Generated Specification Document

INTRODUCTION

-- Purpose and Scope --

Purpose
- Provide a synthesizable, top-level USB 2.0 device (function) core that bridges a UTMI PHY, a Wishbone host interface, and a 32-bit synchronous external SSRAM used as endpoint buffer memory. The core delivers end-to-end device functionality including link management, packet processing, endpoint policy, memory-backed buffering, and software-visible control/interrupts.

Scope
- Interfaces and integration:
- UTMI PHY front end with link/power-state control: attach debounce, reset detection, suspend/resume, and FS/HS negotiation; drives UTMI control pins and adapts byte-wide Rx/Tx with proper TxValid/TxReady handshakes.
- Wishbone B3-style slave for control/status and for software access to the external buffer memory; single-cycle ack pulses per access.
- External 32-bit synchronous SSRAM as the endpoint buffer store, arbitrated with fixed priority to the USB side to meet bus timing.
- Core functionality included:
- Protocol layer: packet decode/encode (PID/CRC5/CRC16), OUT/IN handshakes (ACK/NAK/STALL/NYET), data toggle management, transfer sizing, HS/FS timing windows, and SOF/microframe timing snapshot (frm_nat).
- Internal byte↔word DMA to/from SSRAM for payload moves; per-endpoint DMA request/ack exposure to the system.
- Register file (RF): global/system CSRs (function address, link status, vendor UTMI access, frame timing, interrupt masks/sources) and up to 16 endpoint windows (CSR/INT/BUF0/BUF1), with masked interrupt aggregation onto inta_o/intb_o.
- Memory arbiter: single-port, 32-bit word interface shared between protocol (master, highest priority) and Wishbone (when idle), with read-data broadcast and W-side pulsed/toggling acks.
- Clocking and CDC:
- Two clock domains: phy_clk (UTMI/link/protocol/mem arbiter) and clk_i (Wishbone/RF). All necessary crossings are handled inside submodules (utmi_if, rf, wb).
- Configuration and addressing:
- Compile-time split between RF vs MEM regions via a single high address bit (HADR = 17 in test builds, 12 otherwise).
- Buffer memory size parameterized by SSRAM_HADR; memory is word-addressed (byte lanes [1:0] dropped).
- Endpoint presence via macros; EP0 always present; enabled EPs must be contiguous from 0.
- Power and management signals:
- Suspend indication (susp_o), remote-wakeup request via external input (resume_req_i) and software-triggered pulse (RF), line-state and vendor-status latching for CPU visibility.

Out of scope/assumptions

- The UTMI PHY, the external SSRAM device, and any system DMA engine beyond the provided per-EP req/ack signals are external to this core.
- Application/class-specific firmware, USB descriptors, and detailed EP0 request handling policies are software responsibilities configured via the RF.
- Wishbone optional features (bursts, byte selects, error/retry, stalls) are not implemented; accesses are classic, single-beat with one-shot ack.
- Certification-oriented HS test modes and other non-functional test features are not explicitly implemented here.

Intended use
- Integrate with a UTMI-compliant PHY, a Wishbone-based CPU/SoC, and a single-port 32-bit SSRAM. Software configures endpoints and buffers via the RF region, moves/inspects payloads via the MEM region when the USB master is idle, and services interrupts on inta_o/intb_o. The arbiter ensures USB timing by preempting CPU memory access when required.

-- Features and Capabilities --

Features and Capabilities

Core integration
- UTMI PHY interface (8-bit data): TxValid/TxReady, RxValid/RxActive/RxError; LineState, XcvSelect, TermSel, OpMode[1:0], SuspendM
- Wishbone B3-style slave for control/status and memory access; single-cycle ack pulse per access
- External 32-bit synchronous SSRAM endpoint buffer memory with fixed-priority arbiter (USB side preempts CPU side)

USB link and power management
- Full-Speed and High-Speed device operation with HS negotiation (chirp K/J counting, auto FS/HS select)
- Suspend detection (>3 ms idle), host-initiated resume, and device-initiated remote wakeup
- Debounced attach detection, USB reset detection and status export
- UTMI vendor sideband: 4-bit VControl write with load strobe; 8-bit VStatus readback

Protocol layer
- Packet decode/encode with PID integrity check, CRC5 (tokens) and CRC16 (data)
- Token filtering (address match, CRC clean, ignore hub-only/unsupported PIDs; PING gated to HS)
- Handshakes: ACK/NAK/STALL/NYET; ZLP support
- Data toggle management (selection and update) across IN/OUT/control flows
- SOF and HS microframe tracking; frm_nat snapshot (frame, microframe index, time since SOF)
- Speed-aware timing windows and response timeouts (HS/FS)

Endpoints and DMA
- Up to 16 endpoints (EP0 always present; others via compile-time macros); contiguous EP numbering enforced
- Per-endpoint windows: CSR, INT (read-to-clear), BUF0, BUF1 (32-bit descriptors)
- Internal byte↔word DMA (idma) between USB byte stream and 32-bit memory; handles unaligned starts and partial last words; ring wrap option
- Per-endpoint external DMA request/ack exposure (dma_req_o/dma_ack_i) with flow watermarks (dma_in_buf_sz1, dma_out_buf_avail)

Memory/arbiter
- Single-port 32-bit bus to SSRAM; M-side (protocol) zero-wait mack=mreq; W-side (Wishbone)

pulsed/toggling ack when M idle
- Read data broadcast to both masters; word-addressed W-side (drop [1:0])

Registers, interrupts, and software model
- System RF bank: MAIN_CSR (status + SW remote wake), FUNCTION_ADDRESS, INT masks, RF/system INT source (read-to-clear), frm_nat, UTMI vendor access
- Per-EP INTs (read-to-clear) with masks inside EP; two aggregated interrupt outputs (inta_o, intb_o) combine per-EP and masked RF/system events
- RF/MEM address space split by a single high address bit (parameterized HADR)

Clocking and CDC
- Two clock domains: phy_clk (UTMI/link/protocol/arbiter) and clk_i (Wishbone/RF)
- Safe crossings inside submodules for vendor writes, resume requests, RF/system event edges, and WB ack

Configurability and sizing
- Parameter SSRAM_HADR sets buffer memory address width and total size (bytes = 2^(SSRAM_HADR+3))
- Compile-time selection of RF/MEM split address bit (HADR) and enabled endpoints

Diagnostics and simulation aids
- frm_nat timing snapshot for scheduling/diagnostics
- Simulation prints: enabled endpoints, address sizes, computed buffer size; checks for EP contiguity

Non-features/limits (by design)
- Wishbone optional features not implemented: byte selects, bursts, error/retry, pipeline/STALL
- No explicit HS test modes (Test_J/Test_K/SE0_NAK/Test_Packet/Force_Enable) in this top
- EP0/request policy specifics are software-driven via RF; hardware provides mechanisms (handshakes, DMA, descriptors)

-- Standards Compliance --

Standards Compliance

USB 2.0 device (partial compliance)
- Implemented behaviors aligned with USB 2.0:
- UTMI 8-bit PHY interface: LineState, XcvSelect, TermSel, OpMode[1:0], SuspendM; TxValid/TxReady, RxValid/RxActive/RxError
- Link state handling: attach debounce, reset detection (SE0), suspend (>3 ms idle), host/device resume (remote wake timing), HS negotiation via chirp K/J counting
- Transaction layer: PID integrity check, token CRC5 and data CRC16, address/endpoint match gating, ACK/NAK/STALL/NYET handshakes, data toggle management, SOF/frame and HS microframe tracking, HS/FS response windows and timeouts, ZLP support
- Error handling: ignore on CRC/protocol errors; event/interrupt reporting for pid_cs_err, crc5_err/crc16_err, timeouts
- Optional/HS features present: NYET (HS), PING acceptance gated to HS, remote wake signaling capability (device-initiated resume)
- Known gaps vs full USB-IF certification:
- HS electrical test modes not implemented (Test_J, Test_K, SE0_NAK, Test_Packet, Force_Enable)
- Default control pipe (EP0) strict hardware rules not fully enforced in RTL (e.g., guaranteed SETUP acceptance while halted/buffers full; length-mismatch STALL vs NAK; explicit Status stage ZLP handling)
- Function address not forcibly cleared to 0 on bus reset in hardware

- Remote-wakeup enable feature gating not enforced in hardware
- Interoperability assumptions:
- Compliance depends on an external UTMI-compliant PHY; UTMI vendor sideband interface (VControl/VStatus) supported but vendor-specific
- USB enumeration, descriptors, and class/policy for EP0 handled by firmware via the RF register set

Wishbone bus
- Closest match: Wishbone B3 (classic) slave
- Base signal set and classic one-beat ACK handshake implemented (CLK_I, RST_I, ADR_I[n:2], DAT_I/O, WE_I, CYC_I, STB_I, ACK_O)
- Read data qualified by ACK_O; writes complete on ACK_O; synchronous to clk_i at the WB boundary
- Not implemented (acceptable for B3 classic): SEL_I, ERR_O/RTY_O, STALL_I, CTI/BTE bursts, tags, locks, byte enables
- Not WB B4 compliant (stricter CDC/reset/CYC-qualification expectations not met by design)

Other interfaces
- External memory: generic 32-bit synchronous SRAM-like interface (no formal standard claimed)

Cryptographic/CRC conformity
- Token CRC5 polynomial x^5 + x^2 + 1 and Data CRC16 polynomial x^16 + x^15 + x^2 + 1 implemented with USB LSB-first convention

Certification note
- The core targets USB 2.0 device behavior and WB B3 classic bus semantics. Passing USB-IF certification will require addressing the noted gaps (HS test modes, EP0/reset rules) and validating with a compliant UTMI PHY.

-- External Dependencies and Assumptions --

External Dependencies and Assumptions

External hardware
- UTMI PHY (8-bit) required
- Must be UTMI-compliant and provide phy_clk_pad_i appropriate to operating speed (e.g., HS 60 MHz; FS per PHY).
- Exposes LineState[1:0], RxValid/RxActive/RxError, TxReady; accepts DataOut[7:0]/TxValid and control pins XcvSelect, TermSel, OpMode[1:0], SuspendM.
- usb_vbus_pad_i must reflect VBUS presence (used as POR source in the link FSM).
- Optional vendor sideband: VControl_pad_o[3:0]/VControl_Load_pad_o and VStatus_pad_i[7:0]; functionality is PHY-specific.
- External 32-bit synchronous SSRAM (endpoint buffer memory)
- Single-port, word-wide interface: sram_adr_o[SSRAM_HADR:0], sram_data_i[31:0], sram_data_o[31:0], sram_we_o, sram_re_o.
- Assumed single-cycle synchronous behavior in phy_clk domain; read data must be valid in the same cycle as the arbiter’s ack (M side uses mack=mreq) and tolerate sram_re_o held asserted.
- No wait-state/ready signaling is supported; writes occur when sram_we_o is asserted.
- Wishbone host (CPU/SoC)
- Classic B3-style single-beat slave usage with signals clk_i, rst_i, wb_adr/dat/we/stb/cyc/ack.
- Master must hold address/data/request stable until wb_ack_o one-shot pulse; no byte enables, bursts, errors/retries, or stalls supported.

External system blocks (optional)

- System DMA: may consume dma_req_o[15:0] and drive dma_ack_i[15:0] per endpoint.
- Handshake is in the wclk/clk_i domain (as implemented in usbf_rf); dma_ack_i should be clean, single-cycle pulses synchronous to clk_i.

Clocking and CDC assumptions
- Two independent clock domains: phy_clk_pad_i (UTMI/link/protocol/memory arbiter) and clk_i (Wishbone/RF/CPU).
- wb↔phy crossings use simple level/pulse synchronization; correctness assumes WB master keeps requests stable through ack and that ma_din/rf_din are valid when wb_ack_o pulses.
- usbf_mem_arb operates only on phy_clk; its W-side requester must already be synchronous to phy_clk (no CDC inside the arbiter).

Firmware/software responsibilities
- USB enumeration, descriptors, and EP0 policy (SETUP/DATA/STATUS rules) are handled in software via the RF register set.
- Remote wakeup policy gating (DEVICE_REMOTE_WAKEUP feature) is a firmware responsibility; hardware provides resume request mechanisms only.
- Programming of endpoint CSRs, BUF0/BUF1 descriptors, and memory payload management in the MEM region.
- Servicing interrupts on inta_o/intb_o; clearing read-to-clear INT sources (system and per-EP).

Build-time configuration dependencies
- Parameters/macros must be set consistently:
- SSRAM_HADR (buffer memory size/width), USBF_UFC_HADR (RF/MEM address split bit), USBF_HAVE_EPn (enabled endpoints must be contiguous from 0; EP0 always present).
- Optional USBF_ASYNC_RESET alters reset style; system reset strategy must match.

Reset and power assumptions
- rst_i drives phy_rst_pad_o and resets top-level logic; utmi link FSM additionally uses usb_vbus as POR.
- Proper VBUS indication and power sequencing are required for correct attach/debounce and reset detection.

Arbitration and performance expectations
- USB side (protocol layer) preempts CPU/Wishbone memory access at any time to meet USB timing; software must tolerate variable latency and a pulsed/toggling W-side ack cadence from the arbiter.
- Read data from SSRAM is broadcast to both masters; only the selected side should sample it on its ack.

Data formatting/Addressing
- Wishbone MEM region is word-addressed (byte lanes [1:0] dropped by the arbiter); RF region uses 32-bit word addresses.
- BUF descriptors must be word-aligned; sizes are in bytes; idma packs/unpacks bytes into 32-bit words internally (endianness toward memory is implementation-defined but consistent within the core).

Compliance-related assumptions/limits
- Passing USB-IF HS certification requires an external UTMI PHY and addressing noted gaps (e.g., HS test modes, strict EP0/reset rules) in the overall system.
- No formal standard is claimed for the external SRAM interface beyond a generic synchronous single-port contract.

Diagnostics/simulation

- Initial reporting (simulation) depends on macros and prints configuration, memory size, and endpoint contiguity; not synthesized.

-- Supported Modes and Speeds --

Supported Modes and Speeds

USB link speeds
- High-Speed (HS, 480 Mb/s): Auto-negotiated after reset via HS chirp (K/J counting to 3 pairs). mode_hs=1, HS microframes (125 µs) tracked; NYET and PING (HS-only) supported.
- Full-Speed (FS, 12 Mb/s): Operates when HS negotiation is not completed; SOF frames at 1 ms. mode_hs=0.
- Low-Speed (LS): Not supported by the core (UTMI FS/LS transceiver control is present, but protocol engine targets FS/HS).

USB device and link states
- Device role only (no host/OTG).
- States: DETACHED/ATTACH (VBUS debounce) → NORMAL (active) → SUSPEND (>3 ms idle) → RESUME (host or device-initiated) → RESET (SE0 detection and handling).
- Device-initiated remote wakeup supported (resume signaling after >5 ms idle); enable/policy handled by software.

UTMI/PHY operating modes
- Transceiver select: FS/LS vs HS selection per negotiated speed.
- Termination control: FS TermSel on/off as required.
- OpMode: normal vs non-driving used during reset/resume/chirp sequences.

Endpoint/transfer modes
- Endpoint types: Control (EP0), Bulk, Isochronous, Interrupt.
- Directions: IN/OUT; ZLP supported.
- Up to 16 endpoints (EP0 mandatory; others build-time selectable, contiguous from 0).

Timing granularity and accounting
- HS microframes (125 µs) and FS frames (1 ms) supported; frm_nat captures microframe index, frame number, and time since SOF.
- Speed-aware response windows and timeouts (distinct HS/FS thresholds) for ACK/data.

Bus/memory/DMA operating modes
- Wishbone: Classic B3 single-beat slave (no bursts/byte-select/stall); one-shot ACK per access.
- Memory: Single-port 32-bit SSRAM at phy_clk; USB-side (M) has preemptive priority; CPU-side (W) serviced when idle.
- DMA: Internal byte↔word DMA (idma) for payload moves; optional external per-EP DMA handshake (dma_req/dma_ack).

Unsupported/omitted modes
- HS electrical test modes (Test_J/Test_K/SE0_NAK/Test_Packet/Force_Enable): Not implemented.
- Low-Speed device operation (protocol level): Not supported.
- Host/OTG operation: Not supported.

Clocking expectations by mode
- phy_clk sourced by UTMI PHY (e.g., 60 MHz at HS; FS as provided by PHY); core logic for link/protocol/memory runs on phy_clk.

- clk_i (Wishbone/RF) is independent; CDC bridges provided internally.

-- Use Cases and System Context --

Use Cases and System Context

Typical system integration
- Device-side SoC with Wishbone CPU connects to:
- A UTMI-compliant USB 2.0 PHY (8-bit) for line-state, Rx/Tx, suspend/resume, and HS/FS negotiation
- A single-port 32-bit synchronous external SSRAM as endpoint buffer memory
- Optional system DMA engine consuming per-endpoint dma_req_o[15:0] and returning dma_ack_i[15:0]
- Two clock domains: phy_clk (UTMI/link/protocol/memory arbiter) and clk_i (Wishbone/register file). Internal bridges handle CDC.
- Software configures the core via the RF (register file) over Wishbone, programs endpoint descriptors, moves or inspects payloads in the MEM region, and services interrupts on inta_o/intb_o.

Representative use cases (device functions)
- Mass storage–like bulk transfers (HS or FS):
- Use bulk IN/OUT endpoints, double buffering via BUF0/BUF1, large payloads staged in SSRAM
- Optional external DMA to sustain throughput; CPU manages descriptors and status via RF
- CDC/ACM (virtual COM) and HID devices:
- Control EP0 plus interrupt/bulk endpoints; low-latency event delivery via inta_o/intb_o
- Firmware upgrade/bootloader over USB:
- EP0 control and bulk endpoints for image transfer; software-driven policy and integrity checks
- Isochronous audio/video streaming (HS/FS):
- Isochronous IN/OUT endpoints using frm_nat microframe timing for scheduling
- Internal idma for sustained byte↔word moves to/from SSRAM
- Vendor-specific/device-test interfaces:
- Flexible endpoint mix and UTMI vendor sideband access (VControl/VStatus) for PHY-specific features

Operational context and flows
- Enumeration and policy:
- Software provides descriptors and EP0 control-state handling; hardware enforces PIDs/CRCs/handshakes and data toggles
- Data path:
- USB protocol layer masters the memory arbiter (preemptive priority) to meet link timing; CPU/Wishbone accesses SSRAM when USB master is idle
- Internal idma handles byte↔word packing, unaligned bursts, and partial words
- Power management:
- Suspend detection (>3 ms idle) and host/device-initiated resume (remote wake request via RF or resume_req_i)
- Speed modes:
- HS auto-negotiation via chirp; FS operation when HS not negotiated; LS not targeted at the protocol layer

Integration variants
- With external DMA: use dma_req_o/dma_ack_i per EP to offload CPU; RF exposes watermarks (dma_in_buf_sz1, dma_out_buf_avail)
- CPU-only: Wishbone software reads/writes MEM region opportunistically (ack pulses/toggling cadence from arbiter) while respecting USB preemption
- Configurable capacity and footprint: SSRAM_HADR sets buffer size; compile-time macros select

number of endpoints (contiguous from 0)

System constraints and expectations
- Certification-oriented HS test modes are not implemented; strict EP0/reset behaviors are software-managed
- External SSRAM is assumed synchronous single-port; UTMI PHY supplies phy_clk and line-state semantics
- Wishbone interface is classic B3 single-beat (no bursts/byte-select/stall); address/data must be held stable until ack

Target platforms
- FPGA or ASIC SoCs with Wishbone-based CPUs (e.g., open-source RISC-V/OpenRISC) and an external UTMI PHY; evaluation boards or production designs requiring USB 2.0 device connectivity

-- Limitations and Out-of-Scope --

Limitations and Out-of-Scope

USB protocol/features
- Partial USB 2.0 device compliance: default control pipe (EP0) policy is firmware-driven; hardware does not enforce all mandatory EP0/reset rules (e.g., guaranteed SETUP acceptance when halted, length-mismatch STALL behavior, automatic function-address clear to 0 on reset)
- High-Speed electrical test modes not implemented: no Test_J, Test_K, SE0_NAK, Test_Packet, or Force_Enable
- Low-Speed device operation not targeted by the protocol layer (FS/HS only)
- Device role only: no host or OTG functionality
- Remote-wakeup feature gating (DEVICE_REMOTE_WAKEUP) not enforced in hardware; software must manage policy
- Advanced power management (USB LPM/U1/U2) not implemented

PHY and interface scope
- UTMI 8-bit PHY interface only (no ULPI/UTMI+ direct support); vendor sideband (VControl/VStatus) is PHY-specific and optional

Endpoint/memory/DMA limits
- Maximum of 16 endpoints; non-EP0 endpoints must be enabled contiguously from EP1 upward at build time
- Internal DMA sizing limits: TX size counter 14 bits (up to 16 KB), RX byte counter 11 bits (up to 2 KB)
- External buffer memory must be single-port, synchronous, 32-bit wide; no wait-state/ready handshake supported (M-side is zero-wait, W-side paced by toggling ack)
- Wishbone MEM region is word-addressed (byte lanes [1:0] dropped); no byte-level writes/reads to buffer memory via WB

Wishbone bus constraints
- Classic B3 single-beat only: no bursts (CTI/BTE), no byte selects (SEL_I), no pipeline/STALL, no error/retry (ERR/RTY)
- WB-to-phy clock crossing uses simple level/pulse schemes; masters must hold address/data stable until ack; fully asynchronous clocking across domains may require system-level constraints

Scheduling/timing
- Speed-aware timeouts and response windows are fixed by compile-time constants; no dynamic reconfiguration at runtime
- Isochronous scheduling assistance limited to frm_nat timing snapshot; no hardware

bandwidth/reservation manager

Functionality explicitly out-of-scope
- USB class-specific logic, descriptor storage, and enumeration policy (firmware responsibility)
- Security/encryption/compression beyond USB CRCs
- Hub functionality, test harnesses, and certification utilities
- Multi-port/multi-PHY or multi-function aggregation beyond provided endpoints

Dependence on external components
- Correct operation requires a standards-compliant UTMI PHY (supplying phy_clk and line-state semantics) and a synchronous SSRAM; deviations from these assumptions are out of scope

ARCHITECTURE

-- Top-Level Block Diagram and Composition --

Top-Level Block Diagram and Composition

Blocks and roles
- UTMI PHY (external)
- 8-bit data path; LineState[1:0], RxValid/RxActive/RxError, TxReady; control pins XcvSelect, TermSel, SuspendM, OpMode[1:0]; vendor VStatus[7:0]
- usbf_utmi_if (UTMI front-end + link/power)
- Adapts UTMI to core; provides rx_data/valid/active/err and tx_ready; accepts tx_data/tx_valid/tx_valid_last/tx_first
- Link state machine (via usbf_utmi_ls): attach debounce, reset detect, HS chirp, suspend/resume; drives UTMI control pins; exposes mode_hs, usb_reset, usb_suspend, usb_attached; generates suspend_clr
- usbf_pl (protocol layer)
- Packet decode/encode, endpoint policy, HS/FS timing and SOF/microframe accounting (frm_nat)
- Sub-functions: usbf_pd (RX decode), usbf_pa (TX assemble + CRC16), usbf_idma (byte↔word DMA), usbf_pe (policy/endpoint engine)
- usbf_mem_arb (SSRAM arbiter)
- Fixed-priority mux to external 32-bit SSRAM; M-side (protocol) preempts W-side (Wishbone)
- usbf_rf (register file + endpoint mux + interrupts/DMA)
- System regs, per-EP windows (CSR/INT/BUF0/BUF1), aggregated interrupts (inta_o/intb_o), vendor UTMI access, per-EP DMA conduit
- usbf_wb (Wishbone bridge)
- Decodes RF vs MEM region (by HADR bit), sequences RF strobes or forwards memory requests to arbiter W-side; generates one-shot WB ACK
- External SSRAM (endpoint buffer memory)
- Single-port, 32-bit, synchronous
- Wishbone host (external)
- Classic B3 master to configure RF and access MEM region
- Optional external DMA (system)
- Per-EP dma_req_o/dma_ack_i handshake

