/B via its INT register masks; clearing is by reading the endpoint INT register.
• RF (global) group: OR of enabled RF sticky events. Events latched sticky in RF and independently maskable onto A and/or B.
- RF sticky event set (bit indices): [8] usb_reset, [7] rx_err, [6] deattach, [5] attach, [4] suspend_end, [3] suspend_start, [2] nse_err, [1] pid_cs_err, [0] crc5_err.
- Software interface:
• RF 0x02 (mask): inta_msk[8:0], intb_msk[8:0] gate which RF sticky events contribute to inta_o/intb_o. Masks do not affect latching.
• RF 0x03 (sources): int_srca[15:0] = live per-EP summary (bit N = EP N’s (inta|intb)), read has no side-effects; int_srcb[8:0] = RF sticky bits, read-to-clear (reading 0x03 clears all RF sticky bits).

• Per-EP INT (0x10+4*N+1): read-to-clear per-EP sticky status; writable iena/ienb route EP events to A and/or B.
- Timing/domains: Endpoint/protocol/UTMI events originate in the PHY/protocol clock domain and are synchronized in usbf_rf into clk_i before contributing to inta_o/intb_o.
- Reset behavior: RF masks clear (no RF events routed), RF sticky bits clear, endpoint masks/flags clear, and inta_o/intb_o deassert.
- Notes: DMA request lines are not interrupts and do not affect inta_o/intb_o.

-- Power and Status Signals --

- Top-level power/link controls (UTMI side)
- rst_i: Core reset; forwarded to phy_rst_pad_o (resets the PHY). Reset is synchronous unless USBF_ASYNC_RESET is defined.
- phy_rst_pad_o: UTMI PHY reset = rst_i.
- UTMI control outputs: XcvSelect_pad_o (HS/FS Rx select), TermSel_pad_o (FS termination), SuspendM_pad_o (1 = not suspended; deasserted only during suspend), OpMode_pad_o[1:0] (00=normal; special modes during reset/resume/chirp).
- usb_vbus (in): VBUS presence; used to force POR/ATTACH entry and generate debounced attach status.

- Suspend, resume, and remote wake
- usb_suspend (phy clock domain): Indicates core is in suspend; mirrored to clk_i as susp_o (software-visible).
- susp_o: Registered mirror of usb_suspend in clk_i domain.
- resume_req_i (clk_i domain): Firmware-driven remote wake request (RF 0x00 write bit 5). Latched internally until cleared.
- suspend_clr (from UTMI/link FSM): One-shot on suspend exit; used to clear the latched resume request.
- Hardware enforces USB timing: idle detect (>3 ms), remote-wake eligibility (≥5 ms in suspend), resume K drive (~1 ms), attach debounce (~100 ms).

- Link state/status exports (live)
- mode_hs: 1=High-Speed, 0=Full-Speed.
- usb_reset: Asserted on bus reset detection/handling.
- usb_attached: Debounced attach state after VBUS/line debounce.
- LineState_r[1:0]: Registered UTMI LineState (phy domain), software-readable.
- VStatus_r[7:0]: Registered UTMI vendor status (phy domain), software-readable.

- Software-visible registers (RF via Wishbone)
- main_csr (addr 0x00 read): Packs suspend (usb_suspend), mode_hs, usb_attached, line_state[1:0].
- main_csr (addr 0x00 write): Bit 5 = remote-wake request (drives resume_req_i).
- frm_nat (addr 0x04 read): 32-bit SOF/microframe timestamp for diagnostics.
- utmi vendor port (addr 0x05): Read VStatus_r[7:0]; write vendor control [3:0] with strobe into phy domain.
- RF sticky events (addr 0x03 read/clear): usb_reset, attach/detach, suspend entry/exit, protocol error flags; contribute to top-level inta_o/intb_o via masks.

- Interrupts (aggregation and per-endpoint)
- Top-level inta_o/intb_o: OR of RF sticky events and all endpoint INT sources, masked per-bank.
- Per-endpoint (usbf_ep_rf) INT register: Sticky bits (read-to-clear) sourced from protocol/DMA events:
• out_to_small, seqerr, buf1_done, buf0_done, upid_unexpected, crc16_err, timeout.
- Per-endpoint inta/intb: Level = OR(int_stat & iena/ienb). usbf_rf also exposes a live per-EP summary

int_srca[15:0].

- Per-endpoint configuration/status (usbf_ep_rf)
- CSR (adr==0): Direction, endpoint number, DMA enable, max packet size, ots_stop, user fields uc_bsel/uc_dpd.
- INT (adr==1): Read returns {iena, ienb, int_stat}; read clears int_stat. Write updates enables.
- BUF0/BUF1 (adr==2/3): Buffer descriptor/status words (buf0_orig tracks last SW write).
- ep_match (clk): 1 when selected endpoint number matches; parent uses to mux live csr/buf0/buf1 upward.
- Live DMA readiness (clk): dma_in_buf_sz1 (IN data >= 1 MPS), dma_out_buf_avail (OUT space for ≥1 MPS).
- dma_req (wclk): Per-EP DMA service request to system DMA.
- Reset defaults: iena/ienb/int_stat=0; buf0/buf1/buf0_orig=0xffff_ffff.

- RX protocol/CRC status (usbf_pd, phy clock domain)
- pid_cks_err: Invalid PID nibble complement.
- token_valid: 1-cycle strobe on TOKEN complete or ACK PID; qualifies token fields and crc5_err.
- crc5_err: Token CRC5 mismatch (1-cycle at token_valid).
- rx_data_valid: Qualifies DATA payload bytes.
- rx_data_done: 1-cycle strobe at end of DATA payload.
- crc16_err: 1-cycle at rx_data_done when CRC16 residue != 16'h800d.
- seq_err: Protocol/sequence anomaly (suppressed on PHY rx_err).

- TX/RX DMA status (usbf_idma, phy clock domain)
- send_data: High while TX data available (or ZLP requested); deasserts after last byte.
- idma_done: RX path pulses on rx_data_done; TX path sticks high when size reaches zero until next tx start/reset.
- sizu_c[10:0]: Live RX byte count for current packet.

- Endpoint protocol engine status (usbf_pe, phy clock domain)
- Interrupt pulses (1-cycle): int_buf0_set, int_buf1_set, int_upid_set, int_crc16_set, int_to_set, int_seqerr_set (iso OUT only).
- Flags: nse_err (not-selected endpoint token), out_to_small (HS OUT short packet in DMA mode).
- Uses mode_hs to select timeout thresholds and enable HS-only features (PING/NYET).

- Memory arbitration and bus status (usbf_mem_arb)
- mack: Combinational accept for M master (mack=mreq).
- wack: One-cycle pulse for W master when M is idle (every other cycle if wreq held and M idle).
- mdout/wdout: Read data broadcast = sram_din; sample with respective acks.
- No software-visible status; sram_re tied high (external CE/CS should gate device power).

- Wishbone front-end (usbf_wb)
- wb_ack_o: 1-cycle pulse on completion; RF region acks immediately, memory region acks on ma_ack.
- wb_data_o: RF = rf_din (e.g., main_csr, frm_nat, VStatus, endpoint windows); MEM = ma_din.
- Carries all software-visible power/link status via RF; no UTMI power control generated here.

- Clock/reset domains and CDC
- phy_clk_pad_i: UTMI/link, protocol engines, DMA, memory arbiter, most status producers.
- clk_i (wclk): Wishbone/RF bus; mirrors usb_suspend as susp_o and exposes registers/interrupts.
- Key CDC paths: resume_req (clk_i→phy), suspend_clr (phy→clk_i), RF sticky/INT bits (phy→wclk), wb_req sampling into phy (caution if clocks unrelated), wb_ack back into wb_clk (safed with multi-FF edge detect).

- Polarity/implementation notes
- SuspendM is active-high for not-suspended per UTMI; verify target PHY polarity.
- Attach debounce, suspend/resume, HS negotiation, and reset/chirp timing handled in UTMI/link FSM; HS Test Modes not implemented.
- Modules usbf_crc5/usbf_crc16 and usbf_ep_rf_dummy produce no power or status signals.

-- Timing and Handshake Requirements --

- Clocks, resets, and CDC
- phy_clk domain: UTMI interface (usbf_utmi_if/ls), protocol (usbf_pl/usbf_pd/usbf_pe), IDMA, memory arbiter, RF endpoint cores. wb_clk domain: Wishbone slave and RF bus aggregation. Prefer wb_clk == phy_clk; otherwise add explicit CDC for WB request/address/data and RF pulses.
- Resets are synchronous unless a build option enables async in selected flops. rst_i resets core logic; usbf_utmi_ls also forces POR on VBUS attach.

- UTMI RX (PHY→core)
- Sample RxValid, RxActive, RxError, DataIn on each phy_clk; expose as registered rx_* and rx_data. No RX backpressure—consume bytes whenever rx_valid=1.

- UTMI TX (core→PHY)
- DataOut update: drive tx_data when (TxReady || tx_first). During drive_k, drive 8'h00 if not updating data.
- TxValid protocol: assert when (tx_valid || tx_valid_last || drive_k). Hold TxValid until TxReady unless the previous cycle was drive_k (bounded pulse). Requirements: assert tx_first exactly 1 cycle at SOP; assert tx_valid_last on the final payload beat.
- Transmit pacing: upstream must only advance data when the byte is accepted (tx_ready && tx_valid). rd_next must pulse once per accepted byte.

- UTMI line state, speeds, and special modes (usbf_utmi_ls)
- Control pins: XcvSelect selects HS/FS, TermSel enables FS term, SuspendM is active-high not-suspended, OpMode=00 normal, OpMode=10 disables bit-stuff/NRZI during reset/resume/chirps. drive_k asserted during resume and HS chirp.
- Timing windows (approximate, derived from 250 ns ticks): attach debounce ~100 ms; suspend after idle >~3 ms; device remote-wake allowed only after ≥~50 ms in suspend, then ~5 ms wakeup delay, drive K ~1 ms, settle ~100 µs; host resume similar settle; reset on qualified SE0; HS chirp: device K ~1.2 ms, require 6 alternating host K/J chirps else fall back to FS.
- Status handshakes: mode_hs latched on HS entry; usb_reset asserted during reset; usb_suspend set in suspend; usb_attached after debounce; suspend_clr pulses on suspend exit to clear queued resume requests.

- USB protocol engine handshakes (usbf_pl/usbf_pe)
- Token gating: New matched token (CRC5 OK, address/EP match) starts a transaction; any new match preempts the current state to IDLE on the next clk.
- Handshake PIDs: send_token asserts exactly 1 clk to emit ACK/NAK/STALL/NYET (NYET only in HS). HS PING elicits ACK via the same one-cycle strobe.
- IN (device→host): start TX DMA with one-cycle tx_dma_en; hold data_pid_sel constant for packet. After transmit completes, wait for ACK (token_valid && pid_ACK) within timeout: HS=22 cycles, FS=36 cycles. On timeout, abort and raise int_to_set.
- OUT/SETUP (host→device): start RX DMA with one-cycle rx_dma_en. No RX backpressure; rx_data_valid marks payload bytes; rx_data_done marks EOP. Abort without handshake on data-activity timeout (HS=22, FS=36 cycles), crc16_err, or explicit abort. After rx_data_done, choose

handshake per policy: NAK on size policy violations; ACK on seq err (non-iso) else NYET (HS+no_bufs) else ACK.
- Descriptor/flag updates: UPDATE and UPDATE2 are single-cycle phases to release/commit buffers, toggle next_bsel/next_dpid, and strobe interrupts; consumers must edge-detect these pulses.

- CRC requirements
- CRC5 (combinational): din mapping LSB-first; seed 5'b11111; compare ones' complement remainder against token1[7:3]. Must meet single phy_clk cycle from token1 capture to token_valid (next cycle). Use only when token_valid and PID is TOKEN.
- CRC16 (combinational): seed 16'hffff at rx_active rising edge; update 1 byte/clk when data_valid_d=1; residue 16'h800d must be checked only on rx_data_done. CRC check is independent of rx_data_valid timing (which is 2-cycle delayed payload stream).

- IDMA (memory↔USB streaming)
- Start/parameter capture: tx_dma_en/rx_dma_en are 1-cycle pulses; present stable adr, size, buf_size, and send_zero_length (TX) during that cycle. Address splits into word and byte-offset; ring wrap on word boundaries.
- RX path: no backpressure; read-modify-write first unaligned word; assert mreq/mwe per word; flush partial word at packet end; idma_done pulses 1 clk after rx_data_done.
- TX path: send_data=1 indicates valid tx_data_st; consumer must not assert rd_next until send_data=1 and must assert rd_next exactly once per consumed byte. Ping-pong prefetch hides read latency; idma_done when size drains (or immediate ZLP via send_zero_length).

- Memory arbiter (single-port SRAM)
- Priority: M-side (protocol/IDMA) has absolute priority over W-side (Wishbone). If mreq=1, W is preempted immediately.
- Acks: M-side mack=mreq (no waits inserted); W-side wack is a 1-cycle pulse, at most every other phy_clk, and only while M is idle. A W ack can be suppressed if M asserts mreq that cycle.
- Read data broadcast: sram_din drives both mdout and wdout; each master must sample only on its own ack. sram_re is tied high; sram_we asserted only by the selected master.