Interconnect (data/control flow)

- USB RX path: UTMI PHY -> usbf_utmi_if (rx_data/valid/active/err) -> usbf_pl.usbf_pd -> usbf_pl.usbf_idma (mreq/mwe) -> usbf_mem_arb (M) -> SSRAM
- USB TX path: SSRAM -> usbf_mem_arb (M) -> usbf_pl.usbf_idma -> usbf_pl.usbf_pa (Data PID + CRC16) -> usbf_utmi_if (tx_data/tx_valid/tx_first/tx_valid_last, tx_ready) -> UTMI PHY
- Endpoint control/status:
- usbf_rf multiplexes per-EP CSR/BUF0/BUF1 and flow flags to usbf_pl (csr, buf0, buf1, dma_in_buf_sz1, dma_out_buf_avail)
- usbf_pl broadcasts idin[31:0] + strobes (buf0_set, buf1_set, uc_bsel_set, uc_dpd_set, buf0_rl; int_*_set) back to usbf_rf; only the selected EP (ep_match) latches updates
- Interrupt aggregation:
- per-EP inta/intb (level) ORed in usbf_rf with masked RF/system events -> inta_o, intb_o
- Wishbone access path:
- Wishbone host -> usbf_wb
- RF region (RF_SEL): rf_re/rf_we_d strobes to usbf_rf; rf_dout -> wb_data_o
- MEM region (MEM_SEL): ma_req/ma_we to usbf_mem_arb W-side; ma_din -> wb_data_o; W-side handshake via wreq/wack
- Memory arbiter:
- M-side (protocol layer): madr/mdin/mdout/mwe/mreq/mack (mack=mreq) to SSRAM
- W-side (Wishbone): wadr/wdin/wdout/wwe/wreq/wack to SSRAM
- sram_adr/sram_data_o/sram_data_i/sram_we/sram_re connect to external SSRAM
- Power and vendor sideband:
- resume_req_i and rf_resume_req feed remote-wakeup; usbf_utmi_if returns suspend_clr
- usbf_rf drives VControl[3:0] and VControl_Load to UTMI PHY; latches VStatus[7:0] for CPU readback

Addressing and regions
- RF vs MEM split by HADR bit (compile-time: 17 test, 12 non-test)
- RF region: system regs and 16 EP windows (each 4 words: CSR/INT/BUF0/BUF1)
- MEM region: external SSRAM buffer space; Wishbone addresses are word-based (drop [1:0])

Clocks and resets (CDC)
- phy_clk_pad_i domain: usbf_utmi_if, usbf_pl (incl. idma), usbf_mem_arb, wb control FSM
- clk_i domain: usbf_rf and Wishbone-visible flops; usbf_wb ACK generator
- CDC bridges:
- wb_req/ack between wb_clk (clk_i) and phy_clk FSM; one-shot ack returned to WB
- rf vendor write and resume request crossed wclk->clk (and into phy_clk via utmi_if)
- RF/system event edges crossed into wclk for interrupt latching
- Reset: phy_rst_pad_o tied to rst_i; utmi_ls also considers usb_vbus as POR source

Top-level I/O groupings
- Wishbone/system: clk_i, rst_i, wb_adr/dat_i/dat_o/we/stb/cyc/ack, inta_o, intb_o
- UTMI PHY: phy_clk_pad_i, phy_rst_pad_o, DataIn/DataOut[7:0], TxValid/TxReady, RxValid/RxActive/RxError, XcvSelect, TermSel, SuspendM, OpMode[1:0], LineState[1:0], usb_vbus_pad_i
- UTMI vendor: VControl_Load_pad_o, VControl_pad_o[3:0], VStatus_pad_i[7:0]
- External SSRAM: sram_adr_o[SSRAM_HADR:0], sram_data_o[31:0], sram_data_i[31:0], sram_we_o, sram_re_o
- DMA: dma_req_o[15:0], dma_ack_i[15:0]
- Power mgmt: susp_o (mirrors usb_suspend), resume_req_i

Priority and arbitration
- USB side (protocol/M) has preemptive priority over Wishbone (W) in usbf_mem_arb to meet link timing

- W-side ack toggles/pulses when M is idle, pacing bulk SW transfers

Build-time configuration hooks
- USBF_UFC_HADR selects RF/MEM split address bit; SSRAM_HADR sets buffer memory size/width
- Endpoint presence via USBF_HAVE_EPn; EPs must be contiguous from 0 (EP0 always present)

Concise block-to-block map (ASCII)
- UTMI PHY <-> usbf_utmi_if <-> usbf_pl (pd/pa/pe/idma)
- usbf_pl (M) <-> usbf_mem_arb <-> External SSRAM
- Wishbone host <-> usbf_wb -> (RF path) usbf_rf <-> usbf_pl; (MEM path) usbf_mem_arb (W)
- usbf_rf <-> per-EP windows, interrupts (inta_o/intb_o), DMA req/ack; vendor sideband to UTMI

-- Data Path Overview --

Data Path Overview

Receive (USB -> Memory)
- UTMI PHY -> usbf_utmi_if
- UTMI RxValid/RxActive/RxError and DataIn[7:0] are registered on phy_clk and exposed as rx_data/valid/active/err.
- usbf_utmi_if -> usbf_pl.usbf_pd
- usbf_pd decodes PID, validates token CRC5, streams DATA payload bytes (CRC16 stripped) and asserts rx_data_valid/done.
- usbf_pl.usbf_pd/usbf_pe -> usbf_pl.usbf_idma
- Policy engine (usbf_pe) decides to accept OUT, then enables RX DMA with address/size; idma packs bytes into 32-bit words, handles unaligned/partial words.
- usbf_pl.usbf_idma (M-side) -> usbf_mem_arb -> External SSRAM
- idma issues mreq/mwe; mem_arb grants immediate M-side priority; 32-bit words written to SSRAM.

Transmit (Memory -> USB)
- External SSRAM -> usbf_mem_arb -> usbf_pl.usbf_idma (M-side)
- idma prefetches/double-buffers 32-bit words and unpacks to bytes on rd_next.
- usbf_pl.usbf_idma -> usbf_pl.usbf_pa
- TX assembler emits Data PID, payload bytes, and CRC16; supports ZLP.
- usbf_pl.usbf_pa -> usbf_utmi_if -> UTMI PHY
- tx_data/tx_valid/tx_first/tx_valid_last aligned to TxReady; utmi_if holds TxValid until accepted.

Control/Status and Descriptor Path (CPU <-> Core)
- Wishbone host -> usbf_wb (RF_SEL) -> usbf_rf
- CPU reads/writes system registers and per-EP windows (CSR/INT/BUF0/BUF1). RF/system INT sources are read-to-clear.
- usbf_rf -> usbf_pl
- RF multiplexes the selected EP’s CSR/BUF0/BUF1 and flow flags (dma_in_buf_sz1, dma_out_buf_avail) into the protocol layer.
- usbf_pl -> usbf_rf
- Protocol broadcasts idin[31:0] with strobes (buf0_set, buf1_set, uc_bsel_set, uc_dpd_set, buf0_rl, int_*_set); only the matched EP latches updates (ep_match).

Software Memory Access Path (CPU <-> Buffer Memory)
- Wishbone host -> usbf_wb (MEM_SEL) -> usbf_mem_arb (W-side) -> External SSRAM
- usbf_wb issues W-side requests; mem_arb serves W when M-side idle; read data is broadcast to both masters; W-side ack pulses/toggles pace transfers.

DMA Handshake Path (optional external system DMA)
- usbf_rf (per-EP) -> dma_req_o[15:0] -> System DMA -> dma_ack_i[15:0] -> usbf_rf
- RF maintains per-EP IN/OUT counters/watermarks; acks drive counters and availability flags feeding the protocol layer.

Interrupt Path
- usbf_pl/usbf_rf (per-EP events) and RF/system events -> usbf_rf aggregator -> inta_o/intb_o
- Per-EP INTs (level) ORed with masked RF/system sources; CPU clears by reading EPx_INT and RF Word 3.

Vendor Sideband and Power Management
- UTMI vendor status: VStatus[7:0] -> usbf_utmi_if -> latched in usbf_top -> usbf_rf for CPU readback.
- Vendor control: CPU writes RF Word 5 -> usbf_rf -> usbf_utmi_if (one-shot load) -> UTMI VControl[3:0].
- Remote wake: resume_req_i (external) and rf_resume_req (from RF) -> usbf_utmi_if; suspend_clr returns to clear requests; susp_o mirrors usb_suspend.

Addressing/Data Formatting
- Buffer memory is word-addressed toward SSRAM (Wishbone drops addr[1:0]); idma packs/unpacks USB bytes into 32-bit words, handling unaligned starts and partial last words.

Arbitration and Priority
- usbf_mem_arb grants fixed, immediate priority to the USB protocol side (M). W-side (Wishbone) receives bus time only when M is idle; W ack pulses/toggles accordingly.

Clock/Reset Domains along Data Paths
- phy_clk domain: utmi_if, protocol layer (pd/pa/pe/idma), mem_arb, usbf_wb control FSM; SSRAM interface signals.
- clk_i (Wishbone) domain: usbf_rf register/interrupt bank and wb data/ack. CDC bridges handle RF vendor write, resume requests, and WB ack crossings.

-- Control and Status Path --

Control and Status Path

External control/status producers and consumers
- CPU/Wishbone master (clk_i): programs and reads the RF (register file) via usbf_wb; services interrupts on inta_o/intb_o; optionally drives remote wake via MAIN_CSR.din[5].
- UTMI PHY (phy_clk): provides LineState[1:0], VStatus[7:0]; accepts vendor VControl[3:0] and load strobe; link status exposed by utmi_if (mode_hs, usb_reset, usb_suspend, usb_attached).
- Protocol layer (phy_clk): consumes per-EP CSR/BUF descriptors and flow flags from RF; broadcasts descriptor/status updates and interrupt set strobes back to RF.

Wishbone to RF/MEM steering (usbf_wb)
- Address MSB split (HADR) selects target region: RF_SEL (!wb_addr_i[HADR]) or MEM_SEL (wb_addr_i[HADR]).
- RF accesses: one-cycle RF read/write strobes (rf_re/rf_we_d) and a one-shot WB ack pulse in clk_i.
- MEM accesses: forwarded to mem_arb W-side; ack returned and converted to a one-shot WB ack.

Register file (usbf_rf, clk_i domain, CPU-visible)

- System/global registers (word offsets):
- 0 MAIN_CSR: [4:3] LineState, [2] attached, [1] HS mode, [0] suspend; write with din[5]=1 generates rf_resume_req (remote wake one-shot).
- 1 FUNCTION_ADDRESS: [6:0] USB device address.
- 2 INTERRUPT MASKS: inta_msk[8:0], intb_msk[8:0] for RF/system events.
- 3 INTERRUPT SOURCE (read-to-clear): int_srcb[8:0] RF/system events; int_srca[15:0] per-EP any-pending mirror.
- 4 FRM_NAT: microframe index, frame number, time-since-SOF snapshot from protocol layer.
- 5 UTMI VENDOR IF: read VStatus[7:0]; write VControl[3:0] with load pulse crossing to phy_clk.
- Per-endpoint windows (EP0..EP15), 4 words each:
- CSR: direction/type/max packet/toggle/stall/DMA enable/endpoint number/OTS state.
- INT: per-EP events (read-to-clear); internal masks drive ep-level inta/intb.
- BUF0/BUF1: base address, size, and flags for buffer descriptors.
- Matched-EP mux (phy_clk side): exposes the selected EP's CSR/BUF0/BUF1 and flow flags (dma_in_buf_sz1, dma_out_buf_avail) to the protocol layer; only the matched EP captures broadcast updates.

Protocol layer <-> RF control/status exchange (phy_clk crossings handled inside RF)
- From RF to protocol: csr, buf0, buf1, dma_in_buf_sz1, dma_out_buf_avail for the matched EP.
- From protocol to RF (broadcast with per-EP capture on ep_match):
- Descriptor/status updates over idin[31:0] via buf0_set, buf1_set, buf0_rl; UC field updates via uc_bsel_set, uc_dpd_set.
- Event/interrupt set strobes: int_buf0_set, int_buf1_set, int_upid_set, int_crc16_set, int_to_set, int_seqerr_set; flow-status out_to_small.

Interrupt aggregation and software servicing
- Per-EP level interrupts (inta_ep, intb_ep) ORed with masked RF/system events (inta_rf, intb_rf) to form inta_o/intb_o.
- Software clears per-EP events by reading EPx_INT and RF/system events by reading Word 3.
- int_srca[15:0] mirrors per-EP any-pending for diagnostic polling.

DMA request/ack status path (optional system DMA)
- usbf_rf generates per-EP dma_req_o[15:0] based on IN/OUT counters and watermarks; consumes dma_ack_i[15:0].
- Acks update internal counters and influence flow flags reported to the protocol layer.

UTMI vendor sideband and link status
- VStatus_pad_i[7:0] latched each phy_clk; CPU reads via RF Word 5.
- CPU writes VControl_pad_o[3:0] via RF Word 5; write asserts VControl_Load_pad_o (one-shot into phy_clk).
- Link status from utmi_if (mode_hs, usb_reset, usb_suspend, usb_attached) reflected into RF/system view; susp_o mirrors usb_suspend on clk_i.

Suspend/resume and remote wake control path
- External resume_req_i is latched in usbf_top (resume_req_r) and presented to utmi_if until suspend_clr pulses.
- Software-triggered rf_resume_req generated by writing MAIN_CSR.din[5]=1; crossed into phy_clk; cleared on suspend_clr.

Top-level latches and visibility
- LineState_r (UTMI LineState) and VStatus_r latched on phy_clk for CPU visibility via RF; phy_rst_pad_o tied to rst_i.

Clock-domain crossings and coherency
- WB control FSM (phy_clk) and WB ack generator (clk_i) form the WB CDC; address/data must be stable until ack.
- RF handles wclk->clk crossings for vendor write and resume request; event edges captured into wclk for interrupt latching.

Software programming model (typical)
- Initialize FUNCTION_ADDRESS; configure global masks.
- For each EP: program CSR (direction, type, max packet, DMA enable), BUF0/BUF1 descriptors.
- Monitor inta_o/intb_o, read EPx_INT and RF Word 3 to identify and clear events.
- Update descriptors and UC fields via protocol-driven writes (observed in EP windows) or CPU rewrites as needed.
- Use FRM_NAT for timing diagnostics; use vendor registers for PHY-specific control; request remote wake via MAIN_CSR.din[5] when allowed.

-- Clock and Reset Domains --

Clock and Reset Domains

Clock domains (top-level)
- phy_clk_pad_i ("phy_clk"): UTMI/link/protocol core domain
- Hosts: usbf_utmi_if (incl. usbf_utmi_ls), usbf_pl (pd/pa/pe/idma), usbf_mem_arb, usbf_wb control FSM, external SSRAM interface
- clk_i ("wb_clk"): Wishbone/CPU-visible domain
- Hosts: usbf_rf CPU register bank and interrupt logic, usbf_wb ACK generator and WB data mux

Intra-module dual-clock usage
- usbf_rf spans both domains
- wclk=clk_i: CPU-visible regs (system bank and EP windows), interrupt aggregation, vendor-write initiation
- clk=phy_clk: core-facing matched-EP mux outputs (csr, buf0, buf1, dma flags) to protocol layer; per-EP capture of broadcast updates
- usbf_wb spans both domains
- phy_clk: request capture and target selection FSM (RF vs MEM)
- clk_i: one-shot ACK generation and read-data registration

Key CDC paths and mechanisms
- Wishbone request/ack (clk_i ↔ phy_clk)
- wb_req_s1 samples (stb&cyc;) into phy_clk; usbf_wb FSM sequences the access
- wb_ack_d crosses back and generates a one-shot wb_ack_o in clk_i
- Assumption: WB master holds address/data stable until ack
- RF control/status (clk_i ↔ phy_clk)
- Vendor control write: CPU write in clk_i produces a one-shot load crossed into phy_clk for UTMI VControl
- Vendor status read: VStatus[7:0] latched each phy_clk; read by CPU via RF Word 5
- Resume request: MAIN_CSR.din[5] in clk_i generates rf_resume_req one-shot crossed to phy_clk; cleared by suspend_clr from utmi_if
- RF/system event edges: attach/detach, suspend start/end, errors are captured into clk_i for interrupt latching (read-to-clear)
- Protocol↔RF (within phy_clk with per-EP gating)
- RF exposes matched EP csr/buf0/buf1 and flow flags to the protocol engine in phy_clk

- Protocol broadcasts idin/strobes; only the matched EP captures, avoiding wide CDC
- DMA handshakes (system DMA optional)
- dma_req_o[15:0] generated in clk_i (RF), dma_ack_i[15:0] received in clk_i
- Internal path reflects acks back into phy_clk via a small synchronizer pipeline (clk_i→phy_clk) to update counters/flags used by the protocol layer
- Suspend indication (phy_clk → clk_i)
- usb_suspend (phy_clk) mirrored as susp_o in clk_i; cleared on resume via suspend_clr pulse from utmi_if

Module-to-domain mapping (summary)
- phy_clk: utmi_if/utmi_ls, pl (pd/pa/pe/idma), mem_arb, wb control FSM, SSRAM bus
- clk_i: rf CPU registers and interrupts, wb ACK and WB-side data mux

Resets and POR behavior
- Global reset input rst_i
- phy_rst_pad_o is tied directly to rst_i (asserts PHY reset pin)
- Default reset style is synchronous to the local clock; optional asynchronous assertion via compile-time macro USBF_ASYNC_RESET in several blocks (utmi_if, pl, mem_arb, rf, wb)
- utmi_ls adds a PHY-side POR source: usb_vbus acts as an additional reset for its state machine
- Reset effects (illustrative)
- utmi_if: clears TxValid, RX indicators; link-state machine to POR
- wb: resets control FSM (phy_clk) and ACK pipeline (clk_i)
- rf: clears masks, event latches, rf_resume_req, vendor-write pulse; initializes EP buffers to known values

Domain-specific interfaces
- External SSRAM interface (sram_*) is driven in phy_clk; mem_arb's wclk input is unused (W-side requests are already synced via usbf_wb)
- UTMI interface (DataIn/Out, TxValid/Ready, RxValid/Active/Error, LineState, control pins) is in phy_clk
- Wishbone interface (wb_*) is in clk_i; address/data must remain valid until wb_ack_o pulses due to CDC

Design assumptions/cautions
- The WB request/ack CDC uses level sampling and one-shot pulses rather than a full handshake synchronizer; fully asynchronous, unrelated clocks are tolerated only if the WB master adheres to the "hold-until-ack" rule and producers keep data stable through ack
- mem_arb gives immediate, preemptive priority to the phy_clk master (protocol), so W-side accesses (originating from clk_i) are served only when M-side is idle

-- Clock-Domain Crossing Strategy --

Clock-Domain Crossing Strategy

Domains
- phy_clk_pad_i (phy_clk): UTMI/link, protocol layer (pd/pa/pe/idma), mem_arb, wb control FSM, SSRAM I/F
- clk_i (wb_clk): CPU/Wishbone-visible: usbf_rf registers/interrupts, usbf_wb ACK generator and WB data mux

Principles and primitives used
- Level sampling + one-shot pulses across domains (no async FIFOs, no Gray pointers)
- Single-/double-flop synchronizers for pulses and status; edge-detect in receiver domain for

read-to-clear latches
- Keep bulk data paths within a single domain; only control/status and request/ack handshakes cross domains

Defined CDC paths
1) Wishbone request/ack (clk_i ↔ phy_clk) via usbf_wb
- Request: wb_stb_i & wb_cyc_i level-sampled into phy_clk (wb_req_s1) to drive the phy_clk FSM (RF vs MEM)
- Completion: wb_ack_d (phy_clk) crosses back to clk_i and is converted to a one-shot wb_ack_o
- Constraint: WB master must hold address/data until ack; producers must keep rf_din/ma_din stable through ack

2) RF control/status crossings (clk_i ↔ phy_clk) via usbf_rf
- Vendor control write: CPU write in clk_i emits a one-shot load pulse into phy_clk (drives UTMI VControl + load)
- Vendor status read: VStatus[7:0] latched in phy_clk each cycle; CPU reads via RF Word 5 in clk_i
- Remote wake: MAIN_CSR.din[5] in clk_i generates rf_resume_req one-shot into phy_clk; utmi_if returns suspend_clr to clear
- RF/system events: attach/detach, suspend start/end, errors sampled/edged into clk_i for read-to-clear interrupt latches

3) Protocol↔RF interface (within phy_clk)
- RF exposes matched EP csr/buf0/buf1 and flow flags to the protocol layer in phy_clk
- Protocol broadcasts idin/strobes; RF gates updates by ep_match (avoids cross-domain buses by keeping this path in phy_clk)

4) System-DMA handshake reflection (optional)
- dma_req_o[15:0] and dma_ack_i[15:0] are in clk_i (RF). A small synchronizer pipeline reflects dma_ack into phy_clk to update counters/flow flags used by protocol

5) Status/PM crossings at top level
- Suspend indication: usb_suspend (phy_clk) mirrored to susp_o in clk_i; cleared by suspend_clr pulse from utmi_if (phy_clk)
- External resume_req_i is latched into phy_clk (resume_req_r) and held until suspend_clr

CDC avoidance by design
- Data plane (USB RX/TX, DMA to SSRAM) is entirely in phy_clk; mem_arb’s W-side is serviced only when M idle, with W requests already synchronized by usbf_wb

Reset crossing strategy
- Default synchronous resets in each domain; optional async assert via USBF_ASYNC_RESET per block (utmi_if, pl, mem_arb, rf, wb)
- UTMI link state adds a PHY-side POR using usb_vbus

Assumptions/limitations
- The WB CDC uses single-flop level sampling for requests and one-shot for acks; with fully asynchronous clocks, correct operation relies on the "hold-until-ack" rule and stable data producers
- No multi-beat CDC buffering; no metastability-hardened FIFOs; timing closure may require system-level constraints for unrelated clocks

-- Memory/Buffer Architecture and Arbitration --

Memory/Buffer Architecture and Arbitration

Physical buffer memory
- External SSRAM: single-port, 32-bit synchronous memory connected via sram_adr_o[SSRAM_HADR:0], sram_data_i/o[31:0], sram_we_o, sram_re_o.
- Size/width: parameter SSRAM_HADR defines address width; total bytes = 2^(SSRAM_HADR+3).
- Access granularity: 32-bit words; read port effectively always enabled (sram_re_o=1'b1).

Address map and host access
- Address regions selected by a high address bit (HADR = USBF_UFC_HADR):
- RF region: !wb_addr_i[HADR] → register file (usbf_rf).
- MEM region: wb_addr_i[HADR] → external SSRAM via mem arbiter W-side.
- Wishbone MEM addressing is word-based inside the arbiter: ma_adr = wb_addr_i[SSRAM_HADR+2:2].
- Host transfers complete when W-side ack pulses/toggles; software must hold address/data until ack.

Masters of the buffer memory
- M-side (USB core, highest priority): usbf_pl/usbf_idma issues madr/mdout/mwe/mreq (mack=mreq) in phy_clk.
- W-side (Wishbone/CPU, lower priority): usbf_wb issues wadr/wdin/wwe/wreq in phy_clk on MEM accesses from clk_i; receives wack pulses/toggles.

Arbitration (usbf_mem_arb)
- Fixed, immediate priority to M-side to meet USB timing.
- Selection: wsel = (wreq | wack) & !mreq; address/data/we muxed between M and W.
- Acknowledges:
- M-side: mack = mreq (zero-wait arbiter path).
- W-side: wack is a one-clock pulse; if wreq held high and M idle, wack toggles every other cycle (half-rate pacing).
- Read data fanout: sram_data_i broadcast to both masters; each samples only on its ack.
- No internal pipelining; upstream must align address/data with ack protocol.