- Wishbone B3 classic slave (usbf_wb)
- One request at a time, sampled only in IDLE. Master must hold CYC/STB/ADR/WE/DAT stable until wb_ack_o pulses (single wb_clk cycle). No ERR/RTY/STALL.
- Address decode: wb_addr_i[17]==0 → RF region; ==1 → Memory region.
- RF access: rf_re/rf_we pulse for 1 phy_clk; wb_ack_o generated via W0→W1→W2 path. Read data must be valid at wb_ack_o; writes must be captured on the single write pulse.
- Memory access: ma_req (and ma_we) held until ma_ack (arbiter wack). wb_ack_o pulses after ma_ack returns through the synchronizer. With M idle, service rate ≤ one W ack every two phy_clk.
- CDC: wb_ack_d (phy_clk) retimed to wb_clk via a 3-FF edge detector. Safest when wb_clk==phy_clk; otherwise add CDC on request/address/data and RF strobes.

- RF/endpoint register file handshakes (usbf_rf/usbf_ep_rf)
- Endpoint selection: match is registered (match_r1) with 1-cycle latency; all in-band set/rl strobes and idin payloads are honored only when ep_match_r==1 and must be asserted exactly 1 clk with idin stable.
- DMA cross-domain: dma_req is held level in wclk until serviced; external DMA must generate 1-cycle wclk dma_ack pulses per beat. Ack is reflected to clk as a single-cycle pulse via a two-flop synchronizer.
- Interrupts: per-EP int_stat bits are sticky level in clk and clear on INT read; inta/intb are level in wclk (OR of int_stat & enables). Software must edge-detect buffer-done strobes and service sticky bits via read-to-clear.

- Top-level observables
- susp_o: registered copy of usb_suspend in clk_i domain (1-cycle latency). LineState_r and VStatus_r captured in phy_clk for software observation.

- External integration requirements
- Ensure external SRAM timing meets immediate M-side mack behavior and W-side every-other-cycle wack pacing; mdin must be valid at ack.
- Keep wb data/address stable until wb_ack_o; RF must accept single-cycle write strobes and present read data valid at ACK.
- For UTMI TX, honor tx_first/tx_valid_last rules to guarantee SOP/EOP acceptance; for RX, never assume backpressure exists.
- If clocks differ, add CDC for WB requests and RF pulses; otherwise constrain operation to wb_clk == phy_clk for safest timing.

-- Signal Polarity and Conventions --

- Global
- Clocking: All sequential logic samples on the rising edge of its clock. Primary domains are phy_clk (UTMI/protocol/memory) and wb_clk/clk_i/wclk (bus/registers/interrupts).
- Reset: rst_i is active-low. Internally rst is active-low (if (!rst) ...). Resets are synchronous by default; USBF_ASYNC_RESET may enable async reset in some builds. phy_rst_pad_o mirrors rst_i (active-low to PHY).
- Assertion level: Unless explicitly noted, requests, enables, qualifiers, handshakes, status, error flags, interrupts, and masks are active-high (1 = asserted).

- Pulse vs level conventions
- One-cycle strobes (active-high in their clock domain): wb_ack_o, wack, rf_re, rf_we, int_*_set, buf0_set, buf1_set, buf0_rl, uc_bsel_set, uc_dpd_set, tx_dma_en, rx_dma_en, token_valid, send_token, tx_first, tx_valid_last, suspend_clr, rf_resume_req, utmi_vend_wr.
- Level-valid qualifiers/flags (active-high): *_valid, *_active, send_data, dma_req[_o], dma_in_buf_sz1, dma_out_buf_avail, inta[_o], intb[_o], idma_done, mode_hs, usb_* status.
- Clear signals: *_clr are active-high clears for associated counters/flags.

- Wishbone/bus view (wb_clk domain unless noted)
- wb_stb_i, wb_cyc_i, wb_we_i: active-high request/WE qualifies.
- wb_ack_o: active-high, single-cycle pulse per completed access; wb_data_o is valid when wb_ack_o=1.
- Address region select: wb_addr_i[17] = 1 selects Memory Array (active-high MA select); 0 selects Register File.

- Memory/arbiter (phy_clk domain)
- External SRAM: sram_we_o active-high; sram_re_o tied high; sram_adr_o/sram_data_o driven by current owner; sram_data_i broadcast to both masters.
- Arbiter handshakes: M-side mreq, mwe active-high; mack = mreq (combinational, active-high). W-side wreq, wwe active-high; wack is an active-high, one-cycle pulse when served; wsel active-high when W owns the port.

- UTMI PHY/link
- Receive: RxValid, RxActive, RxError active-high from PHY; internal rx_valid/active/err are registered, active-high mirrors. DataIn sampled each phy_clk.
- Transmit: TxValid active-high to PHY; TxReady active-high from PHY. tx_first and tx_valid_last are active-high, one-cycle strobes. drive_k active-high forces K signaling; during drive_k, DataOut is forced

to 8'h00 and TxValid behavior is special-case.
- Line control/status: XcvSelect 0=HS, 1=FS (verify with target PHY); TermSel=1 enables FS termination; OpMode 2'b00=normal, 2'b10=special (disable bit-stuff/NRZI); SuspendM per code asserts during suspend or K detect (note: many PHYs define SuspendM=1 as not-suspended—verify/invert as needed). usb_vbus, mode_hs, usb_reset, usb_suspend, usb_attached, suspend_clr, drive_k are active-high.
- LineState qualifiers: "long" qualifiers (k_long, j_long, se0_long, idle_long) assert active-high after two consecutive samples.

- Protocol/endpoint/DMA
- match_o, dma_in_buf_sz1, dma_out_buf_avail: active-high when true for the selected endpoint.
- Handshakes and control: send_token (pulse, active-high) with token_pid_sel encoding (00=ACK,01=NACK,10=STALL,11=NYET); data_pid_sel encoding (00=DATA0,01=DATA1,10=DATA2,11=MDATA) is level-valid; send_zero_length active-high requests ZLP; abort active-high cancels ongoing DMA/transfer.
- DMA handshakes: dma_req_o bits active-high; dma_ack_i bits active-high; idma_done active-high level on completion.

- Interrupts and masks
- Event sources (int_*_set) are active-high, one-cycle pulses; sticky bits set high and clear on read (active-high read-to-clear).
- inta_o/intb_o are active-high levels; masks inta_msk/intb_msk are active-high enables for contribution to the outputs.

- Naming conventions
- Signals ending with _set or _rl are active-high, one-cycle pulses.
- *_msk, *_ena are active-high enable/mask bits.
- *_err flags are active-high error indications.
- *_valid, *_active, *_avail, *_hold, *_sz1 are active-high qualifiers/status.

- Data/ordering conventions (affects interpretation, not polarity)
- Little-endian byte lanes on 32-bit memory: lane0→[7:0], lane1→[15:8], lane2→[23:16], lane3→[31:24].
- CRC flags (pid_cks_err, crc5_err, crc16_err) are active-high; CRC-5/CRC-16 blocks are combinational with LSB-first conventions and ones'-complement compare/append.

## PARAMETERIZATION AND CONFIGURATION

-- Generic Parameters (Address Widths, Buffer Sizes) --

- USBF_UFC_HADR (Wishbone internal address width / RF vs MEM select)
- Purpose: Chooses RF (0) vs MEM (1) via wb_addr_i[USBF_UFC_HADR]; also sets internal WB address width = USBF_UFC_HADR+1.
- This build: 17 → select bit at wb_addr_i[17]; effective WB address width = 18 bits. WB data is 32-bit; accesses are 32-bit aligned (no byte-selects).

- SSRAM_HADR (external SRAM word-address width)
- Purpose: Width of sram_adr[SSRAM_HADR:0] on the 32-bit word-addressed packet-buffer port. Used

consistently by mem_arb/IDMA/PL/PE.
- Capacity: Effective buffer store = 4 × 2^(SSRAM_HADR+1) bytes.
- This build: 14 → sram_adr[14:0]; capacity ≈ 128 KiB.

- ADR_W (descriptor/DMA byte-address width; derived)
- Definition: ADR_W = SSRAM_HADR + 3; IDMA/PE use adr[SSRAM_HADR+2:0] as a byte address while the SRAM port remains word-addressed.
- This build: 17 bits (adr[16:0]).

- SZ_W (descriptor/buffer size field width)
- Fixed: 14 bits (size[13:0]). Max single programmed transfer per descriptor = 16 KiB.

- RF_AW (register-file address width)
- Fixed: 7 bits (adr[6:0]) covering globals and per-EP windows.

- MPS_W (Max Packet Size field width)
- Fixed: 11 bits; used to bound per-transaction sizes/address increments.

- BUF0 capacity slice (per-endpoint capacity/limit)
- Fixed: 12 bits in words (buf0_orig[30:19]) → up to ~4096 words (≈16 KiB) tracked per buffer.

- Endpoint presence/limits (addressing context)
- Max endpoints supported: 16 (vectors 16-bit). This build enables EP0..EP3.

- Constraints/consistency
- SSRAM_HADR must match across usbf_mem_arb/usbf_pl/usbf_pe/usbf_idma; it also sets ADR_W.
- Descriptor packing: size[13:0] shares a 32-bit word with address high bits; practical upper bound SSRAM_HADR ≤ 14 to avoid overlap.
- Addressing conventions: External SRAM is 32-bit word-addressed; IDMA operates on byte addresses and handles alignment/RMW as needed.

-- Build-Time Macros and Options --

- Address map and buffer memory
- USBF_TEST_IMPL
- Purpose: Selects expanded test/integration map.
- Effect: RF/MEM region select uses wb_addr_i[17]; wider Wishbone space pairs with larger external SRAM.
- Affects: usbf_wb decode and readback mux; system address map.
- This build: defined (region select at wb_addr_i[17]).
- USBF_UFC_HADR
- Purpose: Bit index for RF vs MEM select when not using TEST_IMPL.
- Effect: RF/MEM select is wb_addr_i[USBF_UFC_HADR].
- Affects: usbf_wb decode; interconnect/driver mapping.
- This build: not used (TEST_IMPL drives bit 17).
- SSRAM_HADR
- Purpose: External SRAM word-address width; sets buffer memory size and bus widths.
- Effect: Sizes sram_adr[SSRAM_HADR:0]; scales buffer space; drives internal address math.
- Affects: usbf_mem_arb, usbf_pl/usbf_idma/usbf_pe address/counter widths; top-level SRAM port; software descriptor fields.
- This build: 14.

- Endpoint presence

- USBF_HAVE_EPx (EP1..EP15; EP0 always present)
- Purpose: Per-endpoint compile-time inclusion.
- Effect: Enabled EPs instantiate usbf_ep_rf/usbf_pl/usbf_pe; disabled EPs use usbf_ep_rf_dummy. Must be contiguous from EP1 upward.
- Affects: usbf_rf endpoint instantiation and INT/DMA vector widths; software-visible EP windows remain fixed.
- This build: EP1..EP3 enabled; EP4..EP15 disabled.

- Reset style
- USBF_ASYNC_RESET
- Purpose: Switch selected flops to asynchronous reset (active-low) instead of synchronous.
- Effect: Applies in guarded logic (e.g., usbf_utmi_if Rx/TxValid flops, usbf_wb FSM, usbf_mem_arb wack_r). Others remain synchronous as coded.
- Affects: Reset behavior in phy_clk/wb_clk domains; integration sequencing.
- This build: undefined (synchronous reset).

- UTMI/link-state timing and clocks
- USBF_T1_* and USBF_T2_* families
- Purpose: Parameterize USB 2.0 timing windows (reset detect, suspend/resume, HS chirp, wakeup, debounce) in usbf_utmi_ls.
- Effect: Real-time durations depend on phy_clk and prescalers; adjust defines or parameters to maintain compliance on different clocks.
- Affects: usbf_utmi_if/usbf_utmi_ls control of XcvSelect, OpMode, TermSel, SuspendM and status (mode_hs, usb_reset, usb_suspend).
- USBF_HMS_DEL
- Purpose: Sub-SOF divider feeding hms_clk/sof_time for frm_nat timestamping.
- Affects: usbf_pl SOF/microframe timing capture.
- Note: Some protocol timeouts (e.g., ACK wait, data-activity) are fixed in RTL in usbf_pe and are not macro-parameterized.

- Optional decode helpers
- USBF_RF_SEL / USBF_MEM_SEL
- Purpose: Project macros to abstract RF/MEM decode.
- Effect: Alternative to direct wb_addr_i bit check in usbf_wb; keeps portability across maps.
- This build: direct use of wb_addr_i[17].

- Modules without build-time knobs (for clarity)
- usbf_crc5/usbf_crc16: Polynomial, bit order, seeds, and residues are fixed; no macros.
- usbf_idma: DMA semantics, prefetch depth, and size-counter widths are fixed; only SSRAM_HADR affects address widths.
- usbf_mem_arb: Fixed-priority M>W and always-on sram_re; no macros beyond reset and SSRAM_HADR width propagation.
- usbf_pe: HS/FS timeout cycle counts are hard-coded; no build-time parameters.

- Integration cautions
- Keep SSRAM_HADR identical across usbf_pl, usbf_idma, usbf_mem_arb, usbf_pe, top-level ports, and software descriptors.
- RF/MEM region-select (bit index) must match software and interconnect decode.
- UTMI timing defines must be retuned if phy_clk differs from reference.
- Wishbone CDC is not macro-controlled; best when wb_clk == phy_clk or add external CDC.

- This build summary

- USBF_TEST_IMPL: defined (RF/MEM at wb_addr_i[17]).
- USBF_UFC_HADR: unused.
- SSRAM_HADR: 14.
- USBF_HAVE_EPx: EP1..EP3 enabled; others disabled.
- USBF_ASYNC_RESET: undefined (synchronous).