Endpoint buffer model (per EP via usbf_rf/usbf_pe)
- Two descriptors per EP: BUF0 and BUF1 (32-bit each) hold:
- Base address: bufX[SSRAM_HADR+2:0] (start byte address; low nibble may carry UC/DPD bits when written by core).
- Size: bufX[30:17] (bytes). Additional control/status flags encoded per integration.
- Selection/policy (high level):
- DMA enabled endpoints typically use BUF0; control EPs select by direction; otherwise rotation/availability rules choose BUF1 when applicable.
- Buffer updates performed by protocol via a broadcast write bus (idin[31:0]) with strobes: buf0_set, buf1_set, buf0_rl (reload from last CPU write), uc_bsel_set, uc_dpd_set. Only the matched EP latches updates.
- Completion rules (examples):
- IN: buffer_done when buffer becomes empty after transfer.
- OUT: buffer_done when received data is "full enough" versus max packet size (or exact size if policy demands).

Internal DMA engine (usbf_idma) on M-side
- RX (USB→memory):
- Packs incoming bytes into 32-bit words; pre-reads destination word for unaligned starts (read-modify-write); writes full words and a partial last word if needed.
- Start byte address adr split into madr=adr[SSRAM_HADR+2:2] and adr_cb=adr[2:0]; word address

increments on mack; byte index tracks position.
- Optional ring wrap: when adrw_next reaches last_buf_adr=(adr+buf_size), wrap to 0 if dma_en is set.
- TX (memory→USB):
- Double-buffers 32-bit words; unpacks to bytes on rd_next; decrements size counter; supports zero-length packet without memory access.
- Handshake: mreq level for reads; pulsed on word-ready for writes; mack sampled (registered) to advance.

Flow/availability accounting (inside usbf_rf)
- Shadows original CPU-programmed BUF0 (buf0_orig) to define total word budget.
- Counters (clk_i domain, reflected to core):
- OUT: dma_out_cnt tracks words remaining to pull from host memory; dma_out_buf_avail flags room for another OUT packet.
- IN: dma_in_cnt tracks words produced for IN; dma_in_buf_sz1 flags at least one packet-sized chunk present.
- Optional external DMA: per-EP dma_req_o/dma_ack_i handshakes are generated/consumed in usbf_rf, with ack crossings reflected back to phy_clk for protocol flow decisions.

Throughput and timing behavior
- USB-side (M) preempts W-side instantly to honor inter-packet timing; M-side writes/reads see arbiter zero-wait ack (mack=mreq) from the arbiter perspective.
- W-side host memory transfers proceed only when M is idle and are paced by pulsed/toggling wack; continuous wreq achieves at most every-other-cycle acceptance while M is idle.

Software usage summary
- Program per-EP CSR, then BUF0/BUF1 base and size pointing into MEM region; fill/read payloads in MEM region.
- Protocol engine moves data via idma and updates descriptors/status through RF strobes; software observes per-EP INTs and RF/system events to service buffers.
- Host direct MEM accesses go through usbf_wb→usbf_mem_arb (W-side); RF vs MEM chosen by wb_addr_i[HADR].

-- Address Map Partitioning --

Address Map Partitioning

Top-level partition (Wishbone aperture)
- Address-space split by a single high address bit HADR = `USBF_UFC_HADR`:
- Test build: HADR=17
- Non-test build: HADR=12
- Region select:
- RF region (register file): RF_SEL = !wb_addr_i[HADR]
- MEM region (buffer memory/SSRAM via arbiter): MEM_SEL = wb_addr_i[HADR]
- All software-visible addresses are 32-bit word addresses; increment by 1 per 4 bytes.

Register-file (RF) region layout (word addressing)
- usbf_wb passes wb_addr_i to usbf_rf as adr[8:2] (7-bit word index). Reads/writes are 32-bit words.
- System/global bank (aliased reads): adr[6:2] == 0 or 1 return the same contents; writes must target adr[6:2] == 0.
- Word 0: MAIN_CSR (status; write din[5]=1 issues remote-wakeup one-shot)
- Word 1: FUNCTION_ADDRESS [6:0]
- Word 2: INTERRUPT MASKS {intb_msk[8:0], inta_msk[8:0]}
- Word 3: INTERRUPT SOURCE (read-to-clear) {int_srcb[8:0], int_srca[15:0]}

- Word 4: FRM_NAT [31:0] (frame/microframe/time-since-SOF snapshot)
- Word 5: UTMI VENDOR IF (read VStatus[7:0]; write VControl[3:0] + one-shot load)
- Per-endpoint windows (EP0..EP15): adr[6:2] ranges 0x04..0x13 (word-aligned). For EPn:
- Base word address = 4 + 4*n
- Offsets within EP window:
- +0: EPx_CSR (control/status)
- +1: EPx_INT (latched, read-to-clear events)
- +2: EPx_BUF0 (descriptor)
- +3: EPx_BUF1 (descriptor)
- Disabled endpoints still decode their window; a dummy block drives constant outputs (reads zero for CSR/INT, 0xFFFF_FFFF for BUF0/BUF1) and never interrupts/DMAs.

Buffer-memory (MEM) region layout
- Selected by wb_addr_i[HADR]=1; Wishbone accesses are forwarded to usbf_mem_arb W-side.
- Word addressing into SSRAM: ma_adr = wb_addr_i[SSRAM_HADR+2:2]; low byte bits [1:0] are dropped.
- External SSRAM size: bytes = 2^(SSRAM_HADR+3); 32-bit word granularity; sram_re_o held asserted, data returned on ack.

Descriptor/address linkage
- EPx_BUF0/BUF1 hold base byte addresses into the MEM region: bufX[SSRAM_HADR+2:0]. The protocol/DMA engine splits these into word address (madr=adr[SSRAM_HADR+2:2]) and byte-in-word index (adr[2:0]).
- Descriptor size fields: bufX[30:17] carry sizes in bytes; additional control/status bits are encoded per integration.

Access semantics and notes
- RF/system event registers (Word 3) and EPx_INT are read-to-clear.
- All RF addresses are word-based; software increments addresses by 1 for 4-byte steps.
- MEM region is accessible to the host only when the USB M-side is idle; arbiter paces W-side with pulsed/toggling acks.
- No byte enables on Wishbone: software should use word-aligned accesses and manage sub-word updates in software if needed.

-- DMA Architecture --

DMA Architecture

Overview
- Two complementary DMA mechanisms:
1) Internal IDMA (in protocol core, phy_clk): byte↔word bridge moving payloads between the USB byte stream and 32-bit external SSRAM.
2) Optional system DMA handshake (in usbf_rf, clk_i): per-endpoint dma_req_o/dma_ack_i interface for a platform DMA to assist buffer movement/pipelining at the system level.

Internal IDMA (usbf_idma, phy_clk)
- Purpose: Efficiently move endpoint data between the UTMI-side byte stream and 32-bit word SSRAM with alignment, partial-word, and ring-buffer handling.
- Interfaces:
- USB RX in: rx_data_st[7:0], rx_data_valid, rx_data_done.
- USB TX out: tx_data_st[7:0], rd_next, send_data.
- Memory: madr[SSRAM_HADR:0], mdout[31:0], mdin[31:0], mwe, mreq, mack.

- Control: rx_dma_en/tx_dma_en/abort, adr (start byte addr), buf_size, size (TX bytes), dma_en (wrap enable).
- RX (USB→memory):
- Packs bytes into 32-bit words; performs read-modify-write pre-read for unaligned start; writes full words and a partial last word when required.
- Addressing split: madr=adr[SSRAM_HADR+2:2], adr_cb=adr[2:0]; madr increments on mack; adr_cb tracks byte position.
- Ring wrap to address 0 when adrw_next reaches last_buf_adr=(adr+buf_size) and dma_en=1.
- TX (memory→USB):
- Double-buffers two words; unpacks to bytes on rd_next; decrements size counter; supports zero-length packet (ZLP) without memory access.
- Handshake: mreq held for reads until mack; pulsed per completed word for writes; mack registered; idma_done on rx_data_done (RX) or size==0 (TX).
- Sizes/params: TX size counter 14 bits (≤16 KB); RX byte counter 11 bits (≤2 KB); clock/reset synchronous (optional USBF_ASYNC_RESET).

Per-Endpoint DMA control (usbf_rf, clk_i)
- Role: Endpoint-level accounting, flow signaling, and optional external DMA handshake generation.
- Descriptors: EPx_BUF0/BUF1 hold base byte address (bufX[SSRAM_HADR+2:0]) and size (bufX[30:17]); protocol updates via broadcast idin bus (buf0_set/buf1_set/buf0_rl, uc_bsel_set/uc_dpd_set).
- Counters/flow flags:
- OUT path: dma_out_cnt tracks words remaining; dma_out_buf_avail asserts when another OUT packet fits; updates on dma_ack_i and buffer (re)arm.
- IN path: dma_in_cnt tracks words filled; dma_in_buf_sz1 asserts when at least one packet-sized word count is available; updates on dma_ack_i and (re)arm.
- External DMA handshake (optional system DMA):
- Generates per-EP dma_req_o[15:0] based on counters and DMA enable; deassert policy uses hold thresholds to avoid chatter.
- Consumes dma_ack_i[15:0]; acks cross back into phy_clk via a small synchronizer pipeline to update protocol-visible flow flags/counters.

Protocol-layer DMA orchestration (usbf_pl/usbf_pe, phy_clk)
- Start conditions:
- usbf_pe asserts rx_dma_en for accepted OUT/SETUP (policy-checked) or tx_dma_en for IN; determines transfer geometry (size_next=min(buf_size,max_pl_sz)) from CSR/BUF and mode/speed.
- Completion and status:
- On idma_done (and policy outcomes like ACKed IN, valid OUT), protocol updates descriptors (buf*_set or buf0_rl), UC fields (uc_*_set), raises interrupts (int_*_set), and may release DMA buffer.
- Timeouts and error conditions (ACK wait, data wait, CRC/SEQ) set corresponding INT bits and influence handshakes (ACK/NACK/STALL/NYET).

Memory/arbiter interaction (usbf_mem_arb, phy_clk)
- M-side (IDMA/protocol) has fixed, immediate priority; mack=mreq gives zero-wait arbitration from the arbiter's perspective.
- W-side (Wishbone host) granted only when M is idle; wack pulses/toggles pace host bulk transfers; word addressing ma_adr=wb_addr_i[SSRAM_HADR+2:2].

Clock-domain crossings
- IDMA and protocol run in phy_clk; RF registers and external DMA handshake live in clk_i.
- dma_ack_i crosses from clk_i→phy_clk (two-step synchronizer inside RF) to keep core counters coherent; vendor/resume pulses cross clk_i→phy_clk; WB req/ack uses level+one-shot scheme.

Throughput/latency characteristics
- USB path favored: M-side preempts W-side instantly to meet inter-packet timing.
- TX prefetch and double-buffering reduce stall risk; RX pre-read preserves unaligned bytes; partial-word finalization ensures no data loss.
- External DMA acks can be held-off by policy (hold thresholds) to reduce req/ack chatter while keeping pipes full.

Programming model tie-in
- Software programs EPx_CSR and BUF0/BUF1 (base/size/flags), manages payloads in MEM region, and services per-EP/RF interrupts.
- Optional system DMA may honor dma_req_o to accelerate buffer movement; otherwise, the internal IDMA alone sustains USB transfers backed by SSRAM.

-- Interrupt Architecture --

Interrupt Architecture

Overview and domains
- Aggregation and software-visible registers reside in the Wishbone/CPU clock domain (clk_i) inside usbf_rf.
- Interrupt sources originate from two places:
1) Per-endpoint events from the protocol layer (phy_clk), broadcast as set strobes and captured per-EP in usbf_ep_rf.
2) RF/system events (link/power/errors) generated in the UTMI/link/protocol side (phy_clk) and edge/level captured into clk_i in usbf_rf.
- Two top-level interrupt outputs: inta_o and intb_o (clk_i domain).

Top-level interrupt outputs
- inta_o = OR(per-EP inta) | OR(int_srcb & inta_msk)
- intb_o = OR(per-EP intb) | OR(int_srcb & intb_msk)
- RF/system mask registers (Word 2 in RF region):
- inta_msk[8:0] maps RF/system events into inta_o
- intb_msk[8:0] maps RF/system events into intb_o

RF/system event latch (global)
- Word 3 (read-to-clear): int_srcb[8:0] (RF/system events) and int_srca[15:0] (mirror of per-EP pending)
- int_srcb bits:
[8] usb_reset
[7] rx_err
[6] detach (falling edge of usb_attached)
[5] attach (rising edge of usb_attached)
[4] suspend_end
[3] suspend_start
[2] nse_err
[1] pid_cs_err
[0] crc5_err
- Behavior:
- int_srcb is set on detected edges/levels synchronized into clk_i; reading Word 3 clears all int_srcb bits.
- int_srca[15:0] reflects per-EP any-pending (ep_inta | ep_intb) and is software-visible only (not read-to-clear).

Per-endpoint interrupt architecture (EPx)
- Register: EPx_INT (offset +1 within each EP window) is read-to-clear.
- Event sources (latched from protocol strobes when EP matches): examples include
- BUF0_DONE, BUF1_DONE (buf*_set events)
- Unexpected PID (int_upid_set)
- CRC16 error (int_crc16_set)
- Timeouts (int_to_set: IN ACK wait / OUT data wait)
- Sequence error (int_seqerr_set)
- OUT too small (out_to_small)
- Per-EP mask control:
- Two 6-bit mask sets (iena/ienb) inside EPx gate which latched bits contribute to per-EP inta/intb level outputs.
- Output levels:
- epX_inta and epX_intb (wclk/clk_i domain) are level ORs of (latched events & masks); aggregated across all EPs into inta_ep/intb_ep.
- Read-to-clear semantics:
- Reading EPx_INT clears the EP's latched event bits (does not affect RF/system latch).

Masking and aggregation summary
- Per-EP: EPx_INT holds events; iena/ienb determine contribution to epX_inta/intb.
- Global: RF Word 2 masks (inta_msk/intb_msk) control which RF/system events contribute to inta_o/intb_o.
- Final outputs: OR reduction of all enabled per-EP levels plus masked RF/system events per line.

Clock-domain crossing considerations
- Protocol event strobes (phy_clk) are broadcast and captured only by the matched EP in the core (clk domain), then reflected as level signals in clk_i.
- RF/system status (attach/detach, suspend start/end, usb_reset, error flags) are synchronized into clk_i and edge-detected for int_srcb.
- Interrupt outputs and all software-visible latches/masks are in clk_i.

Software programming model
- Configure RF/system masks at RF Word 2.
- Service RF/system events by reading RF Word 3 (clears int_srcb).
- Service per-EP events by reading EPx_INT (offset +1; clears EP latched bits) and by updating per-EP masks via writes to EPx_INT.
- int_srca provides a quick per-EP pending summary (software-visible mirror only).

Related status registers
- MAIN_CSR (Word 0) exposes live link status (line state, attached, HS mode, suspend); write din[5]=1 issues a remote-wakeup pulse (not an interrupt by itself).
- UTMI VENDOR IF (Word 5) readback provides vendor status (not an interrupt source).

-- Power Management and Link State Control --

Power Management and Link State Control

Domains and status visibility
- Link/power FSM runs in phy_clk inside usbf_utmi_ls (instantiated by usbf_utmi_if).
- Software-visible mirrors are provided in clk_i via usbf_rf:
- MAIN_CSR[4:0] = {LineState[1:0], attached, mode_hs, suspend} (read-only snapshot)

- susp_o mirrors usb_suspend (phy→clk_i crossing).

Attach/detach and POR
- usb_vbus is used as an additional POR source for the link FSM (forces state=POR).
- ATTACH sequence:
- Debounce for ~100 ms; on success set usb_attached=1 and enter NORMAL.
- Detach event detected on falling edge of attached; both edges latched into RF int_srcb (Word 3).

Suspend entry/hold/exit
- Idle qualification depends on speed:
- FS idle = J; HS idle = SE0; filtered to generate idle_long.
- Enter suspend when idle > 3 ms:
- FS: NORMAL → SUSPEND.
- HS: NORMAL → RES_SUSP (intermediate) then SUSPEND (if J) or RESET (if SE0).
- Suspend outputs/controls:
- usb_suspend asserted in SUSPEND.
- SuspendM = (usb_suspend & !resume_req_s) | (LineState==K) (abstract control; polarity PHY-dependent).
- OpMode typically non-driving; termination per state/speed.
- Exit suspend:
- Host resume: detect K (k_long) → RESUME.
- Device-initiated resume (remote wake): allowed only if idle > 5 ms; see below.
- RF/system events:
- suspend_start and suspend_end edges latched in int_srcb (read-to-clear).

Device-initiated remote wakeup (resume)
- Sources:
- External pin: resume_req_i (top) latched to resume_req_r (phy_clk) until suspend_clr pulse.
- Software: write MAIN_CSR with din[5]=1 to generate rf_resume_req (clk_i→phy_clk one-shot) for integration.
- FSM behavior:
- After T2_wakeup (~>5 ms in suspend), enter RESUME_REQUEST: prepare PHY (FS term on, OpMode=non-driving).
- RESUME_SIG: drive K (drive_k=1) for ≥1 ms.
- RESUME: clear suspend; when LineState=SE0, restore proper transceiver selection/termination, OpMode normal; then RESUME_WAIT (>100 µs) → NORMAL.
- suspend_clr: one-shot pulse from utmi_if clears pending resume requests and software/top latches.

Reset detection and speed negotiation
- Reset detection: SE0 > 2.5 µs triggers RESET state; usb_reset asserted during RESET.
- During RESET: force FS interface (XcvSelect=FS, TermSel on, OpMode non-driving).
- HS negotiation after RESET:
- SPEED_NEG: device chirp (drive_k=1) for >1.2 ms; then detect alternating host K/J (SPEED_NEG_K/J).
- Commit HS (mode_hs=1) after 6 edges (3 K/J pairs) → SPEED_NEG_HS; otherwise remain FS (SPEED_NEG_FS).
- On post-negotiation idle (SE0), return to NORMAL.
- mode_hs is sticky and only updated in SPEED_NEG_{HS,FS} or forced to FS in RESET.

UTMI PHY control mapping (driven by link FSM)
- XcvSelect: 1=FS/LS transceiver, 0=HS.
- TermSel: 1 enables FS termination, 0 disables.

- OpMode[1:0]: 00 normal drive, 10 non-driving (used in suspend/reset/resume prep).
- SuspendM: asserted per expression above.
- drive_k: asserted during resume signaling and HS chirp.

Software interface tie-in
- MAIN_CSR (Word 0) exposes: suspend, mode_hs, attached, LineState; write din[5]=1 requests remote wake (rf_resume_req one-shot).
- INT SOURCE (Word 3) latches RF/system link events: usb_reset, suspend start/end, attach/detach, and error flags (read-to-clear).
- susp_o (top output) mirrors usb_suspend for system-level PM.

Clock-domain crossings and latching
- Link FSM and UTMI signals in phy_clk; status/edges are synchronized into clk_i within usbf_rf.
- resume_req paths (external and software) are synchronized into phy_clk; suspend_clr returns to clear sources.

Notes and constraints
- Idle and timing thresholds implemented with internal counters (e.g., 3 ms suspend, >5 ms wake eligibility, ≥1 ms resume K, >100 µs resume settle, HS chirp timing and edge count).
- SuspendM polarity vs. UTMI spec may be PHY-dependent; this block provides the abstract control as defined above.

OPERATION

-- Reset and Initialization Sequence --

Reset and Initialization Sequence

Reset sources and clock domains
- Two independent clock domains:
- phy_clk: UTMI/link/protocol/memory arbiter
- clk_i: CPU/Wishbone/RF
- Global system reset rst_i:
- Drives phy_rst_pad_o (resets external UTMI PHY as integrated).
- Resets Wishbone-side logic (usbf_wb in phy_clk; usbf_rf in clk_i) and all core FSMs/registers.
- Optional async reset style:
- Default: synchronous resets per domain. With USBF_ASYNC_RESET defined: async assert, sync deassert in modules that support it (e.g., utmi_if, mem_arb, idma, wb).
- Additional POR source:
- usb_vbus (from PHY) forces state=POR in the link FSM (usbf_utmi_ls).

Power-on/system reset deassertion sequence (rst_i→1)
1) UTMI/link domain (phy_clk):
- usbf_utmi_ls enters POR; initializes to FS: XcvSelect=FS, TermSel=1, mode_hs=0, usb_suspend=0, usb_attached=0, OpMode non-driving.
- Proceeds to ATTACH; debounces ~100 ms, then sets usb_attached=1 and enters NORMAL. drive_k=0; SuspendM follows its expression.

2) RF/CPU domain (clk_i):
- usbf_rf defaults:
- FUNCTION_ADDRESS=0; inta_msk/intb_msk=0; int_srcb=0; rf_resume_req=0; vendor write pulse deasserted.
- Per-EP: masks/counters cleared; BUF0/BUF1=0xFFFF_FFFF (invalid); CSR defaults cleared.
3) Wishbone bridge (usbf_wb, phy_clk): FSM→IDLE; one-shot ack generator cleared.
4) Memory arbiter (usbf_mem_arb, phy_clk): wack_r cleared; M-side priority idle; sram_re_o asserted.
5) Protocol/IDMA (phy_clk): usbf_pl/usbf_idma FSMs→IDLE; size/byte counters cleared.

Software-visible status after POR
- MAIN_CSR: suspend=0, mode_hs=0, attached=0 until ATTACH completes; LineState snapshot valid.
- After ~100 ms debounce: attached=1; RF int_srcb[5] (attach) set (read-to-clear).

Recommended initialization (host/CPU)
1) Wait for attach: poll MAIN_CSR[2] or service attach interrupt (RF Word 3).
2) Program FUNCTION_ADDRESS after SET_ADDRESS.
3) Configure endpoints: write EPx_CSR (dir/type/maxpkt/DMA), EPx_BUFx (base/size/flags); stage payloads in MEM region as needed.
4) Enable desired RF/system and per-EP interrupt masks.

Bus reset (runtime)
- Detection and link re-init:
- SE0 > 2.5 µs → RESET: usb_reset=1; force FS interface (XcvSelect=FS, TermSel=1, OpMode non-driving).
- After >1.0 ms → speed negotiation (device chirp); return to NORMAL in HS (mode_hs=1 after 6 K/J edges) or FS (mode_hs=0).
- Software:
- usb_reset latched in RF int_srcb[8] (read Word 3 to clear).
- Function address and EP state are not auto-restored; firmware must reinitialize per USB spec and system policy.

Suspend/resume interaction
- Enter suspend after idle >3 ms (speed-qualified); MAIN_CSR[0] reflects suspend.
- Device-initiated remote wake:
- External: resume_req_i latched (phy_clk) until suspend_clr.
- Software: write MAIN_CSR[5]=1 to generate rf_resume_req (clk_i→phy_clk one-shot).
- FSM enforces >5 ms in suspend; drives K ≥1 ms; restores normal; suspend_clr pulse clears pending requests.

Clock-domain/CDC notes
- Link FSM status (usb_reset/suspend/attached/mode_hs, LineState) synchronized into clk_i for RF latching.
- Resume request pulses cross clk_i→phy_clk; suspend_clr crosses back to clear sources.
- WB ack: phy_clk strobe → clk_i one-shot; WB master must hold address/data until ack.

Simulation-only initialization
- Initial block prints enabled endpoints, bus address sizes, computed SSRAM size; checks EP numbering continuity (no hardware effect).

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

Attach/Detach, USB Reset, and Speed Negotiation

Attach/detach detection
- The link controller (usbf_utmi_ls, phy_clk) monitors UTMI LineState and usb_vbus.
- After POR/reset, the FSM enters ATTACH and debounces presence for ~100 ms.
- On successful debounce, usb_attached=1 and the FSM transitions to NORMAL (device attached).
- Detach is detected on loss of attach; attach (rise) and detach (fall) are latched in RF Word 3 int_srcb: [5]=attach, [6]=detach (read-to-clear). MAIN_CSR[2] mirrors the current attached state.

USB reset detection and handling
- Reset detection: SE0 asserted for >2.5 µs drives the FSM to RESET and asserts usb_reset.
- During RESET the interface is forced to FS: XcvSelect=FS, TermSel=1 (FS termination on), OpMode=non-driving.
- After >1.0 ms in RESET, the FSM proceeds to speed negotiation (device chirp) and then returns to NORMAL.
- Software visibility: usb_reset is latched in RF Word 3 int_srcb[8] (read-to-clear). Hardware does not force FUNCTION_ADDRESS to 0; firmware must restore address/EP state per policy.

High-speed chirp negotiation and mode selection
- SPEED_NEG (device chirp): drive_k asserted for >1.2 ms.
- Host response detection: alternating long K/J transitions counted in states SPEED_NEG_K/J.
- Commit to HS (SPEED_NEG_HS) on 6 edges (3 K/J pairs): mode_hs<=1; select HS transceiver (XcvSelect=HS), TermSel=0, OpMode normal.
- If chirp not qualified (e.g., SE0), remain FS (SPEED_NEG_FS): mode_hs<=0; XcvSelect=FS, TermSel=1, OpMode normal.
- After post-negotiation idle (SE0), return to NORMAL. mode_hs is sticky and only updates in SPEED_NEG_{HS,FS} or is forced to FS in RESET. MAIN_CSR[1] mirrors mode_hs.

Software status and event latching
- MAIN_CSR (Word 0): [4:3] LineState snapshot, [2] attached, [1] mode_hs, [0] suspend.
- RF/system event latch (Word 3, read-to-clear): [8] usb_reset, [6] detach, [5] attach (plus other events).
- All link events originate in phy_clk and are synchronized into clk_i for RF visibility/interrupt aggregation.

UTMI control mapping through phases
- XcvSelect: 1=FS/LS, 0=HS; TermSel: 1=FS termination on, 0=off; OpMode: 00=normal, 10=non-driving.
- drive_k asserted during device chirp and resume signaling (not part of attach detection itself).

Key timing thresholds (internal)
- Attach debounce ~100 ms; Reset qualification SE0 >2.5 µs; Reset hold >1.0 ms before negotiation; Device chirp >1.2 ms; HS qualification requires 6 K/J edges (3 pairs).

Summary
- The core debounces attach, exposes attached/LineState to software, detects USB reset via SE0 timing and asserts a reset indication, then executes USB 2.0 HS chirp negotiation to select HS or FS and reflects the result via mode_hs. All key events are latched in RF for software handling and interrupt aggregation.

-- Suspend/Resume and Remote Wakeup --

Suspend/Resume and Remote Wakeup

Suspend entry and state
- Entry condition: The link FSM (usbf_utmi_ls, phy_clk) declares suspend after bus idle >3.0 ms (speed-qualified):
- FS: NORMAL → SUSPEND directly.
- HS: NORMAL → RES_SUSP (>100 µs check) → SUSPEND if J; else RESET if SE0.
- Indicators/outputs:
- usb_suspend asserted while in SUSPEND; top-level susp_o mirrors usb_suspend (crossed to clk_i).
- UTMI SuspendM driven as: SuspendM = (usb_suspend & !resume_req_s) | (LineState == K).
- RF/system events: suspend_start (enter) and suspend_end (exit) latched in RF Word 3 int_srcb[3] and [4] (read-to-clear).

Host-initiated resume
- Detection: K state (k_long) during SUSPEND.
- Sequence:
- RESUME: clear usb_suspend; when LineState becomes SE0, restore transceiver/mode; then RESUME_WAIT (>100 µs) → NORMAL.
- Signals: suspend_clr pulse generated on resume to clear pending remote-wakeup requests in the core and top.

Device-initiated remote wakeup (resume)
- Request sources:
- External: resume_req_i (top input) latched into phy_clk as resume_req_r and held until suspend_clr.
- Software: write MAIN_CSR with din[5]=1 in RF (clk_i) to assert rf_resume_req one-shot (safely crossed to phy_clk).
- Eligibility and sequence (in SUSPEND):
- Must satisfy wake eligibility: idle >5 ms (T2_wakeup).
- RESUME_REQUEST: prepare PHY (FS termination on, OpMode non-driving).
- RESUME_SIG: drive_k asserted to signal K on the bus for ≥1 ms.
- RESUME: clear usb_suspend; upon SE0, restore speed-specific transceiver/termination and OpMode normal; RESUME_WAIT >100 µs → NORMAL.
- Clearing: suspend_clr one-shot clears resume_req sources (both external and software) and de-latches resume_req_r.

Software visibility and controls
- MAIN_CSR (Word 0, read): [0]=suspend, [1]=mode_hs, [2]=attached, [4:3]=LineState snapshot.
- MAIN_CSR (write): din[5]=1 issues rf_resume_req one-shot (software-initiated remote wake request).
- RF/system event latch (Word 3, read-to-clear): [4]=suspend_end, [3]=suspend_start; also holds attach/detach/reset/error events.

UTMI/PHY behavior across suspend/resume
- In SUSPEND: OpMode moves to non-driving, FS term control per state; SuspendM reflects usb_suspend/K detection.
- During device resume signaling: drive_k keeps UTMI transmit context while PHY drives K; DataOut is held to 0x00 during drive_k.

Clock-domain crossings
- usb_suspend/mode_hs/attached/LineState originate in phy_clk and are synchronized into clk_i for RF exposure.
- rf_resume_req (clk_i) and resume_req_i (top) are synchronized into phy_clk; suspend_clr crosses back to clear requesters.

Timing thresholds (internal implementation)

- Suspend entry: idle >3.0 ms (speed-qualified).
- Remote wake eligibility: >5 ms in suspend.
- Device resume signaling (K): ≥1.0 ms.
- Post-resume settle: >100 µs before returning to NORMAL.

Summary
- The core detects and latches suspend entry/exit, exposes live suspend status (susp_o/MAIN_CSR), supports both host-driven resume and device-initiated remote wake with proper eligibility and timing, and safely handles request/clear pulses across clock domains while driving UTMI SuspendM/OpMode/drive_k appropriately.

-- Receive Path Operation --

Receive Path Operation

Overview
- The receive path converts UTMI RX signaling into validated USB tokens/data, performs PID and CRC checks, routes transactions to the targeted endpoint, and (for OUT) stores payload into external SSRAM via a byte→word DMA. It latches status/interrupts for software and enforces endpoint policy (accept/ignore, handshake).

UTMI capture (PHY → core)
- usbf_utmi_if registers UTMI RxValid, RxActive, RxError, and DataIn[7:0] on phy_clk to produce clean, 1-cycle-latency signals: rx_valid, rx_active, rx_err, rx_data.

Packet decode and validation (usbf_pd)
- First non-idle RX byte is latched as PID; the packet is classified as TOKEN, DATA, or HANDSHAKE.
- TOKEN: captures address[6:0], endpoint[3:0], and (for SOF) frame_no[10:0]; computes CRC5 and flags crc5_err on token_valid.
- DATA: streams payload bytes on rx_data_st with CRC16 trailer stripped; computes rx CRC16 residue and flags crc16_err at rx_data_done.
- Sequencing errors (unexpected/aborted packets, early EOP) set seq_err.
- token_valid pulses after a TOKEN and on received ACK handshakes.

Address/match gating and policy entry (usbf_pl)
- Front-end filters disallowed PIDs (hub-only, PRE/ERR, SPLIT; PING only in HS). A function-address match is required.
- match_o = valid & CRC-clean & addressed TOKEN & external endpoint match; this qualifies the packet for policy processing (usbf_pe).

Endpoint selection and policy (usbf_pe)
- Uses decoded endpoint number and EP configuration (EPx_CSR) to decide OUT acceptance and handshakes.
- For OUT/SETUP:
- Checks buffer availability and size policy (sml_ok/lrg_ok), data PID sequence for non-ISO, and CRC status.
- Enables RX DMA when accepting the transaction; on completion selects ACK/NAK/STALL/NYET (HS conditions) per policy.
- For control, tracks direction stage and updates toggles/buffer select per rules.

OUT payload move to memory (usbf_idma, RX path)
- Byte→word pack with read–modify–write for unaligned starts:

- Pre-reads destination word on entry (WAIT_MRD) to seed unchanged bytes.
- Places each rx_data_valid byte into the correct byte lane; increments adr_cb per byte.
- Writes a full word on every 4th byte or a partial word at end-of-packet (wr_last), waiting for final ack (MEM_WR2).
- Counts received bytes in sizu_c; asserts idma_done on rx_data_done.
- Generates memory requests (mreq/mwe) and accepts acks (mack); usbf_mem_arb gives M-side priority so RX meets timing.

Descriptor/status updates and handshakes
- On RX completion, usbf_pe updates the current buffer descriptor (BUFx) with size/next address, may rotate/reload buffers (buf0_set/buf1_set/buf0_rl), updates UC fields (bsel/dpid), and asserts per-EP INT events (e.g., BUF0_DONE/BUF1_DONE).
- Non-ISO sequence or size violations map to NACK or status as configured; CRC16 errors recorded via int_crc16_set.

Errors and protective gating
- pid_cks_err and crc5_err (token-level) are latched as global RF/system events (Word 3 bits [1:0]).
- DATA CRC failures flagged per-EP (int_crc16_set) at rx_data_done.
- Protocol sequencing errors (seq_err) reflected to per-EP status/INT.
- PHY rx_err aborts current packet and prevents handshakes on corrupted traffic (per USB rules).

SOF and timing capture
- SOF frames update frm_nat: microframe index (HS), 11-bit frame number, and time since last SOF (sub-µs tick), aiding scheduling/diagnostics.

DMA and memory arbitration
- usbf_idma is the sole RX writer; usbf_mem_arb gives M-side fixed, immediate priority. W-side (CPU) may access buffer memory only when M is idle; read data is broadcast to both sides but only sampled on their respective acks.

Software visibility and interrupts
- Per-EP: EPx_INT (read-to-clear) holds RX-related events (buffer done, CRC16 error, timeouts, sequence error, unexpected PID, OUT too small).
- Global RF/system (Word 3): pid_cs_err, crc5_err, attach/detach, usb_reset, suspend_start/end (read-to-clear).
- Aggregated inta_o/intb_o = OR(per-EP level) | OR(masked RF/system events).

Latency and CDC notes
- RX indicators are synchronized in phy_clk; decode, policy, and DMA all run in phy_clk for deterministic timing.
- CPU-visible mirrors (MAIN_CSR, INT SOURCE, EP windows) are in clk_i; event edges are safely crossed and latched for software.

Corner behaviors
- RX is maintained across HS/FS modes; token filters adapt (e.g., PING HS-only).
- Control EP handles SETUP and subsequent DATA/STATUS stages with appropriate toggle and buffer selection.
- Memory alignment and partial final word writes are handled transparently by the RX DMA.

-- Transmit Path Operation --

Transmit Path Operation

Overview
- The transmit path takes endpoint policy decisions, fetches payload bytes from external SSRAM via a byte-unpack DMA, assembles USB data packets (Data PID + payload + CRC16) or single-byte handshake tokens, and drives the UTMI TX interface with correct valid/ready handshaking and first/last markers.

Policy decision and start conditions (usbf_pe)
- On an addressed IN (or control-IN) transaction accepted by policy, usbf_pe:
- Chooses the Data PID (this_dpid) per endpoint type, speed, and toggle rules; may request a zero-length packet (ZLP) if configured.
- Enables TX DMA (tx_dma_en) with source address/size from the selected buffer descriptor (EPx_BUF0/BUF1) and enforces size/policy constraints.
- For handshake-only responses (ACK/NACK/STALL/NYET), asserts send_token with token_pid_sel.
- Isochronous IN: no host ACK wait; bulk/interrupt IN: host ACK wait with speed-dependent timeout (HS=22, FS=36 ticks).

Payload fetch from memory (usbf_idma, TX path)
- Double-buffered prefetch of 32-bit words from SSRAM:
- MEM_RD1/MEM_RD2: issue reads; fill alternating buffers on each mack.
- STREAM: unpack bytes to tx_data_st, advancing on rd_next from the packet assembler.
- Size control:
- sizd_c loaded from the descriptor size; decremented per rd_next; idma_done asserts when it reaches zero.
- ZLP support: send_zero_length_r allows a data packet with no payload fetch.
- Memory arbitration: usbf_mem_arb gives M-side (USB) fixed, immediate priority so TX meets timing; W-side (CPU) accesses only when M is idle.

Packet assembly and CRC (usbf_pa)
- States: IDLE → DATA → CRC1 → CRC2 (or WAIT for ZLP) → IDLE.
- Data packet:
- Cycle 0: send Data PID; tx_valid=1; tx_first pulses.
- Payload phase: drive tx_data_st bytes; assert rd_next when (tx_ready && tx_valid_r) to pull next byte.
- End-of-payload: inject first CRC16 byte in the same cycle end is detected with ready; then second CRC16 byte with tx_valid_last=1.
- ZLP: Data PID then CRC16 bytes without payload.
- CRC16: running LFSR initialized to 16'hFFFF; ones-complement reflected ordering sent low byte then high byte.

Handshake-only tokens
- On send_token: output selected PID (ACK/NACK/STALL/NYET) as a single beat; held stable until TxReady=1; tx_valid_last asserted for that beat.

UTMI interface adaptation (usbf_utmi_if)
- DataOut driving:
- If TxReady or tx_first: DataOut <= tx_data.
- During drive_k (resume/chirp phases), DataOut forced to 0x00 (not used for normal TX).
- TxValid generation/holding:
- TxValid asserted for tx_valid, drive_k, or last-beat marking; held until the PHY accepts the byte (TxReady=1), guaranteeing clean UTMI handshaking.
- tx_ready is a registered pass-through of UTMI TxReady back to the core.

Post-transmit updates and interrupts (usbf_pe/usbf_rf)
- After successful IN:
- Update buffer descriptor (size/next address), rotate/reload buffers as configured, and update UC fields (next_bsel/next_dpid).
- Assert per-EP interrupts (e.g., BUF0_DONE/BUF1_DONE); on timeout (no ACK) raise int_to_set.
- Software reads per-EP INT (read-to-clear) and RF/system latches as needed; aggregated inta_o/intb_o reflect OR of per-EP and masked RF/system events.

Flow control and timing
- IN path waits for host ACK in non-iso; timeout sets int_to_set and ends the attempt.
- Speed-aware behavior: PING filtering and timing windows (ACK/data) depend on mode_hs.
- tx_first preloads the first byte to minimize turnaround; tx_valid_last marks the final byte.

CDC and visibility
- All TX datapath/control (policy/assembler/DMA) run in phy_clk.
- CPU-visible mirrors and interrupt latches are in clk_i; events cross domains with one-shots/edge detectors inside usbf_rf.

Corner/robustness behaviors
- Zero-length IN packets supported without memory access.
- Handshake PIDs are held until TxReady to avoid loss on temporary sink backpressure.
- Memory-unpack and rd_next alignment ensure bytes are presented only when UTMI will accept them, preventing under/overruns.

-- Endpoint Policy and Buffer Management --

Endpoint Policy and Buffer Management

Overview
- Endpoint policy and buffering are implemented jointly by the protocol layer (usbf_pe), the per-endpoint register files (usbf_ep_rf), and the internal DMA (usbf_idma), with CPU-visible control/status exposed through usbf_rf.
- EP0 (control) is always present; additional endpoints are enabled by macros. All endpoints share a broadcast update bus and a matched-EP mux.

Endpoint configuration (EPx_CSR) and selection
- Per-EP CSR fields (examples): direction (IN/OUT/CONTROL), transfer type (iso/bulk/interrupt/control), max packet size, DMA enable, stall, OTS stop, endpoint number, and UC fields (uc_bsel/uc_dpd).
- ep_match = (ep_sel == CSR.endpoint_number). Only the matched endpoint latches broadcast updates (idin + strobes).
- Token gating: Only CRC-clean, addressed tokens of allowed PID class reach policy (match_o = valid & !pid_bad & addr match & !crc5_err).

Policy engine (usbf_pe) — per transfer decisions
- Control EP sequencing: SETUP → DATA_IN/DATA_OUT → STATUS_IN/STATUS_OUT, with proper data PID/toggle rules and timeout/error branches.
- IN policy: choose this_dpid (DATA0/1/2/MDATA as applicable), compute payload size (size_next = min(buf_size, max_pl_sz)), optionally ZLP, start TX DMA; for non-iso, wait for host ACK with HS/FS-specific timeout; update descriptor/toggles on success.
- OUT/SETUP policy: check buffer availability, size policy (sml_ok/lrg_ok), CRC16/sequence status; start RX DMA when accepted; on completion choose handshake (ACK/NAK/STALL/NYET-HS) based

on outcome and remaining buffer capacity.
- Handshake selection summary: STALL if stalled; NACK if no buffers/DMA or size-policy violated; ACK on normal success (and on pid_seq_err for non-iso OUT); NYET (HS-only) if success but no further buffers remain.

Data PID/toggle management
- this_dpid/next_dpid derived from speed (mode_hs), transfer type (iso/bulk/control), tr_fr, uc_dpd override, and received PID legality.
- OUT sequence checking (non-iso) enforces expected PID; iso tolerates sequence differences as status.
- UC field updates (uc_dpd/uc_bsel) are written back by dedicated strobes when the EP is matched.

Buffer descriptors and selection (EPx_BUF0/BUF1)
- Descriptor fields: base address (word-aligned), size in bytes (BUF[30:17]), and control/status flags; all-ones (0xFFFF_FFFF) or [31] used to mark NA.
- Selection policy:
- DMA-enabled endpoints: fixed to BUF0.
- Control EP: select by last token direction (in_token/out_token).
- Others: prefer BUF1 if BUF0 is NA or uc_bsel indicates rotation and BUF1 is available; otherwise BUF0.
- Computed for transfer: adr, buf_size, size_next (min(buf_size,max_pl_sz)), new_adr increment, new_size after transfer, buffer_empty/full, and buffer_done.

Descriptor/status writeback and rotation
- Broadcast idin bus + strobes (buf0_set, buf1_set, buf0_rl, uc_bsel_set, uc_dpd_set) update descriptors/UC fields only in the matched EP.
- buf0_rl: release/reload BUF0 from its original CPU-programmed value (DMA use).
- Status composition on writeback:
- Normal: idin[31:17] = {buffer_done, new_size}; idin[SSRAM_HADR+2:4] = new_adr; idin[3:0] = new_adr[3:0].
- out_to_small (DMA OUT short): idin[31:17] = {0, sizu_c}; address retained.
- Buffer rotation: uc_bsel advances on buffer_done (non-DMA), else unchanged; DMA forces next_bsel=0.

DMA interaction and watermarks (usbf_ep_rf + usbf_idma)
- Counters per EP:
- OUT: dma_out_cnt decremented on dma_ack; incremented by packet words when arming a buffer; dma_out_buf_avail indicates space for next OUT.
- IN: dma_in_cnt incremented on dma_ack; decremented by packet words when arming; dma_in_buf_sz1 indicates at least one packet's worth ready.
- dma_req generation in wclk domain with hold policies avoids chattering; dma_ack crosses back to clk to update counters.
- usbf_idma executes RX packing (read-modify-write for unaligned start/partial last) and TX double-buffered prefetch/unpack; ring wrap supported when enabled.

Interrupts and error reporting
- Per-EP INT (read-to-clear) bits include: BUF0_DONE/BUF1_DONE, unexpected PID, CRC16 error, timeouts (ACK/data), sequence error, OUT too small; mapped to inta/intb via per-EP masks (internal) and aggregated in usbf_rf.
- RF/system (Word 3, read-to-clear): pid_cs_err, crc5_err, attach/detach, suspend start/end, usb_reset, nse_err.

Software programming model

- Typical flow per EP:
1) Program EPx_CSR (dir/type/maxpkt/DMA enable/stall/OTS, endpoint number).
2) Program EPx_BUFx descriptor(s) (base, size, flags) and place payload in MEM region for IN.
3) Optionally enable DMA; for PIO flows, software manages buffer ownership.
4) Monitor EPx_INT and/or aggregated interrupts; service and clear by reading EPx_INT and RF Word 3; update descriptors/CSR for next transfers.

Arbitration and timing
- usbf_mem_arb gives M-side (USB core) fixed, immediate priority over W-side (CPU), ensuring endpoint DMA meets USB timing; CPU accesses buffer memory when M is idle.

Notes/edge cases
- OTS stop + out_to_small forces CSR OTS state (01) and records actual received size.
- Zero-length IN supported without memory touch (policy and TX assembler cooperate).
- Endpoint broadcast/match mechanism guarantees only the addressed EP captures shared updates while the core retains a single write path.

-- Memory Arbitration Behavior --

Memory Arbitration Behavior

Participants and clocking
- Arbiter: usbf_mem_arb runs in phy_clk.
- Masters:
- M side (USB core): usbf_pl/usbf_idma (time■critical).
- W side (CPU/Wishbone): requests sequenced by usbf_wb into phy_clk; wclk is unused by the arbiter.
- External memory: single■port 32■bit synchronous SSRAM; address width parameterized by SSRAM_HADR.

Priority and selection policy
- Fixed, immediate priority to M side:
- If mreq=1, M is selected; W is not selected (wsel=0).
- If mreq=0, W is selected when it asserts wreq and remains selected through its ack cycle via wsel=(wreq|wack) & !mreq.
- Preemption: If M asserts mreq at any time, W selection and ack are immediately suppressed; M takes the bus next cycle.

Handshake/ack semantics
- M side:
- Zero■wait from the arbiter: mack = mreq (combinational).
- Writes occur when (mreq & mwe) are high.
- W side:
- wack is generated by a registered helper (wack_r) and only when M is idle:
- wack = wack_r & !mreq; wack_r <= wreq & !mreq & !wack.
- Behavior:
- Single transfer: a one■cycle wack pulse per accepted request.
- Sustained wreq with M idle: wack toggles every other clock (1,0,1,0,…) yielding ~½■rate transfers.
- Any M request immediately forces wack low (preemption), deferring W.

Bus multiplexing and signaling
- Address/data/we muxed:
- sram_adr = wadr if W selected else madr.

- sram_dout = wdin if W selected else mdin.
- sram_we = (wreq & wwe) when W selected; else (mreq & mwe) for M.
- Read enable: sram_re = 1 (always reading); both masters see sram_din.
- Read data usage: mdout = wdout = sram_din; each master must sample only on its ack (mack or wack).

Addressing conventions
- W side uses word■based addressing: ma_adr[SSRAM_HADR+2:2] (32■bit alignment).
- M side uses madr[SSRAM_HADR:0] as the memory word address.

Throughput, fairness, and timing implications
- Deterministic USB priority: M can continuously occupy the bus; W can be starved under heavy USB load—intentional to meet USB timing.
- When M is idle, W achieves up to one accepted transfer every two phy_clk cycles if it holds wreq high (due to toggling wack).
- No internal pipelining: masters must align address/data with ack; there is no buffering inside the arbiter.

Cross■domain interactions
- The arbiter assumes W■side requests presented in phy_clk (usbf_wb bridges clk_i→phy_clk and returns a one■shot ack to clk_i after ma_ack).
- Read/write completions seen by Wishbone occur only after the arbiter has produced the corresponding ack on phy_clk and usbf_wb has generated wb_ack_o on clk_i.

Reset and parameterization
- Arbiter state (wack_r) is synchronous to phy_clk; optional async assert/sync deassert if compiled.
- SSRAM_HADR sets memory depth; all muxing/handshakes scale accordingly.

Corner cases and guarantees
- Simultaneous requests: M always wins; W receives no ack that cycle.
- Selection hold: W remains selected through its ack cycle via (wreq|wack), even if it drops wreq after seeing wack.
- Always■on read enable requires an SSRAM interface tolerant of continuous reads (common for simple synchronous SSRAMs).

-- Wishbone Transaction Sequencing and Acknowledge --

Wishbone Transaction Sequencing and Acknowledge

Target decode and address map
- Address MSB selects target (compile-time HADR):
- RF_SEL = !wb_addr_i[HADR] → register-file space (usbf_rf)
- MEM_SEL = wb_addr_i[HADR] → memory space (usbf_mem_arb W-side)
- 32-bit word accesses only; wb_addr_i is word-aligned for both regions.