-- Endpoint Presence and Configuration --

Endpoint presence is a build-time property: up to 16 logical endpoints (EP0..EP15) are supported, with EP0 mandatory. Additional endpoints (EP1..EP15) are enabled via USBF_HAVE_EPx macros and must be contiguous starting at EP1; this build instantiates EP0..EP3. Absent endpoints are compiled as dummy stubs: their windows read as zero, they never match on the wire, and they assert no DMA/IRQ, but 16-wide DMA/IRQ vectors remain fixed at the top level. Each instantiated endpoint has a software-programmable endpoint number in its CSR; on-the-wire token_endp[3:0] (validated by CRC-5) and the device address are matched against this number to activate the endpoint for a transaction. Only the matched endpoint participates in transfers and latches in-band updates (descriptor write-back, buffer select and data-PID base updates); firmware must ensure endpoint numbers are unique across active instances. Software configuration per endpoint is exposed in the RF space at base = 0x10 + 4*N (N=0..15): +0 CSR (direction/type, enable/STALL, DMA enable, max packet size, policy bits, programmable endpoint number, optional scheduling hints), +1 INT (per-EP status, read-to-clear; enables also hosted here), +2 BUF0 and +3 BUF1 (buffer descriptors: base address/size). Double buffering is standard; hardware manages buffer select and data-PID base via commit strobes after transfers. A buffer is treated as not available when disabled (e.g., descriptor bit set) or its address field is all ones; with dma_en=1, hardware wraps within the programmed buffer span. The Wishbone front-end (usbf_wb) does not create endpoints; it provides access to endpoint RF (wb_addr[17]=0) and external SRAM buffer space (wb_addr[17]=1). The memory arbiter gives fixed priority to the protocol/DMA side, so software SRAM accesses may stall during activity; configure buffers preferably while idle. Endpoint activation on the wire is gated by: valid PID, token_valid, CRC-5 OK, device address match, and endpoint-number match; token_valid strobes on ACKs as well, but those carry no endpoint info and must be ignored for selection. Speed/link state from UTMI (mode_hs, usb_reset, usb_suspend, usb_attached) does not change presence but informs configuration: firmware should program per-EP max packet size/policy consistent with current speed (e.g., HS enables PING/NYET handling) and must reinitialize CSRs, descriptors, and data-toggle bases after usb_reset. EP0 uses the same CSR/INT/BUF scheme for control transfers; some Chapter 9 policies may require firmware discipline. Software must not program windows of non-instantiated endpoints, should align buffer bases to 32-bit boundaries, and must reapply configuration after resets; there is no runtime register enumerating which endpoints are synthesized.

-- Address Split Configuration --

- Top-level Wishbone address-space split
- Select bit: wb_addr_i[17]. 0 = Register-File (RF) region, 1 = external Memory (SSRAM) region.
- Implemented in usbf_wb (USBF_RF_SEL/USBF_MEM_SEL resolve to wb_addr_i[17] in this build).
- Data steering: RF reads return rf_din, writes pass wb_data_i; MEM reads return ma_din, writes pass wb_data_i.
- ACK: RF accesses ack immediately; MEM accesses ack on ma_ack from usbf_mem_arb.

- RF region map (word-addressed adr[6:0], no byte enables)
- Global window (aliased): adr[6:2] == 0 or 1 select same top-level regs; adr[6:2] == 2/3 reserved (read as 0).
- 0x00: main_csr (R); W bit5 issues rf_resume_req pulse.
- 0x01: function address fa[6:0] (R/W).

- 0x02: interrupt masks A/B (R/W).
- 0x03: interrupt sources (R; read-to-clear sticky RF events).
- 0x04: frm_nat timestamp (R).
- 0x05: UTMI vendor status (R); vendor control + write strobe (W).
- Per-endpoint windows: base = 0x10 + 4*EP (EP=0..15); adr[1:0] = {0:CSR, 1:INT, 2:BUF0, 3:BUF1}.
- Disabled/placeholder EPs read as zeros or dummy constants; writes have no effect.
- Reading INT clears per-EP sticky status; writes program iena/ienb.

- MEM region (external SRAM window)
- When wb_addr_i[17]==1, usbf_wb forwards ma_adr = wb_addr_i to usbf_mem_arb; high bits beyond SRAM width truncate naturally.
- Addressing is 32-bit word-aligned; no byte-selects anywhere on this path. Software must issue aligned 32-bit accesses.
- Parameter SSRAM_HADR sets SRAM word address width (example: 14 => 15-bit word addr, ≈128 KiB total at 32-bit).

- External memory arbitration (address-source split after MEM select)
- usbf_mem_arb muxes single-port SRAM between core DMA master (M) and Wishbone MEM master (W).
- Fixed priority: M (mreq) always wins; W (wreq) is served only when M is idle. wack is a 1-cycle pulse; sustained W traffic achieves ≈1 transfer per 2 phy_clk cycles when M is idle.

- IDMA byte/word address handling (downstream of the split)
- Protocol descriptors provide a byte address adr[SSRAM_HADR+2:0]. IDMA splits into word index adr_cw (madr) and byte offset adr_cb.
- RX uses read-modify-write for first/last partial words; TX prefetches two words and multiplexes byte lanes.
- With dma_en=1, address advancement wraps in a circular buffer anchored at absolute address 0; ring length = buf_size bytes.

- Protocol vs. bus addressing (orthogonal)
- USB device address fa (R/W at RF@0x01) and token endpoint number select on-wire targets; they do not affect the Wishbone RF/MEM decode.
- In-band EP updates are additionally gated by ep_match (token ep_sel vs. CSR[21:18]); bus routing remains purely address-based via usbf_rf.

- Clocking/CDC and transaction model
- usbf_wb samples WB request into phy_clk and returns a synchronized ack to wb_clk; one outstanding request at a time.
- Best used when wb_clk == phy_clk or with closed CDC timing; CDC does not alter the address map.

- Software guidance
- Use wb_addr_i[17]==0 to access RF global and per-EP windows; wb_addr_i[17]==1 for buffer SRAM.
- Only 32-bit aligned accesses are supported in both regions.
- After reset, program fa and endpoint descriptors before relying on addressed USB transactions.

- Non-participating blocks
- usbf_utmi_ls, usbf_crc5, and usbf_crc16 have no bus-facing address map and do not affect the address-space split.

-- Timing and Protocol Constants --

- Clocking, reset, and domains
- Two clocks: phy_clk (UTMI/protocol/memory) and wb_clk/clk_i (Wishbone/RF). Resets are synchronous unless USBF_ASYNC_RESET is enabled.
- CDC highlights: wb_ack_o is a one-cycle pulse in wb_clk generated from a PHY-domain ack; usbf_wb inserts two phy_clk dead cycles (W1/W2) after each ack request.

- UTMI line states and idle
- LineState encodings: J=01, K=10, SE0=00, SE1=11. FS idle=J; HS idle=SE0.
- “Long” qualifiers (k_long, j_long, se0_long, idle_long) require two consecutive samples.

- Time-base generators and thresholds (macro/clock driven)
- Short tick gate ps_cnt: divides clk to ~250 ns ticks while idle_long is true (drives T1_* windows).
- Micro/millisecond chain (T2_*): ~2.5 µs events aggregated to ~0.5 ms, then counted in me_cnt.
- Common threshold flags (approximate real time; exact via USBF_T1_* / USBF_T2_*): >2.5 µs, <3.0 ms window, >3.0 ms, >3.125 ms, >5.0 ms, >100 µs, >1.0 ms, >1.2 ms, and ~100 ms (attach debounce).

- UTMI/link FSM timing behaviors
- Attach: usb_attached asserted after ~100 ms of VBUS presence.
- Reset detect: SE0 qualified > ~2.5 µs (also from SUSPEND) triggers usb_reset and speed renegotiation.
- Suspend entry: FS idle > ~3.0 ms enters SUSPEND. In HS, after ~3.0 ms idle, switch to FS observation and, after ~100 µs, enter SUSPEND on J long.
- Host resume: while suspended, detect K, then wait for SE0 and settle ~100 µs before returning to NORMAL.
- Remote wake (device resume): require ~5 ms in suspend, then drive K for ~1.0 ms, followed by the resume path and ~100 µs settle.
- High-speed negotiation: after reset, device chirps K for ~1.2 ms; accept HS upon six alternating host K/J chirps, else fall back to FS.

- UTMI control encodings (as driven)
- XcvSelect: 0=HS, 1=FS. TermSel: 1=enable FS termination. OpMode: 00=Normal, 10=disable bit-stuff/NRZI (used during reset/resume/chirp).
- SuspendM is driven as (usb_suspend & !resume_req_s) | (LineState==K) and should match PHY polarity.

- UTMI TX handshake hardening
- TxValid is held until TxReady. tx_first preloads SOP; tx_valid_last guarantees the final beat handshake.
- During drive_k, a bounded TxValid pulse is generated; DataOut forced to 0x00 and the hold path is disabled on the next cycle.

- Protocol-layer timing constants (protocol/endpoint engine)
- IN ACK wait timeout (phy_clk cycles): HS=22, FS=36.
- OUT inactivity timeout (phy_clk cycles): HS=22, FS=36.
- int_to_set asserts on IN ACK wait expiry (IN2) or OUT inactivity (OUT) per above thresholds.

- SOF/microframe timing and stamping
- hms_clk asserted when a 5-bit counter equals 5'h1C (28). sof_time[11:0] increments on each hms_clk between SOFs.
- frm_nat[31:0] = {mfm_cnt[3:0], 1'b0, frame_no[10:0], 4'h0, sof_time[11:0]}.

- PID encodings and protocol classes/filters
- Lower-nibble PID encodings: OUT=0001, IN=1001, SOF=0101, SETUP=1101, DATA0=0011, DATA1=1011, DATA2=0111, MDATA=1111, ACK=0010, NACK=1010, STALL=1110, NYET=0110, PRE=1100, ERR=1100, SPLIT=1000, PING=0100, RES=0000.
- TOKEN class = {OUT, IN, SOF, SETUP, PING}. DATA class = {DATA0, DATA1, DATA2, MDATA}.
- Address match gating excludes ACK, NACK, STALL, NYET, PRE, ERR, SPLIT; PING is accepted HS-only and filtered in FS.

- CRC protocol constants and timing of checks
- CRC-5 (TOKEN): polynomial x^5 + x^2 + 1; seed 0x1F; LSB-first over 11 bits {ADDR[6:0], ENDP[3:0]}; bus carries ones' complement of remainder. Checked when token_valid strobes (one clk after second token byte capture).
- CRC-16 (DATA): polynomial x^16 + x^15 + x^2 + 1 (0x8005; reflected 0xA001); seed 16'hFFFF; byte updates LSB-first; receiver residue expected 16'h800D after payload+CRC bytes. crc16_err qualified with data_done.

- Memory arbitration and bus timing constants
- usbf_mem_arb: M-side mack=mreq (combinational); W-side wack is a one-phy_clk pulse and, when M is idle and W is continuous, occurs at most every other phy_clk.
- sram_re tied high; read data broadcast to both masters; each samples only on its own ack.
- usbf_wb: wb_addr_i[17] selects RF (0) vs MEM (1). RF ACK immediate; MEM ACK on mem_arb wack. wb_ack_o is a single wb_clk pulse per transaction via a 3-FF edge detector.

- IDMA/memory-path timing behaviors
- 32-bit word interface; M-side priority hides arbiter latency for protocol/DMA (mack_r registered internally).
- TX prefetch: fixed two-word ping-pong (8 bytes) enables byte-per-phy_clk streaming if memory returns one word per boundary crossing.
- RX first word uses read-modify-write; final partial word flushed on packet end.

- RF/DMA pacing constants (word-based)
- P = floor(MPS/4) words. Availability flags and counters operate on word counts (12-bit).
- DMA request hold thresholds: OUT holds while >3 words outstanding; IN holds when capacity ≥8 words and within a 3-word margin of full.

- Build-time parameters affecting timing/space
- USBF_UFC_HADR=17 selects RF vs MEM region (wb_addr_i[17]).
- SSRAM_HADR=14 (word addressing); effective memory size = (1 << (SSRAM_HADR+1)) × 4 bytes.

- Resolution to real time
- All cycle/count thresholds resolve to real-time values from the implemented phy_clk (commonly 60 MHz for UTMI+) and synthesis-time macros (USBF_T1_*, USBF_T2_*, USBF_HMS_DEL). Extract concrete counts for the target build to finalize microsecond/millisecond numbers.

-- Simulation and Debug Options --

- Build-time options to set before sim
- USBF_ASYNC_RESET (default off): makes some resets asynchronous.
- USBF_TEST_IMPL (on in this build): widens address map; Wishbone region select at wb_addr_i[17].
- USBF_HAVE_EPx: compile-time enable for EPs; enabled EPs must be contiguous or sim stops.
- USBF_UFC_HADR/SSRAM_HADR: address widths for RF decode and external SRAM; must match memory model.

- Timing macros (T1/T2 families): control FS/HS timeouts, suspend/resume, reset, HS chirp counts; may be scaled to speed up sims.

- Clocks, resets, and CDC cautions
- Two primary clocks at top: phy_clk (protocol/mem) and wb_clk (Wishbone). For safest sim, run wb_clk == phy_clk; usbf_wb and mem_arb sample requests in phy_clk with minimal CDC.
- Top-level reset rst_i is active-high; some sub-blocks use active-low FSM resets (e.g., usbf_wb). Hold resets long enough to avoid X pulses on ACK paths.
- usbf_ep_rf has its own wclk and clk with CDC hardening around dma_req/ack; others assume single-domain or aligned clocks.