Request capture and control FSM (phy_clk domain)
- wb_req_s1 <= (wb_cyc_i & wb_stb_i) sampled in phy_clk.
- States: IDLE, MA_WR, MA_RD, W0, W1, W2.
- IDLE:
- MEM write: set ma_req=1, ma_we=1 → MA_WR
- MEM read: set ma_req=1 → MA_RD
- RF write: pulse rf_we_d=1 → W0

- RF read: pulse rf_re=1 → W0
- MA_WR/MA_RD: hold ma_req (and ma_we for write) until ma_ack=1; on ma_ack pulse wb_ack_d=1 → W1
- W0 (RF access): pulse wb_ack_d=1 → W1
- W1 → W2 → IDLE: fixed two-cycle cooldown before accepting a new request (prevents immediate retrigger if CYC/STB still high).

Acknowledge generation (wb_clk domain)
- wb_ack_d (from phy_clk) is sampled into wb_clk and edge-detected to produce a single-cycle wb_ack_o pulse per completed access.
- Classic single-termination protocol: only ACK_O is used (no ERR/RTY/STALL). One pulse per transaction.

Data path and validity
- Read data muxed to wb_clk domain and registered on wb_data_o:
- RF read: wb_data_o <= rf_din; ACK indicates data valid in the same wb_clk.
- MEM read: wb_data_o <= ma_din; ACK asserted only after ma_ack, so data is valid on the ACK pulse.
- Writes complete on:
- RF accesses: single rf_we_d pulse (W0) then WB ACK.
- MEM writes: ma_ack observed, then WB ACK is generated.

CDC assumptions and constraints
- Two CDCs: (wb_cyc_i & wb_stb_i) into phy_clk; wb_ack_d back into wb_clk (edge-detected to a one-shot).
- Master must hold address/data and CYC/STB stable until wb_ack_o asserts; no internal address/data latching.
- Simple level/edge scheme assumes related clocks or adequate stability of producer/consumer signals.

Throughput and sequencing effects
- RF accesses: minimum 1 phy_clk strobe + 2 phy_clk cooldown cycles per transaction; one wb_clk ACK pulse per access.
- MEM accesses: latency dominated by ma_ack from usbf_mem_arb; WB ACK only after ma_ack; repeated MEM requests paced by arbiter/M-side priority.
- Back-to-back transactions require either deasserting CYC/STB or waiting for the FSM to return to IDLE after W2.

Interaction with memory arbiter
- usbf_wb asserts ma_req/ma_we to the arbiter W-side; ack (ma_ack) returns when M-side is idle and the transfer completes.
- USB core (M-side) has fixed priority; W-side (Wishbone) can be delayed/starved under heavy USB traffic.

Unsupported/omissions (intentional for classic WB B3)
- No byte enables (SEL), no bursts/tags, no pipeline/STALL, no error/retry signaling; single-beat classic handshake only.

Reset behavior
- FSM reset synchronous to phy_clk (optional async assert/sync deassert by build macro). wb_clk ack edge-detector registers reset in wb_clk domain.

-- Error Detection, Reporting, and Recovery --

Error Detection, Reporting, and Recovery

Detection (where errors are caught)
- Link/PHY level (usbf_utmi_if/usbf_utmi_ls):
- rx_err from UTMI flags a receive error on the current packet; packet is aborted.
- Line-state monitors detect attach/detach, suspend entry/exit, and USB reset (SE0>2.5 µs). A non-SE0 error (nse_err) is also latched at RF level.
- Packet/protocol level (usbf_pd):
- PID integrity: pid_cks_err (PID nibble complement mismatch).
- Token CRC5: crc5_err on token_valid for TOKENs (ADDR/ENDP/CRC parsed).
- Data CRC16: crc16_err at rx_data_done using a residue check over payload+CRC.
- Sequencing: seq_err on unexpected/unsupported PID, premature EOP, or abort by rx_err.
- Endpoint/policy level (usbf_pe):
- Unexpected PID relative to endpoint type: int_upid_set.
- PID sequence (data toggle) errors: int_seqerr_set (ISO recorded as status; non-ISO affects handshake policy).
- Size policy violations: to_small, to_large checked against sml_ok/lrg_ok; out_to_small recorded when DMA OUT completed with short size.
- Timeouts: int_to_set on IN ACK-wait expiry (HS=22, FS=36 ticks) or OUT data-wait expiry.
- Abort conditions: buffer_overflow, new token match while not IDLE, or to_large detected late.
- DMA/memory path (usbf_idma/usbf_mem_arb):
- Safe RX write completion: final partial word write (wr_last) ensured before exit.
- TX size underflow/complete: idma_done when size reaches zero; abort input allows immediate cancellation on policy errors.
- Arbiter prevents W-side interference; not an error detector but avoids bus hazards.
- Wishbone bridge (usbf_wb):
- No bus ERR/RTY; relies on stable CYC/STB/data until ack. CDC scheme assumes producer stability; no explicit WB error signaling.

Reporting (how software is notified)
- Per-endpoint INT (EPx_INT, read-to-clear):
- BUF0_DONE/BUF1_DONE, CRC16 error (int_crc16_set), timeout (int_to_set), sequence error (int_seqerr_set), unexpected PID (int_upid_set), OUT too small (out_to_small), plus buffer/UC updates as events.
- Per-EP masks internal to the EP gate inta/intb level outputs.
- RF/system INTERRUPT SOURCE (Word 3, read-to-clear):
- [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.
- int_srca[15:0] mirrors any-pending per-EP interrupts for visibility.
- Aggregation:
- inta_o = OR(per-EP inta) | OR(masked RF/system inta_msk & int_srcb).
- intb_o = OR(per-EP intb) | OR(masked RF/system intb_msk & int_srcb).

Recovery/mitigation (automatic hardware actions)
- Corrupted traffic handling:
- rx_err and CRC failures: data is ignored; no success handshake sent (per USB). usbf_pe refrains from ACK on CRC/bad packet conditions.
- Handshake policy on errors (usbf_pe):
- STALL if endpoint is stalled.
- NACK on no buffers/DMA or size-policy violation (to_small/to_large); NYET (HS) when success but no

further buffers remain.
- ACK on normal success and, for non-ISO OUT, when pid_seq_err is detected (host retry logic).
- Timeout recovery:
- IN ACK timeout or OUT data timeout raises int_to_set; transfer is terminated and endpoint returns to IDLE.
- Toggle/state hygiene:
- Data toggles (this_dpid/next_dpid) update only on successful transfers; sequence errors do not advance toggle.
- Descriptor/status correction:
- OUT too small (DMA): descriptor written with actual sizu_c (buffer_done=0), address retained; out_to_small flagged.
- On buffer_done with DMA: buf0_rl releases/rotates buffer; UC fields (next_bsel/next_dpid) updated via uc_*_set.
- DMA abort and completion guarantees:
- usbf_idma supports abort to stop active transfers on policy/link errors; ensures final partial RX write (MEM_WR2) is acknowledged before exit.
- Link-level recovery:
- USB reset detected forces link state machine to RESET then speed negotiation; suspend/resume handled with proper K signaling and suspend_clr pulse to clear core suspend.

Protective gating (preventing bad packets from acting)
- Token gating: match_o requires valid addressed tokens with clean CRC5 and allowed PID class; HS-only PING filtered in FS.
- Handshake suppression on PHY error: corrupted/aborted packets do not produce success handshakes.
- Memory arbitration: fixed M-side priority prevents WB interference with time-critical USB transfers.

Software responsibilities and clear/ack flow
- Read-to-clear semantics:
- EPx_INT and RF/system Word 3 are cleared by read; software should snapshot before clearing if needed.
- Service path:
- On inta_o/intb_o, read RF Word 3 and the implicated EPx_INT to determine cause; update EPx_CSR/BUFx as required (e.g., re-arm buffers, adjust policies).
- Limitations (interface-level):
- Wishbone port does not signal bus errors (no ERR/RTY/STALL). Correct operation requires the master to hold request/address/data stable until wb_ack_o.

Diagnostics
- frm_nat provides frame/microframe index and time since SOF for correlating errors/timeouts with bus timing.

-- Timing and Latency Considerations --

Timing and Latency Considerations

Clock domains and CDC
- Two primary clocks: phy_clk (UTMI/link/protocol/mem_arbiter) and clk_i (Wishbone/RF CPU-visible regs).
- Crossings use level strobes + one-shots and simple synchronizers:
- Wishbone: wb_req (clk_i→phy_clk), wb_ack_d (phy_clk→clk_i one-shot).
- RF events and vendor writes: wclk→clk crossings with read-to-clear latches.
- Expect 1–2 cycles of crossing latency in each direction for control/ack pulses.

UTMI interface latency (usbf_utmi_if/usbf_utmi_ls)
- RX path: rx_valid/active/err and rx_data are registered once on phy_clk → fixed 1-cycle RX latency.
- TX path: DataOut updates when TxReady=1 or on tx_first (preload); TxValid is held until TxReady=1, guaranteeing byte acceptance without bubbles.
- Link timing windows (approximate):
- Reset detect: SE0 > 2.5 µs.
- Suspend entry: idle > 3.0 ms; resume K signaling: ≥1.0 ms; resume settle: >100 µs.
- HS chirp negotiation: device K chirp >1.2 ms; commit after 3 K/J pairs (6 edges).
- Attach debounce: ~100 ms.

Protocol layer timing (usbf_pd/usbf_pa)
- RX decode (usbf_pd):
- token_valid: 1-cycle strobe after second TOKEN byte.
- DATA payload pipeline: 2-cycle pipeline (d0/d1/d2) before rx_data_valid; CRC16 residue check at rx_data_done (no extra cycles beyond packet end).
- TX assemble (usbf_pa):
- Cycle 0 of send_data: Data PID sent; tx_first asserted for preload.
- CRC insertion: first CRC byte injected in the same cycle the last payload byte is accepted (when tx_ready && tx_valid_r); second CRC byte on the next tx_ready with tx_valid_last.
- Handshake tokens (ACK/NAK/STALL/NYET): single beat; held until TxReady.

Timeouts and USB-transaction windows (usbf_pe)
- IN ACK wait (non-iso): HS=22 ticks, FS=36 ticks (tick base in phy_clk domain per implementation constants).
- OUT data wait: same HS/FS thresholds; counters pause/clear while activity qualifies.
- On expiry: transfer aborted; int_to_set asserted.

DMA and buffer movement timing (usbf_idma)
- RX (USB→memory): pre-read (read-modify-write) of first word adds one memory read latency before first write; final partial word write (wr_last) guaranteed and may add one extra ack.
- TX (memory→USB): double-buffered prefetch (two 32-bit words) hides most memory read latency; send_data asserts when first word is accepted (rd_first & mack_r).
- Handshakes: mreq held until mack_r; idma_done at rx_data_done (RX) or when sizd_c decrements to zero (TX).

Memory arbitration throughput (usbf_mem_arb)
- M-side (USB) priority: mack = mreq (combinational zero-wait from arbiter's perspective).
- W-side (Wishbone) service cadence when M is idle: wack pulses every other phy_clk (≈½ rate) if wreq held high; immediate preemption on any mreq.
- Read data is broadcast; masters must only capture on their ack.

Wishbone transaction latency (usbf_wb)
- RF accesses: single phy_clk strobe (W0) + two-cycle cooldown (W1/W2) → wb_ack_o one-shot in clk_i with minimal latency.
- MEM accesses: wb_ack_o issued only after ma_ack from arbiter; overall latency = request crossing + arbiter selection + SRAM ack + ack crossing.
- Back-to-back WB ops: require deasserting CYC/STB or waiting for FSM to return to IDLE after W2.

Frame/microframe timing (frm_nat)
- Half-microsecond tick (hms_clk) accumulates since last SOF; microframe index (0..7) derived from repeated HS SOFs.

- frm_nat packs {microframe index, frame number, time since SOF} for scheduling/diagnostics and correlating timeouts.

First-byte turnaround optimizations
- tx_first permits preloading the first byte into DataOut even if TxReady is low, minimizing start-of-packet latency.
- CRC insertion in the same cycle as last data acceptance avoids extra bubble cycles at packet tail.

Throughput and head-of-line considerations
- USB-side transfers can run continuously when memory acks keep pace; W-side can be starved under load by design to meet USB timing.
- RX data pipeline implies a fixed 2-cycle latency to rx_data_valid; sustained throughput remains 1 byte/phy_clk when UTMI provides continuous bytes.

Reset and build-time options impacting timing
- Default synchronous reset; optional USBF_ASYNC_RESET enables async assert/sync deassert.
- Parameters (e.g., SSRAM_HADR) scale memory address widths; timing of external SRAM/PHY dominates absolute latencies.

Interrupt visibility latency
- Per-EP events latch in clk (core) or wclk (RF) domains and are visible to software after domain-crossing and register update—typically 1–2 local cycles plus CDC.

Summary
- Deterministic 1-cycle UTMI RX latency, 2-cycle RX payload pipeline, immediate TX PID emission with preload, HS/FS timeout windows for ACK/data, and fixed-priority memory arbitration shape end-to-end timing. Wishbone interactions add small fixed FSM delays and CDC edge-detected acks; under heavy USB load, W-side bandwidth reduces to ~½ phy_clk when serviceable.

-- Corner Cases and Sequencing Rules --

Corner Cases and Sequencing Rules

Link/power-state sequencing (UTMI/link)
- Suspend/resume:
- Device enters suspend after bus idle >3.0 ms; resume requires K signaling ≥1.0 ms, with >100 µs settle before returning to NORMAL.
- Software-initiated remote wake (rf_resume_req or resume_req_i) is latched and honored only after >5 ms idle; it auto-clears on suspend_clr.
- Reset and HS negotiation:
- USB reset detected on SE0 >2.5 µs; device forces FS interface during RESET then runs HS chirp. Commit to HS after 3 K/J pairs (6 edges). If no chirp, remain FS.
- Attach/debounce:
- usb_attached asserted only after ~100 ms debounce in ATTACH state.

Token filtering and acceptance gates
- Only CRC-clean, addressed, and allowed PIDs reach policy: match_o = !pid_bad & fsel & match & token_valid & !crc5_err.
- Disallowed PIDs are filtered before policy:
- PRE/ERR/SPLIT always filtered; PING filtered in FS (mode_hs=0).
- ACK handshakes observed on RX cause a token_valid pulse but do not carry data.

Control-transfer rules (EP0)
- Direction tracking (in_token/out_token/setup_token) gates which buffer/direction is active in DATA and STATUS stages.
- Zero-length STATUS stages are supported; zero-length IN packets can be emitted without memory touch via send_zero_length.

Data PID/toggle rules
- this_dpid selects TX data PID; next_dpid is stored for the subsequent transfer.
- Toggle advances only on successful (ACKed) non-iso transfers; sequence errors do not update toggle.
- Non-iso OUT with pid_seq_err: device still returns ACK (host corrects on retry); ISO records seqerr as status only.
- HS/ISO rules allow MDATA/DATA2 per tr_fr and mode_hs.

Handshake selection ordering (usbf_pe)
- Priority: STALL if endpoint stalled; else NAK for no buffers/DMA or size-policy violations; else ACK on success; NYET (HS-only) if success but no further buffers are available.
- TOKEN state emits the chosen handshake as a single beat and returns to IDLE next cycle.

Timeout sequencing
- IN (non-iso): wait for host ACK up to HS=22 or FS=36 tick windows; on expiry, assert int_to_set and return to IDLE.
- OUT: if no data within timeout window (HS/FS thresholds), assert int_to_set and return to IDLE.

OUT size-policy corner cases
- to_small: sizu_c < max_pl_sz and sml_ok=0 → NAK and no descriptor advance.
- to_large: sizu_c > max_pl_sz and lrg_ok=0 → NAK or abort depending on detection point.
- DMA OUT short (out_to_small): write descriptor with actual sizu_c, buffer_done=0, and retain address; optionally force OTS state=01 if ots_stop=1.

Descriptor and UC-field writeback ordering
- Buffer updates vs UC status are separate strobes:
- buf*_set writes updated size/address (idin[31:17]={buffer_done,new_size}; idin[SSRAM_HADR+2:4]=new_adr; idin[3:0]=new_adr[3:0]).
- uc_*_set writes {next_dpid,next_bsel} in idin[3:0].
- Low-nibble reuse rule: the meaning of idin[3:0] depends on which strobe is active; software must not assume a single semantic.
- buf0_rl is used to reload BUF0 from CPU-programmed buf0_orig for DMA cycles when buffer_done.

Buffer selection and rotation rules
- DMA-enabled endpoints: always use BUF0; next_bsel forced to 0 on UC update.
- Control EP: buffer selected by last token direction (in_token/out_token) to match control stages.
- Others: prefer BUF1 if BUF0 is NA or uc_bsel indicates rotation and BUF1 is available; else use BUF0.
- UC buffer select (uc_bsel) advances only when buffer_done on non-DMA endpoints.

Abort and preemption rules
- Transfer is aborted if: buffer_overflow during RX, a new token match arrives while not IDLE, or to_large detected late post-match.
- On abort, in-flight DMA is cancelled via abort; RX path still completes final partial write (wr_last) before exit.

RX/TX pipeline and CRC insertion details
- RX payload emerges after a fixed 2-cycle pipeline; CRC16 residue evaluated at rx_data_done only.
- TX: first CRC byte is injected in the same cycle the last payload byte is accepted (tx_ready && tx_valid_r); second CRC byte follows on next ready with tx_valid_last.
- tx_first allows preloading the first data byte even when TxReady=0 to minimize turnaround.

DMA/read-modify-write corner cases (usbf_idma)
- RX unaligned starts: pre-read of destination word (WAIT_MRD) seeds unchanged bytes; final partial word is written in MEM_WR2 and must be acked before DONE.
- TX streaming: two-word prefetch buffers; send_data asserts only after first read ack (rd_first & mack_r).
- Ring wrap: address wraps to 0 only if dma_en is set and adrw_next reaches last_buf_adr.

Arbitration and Wishbone sequencing rules
- Memory arbiter fixed priority:
- M-side (USB) wins immediately; mack=mreq (combinational). W-side can be preempted any cycle by M-side.
- W-side selection held for the ack cycle via (wreq|wack) & !mreq; sustained wreq yields a wack every other phy_clk while M is idle.
- Wishbone slave FSM:
- RF access: 1-cycle strobe (W0) then two-cycle cooldown (W1/W2). MEM access: wb_ack issued only after ma_ack from arbiter.
- wb_ack_o is a one-shot pulse in clk_i; master must hold ADR/DAT/CYC/STB stable until ack.
- No ERR/RTY/STALL; misuse (early deassertion) is undefined.

Interrupt latching/clear sequencing
- Read-to-clear:
- EPx_INT clears on read of the endpoint INT register (adr offset +1).
- RF/system events (Word 3) clear on read; does not affect per-EP INTs.
- Recommended service order: read EPx_INT to gather per-EP detail, then RF Word 3 for system edges, to avoid losing context.
- int_srca mirrors per-EP any-pending and is for visibility only (no clear effect).

Build and address-alignment caveats
- Endpoint enables must be contiguous from 0; top-level initial block warns if not.
- W-side memory addresses are word-based (drop [1:0]); software must align buffer base addresses accordingly.

General safeguards
- Always sample SRAM read data only on the respective ack (mack or wack); sram_re is always asserted.
- Broadcast writeback bus updates (idin + strobes) are captured only by the ep_match endpoint; others ignore, preventing cross-EP corruption.
- Toggles and descriptor advances occur only after successful policy decisions; error paths suppress state advancement.

REGISTER MAP

-- Addressing Scheme and Regions --

Addressing Scheme and Regions

Top-level regions (Wishbone address decode)
- Single MSB selects region (compile-time HADR):
- Test build: HADR=17 → RF region when !wb_addr_i[17], MEM region when wb_addr_i[17].
- Non-test: HADR=12 → RF region when !wb_addr_i[12], MEM region when wb_addr_i[12].
- 32-bit port, word-aligned accesses: addresses inside the RF region are word addresses; byte lanes [1:0] are not used.

RF region (register-file space)
- Addressing into RF (word addressing): usbf_wb presents adr[8:2] (7-bit word index) to usbf_rf.
- System bank (aliased reads):
- adr[6:2]==0 or 1 → same read contents; writes use exact compares to adr[6:2]==0.
- Word map (by adr[2:0]):
- 0: MAIN_CSR (status; write bit[5] issues rf_resume_req)
- 1: FUNCTION_ADDRESS [6:0]
- 2: INTERRUPT MASKS (inta_msk[8:0], intb_msk[8:0])
- 3: INTERRUPT SOURCE (int_srcb[8:0] read-to-clear, int_srca[15:0] mirror)
- 4: FRM_NAT (frame/microframe snapshot)
- 5: UTMI VENDOR IF (read status, write control with load pulse)
- Per-endpoint windows (EP0..EP15):
- Base word address for EPn = 4 + 4*n (each EP occupies 4 words / 16 bytes).
- Offsets within EP window:
- +0: EPx_CSR
- +1: EPx_INT (read-to-clear event bits; EP-local masks gate level outputs)
- +2: EPx_BUF0 descriptor
- +3: EPx_BUF1 descriptor
- Notes:
- All RF addresses are 32-bit words; increment by 1 per 4 bytes.
- RF/system (Word 3) and EPx_INT are read-to-clear; int_srca is visibility-only and not cleared by reads.

MEM region (external buffer memory space)
- Region selection: wb_addr_i[HADR]=1 routes to usbf_mem_arb W-side for SSRAM access.
- W-side addressing is word-based: byte address [1:0] dropped. ma_adr = wb_addr_i[HADR-1:2].
- External SSRAM address bus: sram_adr_o[SSRAM_HADR:0] (word address into memory; depth = 2^(SSRAM_HADR+1) words).
- Total buffer size (bytes): 2^(SSRAM_HADR+3). Example (test): SSRAM_HADR=14 → 2^(14+3)=131072 bytes (128 KiB).

Internal memory address conventions
- Protocol/DMA start addresses use byte indexing:
- idma adr[SSRAM_HADR+2:0] = start byte address in buffer space.
- Word address to memory: madr = adr[SSRAM_HADR+2:2]; byte-in-word index adr_cb = adr[2:0].
- M-side (USB core) presents word addresses to arbiter: madr[SSRAM_HADR:0].
- W-side (Wishbone) presents word addresses via ma_adr; arbiter muxes wadr/madr to sram_adr.

Endpoint buffer descriptors (address fields)
- buf0/buf1 descriptors carry buffer base and size:

- Address field width: bufX[SSRAM_HADR+2:4] store the word-aligned base (low nibble separated for UC/status use).
- Low nibble multiplexing:
- On buffer descriptor writes (buf*_set): idin[3:0] = new_adr[3:0].
- On UC status writes (uc_*_set): idin[3:0] = {next_dpid, next_bsel}.
- Software must program word-aligned base addresses; low bits may be re-used by UC/status updates per the active strobe.

Addressing rules and caveats
- Word alignment: All RF and MEM accesses are 32-bit word-oriented; WB byte selects are not used.
- Region split is a single MSB; ensure system address map reserves contiguous halves for RF vs MEM according to HADR.
- Endpoint enables must be contiguous starting at EP0 (checked at simulation start), but this does not alter the EP window addressing pattern.
- Read data from SSRAM is broadcast to both masters; only sample on respective ack (mack/wack); sram_re is permanently asserted (interface must tolerate continuous reads).

-- System/Core Registers --

System/Core Registers

Scope and access
- Location: RF (register-file) region selected by !wb_addr_i[HADR] (HADR=17 test, 12 non-test).
- Bus: 32-bit, word-aligned; byte lanes [1:0] unused. usbf_wb presents adr[8:2] (7-bit word index) into usbf_rf.
- System bank addressing: reads are aliased when adr[6:2]==0 or 1 (same contents); writes must target adr[6:2]==0.
- Clock domain: CPU-visible registers in clk_i; status sources from phy_clk are safely crossed/latched.

System register bank (word offsets by adr[2:0])
- Word 0: MAIN_CSR (RW, plus write-side-effect)
- Read fields:
- [4:3] LineState[1:0] (UTMI line state snapshot)
- [2] usb_attached (debounced)
- [1] mode_hs (1=HS, 0=FS)
- [0] suspend (core in suspend)
- Other bits read as 0
- Write side-effect:
- din[5]=1 issues a one-shot rf_resume_req (software-initiated remote wakeup)
- Reset: 0x0000_0000

- Word 1: FUNCTION_ADDRESS (RW)
- [6:0] funct_adr (USB device address)
- Other bits read as 0; writes to reserved bits ignored
- Reset: 0x0000_0000 (address=0)

- Word 2: INTERRUPT MASKS (RW)
- [24:16] intb_msk[8:0] (mask RF/system events into intb_o)
- [8:0] inta_msk[8:0] (mask RF/system events into inta_o)
- Other bits read as 0
- Reset: 0x0000_0000 (all masked off)

- Word 3: INTERRUPT SOURCE (R/RC for RF/system; RO mirror for per-EP)
- [28:20] int_srcb[8:0] (latched RF/system events; read-to-clear)
- [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
- [15:0] int_srca (RO mirror: OR of per-EP inta/intb; visibility only, not cleared here)
- Other bits read as 0
- Reset: 0x0000_0000

- Word 4: FRM_NAT (RO)
- [31:0] frm_nat: snapshot from protocol layer (microframe index, 11-bit frame number, time since SOF)
- Reset: 0x0000_0000

- Word 5: UTMI VENDOR INTERFACE (R/W with write-side-effect)
- Read: [7:0] utmi_vend_stat (latched VStatus_pad_i)
- Write: [3:0] utmi_vend_ctrl (driven to VControl_pad_o); write asserts a one-shot VControl_Load_pad_o pulse (clk_i→phy_clk crossing)
- Other bits read as 0; writes to reserved bits ignored
- Reset: 0x0000_0000

Interrupt aggregation (top-level outputs)
- inta_o = OR(per-EP inta) | OR(int_srcb & inta_msk)
- intb_o = OR(per-EP intb) | OR(int_srcb & intb_msk)

Read/Write semantics and side effects
- Read-to-clear: Word 3 int_srcb[8:0] clears on read; does not affect per-EP INTs.
- One-shot generators:
- MAIN_CSR write din[5]=1 → rf_resume_req pulse in core domain
- UTMI vendor write → VControl_Load_pad_o pulse in PHY domain
- Aliasing: System bank reads at adr[6:2]==0 or 1 return identical data; all writes must be to adr[6:2]==0.

Reset behavior
- On reset: FUNCTION_ADDRESS=0; INTERRUPT MASKS cleared; INT SOURCE cleared; MAIN_CSR, FRM_NAT, UTMI vendor IF cleared; vendor write strobe deasserted; rf_resume_req deasserted.

Access notes
- All addresses are 32-bit words; increment by 1 per 4 bytes.
- Status fields (LineState, attached, suspend, mode_hs) reflect debounced/qualified link indications sourced in phy_clk.
- Per-endpoint CSRs/INT/BUF registers reside in EP windows and are documented separately from the system bank.

-- Interrupt Masks and Status --

Interrupt Masks and Status

Overview
- Two interrupt outputs (clk_i domain):
- inta_o = OR(per-EP inta) | OR(int_srcb & inta_msk)
- intb_o = OR(per-EP intb) | OR(int_srcb & intb_msk)
- Masking layers:
- System/RF masks: inta_msk[8:0], intb_msk[8:0] (system bank Word 2)
- Per-endpoint masks: iena[5:0], ienb[5:0] (in EPx_INT)
- Domains: RF/system latches and aggregation in clk_i (wclk in rf); event sources originate in phy_clk and are crossed safely.

System/RF interrupt masks (Word 2, RW)
- [24:16] intb_msk[8:0]: masks RF/system events into intb_o
- [8:0] inta_msk[8:0]: masks RF/system events into inta_o
- Reset: 0x0000_0000 (all masked)

System/RF interrupt status (Word 3)
- [28:20] int_srcb[8:0] (RC, read-to-clear):
- [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
- [15:0] int_srca (RO mirror): OR of all per-EP pending (ep_inta | ep_intb); visibility only
- Other bits read 0; reset 0x0000_0000
- Note: Reading Word 3 clears only int_srcb; EP INTs are cleared by reading EPx_INT.

Per-endpoint interrupt window (EPx_INT)
- Address: EPn base + 1 (EPn base = 4 + 4*n)
- Status bits (int_stat[6:0], read-to-clear):
- [6] out_to_small, [5] seqerr, [4] buf1_set, [3] buf0_set, [2] upid,
[1] crc16, [0] timeout
- Masks (RW within EPx_INT): iena[5:0] $\rightarrow$ per-EP inta, ienb[5:0] $\rightarrow$ per-EP intb
- Implementation note: [3] and [4] share mask bit index 3 for level gating
- Level outputs (wclk domain):
- ep_inta = OR(int_stat[i] & iena[i])
- ep_intb = OR(int_stat[i] & ienb[i])
- Read of EPx_INT clears int_stat for that EP; masks preserved

Aggregation and visibility
- inta_o/intb_o = OR of all EP-level masked levels plus masked RF/system events (int_srcb & inta_msk/intb_msk)
- int_srca[15:0] is a software-visible bitmap of EPs with any pending (inta or intb); it does not generate/clear interrupts

Event sourcing and latching
- RF/system events come from qualified edges in rf: attach/detach, suspend start/end, usb_reset, rx_err, pid_cs_err, crc5_err, nse_err
- Per-EP events are set by protocol-layer strobes (phy_clk domain) captured in the matched EP block

Reset state
- inta_msk/intb_msk = 0; int_srcb = 0; int_srca = 0
- Per-EP int_stat cleared; per-EP masks reset to 0

Recommended software service sequence
1) On inta_o/intb_o, read EPx_INT for implicated EPs (guided by int_srca or polling) to clear EP events
2) Read RF Word 3 to collect/clear RF/system events
3) Re-arm descriptors and masks as needed

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

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

Addressing and layout
- EP window size: 4 words (16 bytes) per endpoint.
- EPn base word address = 4 + 4*n (n=0..15).
- Offsets within each EP window:
- +0: EPx_CSR (Control/Status)
- +1: EPx_INT (Interrupts: status + masks; read-to-clear status)
- +2: EPx_BUF0 (Buffer 0 descriptor)
- +3: EPx_BUF1 (Buffer 1 descriptor)
- Presence: EP0 always present; EP1..EP15 enabled by macros. Disabled EPs map to a dummy block (no DMA/INT; CSR=0, BUF*=0xFFFF_FFFF).

EPx_CSR (Control/Status, 32-bit)
- Key configuration fields:
- Direction: csr[27:26] = 01 IN, 10 OUT (00 control used by EP0)
- Endpoint number for match compare: csr[21:18]
- Transfer type: csr[25:24] (01 iso, 10 bulk; others by design)
- Max packet size (bytes): csr[10:0]
- DMA enable: csr[15] (not used for control EP)
- Stall/disable/status: csr[23:22] (01 disabled, 10 stall)
- OTS stop: csr[13] (OUT-too-small handling hooks)
- UC (user control) fields (updated by core strobes):
- uc_bsel[1:0] (buffer select hint/rotation)
- uc_dpd[1:0] (data PID/toggle control)
- Notes:
- For non-DMA EPs, uc_bsel advances on buffer_done.
- For DMA EPs, BUF0 is used; next_bsel forced to 0 by core on updates.

EPx_INT (Interrupts, 32-bit)
- Read-to-clear status (set-once) bits int_stat[6:0]:
- [6] out_to_small
- [5] seqerr (sequence error)
- [4] buf1_set event
- [3] buf0_set event
- [2] upid (unexpected PID)
- [1] crc16 (data CRC error)
- [0] timeout (IN ACK wait or OUT data wait)
- Per-EP masks (RW):
- iena[5:0] gates EP events into per-EP inta
- ienb[5:0] gates EP events into per-EP intb
- Implementation note: [3] and [4] share the same mask index for level gating as coded.
- Reading EPx_INT clears int_stat; masks are preserved.

EPx_BUF0 / EPx_BUF1 (Buffer descriptors, 32-bit)

- Contents (conceptual): word-aligned base address into external SSRAM, programmed size (bytes), and control/status flags (e.g., own/pending, interrupt-on-done, PID/toggle hints).
- Address fields:
- Core writes updated descriptors via idin bus; word-aligned base stored in bufX[SSRAM_HADR+2:4].
- Low nibble reuse rule:
- On descriptor writes (buf*_set): idin[3:0] = new_adr[3:0]
- On UC status writes (uc_*_set): idin[3:0] = {next_dpid, next_bsel}
- Core update strobes (captured only when ep_match is true):
- buf0_set / buf1_set: write updated descriptor from idin
- buf0_rl: reload BUF0 from last CPU-programmed buf0_orig (DMA release)
- uc_bsel_set / uc_dpd_set: write UC fields from idin[3:0]
- Reset defaults: buf0 = buf1 = 0xFFFF_FFFF; buf0_orig captures last CPU write to BUF0 for DMA accounting.

Endpoint match and broadcast update gating
- ep_match = (ep_sel[3:0] == csr[21:18])
- Only the matched endpoint captures broadcast idin/strobes; unmatched endpoints ignore, preventing cross-EP corruption.
- Core-facing mux returns the matched EP's csr, buf0, buf1, dma_in_buf_sz1, dma_out_buf_avail for policy decisions.

DMA helpers and flow indicators
- dma_req (per EP) is generated internally (wclk domain) based on counters and DMA enable.
- dma_in_buf_sz1: indicates at least one max-packet worth of IN data is present (nonzero max_pl_sz and counter threshold met).
- dma_out_buf_avail: indicates sufficient space remains for another OUT packet (words left ≥ one packet).

Programming and service model (typical)
1) CPU programs EPx_CSR (dir/type/maxpkt/DMA) and BUF descriptors (BUF0 and optionally BUF1) in RF space.
2) Software fills/reads payloads in MEM region (external SSRAM) for IN/OUT respectively.
3) Core moves data and updates BUF descriptors via buf*_set/buf0_rl and UC fields via uc_*_set; EPx_INT latches events (read-to-clear).
4) Software services EPx_INT (clear-on-read), then updates descriptors for subsequent transfers.

Notes and caveats
- All EP window accesses are 32-bit word addressed; increment by 1 per 4 bytes.
- Disabled endpoints (dummy) read as CSR=0, BUF*=0xFFFF_FFFF, never assert DMA/INT.
- Low-nibble semantics depend on which strobe is active (descriptor vs UC writeback).

-- Vendor/PHY Access Registers --

Vendor/PHY Access Registers

Scope
- Provides software access to UTMI vendor sideband signals and key PHY/link state snapshots.
- Lives in the RF (register-file) region; all accesses are 32-bit, word-aligned in clk_i domain with safe crossings to/from the UTMI phy_clk domain.

Register map (system bank)
- Word 0: MAIN_CSR (RW; write has side-effect)

- Read-only PHY/link status fields:
- [4:3] LineState[1:0] (latched UTMI LineState)
- [2] usb_attached (debounced attach)
- [1] mode_hs (1=HS, 0=FS)
- [0] suspend
- Write side-effect:
- din[5]=1 issues a one-shot rf_resume_req (remote wake) toward the UTMI/link FSM (clk_i→phy_clk crossed)
- Reset: 0x0000_0000

- Word 5: UTMI VENDOR INTERFACE (R/W; write has side-effect)
- Read:
- [7:0] utmi_vend_stat: vendor status (VStatus_pad_i[7:0]) latched in phy_clk and made visible in clk_i
- Upper bits read as 0
- Write:
- [3:0] utmi_vend_ctrl: driven to VControl_pad_o[3:0]
- Writing asserts a one-shot VControl_Load_pad_o pulse to the PHY (clk_i→phy_clk crossing)
- Other bits ignored on write
- Reset: 0x0000_0000

Clock-domain crossing and latching
- Vendor control write (Word 5):
- CPU write in clk_i sets utmi_vend_ctrl[3:0] and generates a one-shot write pulse.
- Pulse is safely crossed into phy_clk to drive VControl_Load_pad_o; no data is latched without this pulse.
- Vendor status read (Word 5):
- VStatus_pad_i[7:0] is latched each phy_clk into a stable register and read back in clk_i.
- Link state read (Word 0):
- LineState and attach/suspend/HS mode are produced in the UTMI/link logic (phy_clk), synchronized and reflected in MAIN_CSR.

Top-level UTMI pins affected
- Outputs: VControl_pad_o[3:0] (from utmi_vend_ctrl), VControl_Load_pad_o (one-shot on write)
- Inputs: VStatus_pad_i[7:0] (reflected in utmi_vend_stat)

Reset behavior
- MAIN_CSR, UTMI VENDOR IF fields reset to 0.
- Vendor write pulse deasserted; no pending vendor transaction after reset.

Usage notes and sequencing
- To write vendor controls:
1) Write Word 5 with desired utmi_vend_ctrl[3:0].
2) The core generates VControl_Load_pad_o to commit the value at the PHY boundary.
- To read vendor status: read Word 5; value reflects the most recent phy_clk latched VStatus.
- Remote wake (PHY resume K signaling) is requested by writing MAIN_CSR with din[5]=1; request is honored by the UTMI/link FSM per USB timing (requires >5 ms idle before resume).
- MAIN_CSR reflects instantaneous PHY/link state useful for diagnostics and gating vendor operations (e.g., only issue vendor writes when not in RESET/SUSPEND, per PHY requirements).

Caveats
- Reserved/unused bits read as 0 and are ignored on write.
- Vendor interface semantics are PHY-specific; software must follow the attached PHY’s vendor

register protocol using utmi_vend_ctrl/load and utmi_vend_stat.
- All sideband handshakes are level-to-pulse converted; repeated writes may be required by the PHY protocol if an acknowledge mechanism is defined externally.

-- Resume/Wakeup Control --

Resume/Wakeup Control

Overview
- Supports both host-initiated resume and device-initiated remote wakeup.
- Core suspend state is reflected on susp_o; entry/exit edges are latched as RF/system interrupts (suspend_start, suspend_end) in INT SOURCE (Word 3; read-to-clear).

Ways to request device-initiated resume
1) External pin: resume_req_i (top-level input)
- Latched as resume_req_r and held until cleared by suspend_clr from the UTMI/link block.
2) Software write: MAIN_CSR (system bank Word 0)
- Writing din[5]=1 generates rf_resume_req (one-shot) in usbf_rf; safely crossed into the core domain.
- Both sources feed the UTMI/link FSM resume_req input (OR-equivalent), producing a single-shot request; repeated writes while suspended are allowed and coalesce until handled.

UTMI/link FSM behavior (device-initiated resume)
- Preconditions: device is in suspend and bus idle >5 ms (USB remote-wakeup requirement, enforced by timers in usbf_utmi_ls).
- Sequence:
1) RESUME_REQUEST: after wake timing (T2_wakeup ≈ >5 ms), prepare PHY (FS termination on, OpMode=non-driving).
2) RESUME_SIG: drive K on the bus for ≥1.0 ms (drive_k asserted); UTMI TxValid held as needed.
3) RESUME: clear suspend; reconfigure transceiver selection/termination per negotiated speed; then RESUME_WAIT ≥100 µs and return to NORMAL.
- A suspend_clr pulse is issued on resume to clear internal/latching resume requests (clears resume_req_r; rf one-shot auto-clears when seen).

Host-initiated resume
- Detection: K on the bus while suspended (k_long) transitions to RESUME without requiring a device request.
- Outputs: same RESUME and RESUME_WAIT phases; suspend_end interrupt is set (Word 3 [4]).

Registers and status involved
- MAIN_CSR (Word 0):
- Read: [4:3] LineState, [2] attached, [1] mode_hs, [0] suspend.
- Write: din[5]=1 → rf_resume_req one-shot (software remote-wakeup request).
- INT SOURCE (Word 3; read-to-clear RF/system bits):
- [4] suspend_end (exit suspend)
- [3] suspend_start (enter suspend)
- susp_o: mirrors usb_suspend for system power management.

CDC and one-shot handling
- Software resume request (din[5]) is latched in clk_i (wclk) and converted to a one-shot pulse in the core/phy clock domain.
- External resume_req_i is level-latched and automatically cleared by suspend_clr from UTMI/link on successful resume entry.

UTMI/PHY signaling during resume
- drive_k asserted during RESUME_SIG to generate the K state.
- PHY control pins managed by UTMI/link FSM:
- TermSel, XcvSelect, OpMode, SuspendM adjusted per state/speed; SuspendM deasserts on resume.

Interrupt and software flow (recommended)
1) On entering suspend: Word 3[3]=1 (suspend_start); susp_o=1.
2) To remote-wake:
- Ensure remote-wakeup is allowed by policy; write MAIN_CSR with din[5]=1 or assert resume_req_i.
- Wait for suspend_end (Word 3[4]) and/or susp_o deassertion.
3) Service Word 3 to clear latched RF events; proceed with normal operation.

Notes and caveats
- Hardware does not enforce USB DEVICE_REMOTE_WAKEUP feature enable; firmware must gate remote-wakeup requests to meet USB policy.
- All RF/system event bits are cleared by reading INT SOURCE (Word 3); per-endpoint INTs are cleared by reading EPx_INT.
- Timings (≥5 ms idle before remote-wakeup, ≥1 ms K, ≥100 µs settle) are implemented in the UTMI/link FSM timers.

-- Memory Access Window and Alignment --

Memory Access Window and Alignment

Region selection and size
- Address split by a single high address bit HADR (compile-time):
- MEM region selected when wb_addr_i[HADR] = 1 (MEM_SEL)
- RF region selected when wb_addr_i[HADR] = 0 (RF_SEL)
- External buffer memory size (parameterized): bytes = 2^(SSRAM_HADR+3)
- Word count = 2^(SSRAM_HADR+1); data width = 32 bits

Address formation to SRAM (W-side)
- The arbiter drives a word address to the SRAM:
- sram_adr_o <= wb_addr_i[SSRAM_HADR+2:2]
- wb_addr_i[1:0] are ignored (dropped) → 32-bit word addressing
- Read data is broadcast; only the selected master samples on its ack

Alignment and bus capabilities (Wishbone → MEM)
- 32-bit word-aligned accesses only (no byte enables/SEL)
- Software must present 4-byte aligned addresses and full-word writes
- Partial/byte writes are not supported at the MEM window
- Access granularity: one transfer per ack; no burst/pipeline support

Arbitration and timing
- Fixed priority: USB core (M-side) preempts the Wishbone (W-side)
- W-side accesses complete only when M-side is idle
- W ack is a one-clock pulse; if wreq is held and M remains idle, ack toggles every other phy_clk
- Read data valid and write completion are qualified by the W-side ack; hold address/data stable until ack

Address wrap/modulo behavior
- Only wb_addr_i[SSRAM_HADR+2:2] reach the SRAM; higher bits are effectively truncated

- Software must constrain MEM addresses within the physical size (2^(SSRAM_HADR+3) bytes) or accept modulo addressing

Endpoint buffer descriptor alignment (USB side)
- Buffer base addresses used by the DMA are byte-addressed:
- Internally split into word address (adr[SSRAM_HADR+2:2]) + byte-in-word index (adr[1:0])
- RX path handles unaligned starts via pre-read (read-modify-write) of the first word
- TX path unpacks bytes from 32-bit words (prefetch/double-buffer)
- Descriptor fields retain low nibble of the byte address; upper bits store the word address

Usage guidance
- For CPU payload access via MEM window: perform 32-bit aligned reads/writes; avoid partial-word updates
- For endpoint transfers: program BUF descriptors with byte-resolved base/size; the DMA handles sub-word alignment
- Expect variable W-side latency due to M-side preemption; poll on ack rather than assume fixed access time

INTERFACE SPECIFICATIONS

-- Clock and Reset Interfaces --

Clock and Reset Interfaces

Clock domains and sources
- clk_i (Wishbone/CPU domain)
- Drives usbf_wb Wishbone-side data/ack, and usbf_rf CPU-visible registers, masks, and aggregated interrupt outputs (inta_o, intb_o).
- phy_clk_pad_i (UTMI/CORE domain)
- Drives UTMI front-end (usbf_utmi_if), protocol layer (usbf_pl), memory arbiter (usbf_mem_arb), and the usbf_wb control FSM. Most link/protocol logic (packet/TX/RX/DMA) runs here.

Cross-domain handshakes (CDC)
- Wishbone to MEM (clk_i ↔ phy_clk_pad_i):
- usbf_wb samples WB requests into phy_clk, sequences memory/RF strobes, then returns a one-shot wb_ack_o in clk_i.
- Read data is muxed and must be sampled by software on ack; no pipelining.
- Vendor/PHY sideband (clk_i → phy_clk):
- UTMI vendor control writes (Word 5) are level-captured in clk_i and sent as a one-shot pulse (VControl_Load) into phy_clk.
- Remote wake (clk_i → phy_clk):
- MAIN_CSR write with din[5]=1 generates rf_resume_req one-shot into the core domain.
- Status return (phy_clk → clk_i):
- LineState and VStatus are latched on phy_clk and read back in clk_i; suspend state (usb_suspend) is reflected to susp_o for system PM.
- Endpoint/core muxing (inside rf):
- Core-facing matched-EP outputs (csr, buf0/1, flow flags) live in the core (phy_clk) domain; CPU

windows/INTs live in clk_i.

Reset signals and behavior
- rst_i (top-level system reset)
- phy_rst_pad_o is tied directly to rst_i (forwarded to the PHY reset pad).
- Used throughout submodules; default reset style is synchronous unless USBF_ASYNC_RESET is defined.
- Reset styles by block
- usbf_utmi_if/usbf_utmi_ls (phy_clk): synchronous reset by default; optional asynchronous if USBF_ASYNC_RESET is enabled. On reset, TxValid=0; link FSM enters POR (FS selected, FS term on, attached cleared).
- usbf_pl/usbf_idma (phy_clk): synchronous reset; clears DMA counters/state and TX/RX control.
- usbf_mem_arb (phy_clk): rst is active-low; arbitration/ack helper state clears; M has priority post-reset.
- usbf_wb: rst is active-low; default synchronous to phy_clk (optional async with USBF_ASYNC_RESET). Ack pipeline/state cleared; produces no stray acks after reset.
- usbf_rf: default synchronous to clk_i (wclk). On reset: function address=0, RF masks=0, RF event latches cleared, rf_resume_req deasserted, utmi_vend_wr deasserted. Per-EP blocks reset CSR/masks/counters; BUF0/BUF1 init to 0xFFFF_FFFF (unallocated).

Polarity and domain notes
- Active-low resets explicitly documented for usbf_mem_arb and usbf_wb. Other blocks use project-wide reset conventions (synchronous by default; async-assert/sync-deassert with USBF_ASYNC_RESET when enabled).
- Both clock domains must be reset coherently; CDC paths rely on one-shot pulses and level sampling to avoid metastability during normal operation.

Integration guidance
- Provide clk_i and phy_clk_pad_i from independent or related sources; no fixed phase relationship is assumed.
- Drive rst_i to place both domains in a known state; if USBF_ASYNC_RESET is used, ensure proper async-assert/sync-deassert timing to each domain.
- Software should rely on RF readbacks (MAIN_CSR, INT SOURCE) to confirm post-reset link status and clear latched events before enabling traffic.

-- Wishbone Slave Interface --

Wishbone Slave Interface

Port and clocking
- Wishbone signals: clk_i, rst_i, wb_adr_i, wb_dat_i, wb_dat_o, wb_we_i, wb_stb_i, wb_cyc_i, wb_ack_o.
- All Wishbone-visible logic (data and ack) is synchronous to clk_i. A small control FSM runs in phy_clk to sequence accesses; CDC is handled internally.

Address decoding and regions
- One compile-time-selected high address bit (HADR) splits the map:
- RF region (register file): RF_SEL = !wb_adr_i[HADR]
- MEM region (buffer memory window): MEM_SEL = wb_adr_i[HADR]
- HADR values: 17 (test build) or 12 (non-test build).