- External models required
- UTMI PHY behavioral model: drive LineState, RxValid/Active/Error/DataIn, respond with TxReady; observe TxValid/DataOut during resume/chirps.
- SSRAM model (32-bit): continuous read enable (sram_re tied high), drives sram_data_i at all times; honors sram_we. Zero-wait preferred (mack=mreq) on protocol side.

- Wishbone and memory access (debug tips)
- Region select: wb_addr_i[17]==0 → RF; ==1 → external memory via mem_arb.
- Single outstanding transfer; RF ACK is immediate; memory ACK gated by mem_arb wack. wb_ack_o is a single-cycle pulse; data must be valid on that pulse.
- mem_arb: fixed priority to M-side (protocol). W-side wack pulses every other phy_clk when M idle; W starves when M busy.

- Link/UTMI simulation knobs and observability
- Drive LineState to emulate J/K/SE0; hold ≥2 clk cycles to register. Model attach (~100 ms), reset (SE0 long), suspend (>~3 ms), resume (K long), and HS chirps (6 K/J pairs).
- Observe: mode_hs, usb_reset, usb_suspend, usb_attached, suspend_clr, drive_k, XcvSelect, TermSel, OpMode, SuspendM, LineState_r, VStatus_r, VControl/VControl_Load.
- Remote wake: assert resume_req_i or write RF 0x00 bit5; watch drive_k and auto-clear on suspend exit.

- Protocol/IDMA/EP RF debug hooks
- Protocol engine: watch send_token, token_pid_sel (ACK/NACK/STALL/NYET), data_pid_sel/this_dpid/next_dpid, tx_dma_en/rx_dma_en, rx_ack_to/tx_data_to, int_*_set, buf*_set, uc_*_set, buf0_rl, abort/out_to_small.
- IDMA: exercise RX/TX flows, ring wrap, unaligned starts, aborts; watch mreq/mack, mwe/madr/mdout/mdin, send_data, rd_next, idma_done, size counters.
- Endpoint RF: program CSR/INT/BUF0/BUF1; use dma_req/ack flow control; read INT (sticky, clear-on-read); in-band updates only when ep_match=1. Use usbf_ep_rf_dummy to disable EPs safely.

- CRC debug options
- CRC-5 (token): use combinational usbf_crc5; seed 5'h1f; LSB-first; compare ~crc_out to received field; gate checks with pid_TOKEN.
- CRC-16 (data): seed 16'hffff; byte-parallel LSB-first; include payload and the two CRC bytes; expect residue 16'h800d at packet end.

- RF address map highlights
- 0x00 main_csr (read link state; write bit5 to request remote wake), 0x01 function address, 0x02 interrupt masks, 0x03 RF sticky causes (read-to-clear), 0x04 frame/microframe time, 0x05 UTMI vendor status/control. Per-EP windows start at 0x10 (CSR/INT/BUF0/BUF1).

- Built-in checks/messages and X-handling
- Top prints EP config and stops on non-contiguous EPs. Multiple FSMs perform X-detection on key controls (e.g., enables, acks, size flags) and may stop sim on unknowns. Initialize all inputs and hold handshakes stable during/after reset.

- Recommended waveforms
- Bus/mem: wb_stb/cyc/we/addr/data/ack, rf_re/we/din/dout, ma_req/we/ack/din/dout, mem_arb mreq/mack/wreq/wack/wsel/sram_*.
- Link/UTMI: mode_hs, usb_reset/suspend/attached, drive_k, XcvSelect/TermSel/OpMode/SuspendM, LineState_r, TxValid/TxReady/DataOut.
- Protocol/IDMA/EP: send_token/token_pid_sel, data_pid_sel, tx/rx_dma_en, rd_next/send_data, idma_done, buf*_set/uc_*_set/buf0_rl, int_*_set, rx_ack_to/tx_data_to, abort/out_to_small.
- CRC: token_valid, pid_* qualifiers, token_fadr/endp/crc5, crc5_out(~), data_valid_d, rx_data_done, crc16_sum/out, crc16_err.

- Directed test ideas
- Link: attach→reset→HS chirp→FS fallback; suspend/resume (host and remote wake); VBUS toggle (POR) and re-attach.
- WB/RF: RF read/write immediacy; memory read/write via mem_arb with/without M contention; back-to-back access spacing.
- IN/OUT: normal transfers with ACK, timeout cases (no ACK or no data), policy NACK/STALL, NYET on HS no-buf, ZLP, OUT short packet handling.
- DMA: unaligned starts, ring wrap, abort mid-transfer, latency variations (1-cycle mack) for robustness.
- CRC: deliberate bit flips to trip crc5_err/crc16_err exactly at token_valid/data_done.

- Assertions (suggested)
- One ACK per WB request; no ACK without request. ma_ack implies ma_req; mack==mreq; wack implies wreq & !mreq; mutual exclusion on mem muxing.
- CRC-16 residue == 16'h800d at packet end; CRC-5 matches ~remainder for tokens.
- IDMA RMW correctness on partial first word; ring wrap address bounds; no mwe without word_done.

- Performance notes
- M-side assumes zero-wait memory (mack=mreq). W-side achieves ~0.5 transfers/phy_clk when M idle; can starve under M load. Ensure sim clocks/timescale cover ms-range link timers or scale timing macros.

-- Configuration Constraints and Dependencies --

- Clocks and resets
- Two primary clocks: phy_clk (USB/UTMI/link, data path) and wb_clk (Wishbone). The design assumes wb_clk == phy_clk for safe operation; otherwise add a CDC-safe bridge around usbf_wb and synchronize RF/INT paths.
- Reset is active-low. Reset style (sync vs async) is controlled by USBF_ASYNC_RESET; keep consistent across all instantiated modules.

- UTMI PHY interface and timing
- 8-bit UTMI required. Typical clocks: 60 MHz (HS), 12 MHz (FS). All USB timing constants (reset/resume/suspend/chirp windows, ACK/data timeouts) are tuned to these rates; retune if phy_clk differs.
- LineState encoding assumed: 00=SE0, 01=J, 10=K, 11=SE1. Verify XcvSelect/TermSel widths and polarities; OpMode usage may need remap for PHYs expecting Non-Driving instead of Disable bit-stuff.
- SuspendM polarity/semantics must match your PHY. Ensure the PHY ignores DataOut during special

signaling (OpMode!=Normal).
- VBUS polarity must match RTL expectation (POR gating). Vendor sideband (VStatus/VControl/VControl_Load) is optional; tie off safely if unused.

- External SRAM and memory arbitration
- Single-port, synchronous, 32-bit SRAM assumed. Zero-wait behavior required for the core-side M port: data must be valid with ack in the same phy_clk cycle. If your memory needs CE/OE/latency, insert a wrapper that generates proper ready/ack and aligns read data.
- sram_re is tied high; confirm memory tolerates continuous read-enable.
- SSRAM_HADR sets external SRAM address width; keep consistent across usbf_top/usbf_mem_arb/usbf_idma/usbf_pe/software.
- usbf_mem_arb gives fixed priority to the M (protocol/DMA) port; W (Wishbone) can be starved under load. W sees a one-cycle wack at most every other phy_clk when M is idle.

- Wishbone bus usage
- Classic WB B3 single-transfer only: one outstanding request, no bursts/tags/stall/byte-lanes. wb_ack_o is a one-cycle pulse; capture data on that cycle.
- Address split is compile-time. With USBF_TEST_IMPL enabled, wb_addr_i[17]==0 selects RF, ==1 selects MEM. Update system decode/software if this changes.
- usbf_wb samples WB requests directly into phy_clk and returns ACK through a small pipeline; if wb_clk != phy_clk, use a CDC-safe bridge.

- Addressing, alignment, and ring behavior
- External memory is 32-bit word-addressed internally. Unaligned packet starts/ends are handled via RMW and last-word flush in usbf_idma.
- With dma_en=1, usbf_idma wraps to address 0 when (adr + buf_size) crosses the limit; ring base is hard-coded at 0. Use buf_size multiples of 4 for clean word-aligned wrap.

- Endpoint configuration and RF dependencies
- EP0 is always present. Additional endpoints are compile-time enabled via USBF_HAVE_EPx and must be contiguous starting at EP0. Endpoint numbers programmed in csr[21:18] must be unique.
- CSR legality: csr[27:26] direction 01=IN, 10=OUT, 00=CTRL; CTRL requires csr[15] (dma_en)=0. csr[25:24] type drives handshake/PID policies (iso vs bulk). MPS csr[10:0] must match USB descriptors.
- BUF0/BUF1 descriptors: size in bytes (14 bits), address within SSRAM window; all-ones address is a reserved NA sentinel. For non-CTRL with dma_en=1, only BUF0 is used.
- RF INT mapping: read-to-clear on INT; enable bits gate inta/intb. A known mapping quirk ties int_stat[4] to enable index 3 in both inta/intb.

- DMA and data-path dependencies
- usbf_idma expects near zero-wait memory service; excessive latency can overrun RX at line rate. mdin must be valid when mack asserts; writes are accepted on word_done_r with mack.
- dma_req/dma_ack handshake is one 32-bit word per ack pulse; ack must be a one-cycle pulse. IN prefill and OUT drain counters are in word units; MPS not divisible by 4 is floored in readiness calculations.
- Size limits: TX size max 16 KiB per activation; RX internal counter up to 2047 bytes per packet (enforce via PE/software as needed).

- Protocol engine and timing
- Token match from usbf_pl gates usbf_pe; new matches can abort in-flight updates. PING/NYET are HS-only. Timeouts (ACK/data) depend on mode_hs and are cycle-based; retune with clock changes.

- CRC usage constraints

- CRC-5: LSB-first over 11 bits {ADDR[6:0],ENDP[3:0]}, seed 0b11111, compare ones'-complement of remainder. Mapping must match token byte capture.
- CRC-16: Polynomial 0x8005 (reflected), seed 0xFFFF, LSB-first, residue method assumes CRC bytes are processed; expected residue 0x800d at data_done.

- Interrupt routing and CDC
- inta_o/intb_o are ORs of RF/EP events. If wb_clk != phy_clk, synchronize INT paths or generate INT in a single domain to avoid metastability.

- Software/firmware responsibilities
- Chapter 9 control flow (SET_ADDRESS, descriptors, policies) is firmware-driven; RTL does not cover all certification corner cases or HS Test Modes.
- Program function address, global masks, and handle sticky INT sources. Respect remote wake constraints (host-enabled only; request is latched/cleared by suspend exit). Vendor UTMI control writes are one-shot; follow PHY timing.

- Build-time macros and parameters
- USBF_TEST_IMPL: affects RF/MEM split bit and often SSRAM_HADR; align software/interconnect.
- USBF_ASYNC_RESET: selects reset style.
- USBF_HAVE_EPx: enables contiguous endpoints from EP0.
- UTMI/link timing constants and PE timeouts depend on phy_clk; re-validate if frequency changes.

- Known caveats
- sram_re tied high; confirm power/timing acceptability.
- W arbitration can starve under heavy USB traffic; plan software timeouts.
- Dummy endpoints must occupy only the highest slots after the last active EP; do not interleave with active endpoints.

# MODULE HIERARCHY AND INTEGRATION NOTES

-- Submodule Overview and Responsibilities --

- usbf_utmi_if (UTMI PHY bridge and link management)
- Terminates UTMI PHY and provides clean, registered rx_*/tx_* byte streams to/from the core.
- Owns TxValid hold and last-beat handshakes; supports SOP preload and special-mode neutralization.
- Delegates UTMI line control to usbf_utmi_ls and forwards its control/status to PHY and core.
- Domain: phy_clk.

- usbf_utmi_ls (UTMI line-state and link control)
- Decodes LineState, manages attach/reset/suspend/resume, and negotiates HS/FS.
- Drives UTMI control pins (XcvSelect, TermSel, OpMode, SuspendM); outputs mode_hs, usb_reset/suspend/attached, suspend_clr.
- Asserts drive_k during resume and HS chirps for special signaling.
- Domain: phy_clk.

- usbf_pl (Protocol layer and data path)
- Implements USB 2.0 transaction control (IN/OUT/SETUP), PID/CRC, endpoint policy, and payload

movement to/from external SRAM.
- Sub-blocks:
- usbf_pd: Classifies PIDs, captures token fields (FA/EP/SOF), checks CRC5/CRC16, and streams payload.
- usbf_pe: Central transaction/policy engine; selects handshakes, enforces sequencing and MPS rules, coordinates DMA, and issues descriptor/interrupt updates.
- usbf_pa: Assembles handshakes and DATA packets (PID + payload + CRC16); drives tx_valid/tx_first/tx_valid_last and rd_next.
- usbf_idma: Byte↔word bridge for 32-bit SRAM; RMW/flush on RX, ping–pong prefetch on TX, optional ring/wrap.
- Exposes frm_nat (SOF/microframe timing) and master memory interface to arbiter.
- Domain: phy_clk.

- usbf_crc5 (Token CRC-5 generator)
- Pure combinational CRC-5 over 11 LSB-first bits for token validation (seed 0x1f; inverted compare).

- usbf_crc16 (DATA CRC-16 updater)
- Combinational byte-parallel CRC-16 step (seed 0xffff); used for RX residue checking and TX CRC generation.