Data width and alignment
- 32-bit data port; word-aligned accesses only (no byte enables/SEL).

- RF addressing: usbf_rf sees a 7-bit word address adr[8:2] (increment by 1 per 4 bytes).
- MEM addressing: forwarded to memory arbiter as word address (drop wb_adr_i[1:0]); SRAM sees sram_adr <= wb_adr_i[SSRAM_HADR+2:2].

Transaction types and routing
- RF region: wb access generates a single-cycle RF read (rf_re) or write (rf_we) strobe inside phy_clk; wb_ack_o is returned as a one-shot pulse in clk_i.
- MEM region: wb access is converted to a memory request (ma_req) toward usbf_mem_arb W-side; wb_ack_o is issued after ma_ack returns.
- Read data muxed to wb_dat_o from rf_din (RF) or ma_din (MEM); producer must have data valid by the ack.

Handshake and timing (classic Wishbone)
- Slave responds only when CYC_I and STB_I are asserted.
- Exactly one wb_ack_o pulse per completed access; hold address/data stable until ack.
- Internal phy_clk FSM states: IDLE → (W0 for RF or MA_RD/MA_WR for MEM) → W1 → W2 → IDLE. W1/W2 provide fixed dead cycles to avoid retrigger while WB master drops CYC/STB.
- For MEM accesses, wb_ack_o is generated after ma_ack; for RF, wb_ack_o follows the RF strobe path (next FSM step).

Arbitration effects (MEM window)
- usbf_mem_arb gives fixed, immediate priority to the USB core (M-side). The Wishbone W-side completes only when M is idle.
- W-side “ack” inside the arbiter toggles/pulses; sustained WB requests while M is idle may observe an every-other-phy_clk cadence.

Clock-domain crossings (CDC)
- WB request level (stb&cyc;) is sampled into phy_clk; wb_ack_d is brought back into clk_i and edge-detected to form wb_ack_o (one-shot).
- Assumptions: WB master holds address/data stable until ack; data from RF/MEM is stable at ack.

Unsupported/omitted Wishbone features
- No byte selects (SEL_I), no error/retry (ERR_O/RTY_O), no bursts/tags (CTI/BTE/TG*), no STALL/pipelining. Single-beat, classic handshake only.

Reset behavior
- rst_i present at the Wishbone boundary; usbf_wb state/ack pipeline are reset (active-low, synchronous to phy_clk by default; optional USBF_ASYNC_RESET). No spurious acks after reset.

Address map summary (RF region)
- System bank (aliased reads on adr[6:2]==0 or 1; writes use adr[6:2]==0):
- Word 0 MAIN_CSR (status; write din[5] carries resume one-shot)
- Word 1 FUNCTION_ADDRESS
- Word 2 INTERRUPT MASKS (inta_msk/intb_msk)
- Word 3 INTERRUPT SOURCE (read-to-clear system events)
- Word 4 FRM_NAT (frame/microframe snapshot)
- Word 5 UTMI VENDOR IF (read status, write control with one-shot load)
- Endpoint windows EP0..EP15: 4 words each (CSR, INT [read-to-clear], BUF0, BUF1).

Software guidance
- Use 32-bit aligned accesses only; do not attempt byte writes.
- For RF reads that are read-to-clear (e.g., INT SOURCE, EPx_INT), capture the value on the same

cycle as wb_ack_o.
- Expect variable latency on MEM accesses due to USB-side preemption; poll wb_ack_o rather than assume fixed wait.

-- UTMI PHY Data and Control Interface --

UTMI PHY Data and Control Interface

Ports and clock/reset
- Clock/reset: phy_clk_pad_i (UTMI clock domain), phy_rst_pad_o (tied to rst_i)
- Data RX from PHY: DataIn[7:0], RxValid, RxActive, RxError, LineState[1:0]
- Data TX to PHY: DataOut[7:0], TxValid, TxReady
- Link/PHY control to PHY: XcvSelect, TermSel, SuspendM, OpMode[1:0]
- VBUS input: usb_vbus_pad_i
- Vendor sideband: VControl_pad_o[3:0], VControl_Load_pad_o, VStatus_pad_i[7:0]

Receive path (PHY → core)
- All RX indicators captured on phy_clk_pad_i each cycle:
- rx_data <= DataIn
- rx_valid <= RxValid; rx_active <= RxActive; rx_err <= RxError
- Provides a clean, 1-cycle-latency, synchronous handoff into the protocol layer
- LineState[1:0] is observed by the UTMI link-state controller (usbf_utmi_ls); a snapshot is also latched per-phy_clk for CPU readback via MAIN_CSR[4:3]

Transmit path (core → PHY)
- DataOut update:
- If (TxReady == 1) OR (tx_first == 1), drive DataOut <= tx_data
- Else if drive_k == 1, force DataOut <= 8'h00 (filler while K is being driven by control pins)
- TxValid generation:
- Deasserted on reset
- Assert/hold while any of: tx_valid, drive_k, tx_valid_last, or (prior TxValid & !(TxReady | drive_k_r)) is true
- Ensures the PHY sees a continuous transmit context until a byte is accepted (TxReady=1)
- tx_ready back to core is a registered pass-through of TxReady
- tx_first allows preloading the first byte even if TxReady is low, minimizing turnaround

PHY control pins and link management (driven by usbf_utmi_ls)
- XcvSelect: 1 = FS/LS transceiver selected; 0 = HS transceiver selected
- TermSel: 1 = FS termination on; 0 = off
- OpMode[1:0]: 00 = normal drive; 10 = non-driving (tri-state); others unused here
- SuspendM: asserted during suspend unless a resume request is pending, or whenever instantaneous LineState indicates K per implementation
- Implemented as: SuspendM = (usb_suspend & !resume_req_s) | (LineState == 2'b10)

State-machine–visible status (core outputs from UTMI/link)
- mode_hs: 1 = HS mode, 0 = FS
- usb_reset: asserted during RESET handling
- usb_suspend: core suspend state (mirrored to susp_o for system PM)
- usb_attached: debounced after attach settling
- suspend_clr: pulse to clear suspend and latched remote-wakeup requests at resume entry

Reset, suspend/resume, and speed negotiation (summary)
- USB reset detection: SE0 qualified (>2.5 µs) triggers RESET; FS interface forced during reset

(XcvSelect=FS, TermSel on, OpMode non-driving)
- HS negotiation (after RESET): device chirp K/J detection; commit to HS after 3 K/J pairs (6 edges) else remain FS; set mode_hs accordingly
- Suspend entry: idle >3.0 ms (FS idle=J; HS idle=SE0) → SUSPEND (SuspendM asserted)
- Host-initiated resume: K observed while suspended (k_long) → RESUME
- Device remote-wakeup: after idle >5 ms (T2_wakeup) and resume_req_s asserted → RESUME_REQUEST → drive K ≥1.0 ms (RESUME_SIG) → RESUME → RESUME_WAIT ≥100 µs → NORMAL
- Timers (derived internally): thresholds at ~2.5 µs, 100 µs, 1.0/1.2 ms, 5 ms, 100 ms for attach debounce

VBUS/attach handling
- usb_vbus_pad_i used as an additional POR source for the link FSM
- ATTACH state debounces for ~100 ms before asserting usb_attached and entering NORMAL

Vendor sideband interface (software visible)
- Write RF Word 5: utmi_vend_ctrl[3:0] → VControl_pad_o[3:0]; generates one-shot VControl_Load_pad_o into phy_clk
- Read RF Word 5: utmi_vend_stat[7:0] reflects latched VStatus_pad_i[7:0]

Software-visible snapshots (RF/system)
- MAIN_CSR (Word 0 read): [4:3] LineState, [2] usb_attached, [1] mode_hs, [0] suspend
- Remote wake request: write MAIN_CSR with din[5]=1 → rf_resume_req (one-shot into core/phy domain)

Clocking and reset notes
- All UTMI interface logic (data paths and link/PHY control) is synchronous to phy_clk_pad_i
- phy_rst_pad_o is tied to rst_i (board/PHY reset)
- Optional asynchronous reset support via USBF_ASYNC_RESET; otherwise synchronous resets

Electrical/handshake notes
- Data path is byte-wide, LSB-first on the wire (deserialized by the PHY)
- During drive_k, DataOut is don't-care (0x00 driven); K signaling is produced by XcvSelect/TermSel/OpMode while TxValid remains asserted
- TxValid hold ensures UTMI transmit context continuity across ready edges and during resume/chirp generation

-- External SSRAM Interface --

External SSRAM Interface

Physical interface (top-level ports)
- Address: sram_adr_o[SSRAM_HADR:0] (word address)
- Data to SSRAM: sram_data_o[31:0]
- Data from SSRAM: sram_data_i[31:0]
- Control: sram_we_o (write enable), sram_re_o (read enable)
- Clock domain: phy_clk_pad_i (all SSRAM-side logic synchronous to this clock)

Width, addressing, and size
- Data width: 32 bits; accesses are 32-bit words (no byte enables)
- Word-aligned addressing: lower two byte address bits are dropped (wb_addr_i[1:0] ignored)
- Effective word address to SSRAM: sram_adr_o <= {byte_addr[SSRAM_HADR+2:2]}
- Capacity (parameterized):

- Word address width = SSRAM_HADR+1 bits → 2^(SSRAM_HADR+1) words
- Total bytes = 2^(SSRAM_HADR+3)
- Address wrap: software and DMA must stay within physical size; higher address bits beyond SSRAM range are truncated (modulo wrap)

Arbitration and masters (usbf_mem_arb)
- Two masters share a single-port SSRAM:
- M side (USB core/protocol engine): highest priority, time-critical
- W side (Wishbone/CPU via usbf_wb): lower priority, served only when M is idle
- Fixed-priority muxing of address/data/we; read data is broadcast to both masters; only the selected side samples on its ack
- M-side ack: mack = mreq (combinational) → zero-wait from arbiter perspective
- W-side ack: pulsed/toggling (wack) only when M is idle; with wreq held high, opportunities occur every other phy_clk

Control signaling to SSRAM
- sram_we_o asserted when the selected master is performing a write (M: mreq&mwe; W: wreq&wwe;)
- sram_re_o is held high (continuous read enable) to model a simple synchronous SRAM; masters must qualify data capture with their ack

Wishbone access to SSRAM (MEM window)
- Address decode bit HADR selects MEM region (wb_addr_i[HADR]=1)
- usbf_wb forwards requests to mem_arb W side; wb_ack_o is issued after W-side ack returns from mem_arb
- Software must use 32-bit aligned accesses; partial/byte writes are not supported

USB core DMA behavior (M side)
- RX (USB→memory): read-modify-write used for unaligned starts; partial last words written correctly
- TX (memory→USB): double-buffered 32-bit prefetch, byte-unpack to the TX stream
- Optional ring/wrap addressing controlled by descriptors (dma_en)

Clocking and CDC notes
- Entire SSRAM interface (arbiter and ports) runs in phy_clk_pad_i domain
- Wishbone requests/acks are safely bridged (clk_i ↔ phy_clk) inside usbf_wb; read data must be valid at the time wb_ack_o is observed on clk_i

Integration guidance
- Attach a synchronous, single-cycle style SSRAM or equivalent fabric memory to sram_* ports; ensure data is stable for capture on the masters' ack cycles
- Observe word alignment and physical size constraints; plan for M-side preemption of CPU accesses
- Expect variable latency on CPU (W-side) accesses due to USB-preemptive priority

-- DMA Handshake Interface --

DMA Handshake Interface

Top-level ports and mapping
- dma_req_o[15:0] (output, clk_i domain): One per endpoint (bit 0=EP0 … bit 15=EP15). Asserted level when the endpoint has DMA work pending.
- dma_ack_i[15:0] (input, clk_i domain): One-clock pulse per completed 32-bit word serviced by the external DMA for the corresponding endpoint.

Clocking and reset
- Domain: clk_i (a.k.a. wclk in usbf_rf). Requests and acks are generated/consumed in this CPU/DMA/bus clock domain.
- Reset: All DMA request state deasserts on reset; spurious acks are ignored until a request is active.

When requests assert (per endpoint)
- DMA must be enabled in EPx_CSR[15]; otherwise no requests are generated.
- OUT endpoints (host→device, buffer filled by USB, system DMA drains to system memory):
- dma_req_o[x] asserts while there are words remaining to transfer out of the SSRAM buffer (internal counter dma_out_cnt != 0).
- IN endpoints (device→host, system DMA fills SSRAM for USB to send):
- dma_req_o[x] asserts while the number of words produced by DMA (dma_in_cnt) is less than the programmed buffer word budget (buf0_orig[30:19]).

Hold/deassert policy (chatter suppression)
- Requests are level-held to encourage burst service:
- OUT: held while at least 4 words remain (|dma_out_cnt[11:2]).
- IN: held for “large” transfers and while ≥3 words remain (|buf0_orig[30:21] and (dma_in_cnt < buf0_orig[30:19]-3)).
- Requests deassert after an ack if hold conditions are not met.

Ack semantics (what the external DMA must do)
- Drive a single clk_i-cycle pulse on dma_ack_i[x] for each 32-bit word actually moved by the external DMA for EPx.
- Only acknowledge when dma_req_o[x] is asserted; one ack advances internal accounting by one word:
- OUT: dma_out_cnt decrements by 1.
- IN: dma_in_cnt increments by 1.
- Continue issuing acks until the request deasserts (work completed or hold window closed).

Relationship to buffer descriptors
- EPx_BUF0/BUF1 hold the SSRAM base address and size programmed by software; rf snapshots BUF0 (buf0_orig) for DMA word budgeting.
- The DMA handshake counts words; software/external DMA must perform the actual data moves via the MEM window (Wishbone → usbf_mem_arb → SSRAM) in sync with ack pulses.

Direction and usage guidance
- OUT endpoints: External DMA typically copies received payload from SSRAM to system memory; keep acking per word until request drops.
- IN endpoints: External DMA typically fills SSRAM from system memory; ack per word written until the programmed size is reached and request drops.

CDC and safety
- dma_req_o/ack_i are synchronous to clk_i; no additional CDC is required at the top level.
- Internally, acks are safely crossed back to the core clock for counter updates.

Operational notes
- EP0 (control) usually operates without external DMA; set CSR[15]=0 to suppress requests.
- The request is level; the ack must be a pulse (one word per pulse). Do not assert ack when req=0.
- Expect bursts: due to hold policy, requests may remain asserted across multiple acks for better throughput.

-- Interrupt Outputs --

Interrupt Outputs

Signals and domain
- inta_o, intb_o (outputs, active-high, clk_i domain): aggregated interrupt lines to the host system.
- Synchronous to clk_i (wclk in RF). Reset clears masks and latched system events; endpoints retain their own reset behavior.

Aggregation model
- inta_o = OR(per-EP inta) | OR(int_srcb & inta_msk)
- intb_o = OR(per-EP intb) | OR(int_srcb & intb_msk)
- per-EP inta/intb are level signals generated inside each endpoint block (usbf_ep_rf) per its local masks/status.
- int_srcb are RF/system event latches (see below). inta_msk/intb_msk are top-level 9-bit masks.

RF/system event latches (Word 3, read-to-clear)
- int_srcb[8:0] (clk_i domain), set by core status edges/errors; cleared on read of RF Word 3:
- [8] usb_reset
- [7] rx_err
- [6] detach (falling edge of usb_attached)
- [5] attach (rising edge of usb_attached)
- [4] suspend_end (exit suspend)
- [3] suspend_start (enter suspend)
- [2] nse_err (non-SE0 error)
- [1] pid_cs_err (PID complement mismatch)
- [0] crc5_err (token CRC5 error)
- Mask registers (Word 2):
- inta_msk[8:0] gates int_srcb into inta_o
- intb_msk[8:0] gates int_srcb into intb_o

Per-endpoint interrupts
- Each enabled endpoint exposes two level interrupt outputs (inta/intb) in clk_i domain, reflecting its EPx_INT read-to-clear status and EP-local masks.
- These per-EP lines are OR-reduced across all endpoints and directly contribute to inta_o/intb_o as above.
- Software services per-EP causes by reading EPx_INT (read-to-clear) within that endpoint’s window.

Software-visible mirrors
- int_srca[15:0] (RF Word 3, read-only): mirror of any-pending per-EP (OR of epX_inta|epX_intb) for EP0..EP15; software visibility only.

Programming and service flow
- Enable/route system events via RF Word 2 masks (inta_msk/intb_msk).
- Service per-EP sources by reading EPx_INT (clears latched EP bits).
- Service RF/system events by reading RF Word 3 (clears int_srcb).

Reset behavior
- On reset: inta_msk/intb_msk = 0; int_srcb cleared; per-EP INT state/reset per endpoint logic. No spurious pulses are generated at reset deassert.

Timing/CDC notes

- All interrupt aggregation and latches are synchronous to clk_i. Core-domain status/edges (attach/detach/suspend/reset/errors) are safely synchronized into clk_i before latching.

Notes
- Interrupt polarity is level-high at the outputs; internal RF/system events are latched (read-to-clear), while per-EP contributions are level via EP-local latches/masks.

-- Power and Status Signals --

Power and Status Signals

External power/status pins
- susp_o (output, clk_i domain): Mirrors the core's usb_suspend state. Asserted when the link is in suspend; deasserted on resume (host- or device-initiated).
- resume_req_i (input, clk_i domain): System request to initiate remote wakeup. Latched into resume_req_r and held until a suspend_clr pulse is returned from the UTMI link.
- phy_rst_pad_o (output): Tied directly to rst_i (board/PHY reset).
- usb_vbus_pad_i (input): VBUS presence indication to the UTMI/link FSM; used as an additional POR/attach qualifier.

Core/link status (internal to top, software-visible via RF)
- mode_hs (phy_clk domain): 1=High-Speed, 0=Full-Speed; reported to software in MAIN_CSR[1].
- usb_reset (phy_clk domain): Asserted during USB bus reset; also latched as an RF/system event (int_srcb[8]).
- usb_suspend (phy_clk domain): Link suspend state; reflected to susp_o and MAIN_CSR[0].
- usb_attached (phy_clk domain): Debounced attach status after ~100 ms; reflected in MAIN_CSR[2] and latched as attach/detach RF events (int_srcb[5]/[6]).
- LineState[1:0] (UTMI): Latched each phy clock into LineState_r; software snapshot available at MAIN_CSR[4:3].
- VStatus_pad_i[7:0] (UTMI vendor): Latched each phy clock into VStatus_r; readable at RF Word 5 (UTMI VENDOR INTERFACE).

UTMI power-management/control (driven by link FSM)
- SuspendM, XcvSelect, TermSel, OpMode[1:0] (outputs to PHY): Driven per UTMI/link state to manage suspend entry/exit, HS/FS selection, and termination. SuspendM is asserted during suspend unless a resume request is pending, and while K is detected.

Software control and visibility
- MAIN_CSR (RF Word 0, read):
- [4:3] LineState snapshot
- [2] usb_attached
- [1] mode_hs
- [0] suspend
- MAIN_CSR (RF Word 0, write):
- din[5]=1 issues a CPU-initiated remote wake request (rf_resume_req one-shot) into the core.
- RF/system events (RF Word 3, read-to-clear):
- suspend_start [3], suspend_end [4], attach [5], detach [6], usb_reset [8], plus error flags.
- UTMI vendor access (RF Word 5):
- Read VStatus (status snapshot); write VControl[3:0] with one-shot VControl_Load to the PHY domain.

Remote wakeup sequencing
- External request path: resume_req_i is latched (resume_req_r) while suspended; UTMI/link FSM

enforces idle >5 ms before RESUME_REQUEST and drives K ≥1 ms, then issues suspend_clr to release the request.
- Software path: Writing MAIN_CSR with din[5]=1 generates rf_resume_req (one-shot) into the core; cleared on suspend_clr.

Reset and attach behavior
- Global reset (rst_i) drives phy_rst_pad_o and clears RF masks/latches; usb_attached is re-established after attach debounce (~100 ms) when VBUS is present.

Clocking notes
- Link power/status (mode_hs, usb_reset, usb_suspend, usb_attached) and UTMI control pins are in phy_clk domain; susp_o and CPU-visible snapshots/masks are synchronous to clk_i with safe CDC.

-- Timing and Handshake Requirements --

Timing and Handshake Requirements

Clocking and CDC
- Two primary clock domains:
- phy_clk_pad_i: UTMI/link/protocol/memory arbiter. All UTMI Tx/Rx, link FSM timing, and SRAM arbitration occur here.
- clk_i: CPU/Wishbone/RF and external DMA handshake domain.
- Crossings are done internally; integrators must keep external interfaces synchronous to their stated domains:
- Wishbone, dma_req_o/dma_ack_i, susp_o, resume_req_i synchronous to clk_i.
- UTMI pins and SRAM interface synchronous to phy_clk_pad_i.
- Hold external requests/data stable until the corresponding ack/accept pulse is observed in the same clock domain.

UTMI interface (PHY-side) timing
- Receive (PHY→core): rx_data, rx_valid, rx_active, rx_err are sampled and registered; the core observes a 1-cycle latency. UTMI must present stable DataIn when RxValid & RxActive=1.
- Transmit (core→PHY):
- DataOut updates when (TxReady=1) or on tx_first=1 (preload). During drive_k, DataOut is a don’t-care (0x00 driven), while TxValid remains asserted.
- TxValid assertion/hold: core asserts TxValid when data/drive_k/last-byte is pending and holds it high until PHY asserts TxReady=1. PHY must not sample a byte except when TxValid=1 & TxReady=1.
- Link timing thresholds (internal timers, UTMI driven):
- Reset detect: SE0 stable >2.5 µs.
- Suspend entry: bus idle >3.0 ms (FS idle=J, HS idle=SE0).
- Device remote-wakeup: bus idle >5 ms before signaling; K must be driven ≥1.0 ms.
- Resume settle: ≥100 µs before returning to NORMAL.
- HS chirp: device K chirp duration >1.2 ms; host K/J pairs counted (6 edges total) before HS commit.
- LineState qualification: J/K/SE0 “long” qualifiers require ≥2 phy_clk samples stable.

Protocol-layer handshakes and timeouts
- Token filtering requires address match and CRC5 clean before action; malformed/host-only PIDs are ignored.
- Data integrity: CRC16 checked on RX; TX appends CRC16. Errors suppress success handshakes.
- Endpoint engine timeouts (internal counters):
- IN transaction ACK wait (state IN2): speed-dependent thresholds (HS=22, FS=36 internal counts) before timeout event.
- OUT data wait (state OUT): similar speed-dependent thresholds while rx_active=0; timeout raises

int_to_set.
- Handshake selection rules:
- STALL when endpoint stalled; NACK on no buffers/DMA or size-policy violation; ACK on success; NYET (HS-only) when success but no further buffers; ACK also sent for pid_seq_err on OUT (host received data but toggle wrong).

External DMA handshake (clk_i domain)
- dma_req_o[x]: level request per endpoint; asserts when work is pending (IN: produced words < budget; OUT: remaining words > 0). Held to encourage bursts per internal hold policy (e.g., OUT while ≥4 words remain; IN while large transfers have ≥3 words left).
- dma_ack_i[x]: single clk_i-cycle pulse per 32-bit word serviced; only assert when dma_req_o[x]=1. Each pulse advances internal word counters by one.
- Acks must not be asserted when req=0; pulse width = 1 clk_i; consecutive pulses allowed for burst service.

SRAM arbiter and memory access (phy_clk domain)
- M side (protocol) priority: mack = mreq (combinational). M write only when mreq & mwe=1.
- W side (Wishbone path): wack is a one-clock pulse when M is idle; if wreq is held, arbiter provides a transfer opportunity every other phy_clk. Deassert wreq after wack for single transfers; expect preemption if M asserts mreq.
- sram_re_o is held high (always-read model); masters must capture mdout/wdout only on their ack.

Wishbone slave handshake (clk_i domain, bridged via phy_clk FSM)
- Classic single-beat timing:
- Master must hold CYC_I & STB_I, address, and write data stable until wb_ack_o pulses (one clk_i cycle).
- Reads: data valid when wb_ack_o=1.
- Writes: complete on wb_ack_o=1.
- The internal FSM inserts two phy_clk “dead” cycles after ack to avoid immediate retrigger; masters should drop STB_I/CYC_I promptly after ack.