- usbf_mem_arb (Single-port SRAM arbiter/mux)
- Shares one 32-bit SRAM between core master (M) and Wishbone/software (W).
- Fixed priority to M; W granted only when M idle; W acks paced (every other phy_clk when served).
- Broadcasts read data to both; mack=mreq (combinational), wack is a 1-cycle pulse.
- Domain: phy_clk.

- usbf_rf (Register file, per-EP control, interrupts, DMA fanout)
- Software-visible global regs (CSR, function address, RF masks/sources, frm_nat, UTMI vendor bridge, remote-wake) and per-EP windows (CSR/INT/BUF0/BUF1).
- Provides live, matched-endpoint view to protocol layer; aggregates A/B interrupts and per-EP DMA req/ack.
- Submodules:
- usbf_ep_rf: Per-EP RF/INT/DMA accounting and match-gated in-band updates.
- usbf_ep_rf_dummy: Safe tie-off for disabled endpoints (no match/IRQ/DMA behavior).
- Domains: wclk (bus) and phy_clk/core (with CDC one-shots/edge-detect).

- usbf_wb (Wishbone front-end and RF/memory decoder)
- Wishbone B.3-classic slave; decodes address space into RF vs memory regions.
- Sequences one transaction at a time in phy_clk; generates single-cycle wb_ack_o in wb_clk via CDC edge detect.
- Drives rf_re/rf_we or memory ma_req/ma_we; returns rf_din/ma_din on reads.
- Domains: wb_clk and phy_clk.

- Top-level glue (usbf_top)
- Distributes resets (rst_i, phy_rst_pad_o), registers select status across domains, and latches/clears remote-wake requests.
- Wires UTMI pins to utmi_if, byte streams between utmi_if and protocol, memory buses to mem_arb, and RF/wishbone control/status paths.

-- Hierarchical Connectivity and Interfaces --

- Module hierarchy (parent -> children)
- usbf_top

- usbf_utmi_if (contains usbf_utmi_ls)
- usbf_pl
- usbf_pd (instantiates usbf_crc5 and usbf_crc16)
- usbf_pa (instantiates usbf_crc16)
- usbf_idma
- usbf_mem_arb
- usbf_rf
- up to 16x usbf_ep_rf (and/or usbf_ep_rf_dummy)
- usbf_wb

- Top-level external interfaces (usbf_top)
- Wishbone slave (clk_i domain)
- In: wb_clk=clk_i, rst_i, wb_addr_i[17:0] (bit[17] region select), wb_data_i[31:0], wb_we_i, wb_stb_i, wb_cyc_i
- Out: wb_data_o[31:0], wb_ack_o
- UTMI PHY
- RX in: DataIn_pad_i[7:0], RxValid_pad_i, RxActive_pad_i, RxError_pad_i, LineState_pad_i[1:0], usb_vbus_pad_i, VStatus_pad_i[7:0]
- TX/control out: DataOut_pad_o[7:0], TxValid_pad_o, XcvSelect_pad_o, TermSel_pad_o, SuspendM_pad_o, OpMode_pad_o[1:0]
- Handshake in: TxReady_pad_i
- Vendor sideband: VControl_pad_o[3:0], VControl_Load_pad_o
- External synchronous SRAM (phy_clk domain)
- Out: sram_adr_o[SSRAM_HADR:0], sram_data_o[31:0], sram_we_o, sram_re_o=1
- In: sram_data_i[31:0]
- Interrupts/DMA/Power
- Out: inta_o, intb_o, dma_req_o[15:0], susp_o
- In: dma_ack_i[15:0], resume_req_i
- Clocks/resets
- In: phy_clk_pad_i, clk_i, rst_i
- Out: phy_rst_pad_o

- Internal connectivity (usbf_top -> children)
- usbf_utmi_if <-> UTMI PHY pads (all UTMI pins listed above). Exports to core: rx_data[7:0], rx_valid, rx_active, rx_err, tx_ready; accepts from core: tx_data[7:0], tx_valid, tx_valid_last, tx_first. Provides link status/control: mode_hs, usb_reset, usb_suspend, usb_attached, suspend_clr. Accepts resume_req_r from top (latched from resume_req_i or SW) and forwards vendor controls to pads; latches VStatus to RF.
- usbf_pl <-> usbf_utmi_if
- From UTMI IF: rx_data[7:0], rx_valid, rx_active, rx_err, tx_ready, mode_hs
- To UTMI IF: tx_data[7:0], tx_valid, tx_valid_last, tx_first
- usbf_pl <-> usbf_mem_arb (M-port, phy_clk)
- To arb: mreq, mwe, mdin[31:0], madr[SSRAM_HADR:0]
- From arb: mack, mdout[31:0]
- usbf_pl <-> usbf_rf (core view, phy_clk)
- From RF: csr[31:0], buf0[31:0], buf1[31:0], dma_in_buf_sz1, dma_out_buf_avail, match, fa[6:0] (function address)
- To RF: idin[31:0], buf0_set, buf1_set, uc_bsel_set, uc_dpd_set, buf0_rl, int_buf0_set, int_buf1_set, int_crc16_set, int_to_set, int_seqerr_set, int_upid_set, frm_nat[31:0], status flags (pid_cs_err, crc5_err, nse_err, out_to_small)
- usbf_mem_arb <-> external SRAM (phy_clk)
- sram_adr, sram_dout, sram_din, sram_we, sram_re=1. M/W mux with fixed priority to M; mack

mirrors mreq; wack is paced pulse when granted.
- usbf_wb (wb_clk front-end, phy_clk sequencer)
- Wishbone: as above
- RF bus (phy_clk into usbf_rf): rf_re, rf_we, rf_dout[31:0], rf_din[31:0]
- Memory W-port (phy_clk into usbf_mem_arb): ma_adr, ma_dout[31:0], ma_req, ma_we; from arb: ma_ack, ma_din[31:0]
- usbf_rf (aggregation, two views)
- Bus view (from usbf_wb, phy_clk strobes): adr[6:0], re, we, din[31:0] -> muxed dout[31:0]
- Core view (to usbf_pl): live matched EP exports listed above
- Interrupts/DMA: inta_o/intb_o = OR of masked per-EP events and RF sticky sources; dma_req_o[15:0] out; dma_ack_i[15:0] in
- UTMI vendor/status: captures VStatus from top; drives VControl and Load to top pads; rf_resume_req to top (to form resume_req_r)

- Intra-block child connectivity
- usbf_utmi_if <-> usbf_utmi_ls (phy_clk)
- Inputs to ls: LineState[1:0], usb_vbus, resume_req, clk, rst (rx_active/tx_ready present but unused)
- Outputs from ls: XcvSelect, TermSel, OpMode[1:0], SuspendM, mode_hs, usb_reset, usb_suspend, usb_attached, suspend_clr, drive_k (consumed by utmi_if to force K and special signaling)
- usbf_pl sub-blocks (all phy_clk)
- usbf_pd (RX front-end) -> usbf_crc5/usbf_crc16: token/data CRC checks; exports pid_*, token_valid, crc5_err, crc16_err, rx_active, rx_data_valid/done
- usbf_pa (TX assembler) -> usbf_crc16: running CRC-16 for TX; consumes data_pid_sel, send_token/token_pid_sel, send_data, rd_next, tx_data_st
- usbf_idma (memory bridge) <-> usbf_pa/usbf_pd/usbf_pe: rx_data_st/valid/done in; tx_data_st/send_data out; rd_next in; mreq/mwe/madr/mdout/mdin/mack to mem_arb; idma_done/sizu_c back to usbf_pe
- usbf_pe (protocol engine) orchestrates: starts rx_dma_en/tx_dma_en, emits handshakes/data_pid, supplies adr/size/buf_size/dma_en to idma, raises write-back strobes and int_* to RF, observes match/mode_hs/pid_* and idma_done/CRC and timeouts

- usbf_rf -> per-endpoint blocks (fan-out/fan-in)
- Fan-out (clk=phy_clk): shared idin[31:0] and strobes (buf0_set, buf1_set, uc_bsel_set, uc_dpd_set, buf0_rl, int_*), out_to_small, ep_sel[3:0]
- Per-EP instances:
- usbf_ep_rf (real EP):
- Bus side (wclk from usbf_wb via phy_clk strobes): adr[1:0], re, we, din -> dout
- Core side (clk): ep_match, csr, buf0, buf1, dma_in_buf_sz1, dma_out_buf_avail
- Int/DMA (wclk): inta, intb, dma_req; input dma_ack
- usbf_ep_rf_dummy (disabled EP): same ports; ep_match=0, dout=0, inta/intb/dma_req=0, csr=0, buf* driven to 0xffff_ffff locally only

- Clock/reset domains and CDC boundaries
- phy_clk: utmi_if(+ls), pl (pd/pa/idma/pe), mem_arb, core view of rf, phy-side of wb FSM
- wb_clk (clk_i): Wishbone I/F registers/ack, bus view (software) of rf, susp_o register
- rst_i drives core and phy_rst_pad_o; synchronous by default
- CDC
- wb_req sampled into phy_clk inside usbf_wb; wb_ack edge returned into wb_clk (3-FF edge detector)
- rf handles its own wclk<->clk handshakes for per-EP DMA/ints and sticky events
- suspend_clr observed in clk_i domain to clear top-level resume_req latch

- Addressing and selection
- Wishbone region select: wb_addr_i[17] = 0 -> RF space; 1 -> external memory (forwarded to

mem_arb W-port)
- RF internal map: adr[6:2] selects EP window (0x10 + 4*EP); adr[1:0]={0:CSR,1:INT,2:BUF0,3:BUF1}. Global RF regs at low addresses (e.g., 0x00..0x05)

- Arbitration and priorities
- mem_arb: fixed priority to M-port (usbf_pl/usbf_idma). W-port (usbf_wb) granted only when M idle; wack pulses every other phy_clk while granted. mack mirrors mreq combinationally

- Parameterization and widths
- SSRAM_HADR defines external SRAM address width (example 14 -> 15-bit word address bus)
- USBF_UFC_HADR/USBF_TEST_IMPL configure Wishbone address sizing and region select bit

- Top-level derived/control signals
- susp_o: clk_i-domain registered copy of usb_suspend from utmi_if
- resume_req_r: clk_i-domain latched OR of resume_req_i and rf_resume_req, held until utmi_if.suspend_clr

-- CDC Boundaries and Synchronization --

- Clock and reset domains
- phy_clk: UTMI (usbf_utmi_if/usbf_utmi_ls), protocol/path (usbf_pl: usbf_pd/usbf_pa/usbf_pe), IDMA (usbf_idma), memory arbiter (usbf_mem_arb), and the phy-side of the Wishbone front-end (usbf_wb FSM and W-side memory interface).
- wclk (aka wb_clk/clk_i): Wishbone-visible registers and ACK logic (usbf_wb wb_clk side), register-file (usbf_rf and usbf_ep_rf[*]) and top-level status outputs (e.g., susp_o), software-triggered controls.
- Reset: rst_i is applied within each domain (synchronous by default; USBF_ASYNC_RESET optional). If using asynchronous resets, synchronize deassertion per domain; no reset-domain crossing logic is present.

- CDC boundaries and current synchronization
- wb_clk -> phy_clk (Wishbone request path in usbf_wb):
- wb_req_s1 <= wb_stb_i & wb_cyc_i, plus wb_addr_i, wb_we_i, wb_data_i are sampled directly in phy_clk (no 2-FF sync). Safe only if wb_clk == phy_clk.
- phy_clk -> wb_clk (Wishbone ACK/data return in usbf_wb):
- wb_ack_d synchronized via a 3-FF pipeline with edge-detect to produce a single-cycle wb_ack_o (robust).
- Read data ma_din (phy_clk) is observed in wb_clk without an explicit CDC scheme; relies on alignment with wb_ack_o (unsafe if clocks differ).
- phy_clk -> wclk (status/events into RF):
- attach/suspend: 2-FF synchronizers in wclk with edge-detect to sticky events (robust).
- usb_reset, rx_err, nse_err, pid_cs_err, crc5_err, crc16_err, utmi_vend_stat: single-flop sampling in wclk (no 2-FF; unsafe if clocks differ).
- wclk -> phy_clk (software one-shots and control via usbf_rf):
- rf_resume_req, utmi_vend_wr: level-based set/observe/clear without explicit 2-FF on either side; usbf_utmi_ls adds a 2-FF for resume_req internally. Prefer a single intentional synchronizer; current scheme assumes wb_clk == phy_clk.
- clk_i/wclk -> phy_clk (top-level suspend/resume glue):
- resume_req_r drives usbf_utmi_if resume_req without explicit sync; susp_o samples usb_suspend into clk_i with one flop. Unsafe if clocks differ.
- wclk -> clk (register-file traffic in usbf_ep_rf):
- adr/re/we/din written in wclk directly update clk-domain CSRs/BUF regs without synchronizers (unsafe if clocks differ).
- clk -> wclk (interrupts and write-back from usbf_pe/usbf_ep_rf):

- Single-cycle pulses (int_*_set, nse_err, out_to_small, buf*_set, uc_*_set, buf0_rl) and multi-bit idin can be consumed in wclk; no coherent bus+strobe CDC is implemented (unsafe if clocks differ).
- DMA handshake in usbf_ep_rf (bidirectional):
- wclk -> clk (dma_ack): robust 2-FF set/clear handshake (good).
- clk -> wclk (dma_req): combinational condition in clk is consumed in wclk without synchronization (unsafe if clocks differ).

- Single-domain modules (no internal CDC; observe neighbor crossings)
- usbf_utmi_if/usbf_utmi_ls: entirely in phy_clk; resume_req is explicitly 2-FF synchronized. UTMI Rx/Tx handshakes are local to phy_clk. LineState deglitching occurs in-domain.
- usbf_mem_arb: entirely in phy_clk; W-side and M-side handshakes assume phy_clk. wack is consumed by usbf_wb and returned to wb_clk through a robust ACK synchronizer.
- usbf_idma, usbf_pe: entirely in phy_clk; any export to wclk (interrupts, write-back, counters) requires CDC at the RF/Top boundary.
- usbf_crc5/usbf_crc16: combinational; their error pulses are generated in phy_clk and later sampled in wclk (see status/events above).
- usbf_ep_rf_dummy: pure combinational tie-offs; introduces no CDC.

- Assumptions and risks
- Strong assumption: wb_clk (wclk/clk_i) equals phy_clk (frequency and phase). Under this assumption, single-flop samplers and direct pulses generally work, and the hardened ACK path suffices.
- If clocks are asynchronous or not phase-aligned, current weak crossings may cause metastability, pulse loss, and incoherent multi-bit sampling.

- Required hardening if clocks differ
- Wishbone request into phy_clk: 2-FF synchronize a request/toggle; capture addr/we/wdata into phy_clk shadow regs with a coherent req/ack handshake or use a proper dual-clock WB bridge.
- Memory read data back to wb_clk: latch ma_din in phy_clk on ma_ack; transfer with a toggle/req-ack multi-bit scheme or a small dual-clock FIFO; assert wb_ack_o only when synchronized data-valid is seen in wb_clk.
- RF strobes (phy_clk -> wclk): 2-FF synchronize pulses (or convert to toggle); for multi-bit buses (idin), use a bus+valid handshake with source holding data until destination ack.
- RF register writes/reads (wclk -> clk): synchronize write/read strobes (2-FF or toggle with ack) and capture din coherently in clk; avoid combinational evaluation of wclk signals in clk.
- Software one-shots (wclk -> phy_clk): replace level-based set/clear with a toggle/pulse synchronizer and destination-side edge-detect; keep exactly one synchronizer chain (avoid double-sync with downstream).
- Top-level suspend/resume: 2-FF synchronize usb_suspend into clk_i for susp_o; 2-FF synchronize resume_req_r into phy_clk.
- Status/error pulses (phy_clk -> wclk): 2-FF synchronize and source-side stretch to multiple phy_clk cycles, or capture as sticky levels in phy_clk that software clears; then 2-FF sync the level.
- DMA req (clk -> wclk): generate a single-bit request in clk and synchronize to wclk (toggle/req-ack); or mirror needed counters/state into wclk via CDC and evaluate locally.

- Reset CDC
- Keep rst_i per-domain. With asynchronous resets enabled, add per-domain reset-deassert synchronizers to prevent hazards in first-stage flops of CDC paths and to avoid spurious events (e.g., wb_ack_o) on release.

- Tooling and constraints
- Identify and constrain the 3-FF ACK synchronizer (phy_clk -> wb_clk). Mark false paths to first-stage sync flops and set multicycle where applicable.

- If clocks differ, mark old direct cross-domain combinational fan-in/fan-out as false and insert explicit CDC primitives. Add assertions for pulse width and data-valid alignment across domains.

-- Reset Distribution and Power-Up --

- Reset source, polarity, and style
- Single system reset rst_i, active-low. Default build uses synchronous resets in all clock domains; optional USBF_ASYNC_RESET enables selected asynchronous clears in usbf_utmi_if (and may apply to wack_r in usbf_mem_arb). phy_rst_pad_o mirrors rst_i (active-low) to hold the external UTMI PHY in reset.

- Clock domains and reset distribution
- PHY/UTMI domain (phy_clk_pad_i): usbf_utmi_if, usbf_utmi_ls, usbf_pl (pd/pe/pa/idma), usbf_mem_arb core side, and the core-side view of usbf_rf/ep_rf use rst_i synchronously (unless noted above).
- Wishbone/bus domain (clk_i/wb_clk): usbf_wb control in phy_clk; its ACK edge pipeline and data register are in wb_clk (unreset, idle-safe). The bus-view side of usbf_rf (global regs, sticky events, aggregated interrupts) is in wclk and resets synchronously to rst_i.

- Top-level and link-layer power-up
- phy_rst_pad_o = rst_i keeps the external PHY in reset during system reset.
- usbf_utmi_if: RxValid/Active/Error and TxValid clear on reset; other datapath registers without explicit reset are gated and benign until traffic starts. Optional async clears when USBF_ASYNC_RESET is defined.
- usbf_utmi_ls (line-state FSM): state enters POR on reset (and per VBUS gating as coded), applies FS defaults (XcvSelect=FS, TermSel=1), clears mode_hs/usb_attached/usb_suspend, resets attach timers, and then debounces attach (~100 ms) before NORMAL; OpMode has no explicit reset but is driven on first relevant state.

- Protocol/packet/memory path on reset
- usbf_pl: frame/SOF and microframe counters clear; RX/TX/PE/IDMA FSMs idle; no handshakes or DMA enables issued.
- usbf_pd: state=IDLE; rx-valid pipeline cleared; CRC16 seeded by crc16_clr on first rx_active rising edge; CRC5/CRC16 blocks are combinational (no reset).
- usbf_idma: state=IDLE; send_data_r=0; adr_cb=0; sizu_c=0; sizd_c=0x3fff until loaded on first TX; many staging flops unreset but IDLE gating guarantees mreq=0/mwe=0 and no side effects.
- usbf_pe: state=IDLE; in_token/out_token/setup_token=0; timeout counters either held cleared by IDLE gating or qualified so no false events; no descriptor writes or send_token until valid traffic.
- usbf_mem_arb: only wack_r resets to 0; mack=mreq (combinational); W side acknowledged only when M is idle. Keep mreq/wreq low across reset to avoid unintended cycles.

- Register file, endpoints, and interrupts
- usbf_rf (wclk unless noted): funct_adr=0; inta_msk/intb_msk=0; sticky RF events (int_srcb)=0; utmi_vend_ctrl has no explicit reset (defined after first write); edge detectors for attach/suspend settle a few cycles after reset. USB bus reset (usb_reset) only latches a sticky event; it does not assert rst_i.
- usbf_ep_rf (clk): csr0/csr1/ots_stop/iena/ienb/int_stat/uc_dpd/uc_bsel clear to 0; buf0/buf1/buf0_orig=0xffff_ffff (safe sentinels); DMA enable (csr[15])=0 forcing counters to 0; CDC DMA ack synchronizers settle low within a couple of clk cycles. Aggregated inta/intb evaluate low.
- usbf_ep_rf_dummy: fully combinational tie-off; outputs constant (ints=0, dma_req=0, dout=0), independent of reset.

- usbf_wb (Wishbone front end)

- PHY-domain FSM resets to IDLE (ma_req=0, rf_re/we=0). wb_clk ACK edge pipeline is unreset but stays quiescent if wb_cyc_i/wb_stb_i are low during and after reset; wb_data_o must only be sampled with wb_ack_o.

- USB bus reset versus system reset
- System reset (rst_i low) reinitializes all blocks as above and holds the UTMI PHY in reset. USB bus reset (from UTMI/link) does not assert rst_i; it is reported as a sticky RF event and does not auto-clear funct_adr or endpoint state.

- CDC and unreset register considerations at reset release
- Several flops (ACK pipelines, edge detectors, staging buffers) are not explicitly reset but are safe due to IDLE gating and source signals being 0 at reset. Allow 1–2 local clock edges after rst_i deasserts before relying on these paths. Keep WB master idle and prevent memory requests during reset and the first cycles after release.

- Recommended power-up sequence
- Apply rst_i low while phy_clk_pad_i and clk_i/wb_clk are stable (or as they start); hold long enough to meet the external UTMI PHY reset spec (phy_rst_pad_o mirrors rst_i).
- Keep wb_cyc_i/wb_stb_i, mreq, and wreq deasserted during reset and for a few cycles after deassertion.
- Deassert rst_i; allow a few cycles for POR actions and CDC synchronizers to settle. The UTMI link FSM will debounce attach (~100 ms) before NORMAL.
- Program usbf_rf (device address, masks) and per-endpoint CSRs/buffer descriptors via Wishbone before enabling DMA or expecting traffic. Firmware should handle USB bus reset events explicitly if reinitialization is required.

-- Memory Arbiter Integration --

- Role: usbf_mem_arb multiplexes a single 32-bit synchronous external SRAM between two masters in usbf_top: M (protocol/DMA via usbf_idma) and W (software via usbf_wb). The arbiter’s SRAM pins connect directly to the top-level memory interface.
- Port mapping: M: madr/mdin/mdout/mwe/mreq/mack <-> usbf_idma. W: wadr/wdin/wdout/wwe/wreq/wack <-> usbf_wb (ma_*). SRAM: sram_adr_o[SSRAM_HADR:0], sram_dout_o[31:0], sram_din_i[31:0], sram_we_o, sram_re_o.
- Arbitration: Fixed priority M > W. W is granted only when M is idle. Selection holds W through its ack cycle via wsel = (wreq | wack) & !mreq. Preemption is immediate if mreq asserts. No fairness/parking; W can be starved under sustained M traffic.
- Handshakes: M side mack = mreq (combinational, zero wait states added by the arbiter). W side wack is a one-phy_clk-cycle pulse, emitted at most every other phy_clk while W is being served; usbf_wb issues one outstanding request and waits for ma_ack (= wack).
- Data path: Address/data/we are muxed by the selected master; read data (sram_din_i) is broadcast to both mdout and wdout—masters must sample only on their ack. sram_re is tied high; sram_we qualifies the active master only (M: mwe & mreq; W: wwe & wreq). Ensure the external SRAM tolerates constant RE or map to OE/CE accordingly.
- Clocking/CDC: Arbiter logic runs in phy_clk; only wack_r is registered. usbf_wb bridges Wishbone wb_clk to phy_clk for requests and returns a clean 1-cycle wb_ack_o in wb_clk via a synchronized edge detector. Best when wb_clk == phy_clk; otherwise, rely on the synchronized ack for read-data stability (known CDC caveat).
- Addressing/capacity: 32-bit word-addressed, no byte enables. SSRAM_HADR parameter sets address width (e.g., 14 -> ~128 KiB). Sub-word/unaligned writes must use read-modify-write in the master (handled by usbf_idma on M; W software should use aligned 32-bit accesses).
- Throughput: With M idle, W achieves ~1 accepted transfer every two phy_clk cycles (due to pulsed

wack). Under active USB traffic, M dominates and W may be delayed indefinitely. M’s immediate ack means the arbiter adds no latency; real memory timing must be satisfied outside the arbiter.
- Integration with modules: M port is driven exclusively by usbf_idma (orchestrated by usbf_pe). W port is driven exclusively by usbf_wb. UTMI/CRC blocks do not touch the arbiter directly; ep_rf only influences when IDMA runs (via flow-control flags). Dummy endpoints produce no arbiter traffic.
- Traffic shaping (examples): RX (OUT/SETUP) issues a priming read for RMW, then one write per completed 32-bit word plus a final partial flush; CRC16 trailer bytes are not written. TX (IN) issues continuous reads with two-word prefetch; ZLPs issue no reads. Handshake-only, reset/suspend/chirp phases reduce M activity, giving W bandwidth windows. CRC5 failures reduce accepted transactions, indirectly lowering M load; CRC16 failures occur post-DMA and do not reclaim bandwidth during the packet.
- Reset/robustness: wack_r is reset by rst; other arbitration paths are combinational. No internal timeout on W requests (hold until ack). Ensure SSRAM_HADR matches the physical address bus and memory size; meet external SRAM CE/OE/latency requirements.
- Operational guidance: Sample read data only on ack. Plan software/W-port memory accesses during suspend/reset/idle or light-traffic periods to avoid starvation. If a wclk-side DMA uses Wishbone to the same SRAM, it competes on W and is subject to the same starvation risk.

-- Software Integration Guidelines --

- Audience and scope
- For firmware/drivers integrating the USB device core usbf_top via its Wishbone slave and optional external DMA. Covers clocks/reset, address map, initialization, interrupts, buffer/DMA handling, power management, and safety.

Clocks, reset, and CDC
- Two clocks: clk_i (Wishbone/registers) and phy_clk_pad_i (UTMI/protocol/memory). Best practice: clk_i == phy_clk_pad_i or phase-aligned.
- rst_i resets the core and drives the PHY reset. After reset: device address = 0, reprogram endpoints and buffers before enabling traffic.
- If clocks differ, throttle PIO (no back-to-back cycles) and always wait for ACK; expect variable latency.

Address map (example build)
- Region select: wb_addr[17]=0 -> Register File (RF). wb_addr[17]=1 -> External SRAM (packet buffers).
- Global RF registers 0x00..0x05:
- 0x00 main_csr R: {line_state[1:0], attached, mode_hs, suspend}. W: bit5 resume_req (remote wake request).
- 0x01 funct_adr R/W: 7-bit device address.
- 0x02 RF interrupt masks R/W: inta_msk[8:0], intb_msk[8:0].
- 0x03 interrupt sources R: {int_srcb[8:0] sticky, int_srca[15:0] live OR of per-EP}. Read clears RF stickies (int_srcb).
- 0x04 frm_nat R: SOF/microframe timing snapshot.
- 0x05 UTMI vendor status/control (R status[7:0]; W ctrl[3:0] with write strobe). Do not attempt to drive UTMI mode/lines via vendor port.
- Per-endpoint windows: base 0x10 + 4*EP. Offsets: 0 CSR, 1 INT (read-to-clear int_stat; write enables), 2 BUF0, 3 BUF1.
- Memory window (wb_addr[17]=1): CPU access to shared external 32-bit SRAM for packet payloads.

Initialization sequence
- Hold core in reset or ensure bus idle; connect inta_o/intb_o to the system IRQ controller.
- Poll main_csr.attached=1 before enumeration; after each bus reset, re-check main_csr.mode_hs for speed.

- Set funct_adr=0 after reset; program to assigned address after completing the SET_ADDRESS status stage.
- For each enabled EP:
- Program CSR: direction/type/EP number, MaxPacketSize (per speed), policy bits, optionally dma_en (ring mode only if your driver supports it).
- Allocate 32-bit aligned buffers in external SRAM; program BUF0/BUF1 address/size. For OUT, size = available space; for IN, size = payload length (0 => ZLP).
- Configure per-EP INT enables; clear any pending per-EP INTs by reading INT.
- Configure RF interrupt masks at 0x02.

Interrupt model and ISR flow
- Aggregated outputs: inta_o/intb_o = OR(per-EP) | OR(RF stickies masked).
- RF sticky sources at 0x03.int_srcb: usb_reset, rx_err, detach, attach, suspend_end, suspend_start, nse_err, pid_cs_err, crc5_err. Read 0x03 first in ISR to snapshot and clear stickies.
- int_srca[15:0] reflects live OR of per-EP interrupts; then read each EP INT register (clears per-EP bits) to service.
- Recommended ISR:
1) Read RF 0x03; handle usb_reset (reinit address/EPs), attach/detach, suspend/resume; log rx_err/pid_cs_err/crc5_err.
2) Identify EPs with events (via int_srca or scan EP INTs).
3) For each active EP, read its INT (clears), then service buffer completions, CRC16/seqerr/timeout, and short OUT packets; re-prime descriptors promptly.

Endpoint buffer management
- Ownership: Do not modify a buffer’s descriptor or contents while the hardware owns it. Use per-EP completion interrupts to gate ownership changes.
- Memory format: 32-bit words, little-endian by byte lane; only payload bytes are moved. USB CRC16 is generated/checked in hardware and never appears in memory.
- IN endpoints: write payload to SRAM (PIO or system DMA), program BUF size, and arm EP. ZLP: size=0. On completion, use int_buf0_set/int_buf1_set to re-prime that buffer.
- OUT endpoints: program BUF size = capacity. On completion, consume exactly the actual count written back (short packet possible), then re-prime.
- Ping-pong: use both BUF0 and BUF1 for continuous throughput. In ring mode (csr.dma_en=1, non-CTRL), only BUF0 participates; enable only if your driver is cyclic-buffer aware.

PIO vs. external DMA
- PIO through memory window: single 32-bit transfers; always wait for ACK. Expect stalls because the USB core has priority on the SRAM port and PIO is paced to at most one accepted transfer every other PHY clock when idle.
- External DMA (optional): rf exposes dma_req_o[EP] and accepts dma_ack_i[EP]. Map to SoC DMA channels that move data between system memory and USB SRAM buffers; assert dma_ack_i when that channel’s programmed move completes. If unused, leave dma_ack deasserted and use PIO.

Wishbone access guidelines
- Classic single-beat; one outstanding transfer at a time. RF accesses ACK immediately; memory accesses ACK only when served by the arbiter.
- Because requests are sampled in the PHY domain, align clocks when possible; otherwise avoid back-to-back requests and strictly wait for ACK before issuing the next.

Suspend, resume, and remote wake
- Detect suspend_start/end via RF stickies. After suspend_end, defer heavy traffic ~100 µs for PHY settle.

- Remote wake: only when suspended and host has armed it. Write main_csr.resume_req (bit5) once; hardware enforces timing and auto-clears on suspend exit.

Speed negotiation
- After every bus reset, the controller negotiates speed. Poll main_csr.mode_hs and configure per-EP MaxPacketSize and descriptors for HS or FS before enabling traffic.

Control endpoint (EP0)
- Keep EP0 always ready to accept SETUP; do not stall EP0 in normal operation.
- Reinitialize DATA PID bases to DATA0 after reset and when clearing a halt, per USB 2.0.
- Enforce correct control transfer semantics (8-byte SETUP, proper STALL/NAK behavior). Program address after SET_ADDRESS status stage.

Diagnostics and errors
- TOKEN CRC5 failures raise RF sticky crc5_err (diagnostic only; hardware discards invalid tokens). DATA CRC16 failures are per-EP int_crc16_set; discard payload and re-prime.
- Other per-EP indications: timeout (int_to_set), seqerr, upid, out_to_small. Attribute link faults primarily to rx_err; pid_cs_err indicates PID nibble/complement mismatch.
- Log errors with rate limiting; host typically retries.

Safety and ordering
- Treat external SRAM as shared memory. Complete payload moves before arming endpoints; for OUT, ensure space exists before enabling.
- Do not touch descriptors during active transfers; use per-EP interrupts/write-back as the source of truth.
- Avoid long PIO bursts; interleave short, ACK-gated transfers with other work.

Build-time variations and dummy endpoints
- Region select bit and SRAM address width are build-defined (USBF_UFC_HADR, SSRAM_HADR). Use your SoC’s address map.
- Only access enabled endpoints. Dummy EP windows read as zeros (CSR/INT) or 0xFFFF_FFFF (BUF) and ignore writes; they never generate interrupts or DMA requests.

-- Addressing and Alignment Guidelines --

- Global bus rules
- 32-bit word-only everywhere; no byte enables or sub-word writes. Keep all accesses 4-byte aligned (wb_addr_i[1:0]=2'b00).
- One transfer at a time; ACK is a single-cycle pulse per transaction. Sample read data only when ACK is asserted; write completion is indicated by ACK.

- Address map (this build; parameterized in RTL)
- Region select: wb_addr_i[17]=0 → Register File (RF), wb_addr_i[17]=1 → Packet Memory (MEM). This bit and widths may change with parameters; honor top-level generics.
- RF sub-map (word indices): 0x00..0x05 = globals; 0x10+4*N (N=0..15) = per-endpoint window {0:CSR, 1:INT, 2:BUF0, 3:BUF1}. Reserved windows read as 0.

- External SRAM (packet memory) addressing and data lanes
- sram_adr[SSRAM_HADR:0] is a 32-bit word address (not byte). Data ports are 32-bit.
- Effective capacity (bytes) = 4 × 2^(SSRAM_HADR+1); e.g., SSRAM_HADR=14 → 128 KiB.
- Natural little-endian lane order: lowest byte address → data[7:0], then [15:8], [23:16], [31:24].

- Software access to packet memory
- Use only full 32-bit reads/writes via the MEM window; do not attempt byte/halfword updates.
- Read data is broadcast to both masters internally; always qualify with your ACK to avoid sampling the other master’s beat.

- Arbitration and pacing (mem_arb)
- Fixed priority to the USB core (M-side). Wishbone (W-side) memory requests are serviced only when M is idle and are paced to at most one accepted transfer every other phy_clk.
- Avoid long software bursts into packet memory during heavy USB traffic to reduce preemption latency.

- Buffer descriptors and DMA addressing
- BUF0/BUF1 carry byte addresses and byte sizes for payload buffers; NA if buf[31]=1 or address field is all 1s.
- DMA splits adr (byte) into word address + starting byte offset; memory bus remains 32-bit word-aligned end-to-end.
- RX (USB→mem): unaligned starts use read–modify–write on the first word; a final partial word is flushed at EOP. TX (mem→USB): any starting offset is supported; new reads occur on each 4-byte boundary crossing.
- Ring/wrap mode: when dma_en=1 and end-of-buffer reached, addressing wraps to absolute 0. If a non-zero-base ring is required, keep dma_en=0 and manage wrap in software.

- Alignment best practices (strongly recommended)
- Program buffer base addresses 4-byte aligned and buffer sizes as multiples of 4 bytes to avoid RMW on first/last words and to improve throughput.
- Size payload buffers in multiples of the endpoint MaxPacketSize (CSR[10:0]) for clean packetization and internal flow checks.
- Internal word-based thresholds use P = CSR[10:2] (MPS >> 2); choose MPS divisible by 4 to avoid truncation-induced conservatism.

- Endpoint usage notes affecting addressing
- Control endpoints: OUT/SETUP use BUF0; IN uses BUF1. With dma_en=1 on non-CTRL EPs, data moves use BUF0 (ring/DMA path).

- UTMI/CRC data alignment context (not software-addressable)
- UTMI path is 8-bit, byte-aligned; bytes are LSB-first on the wire. CRC5/CRC16 blocks expect LSB-first bytes; no bit reflection needed. These do not alter software-visible addressing.

- CDC and timing context
- Wishbone request sampling and MEM arbitration occur in phy_clk; ACK returns through a synchronizer into wb_clk. For simplest timing, keep wb_clk == phy_clk or rely on provided CDC logic.

- Integration caveats
- Build-time parameters can shift region-select bit and memory depth; update software decode and capacity calculations accordingly.
- Dummy endpoints occupy address space but read as 0 and do not affect alignment rules.

-- Remote Wakeup Integration --

Purpose and scope
- Provide end-to-end USB 2.0 Remote Wakeup (device-initiated resume) from software/external trigger through UTMI PHY signaling, with standards-aligned timing, safe clock-domain crossings, and

automatic request clearing.

Trigger sources and consolidation
- Software: write main_csr bit5 (RF address 0x00) to generate rf_resume_req (wclk -> clk_i one-shot inside usbf_rf).
- External: resume_req_i top-level pin (clk_i domain).
- Consolidation: typically OR rf_resume_req and resume_req_i, then latch into resume_req_r in usbf_top. In the provided build, resume_req is tied to resume_req_i; recommended design ORs both.

Latch/clear semantics (clk_i domain)
- Set: resume_req_r asserts when any request is observed.
- Hold: remains set while suspended; repeated software writes are benign.
- Clear: auto-cleared on suspend exit via suspend_clr_wr (clk_i-retimed copy of suspend_clr from the UTMI link).

UTMI link behavior (phy_clk domain)
- Synchronization: resume_req is two-flop synchronized (resume_req_s) in usbf_utmi_ls.
- Eligibility: honored only while suspended and after ≥ ~5 ms in suspend (T1_gt_5_0_mS).
- Timing sequence: wait ~5 ms wakeup delay (T2_wakeup), drive K for ~1 ms (T2_gt_1_0_mS), settle ~100 µs (T2_gt_100_uS), then return to NORMAL.
- Line control during signaling: OpMode=2'b10 during RESUME_REQUEST/RESUME_SIG; FS termination enabled; drive_k asserted in RESUME_SIG; SuspendM asserted to leave low power; XcvSelect/TermSel managed per negotiated speed and restored after settle (re-enters HS if previously negotiated).
- Tx data safety: usbf_utmi_if forces DataOut=0x00 and gates a bounded TxValid pulse independent of TxReady while drive_k is active, preventing spurious traffic.
- Suspend exit: clears usb_suspend and emits a one-shot suspend_clr used by system logic to clear resume_req_r.

Host precedence and aborts
- Host-initiated resume: detection of long K (k_long) while suspended triggers host resume; any pending local request is benign and will be cleared on suspend exit.
- Reset: SE0-long during suspend forces RESET, aborting any pending device-initiated resume.

Observability and software integration
- Status: main_csr.suspend reflects link suspend; susp_o mirrors suspend in clk_i for system power management.
- Interrupts: suspend_start (on entering suspend) and suspend_end (on exiting suspend) appear in RF sticky int_srcb[3]/[4]; route via inta_msk/intb_msk. There is no dedicated remote-wakeup-done bit; use suspend_end to detect completion.
- Software policy: firmware must only issue requests after the host enables DEVICE_REMOTE_WAKEUP (SetFeature). RTL does not enforce host authorization.

Clock domains and CDC
- rf_resume_req: wclk -> clk_i as a one-shot inside usbf_rf.
- resume_req: clk_i -> phy_clk via two-flop synchronizer inside usbf_utmi_ls.
- suspend_clr: phy_clk -> clk_i as suspend_clr_wr for clearing the top-level latch.

Subsystem impacts
- Protocol engine (usbf_pe), DMA (usbf_idma), packet decoder and CRC blocks (usbf_pd/usbf_crc5/usbf_crc16) remain idle during suspend and through resume signaling; they require no special resets and resume automatically on the next token/data after settle.
- Memory arbitration (usbf_mem_arb) remains operational; Wishbone accesses proceed when M-side

is idle and are preempted once core traffic restarts after resume.
- Wishbone bridge (usbf_wb) remains active; RF read/writes (including the trigger write) ACK immediately; memory path ACKs are paced via the arbiter.

Timing and PHY considerations
- Compliance windows (~5 ms min in suspend, ~5 ms wake delay, ~1 ms K, ~100 µs settle) are enforced by counters in phy_clk and scale with configuration parameters.
- SuspendM assignment deviates from some PHY conventions; verify polarity with the selected UTMI PHY.
- Typical UTMI PHYs keep phy_clk alive in suspend; if system gates phy_clk in deep-suspend, resume timing depends on clock availability.

Recommended integration pattern
- resume_req_mux = rf_resume_req | resume_req_i.
- resume_req_r <= set on request; clear on suspend_clr_wr.
- Feed resume_req_r to usbf_utmi_if/usbf_utmi_ls.
- Use suspend_start/suspend_end for power coordination and completion detection; no software deassert or extra gating required.

Limitations
- No RTL-provided mask to block unauthorized remote wake requests.
- No explicit remote-wakeup start/complete flags beyond suspend_end; firmware may track issuance if needed.

-- Verification and Bring-Up Notes --

- Clocks, resets, CDC
- Start with a single clock: set clk_i == phy_clk_pad_i == wb_clk (12 MHz FS, 60 MHz HS). Avoid asynchronous clocking until CDC is reviewed/hardened.
- Internal reset is largely active-low (if (!rst)); UTMI PHY reset output (phy_rst_pad_o) is active-high. Confirm external PHY reset polarity/wiring.
- Keep bus/handshake strobes low during and just after reset. In usbf_wb, wb_ack sync flops aren't reset; ensure no spurious ACK by holding the bus idle at reset release.

- UTMI/PHY bring-up
- Provide a valid UTMI byte clock and assert VBUS/LineState per scenario. For usbf_utmi_ls, current RTL gates POR/ATTACH oddly with usb_vbus polarity (usb_vbus==1 holds POR); tie usb_vbus low for simulation unless fixed.
- Model TxValid/TxReady properly; during resume/chirp keep TxReady low so special signaling isn't consumed as data. Verify drive_k forces OpMode=10, TermSel per FS rules, and DataOut=0 in parent.
- Timing checkpoints: ~100 ms attach debounce; ~1 ms bus-reset hold before chirp; ~1.2 ms device K; ~1 ms resume drive; >3 ms idle to enter suspend; ≥5 ms before remote-wake.

- Wishbone and register/memory access
- Classic single-beat WB only; deassert STB between transfers. Region select is wb_addr_i[17] (0=RF, 1=MEM). Verify rf_din/ma_din valid on wb_ack_o.
- RF path ACK is fast (W0); MEM path ACK comes via mem_arb wack. Validate W1/W2 spacing prevents double-ACK with back-to-back requests.
- Smoke test RF registers (e.g., main CSR at 0x00, func addr 0x01, masks/status 0x02/0x03) and MEM window R/W to SRAM before enabling USB traffic.

- Memory arbiter and SRAM
- M (core/DMA) has absolute priority; W (WB) is served only when M is idle. W ack pulses every other

phy_clk when M idle. Expect reduced/zero WB bandwidth under sustained DMA.
- sram_re is tied high; ensure the memory tolerates continuous read-enable. Assumed same-cycle write acceptance (mack=mreq). Add a shim if your memory adds write wait states.

- Endpoint RF and programming
- Program per-EP CSR (direction/type, DMA enable, max packet size). Initialize BUF descriptors (size in words at [30:17], byte address in low bits). EPs must be contiguous; EP0 always present.
- Verify ep_match gating: only the selected endpoint instance responds to shared in-band strobes (uc_*_set, buf*_set, buf0_rl, int_*_set, out_to_small). Dummy endpoints read 0 and never assert DMA/INT.
- Observe dma_in_buf_sz1 (IN ready) and dma_out_buf_avail (OUT space) thresholds; ensure software honors them to avoid NACK/NYET.

- IDMA (8-bit <-> 32-bit bridge)
- TX: prefetch 2 words then stream; throttle with rd_next to emulate backpressure. Size accounting decrements on first read-ack and per rd_next. send_zero_length emits ZLP without reads.
- RX: seed with a read-modify of first word, overwrite lanes per adr_cb[1:0], write on each word_done, and flush partial last word at rx_data_done. Support wrap using buf_size; test unaligned starts and boundary cases.
- Abort must cleanly drop to IDLE from any state; no stray mwe/mreq pulses thereafter.

- PE (protocol engine)
- IN: start DMA, transmit, wait for host ACK (non-iso) with rx_ack_to timeout; update descriptors and uc_* on completion. Handle ZLP when MPS=0.
- OUT/SETUP: receive via IDMA; policy checks (sml_ok/lrg_ok), CRC16 errors, seq errors; ACK/NAK/NYET/STALL per type/speed/buffer availability; update descriptors appropriately (DMA vs non-DMA rules, out_to_small in HS).
- HS PING → ACK path; HS OUT no-buffers → NYET. Timeouts: IN ACK wait and OUT data inactivity per HS/FS constants.

- UTMI line-state FSM (usbf_utmi_ls)
- States: POR→ATTACH→NORMAL; RESET detection on SE0 long; SPEED_NEG with chirp K/J to HS; fallback to FS on missing chirps; SUSPEND on idle long; resume via host K or device remote-wake (after ≥5 ms suspend).
- Invariants: mode_hs implies HS xcv and FS term off in NORMAL; OpMode=10 in reset/resume; drive_k implies OpMode=10 and FS TermSel asserted.

- CRC checks
- CRC-5: polynomial x^5 + x^2 + 1, seed=0x1f, 11 LSB-first bits packed as {ADDR[6:0], ENDP[0], ENDP[3:1]}. Receiver compares ones' complement against token1[7:3] only on token_valid from TOKEN completion.
- CRC-16: polynomial 0x8005 reflected (LSB-first), init=16'hFFFF; include CRC bytes in computation; residue at rx_data_done must be 16'h800d on good packets.

- Interrupts and status
- Per-EP INT register is read-to-clear; enables via iena/ienb. RF sticky events at 0x03 clear on read; mask at 0x02. Aggregate inta_o/intb_o should reflect masked OR of sources.
- Quick health: main_csr shows {attached, mode_hs, suspend, line_state}; frm_nat provides (micro)frame timing; VStatus/VControl at 0x05 for vendor sideband.

- Protocol/transaction verification
- PIDs: DATA0/1/2/MDATA, ACK/NAK/STALL/NYET, PING. Validate PID complement and CRC5/CRC16 detection; no ACK on bad packets.

- Flow control and sequencing: NAK/STALL/NYET behavior, HS PING, seq errors, timeouts, zero-length packets, address/endpoint match and non-selected errors.

- Known gaps/caveats
- No HS Test Modes; EP0 SETUP acceptance and some Ch9 edges not enforced. Plan to verify practical OS enumeration and common transfers first.
- Device address on bus reset may need firmware to rewrite 0 at 0x01. Confirm SuspendM polarity/semantics vs PHY expectations.
- CDC risks: usbf_wb samples WB req in phy_clk; keep clocks common initially or add proper synchronizers before async operation.

- Coverage and stress plan
- Speed modes (FS/HS), all PIDs and error paths (CRC/PID/seq), suspend/resume windows, buffer wrap and alignment, WB/MEM contention, long-run randomized bulk/interrupt transfers with concurrent CPU memory accesses.

- Minimal lab checklist
- Assert VBUS and provide UTMI ref clock; release reset with clocks stable.
- Read main_csr (WB 0x00) → attached=1 after debounce; program function address (0x01); configure only EP0 initially.
- Validate RF vs MEM decode via wb_addr_i[17]; confirm single-cycle WB ACKs and correct read data. Exercise SRAM R/W through MEM window before enabling USB traffic.
- FS: observe SOF and frm_nat; run control transfers on EP0. HS: verify reset detect, chirp, mode_hs=1, stable 60 MHz operation. Suspend/resume: host-induced and remote wake via main CSR bit; confirm suspend_clr pulse.

-- Known Limitations and Best Practices --

- USB 2.0 compliance
- Partial device compliance; EP0/SETUP not guaranteed: SETUP can NAK/STALL, does not auto-clear halt or force DATA0 re-sync; status-stage recovery is firmware-driven.
- HS certification modes absent (Test_J/K/SE0_NAK/Test_Packet).
- Reset/address handling relies on firmware to force address=0 after bus reset.
- ISO specifics (bInterval, HS multi-transactions, ISO IN empty ZLP) not enforced in RTL.

- UTMI/PHY assumptions and timing
- 8-bit UTMI at 60 MHz assumed; all timers derive from this clock. If different, recalibrate macro counts (2.5 µs, 100 µs, 1.0–1.2 ms, 5 ms, 100 ms).
- Known issue: 1.0 ms and 1.2 ms thresholds share the same count; fix or add margin.
- SuspendM/XcvSelect/OpMode polarities/encodings are PHY-dependent; verify and wrap if needed.
- During drive_k, TxValid may assert with DataOut=0x00; ensure PHY tolerates it or mask TxValid in this mode.
- VBUS input gates POR level-sensitively; tie correctly or gate to avoid stuck POR when unused.

- Clocking, reset, and CDC
- Two domains (wb_clk, phy_clk) with minimal CDC. Safest when wb_clk == phy_clk.
- usbf_wb samples WB request in phy_clk without full sync; ACK returned via non-reset flops—hold WB idle for several wb_clk cycles after reset.
- usbf_ep_rf crosses re/we/adr/din (wclk) into clk without synchronizers; also samples multi-bit status back into wclk. Add CDC bridges if clocks differ.
- Top resets are synchronous unless USBF_ASYNC_RESET is used; validate both domains.

- External memory and arbitration
- usbf_mem_arb: fixed priority (M over W) can starve software; wack at most every other phy_clk when M idle.
- M-side mack=mreq (immediate), sram_re=1; no wait states. External SSRAM must meet timing or use a shim that stretches requests and registers data.
- No byte enables; all writes are 32-bit. Read data broadcast to both masters—sample only on own ack.
- Address map: wb_addr_i[17]=1 selects MEM; SSRAM_HADR must match physical memory depth.

- IDMA constraints (usbf_idma)
- Expects single-cycle acks; write acks in same cycle as request or they are missed. Pair with usbf_mem_arb or add a stretcher.
- RMW used for unaligned RX starts and partial ends; word-align buffers and sizes (multiples of 4) to reduce overhead.
- TX: 2-word prefetch; sensitive to read latency. RX primes with a read (requires readable memory).
- Counters limited (TX 14 bits ≈ 16 KB, RX 11 bits ≈ 2 KB per transfer). Abort leaves stale prefetched data—reinit before reuse.
- Little-endian byte-lane mapping on 32-bit side.

- Protocol engine policies (usbf_pe)
- SETUP acceptance/spec recovery not enforced in HW; some control OUT length violations NAK instead of STALL.
- Timeouts use fixed cycle counts; scale if clk ≠ 60 MHz.
- New token preempts in-flight work; glitchy match signals can spuriously abort.
- DMA uses only buf0; HS OUT with DMA expects full MPS—short packet flags out_to_small and freezes address.
- HS PING always ACK; HS NYET returned only when no buffers available.
- uc_* commits can be deferred by new matches; transient pid_seq_err possible; interrupts are pulses (not sticky).

- Endpoint RF (usbf_ep_rf)
- CDC on bus and status; keep clocks common or add synchronizers.
- INT enable quirk: int_stat[4] shares enable index with [3]; no dedicated enable.
- DMA counters are 12-bit words (≤16 KB ring). P=MPS/4; MPS < 4 bytes rounds to 0.
- 0xFFFF_FFFF used as "not available" sentinel; never program real buffers at that address.
- Duplicate EP numbers cause multiple instances to latch in-band updates; avoid duplicates. Avoid SW writes racing with HW updates (e.g., out_to_small into BUF1).

- Wishbone front-end (usbf_wb)
- Single-beat classic cycles only; no bursts/STALL/ERR/RTY. Full-word accesses only.
- No timeouts; MEM access stalls indefinitely if ma_ack is delayed/missing.
- Throughput: ~3 phy_clk cycles per RF transfer; MEM path adds ma_ack latency.

- CRC blocks
- usbf_crc5/usbf_crc16 are combinational with deep XOR trees; pipeline around them if timing is tight.
- Fixed LSB-first polynomials; seed 5'b11111 (CRC5) and 16'hFFFF (CRC16). USB uses ones' complement—handle invert/residue correctly.
- X propagate on unknowns; gate usage until valid token/data captured. In usbf_pd, gate crc5_err with token completion (not ACK) and ensure CRC16 gating includes PHY CRC bytes if using residue method.

- Build-time configuration and observability
- Endpoint set is compile-time; changing EPs requires resynthesis; keep numbering unique and

contiguous as expected by build.
- External SSRAM is 32-bit wide; effective capacity set by SSRAM_HADR.
- Vendor UTMI sideband limited (4-bit control + strobe); ensure width matches PHY needs.
- frm_nat is diagnostic only; not monotonic across resets. susp_o lags UTMI suspend by CDC.

- Software responsibilities and best practices
- Implement full Chapter 9: always accept SETUP, clear halts, re-sync DATA0, enforce 8-byte SETUP and STALL-on-mismatch, manage status-stage ZLPs.
- On bus reset, immediately write address=0 and reinit EP toggles/descriptors.
- Program accurate descriptors (wMaxPacketSize, bInterval); enforce polling and HS high-bandwidth rules in firmware.
- Align buffers to 32-bit and size in multiples of 4; avoid all-ones addresses; size buffers ≥ one MPS; use double-buffering and DMA for HS.
- Avoid heavy WB memory traffic during active USB; prefer IRQ-driven servicing and batching; pre-fill IN, keep OUT space available.
- Remote wake: assert only after ≥5 ms suspend; FSM enforces timing but software policy must comply.
- Validate UTMI pin polarities/encodings and recompute timer constants for actual UTMI clock; test suspend/reset/HS chirp on hardware with an analyzer.
- Interrupt handling: read per-EP INT first (read-to-clear), then RF int_src; consider read-twice on RF stickies to avoid races; remember int_srca is live (not sticky).

- Verification targets
- CDC crossings with unrelated clocks, MEM shim/latency, ring wrap and RMW edge cases, abort-on-new-token, HS PING/NYET, ISO vs non-ISO sequencing, CRC residue/ordering, SETUP and EP0 policy paths.