Software-visible read-to-clear and one-shot semantics
- RF/system events (Word 3) are read-to-clear; service latency should ensure no event loss (latches hold until read).
- EPx_INT are read-to-clear per endpoint.
- MAIN_CSR write with din[5]=1 generates a one-shot rf_resume_req; it is cleared by suspend_clr on resume.

Reset, attach, and VBUS behavior
- rst_i drives phy_rst_pad_o; RF masks/latches are cleared on reset.
- usb_attached is asserted after ~100 ms debounce once VBUS is present; attach/detach edges are latched as RF events.

General integration requirements
- Keep all external synchronous interfaces aligned to their domains (UTMI↔phy_clk_pad_i, DMA/Wishbone↔clk_i).
- Do not glitch TxReady/RxValid/RxActive at UTMI; obey valid/ready semantics strictly.
- For maximum throughput: service dma_req_o bursts with consecutive dma_ack_i pulses; for Wishbone bulk MEM transfers, deassert wreq after each wack to achieve one transfer per two phy_clk when M is idle.

-- Signal Polarity and Conventions --

Signal Polarity and Conventions

Resets and clocks
- rst_i: active-low reset at the Wishbone/top boundary; drives phy_rst_pad_o (active-low PHY reset).
- Internal core resets: synchronous by default; some core/link blocks use an active-high rst internally (optional async via USBF_ASYNC_RESET).
- Clock domains: phy_clk_pad_i (UTMI/link/mem_arb), clk_i (Wishbone/RF/DMA handshakes).

UTMI interface conventions
- RX inputs (RxValid, RxActive, RxError): active-high; DataIn[7:0] sampled when RxValid & RxActive=1 (LSB-first on USB).
- TX handshake: TxValid and TxReady active-high; byte transfers occur on TxValid & TxReady.
- TX data preload: tx_first=1 allows DataOut preload even if TxReady=0.
- drive_k: active-high request to keep TxValid asserted while PHY drives K; DataOut forced to 0x00 during drive_k (don't-care).
- LineState[1:0]: 00=SE0, 01=J, 10=K, 11=SE1 (sampled active-high encoding).
- PHY control pins: XcvSelect=1 selects FS transceiver (0=HS); TermSel=1 enables FS termination; OpMode=00 normal, 10 non-driving; SuspendM asserted high during suspend or while K is detected.
- PID format: 8-bit PIDs are {~PID4, PID4} (upper nibble is bitwise complement of lower nibble), LSB-first on the wire.
- CRC conventions: USB CRCs are LSB-first; CRC5/CRC16 logic operates on LSB-first data; TX emits CRC16 low byte first.

Interrupts and status
- inta_o, intb_o: active-high, level interrupts (clk_i domain) = OR(per-EP level) | masked RF/system events.
- RF/system event latches (int_srcb): active-high bits; read-to-clear on RF Word 3.
- Per-EP INT registers: read-to-clear; per-EP level outputs (ep_inta/ep_intb) are active-high.

DMA and memory handshakes
- dma_req_o[15:0]: active-high, level request per endpoint (clk_i domain).
- dma_ack_i[15:0]: active-high, single-cycle pulse per 32-bit word serviced; only valid when corresponding dma_req_o bit is 1.
- sram_we_o: active-high write enable to SSRAM; asserted only for the selected master.
- sram_re_o: held high (active-high read enable) continuously; masters must capture read data only on their ack.
- mem_arb M-side: mack = mreq (active-high, combinational immediate ack to protocol master).
- mem_arb W-side: wack is an active-high, one-clock pulse; if wreq held high, wack toggles every other phy_clk while M is idle.

Wishbone bus conventions (clk_i domain)
- wb_cyc_i, wb_stb_i, wb_we_i: active-high.
- wb_ack_o: active-high, one-clock pulse per completed transfer (read data valid when wb_ack_o=1).
- Addressing: 32-bit word-aligned (ADR[n:2] used internally); byte lanes/sel not supported.

Register access conventions
- MAIN_CSR (Word 0) read bits active-high: [4:3] LineState, [2] usb_attached, [1] mode_hs, [0] suspend.
- MAIN_CSR write one-shot: din[5]=1 generates an active-high rf_resume_req pulse (auto-cleared by suspend_clr).
- UTMI vendor: VControl_Load_pad_o is an active-high one-shot; VControl_pad_o[3:0] driven as written; VStatus_pad_i[7:0] latched active-high.

Data/addressing conventions
- External SSRAM addressed as 32-bit words: sram_adr_o = byte_addr[SSRAM_HADR+2:2].
- USB data bit ordering is LSB-first; frame/microframe counters and snapshots are reported as active-high fields in RF space.

General levels vs pulses
- Level signals: dma_req_o, inta_o/intb_o, TxValid, TxReady, RxValid/RxActive/RxError, sram_re_o/sram_we_o (when asserted), wreq/mreq.
- Pulse/one-shot signals: dma_ack_i, wb_ack_o, mem_arb wack, VControl_Load_pad_o, rf_resume_req (generated), token_valid, rx_data_done, tx_first, tx_valid_last (beat marker).

PARAMETERIZATION AND CONFIGURATION

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

Generic Parameters (Address Widths, Buffer Sizes)

Top-level address decode and regions
- USBF_UFC_HADR (compile-time): selects the single high address bit that splits RF vs MEM regions on Wishbone.
- Test build: USBF_UFC_HADR=17 → RF_SEL = !wb_addr_i[17], MEM_SEL = wb_addr_i[17]
- Non-test: USBF_UFC_HADR=12 → RF_SEL = !wb_addr_i[12], MEM_SEL = wb_addr_i[12]
- RF region addressing into usbf_rf: 7-bit word address adr[8:2] (32-bit words; increment by 1 per 4 bytes).

External buffer memory sizing (SSRAM)
- SSRAM_HADR (parameter): external buffer memory address width (word addressing).
- sram_adr_o[SSRAM_HADR:0] → (SSRAM_HADR+1) word-address bits
- Total words = 2^(SSRAM_HADR+1)
- Total bytes = 2^(SSRAM_HADR+3)
- Example (test build): SSRAM_HADR=14 → 2^(14+3)=131072 bytes (128 KiB)
- W-side (Wishbone→MEM) address presented to arbiter as ma_adr[SSRAM_HADR+2:2] (byte address stripped to word index).

Endpoint instantiation and vectors
- Endpoints enabled via USBF_HAVE_EPn macros; enabled EP numbers must be contiguous from EP0 upward (checked at init).
- Per-EP vectors sized for up to 16 endpoints:
- dma_req_o[15:0], dma_ack_i[15:0]
- int_srca[15:0] (software mirror of any-pending per-EP INT)

Register file and masks
- RF/system event mask widths: inta_msk[8:0], intb_msk[8:0]
- RF/system event latch width: int_srcb[8:0]
- UTMI vendor interface widths: VControl_pad_o[3:0], VStatus_pad_i[7:0]
- Frame/microframe snapshot: frm_nat[31:0] (composite fields as generated by protocol layer)

Per-endpoint window map (each EP: 4 words)
- Base word address for EPn = 4 + 4*n
- +0: EPx_CSR (32-bit)
- +1: EPx_INT (32-bit, read-to-clear)
- +2: EPx_BUF0 (32-bit)
- +3: EPx_BUF1 (32-bit)

Key field and counter widths (protocol/RF)
- EPx_CSR
- Max packet size: csr[10:0] (11 bits, bytes)
- Direction: csr[27:26] (2 bits)
- Endpoint number (for match): csr[21:18] (4 bits)
- DMA enable: csr[15] (1 bit)
- Endpoint buffer descriptors (BUF0/BUF1)
- Programmed size: bits [30:17] (14 bits, bytes; max 16 KiB)
- Address field captured from idin[SSRAM_HADR+2:4] (word address bits; width = (SSRAM_HADR+2)−4+1 = SSRAM_HADR−1)
- Low nibble idin[3:0] used for UC fields/status per core conventions
- DMA accounting (clk domain)
- Word budget source: buf0_orig[30:19] (12 bits, words; max 4096 words = 16 KiB)
- dma_out_cnt[11:0], dma_in_cnt[11:0] (12-bit word counters)
- Internal IDMA sizing
- TX size counter: 14 bits (up to 16 KiB)
- RX byte counter sizu_c: 11 bits (up to 2 KiB)

Bus/data widths
- Wishbone data: 32-bit; addresses are byte-based but RF uses word addressing (adr[8:2]).
- External SSRAM data: 32-bit
- UTMI data: 8-bit (DataIn/DataOut[7:0])

Clocking domains (for parameter context)
- phy_clk_pad_i: UTMI/link/protocol/memory arbiter domain (SRAM addressing and arbitration sized by SSRAM_HADR)
- clk_i: Wishbone/RF/DMA handshake domain (RF address adr[8:2], masks, DMA req/ack vectors)

Notes and derived limits
- Maximum single transfer size per buffer descriptor is bounded by the 14-bit BUF size field (≤16 KiB).
- Packet-level sizing (csr[10:0]) permits max packet sizes up to 2047 bytes.
- Total buffer space scales solely with SSRAM_HADR; no other top-level parameter alters memory depth.

-- Build-Time Macros and Options --

Build-Time Macros and Options

Address map and decode
- USBF_UFC_HADR (macro): selects the high address bit that splits RF vs MEM regions on Wishbone.
- Test build: 17 → RF_SEL = !wb_addr_i[17], MEM_SEL = wb_addr_i[17]
- Non-test: 12 → RF_SEL = !wb_addr_i[12], MEM_SEL = wb_addr_i[12]
- Effect: Sets the RF/MEM region boundary and influences usbf_wb/rf addressing (RF uses word adr[8:2]).

Endpoint presence
- USBF_HAVE_EPn (macros, n=1..15): per-endpoint enable switches. Disabled endpoints are replaced with usbf_ep_rf_dummy.
- Constraint: Enabled endpoints must be contiguous from EP0 upward (checked at sim start; configuration is printed).
- Effect: Alters which EP blocks are synthesized; per-EP vectors remain 16-wide but unused bits are tied off.

Reset style
- USBF_ASYNC_RESET (macro): select reset implementation style across modules (utmi_if, rf, idma, wb, etc.).
- Defined: async-assert, sync-deassert reset in respective clock domains (phy_clk or wclk/clk_i as applicable).
- Undefined (default): fully synchronous resets in each domain.

Timing and protocol constants
- PID encodings (e.g., T_PID_*): compile-time constants for USB PIDs used by TX/RX logic; do not change protocol behavior beyond encoding.
- Timing constants (examples used by utmi_if/usbf_utmi_ls/usbf_pl): thresholds for reset (>2.5 µs), suspend (>3 ms), resume (≥1 ms K, ≥100 µs settle), HS chirp (>1.2 ms), microframe ticks, and policy timeouts (e.g., IN ACK HS=22, FS=36). These are provided as macros in the source and may be overridden only if maintaining USB 2.0 timing compliance.

Memory/buffer sizing (parameterized, typically set at synthesis)
- SSRAM_HADR (Verilog parameter): external SSRAM word-address width.
- Total words = 2^(SSRAM_HADR+1); total bytes = 2^(SSRAM_HADR+3).
- Example (test): SSRAM_HADR=14 → 128 KiB buffer space.
- Effect: Sets sram_adr_o width, Wishbone→MEM address slicing (ma_adr[SSRAM_HADR+2:2]), and buffer descriptor address field width.

Simulation-time reporting/checks
- On elaboration, the design prints: enabled endpoints, RF/MEM address split, computed SSRAM size; warns if EP numbering is not contiguous.

Integration notes and implications
- Changing USBF_UFC_HADR alters the RF/MEM map; software drivers must align their address decoding to the chosen bit.
- USBF_HAVE_EPn must form a dense set from EP0; holes are not supported.
- USBF_ASYNC_RESET selection must match system reset strategy for both clk_i and phy_clk domains.
- Overriding timing macros should only be done with care to preserve USB 2.0 timing compliance.

-- Endpoint Presence and Configuration --

Endpoint Presence and Configuration

Presence and instantiation
- EP0 is always present (control pipe).
- EP1..EP15 are enabled via USBF_HAVE_EPn build-time macros; disabled EPs are instantiated as usbf_ep_rf_dummy (no DMA/INT/match).
- Constraint: Enabled endpoint indices must be contiguous from EP0 upward; a simulation-time check

prints configuration and warns on gaps.
- Maximum endpoints exposed in top-level vectors: 16 (EP0..EP15). Per-EP vectors (dma_req_o/dma_ack_i/INT mirrors) are always 16 wide; unused bits are tied off when endpoints are disabled.

Per-endpoint register window (RF region)
- Addressing: each EP occupies 4 consecutive 32-bit words. EPn base word address = 4 + 4*n.
- +0: EPx_CSR (Control/Status)
- +1: EPx_INT (latched events, read-to-clear; local masks inside EP logic gate level outputs)
- +2: EPx_BUF0 (buffer 0 descriptor)
- +3: EPx_BUF1 (buffer 1 descriptor)

EPx_CSR — key configuration fields
- Direction: csr[27:26] = 00 control (EP0 context), 01 IN, 10 OUT.
- Transfer type: csr[25:24] (e.g., 01 isochronous, 10 bulk; others per implementation).
- Endpoint number (match compare): csr[21:18] (4 bits).
- Max packet size (bytes): csr[10:0] (11 bits; up to 2047 B).
- DMA enable: csr[15] (1=enables DMA path for the EP; control EP treated specially).
- Stall/status control: csr[23:22] (e.g., 10=stall, 01=disabled).
- OTS (On-The-Spot) policy: ots_stop (csr[13]); forces csr1[8:7]=01 on out_to_small.
- User-control fields: uc_bsel[1:0], uc_dpd[1:0] (buffer select/data PID control; updated by core strobes).

Buffer descriptors (EPx_BUFx)
- Size field: bits [30:17] (14 bits, bytes) — maximum single programmed transfer ≤ 16 KiB.
- Address field: word address captured from idin[SSRAM_HADR+2:4]; width depends on SSRAM_HADR (external buffer memory depth).
- Low nibble idin[3:0] carries UC fields/status (next_bsel/next_dpid) as updated by core.
- buf0_orig retains last CPU-programmed BUF0 for DMA accounting.

Endpoint interrupting and events (EPx_INT)
- Read-to-clear latched events including (non-exhaustive): BUF0_DONE, BUF1_DONE, unexpected PID (upid), CRC16 error, timeout, sequence error, OUT too small.
- Local masks per EP gate level outputs ep_inta/ep_intb (aggregated at top as inta_o/intb_o).

Selection/match and broadcast update scheme
- The protocol layer broadcasts update buses (idin + strobes) to all EPs; only the matching EP latches updates (ep_match = (ep_sel == csr[21:18])).
- Core-facing “matched EP” mux provides the selected EP’s CSR/BUF view and flow flags back to the protocol engine; priority order on simultaneous matches: EP0..EP15.

DMA handshakes and flow flags per EP
- dma_req_o[x] (clk_i domain): asserted when EP has work pending (policy- and counter-driven); held per internal hold policy to encourage bursts.
- dma_ack_i[x] (clk_i domain): single-cycle pulse per 32-bit word serviced; advances EP-local word counters.
- Flow helpers back to protocol layer:
- dma_in_buf_sz1: at least one packet’s worth produced for IN.
- dma_out_buf_avail: space available for another OUT packet.

Policy engine capabilities per EP (summary)
- Control EP sequencing: SETUP → DATA_IN/DATA_OUT → STATUS_IN/STATUS_OUT with timeouts/error branches.

- Bulk/iso/int EPs: autonomous OUT acceptance/IN scheduling, handshake selection (ACK/NAK/STALL; NYET for HS), data PID management (toggle/MDATA/DATA2), size policy (sml_ok/lrg_ok), CRC enforcement, and timeouts (HS/FS aware).

Limits and defaults
- Endpoint count: up to 16; EP0 mandatory.
- Max packet size field width: 11 bits; buffer size field: 14 bits.
- After reset: EPs default to disabled/idle; buffers initialize to 0xFFFF_FFFF; masks cleared (exact defaults per usbf_ep_rf).

Integration notes
- Software config flow per EP: program EPx_CSR → program EPx_BUFx → stage payload in MEM (IN) or allocate space (OUT) → monitor EPx_INT/inta_o/intb_o → service and clear by reading EPx_INT and/or RF Word 3 → update descriptors for next transfers.
- Endpoint presence is fixed at build time via USBF_HAVE_EPn; software must not assume non-contiguous EP numbering.

-- Address Split Configuration --

Address Split Configuration

Region-select bit (build-time)
- USBF_UFC_HADR: single high address bit that partitions the Wishbone address space into:
- RF (Register-File) region when wb_addr_i[HADR] == 0
- MEM (Buffer Memory) region when wb_addr_i[HADR] == 1
- Reference configurations:
- Test build: HADR=17 → RF_SEL = !wb_addr_i[17], MEM_SEL = wb_addr_i[17]
- Non-test: HADR=12 → RF_SEL = !wb_addr_i[12], MEM_SEL = wb_addr_i[12]
- Software impact: Changing HADR shifts the RF/MEM boundary; firmware must decode the same bit when mapping RF vs MEM addresses.

Behavior per region
- RF region (wb_addr_i[HADR]=0): Accesses target usbf_rf (register-file space)
- usbf_wb converts WB byte address to a 7-bit word address adr[8:2] (32-bit words)
- Map within RF:
- System bank (aliased on adr[6:2]==0 or 1): MAIN_CSR, FUNCTION_ADDRESS, INT MASKS, INT SOURCE, FRM_NAT, UTMI VENDOR
- Endpoint windows EP0..EP15: each 4 words; EPn base = 4 + 4*n (CSR, INT, BUF0, BUF1)
- Handshake: single-cycle ack generated after the RF strobe (subject to clk crossings); data returned on wb_ack_o

- MEM region (wb_addr_i[HADR]=1): Accesses target external SSRAM via usbf_mem_arb (W-side)
- usbf_wb forwards request to arbiter W-side; WB byte address is down-converted to word index: ma_adr = wb_addr_i[SSRAM_HADR+2:2]
- Arbiter drives sram_adr_o[SSRAM_HADR:0]; read data is shared; only sample on ack
- Handshake: wb_ack_o issued after arbiter returns ma_ack; note M-side (USB core) has priority and can preempt W-side at any time

Addressing conventions and alignment
- Wishbone addresses are byte-based; all RF addresses are 32-bit word-aligned and accessed as adr[8:2]
- MEM accesses are also 32-bit word-aligned (lower address bits [1:0] dropped before reaching

SSRAM)
- System bank read aliasing: adr[6:2]==0 and 1 return identical contents (writes compare exact addresses in the documented row)

Integration notes
- Choose USBF_UFC_HADR to suit SoC address map; keep software and hardware decode consistent
- All RF offsets documented are relative to the RF region (wb_addr_i[HADR]==0) and use 32-bit word increments
- MEM region size and address bus width are independent of HADR and determined by SSRAM_HADR; software must ensure MEM accesses remain within the external buffer space
- When the USB core (M-side) is active, W-side MEM accesses may experience wait states; RF accesses are unaffected by arbiter preemption

Quick examples
- HADR=17 (test): RF at wb_addr[31:0] with bit17=0; MEM at wb_addr with bit17=1
- HADR=12 (non-test): RF at wb_addr with bit12=0; MEM at wb_addr with bit12=1

-- Timing and Protocol Constants --

Timing and Protocol Constants

UTMI/link-state timing (usbf_utmi_ls)
- Base ticks
- USBF_T1_PS_250_NS ≈ 250 ns tick for idle-qualified timers (T1 path).
- Free-running path (T2): ~2.5 µs tick aggregated to ~0.5 ms and ms/100 ms granularity.
- Thresholds/windows
- Reset detect: SE0 > 2.5 µs.
- Suspend entry: idle > 3.0 ms (FS/HS), with HS intermediate RES_SUSP check after >100 µs.
- Device-initiated resume (remote wake): allowed after idle > 5 ms; drive K for ≥1.0 ms; settle >100 µs before NORMAL.
- HS chirp negotiation: device K chirp hold >1.2 ms; commit to HS after 6 edges (3 K/J pairs); otherwise remain FS.
- Attach debounce: 100 ms before declaring attached.
- Idle definition by speed
- FS idle = J; HS idle = SE0 (gates T1 timers).

Protocol/transaction timing (usbf_pe)
- ACK wait after IN (IN2 state):
- HS: 22 (implementation-specific count units).
- FS: 36 (implementation-specific count units).
- OUT data wait (OUT state):
- HS/FS use same respective thresholds as above; timer clears while rx_active.

SOF/microframe timing (usbf_pl)
- Half-microsecond tick: USBF_HMS_DEL = 5'h1c generates hms_clk used for sof_time (time since last SOF).
- Microframe/frame accounting:
- frame_no captured on SOF; repeated SOFs with same 11-bit frame number increment mfm_cnt (HS microframe index 0..7).
- frm_nat[31:0] = {mfm_cnt, 1'b0, frame_no[10:0], 4'h0, sof_time[11:0]}.

PID format and constants
- PID encodings defined by T_PID_* macros.

- On-wire PID format: 8 bits = {~PID4, PID4} (upper nibble is bitwise complement); LSB-first transmission.
- Unsupported/filtered classes (front-end): PRE/ERR, SPLIT, and PING when not in HS are rejected before policy engine.

CRC conventions
- Token CRC5
- Polynomial: G5(x) = x^5 + x^2 + 1.
- Initial seed: 0x1F; transmitted CRC bits are typically the bitwise complement of the LFSR contents (LSB-first).
- Data CRC16
- Polynomial: G16(x) = x^16 + x^15 + x^2 + 1 (0x8005), LSB-first.
- TX: initialize to 16'hFFFF; append ones' complement of final remainder, low byte first (LSB-first on the wire).
- RX residue check: folding payload+received CRC yields 16'h800D when correct.

Reset style (global build option)
- USBF_ASYNC_RESET macro selects async-assert/sync-deassert behavior; otherwise resets are synchronous to the local clock domains.

Notes
- All timing constants above are implemented via counters/clocks inside utmi_if/usbf_utmi_ls/usbf_pl/usbf_pe and are chosen to meet USB 2.0 FS/HS timing windows.
- The numeric thresholds (e.g., 2.5 µs, 3 ms, 1.0 ms, 100 µs, 1.2 ms, 100 ms; HS=22/FS=36 counts) are fixed in this RTL but may be exposed as macros/constants in source for portability; altering them risks USB compliance.

-- Simulation and Debug Options --

Simulation and Debug Options

Elaboration-time reporting and checks
- On simulation start, the core prints:
- Enabled endpoints (via USBF_HAVE_EPn)
- RF/MEM address split bit (USBF_UFC_HADR)
- Computed external buffer size from SSRAM_HADR (bytes = 2^(SSRAM_HADR+3))
- Continuity check: warns if enabled endpoints are not contiguous from EP0 upward.

Status/diagnostic registers for debug (RF region)
- MAIN_CSR (word 0): read LineState[1:0], usb_attached, mode_hs, suspend; write din[5]=1 injects a remote-wakeup (rf_resume_req) for test.
- FUNCTION_ADDRESS (word 1): read/write device address.
- INTERRUPT MASKS (word 2): set masks for RF/system events into inta_o/intb_o.
- INTERRUPT SOURCE (word 3, read-to-clear):
- int_srcb[8:0] latched RF/system events (usb_reset, rx_err, attach/detach, suspend start/end, nse_err, pid_cs_err, crc5_err)
- int_srca[15:0] mirror of per-EP any-pending (OR of inta/intb)
- FRM_NAT (word 4): 32-bit snapshot {microframe index, frame number, time since SOF} for timing/diagnostics.
- UTMI vendor interface (word 5):
- Read latched VStatus[7:0]
- Write VControl[3:0] with one-shot VControl_Load pulse into the PHY domain

Per-endpoint debug/visibility (EP windows EP0..EP15)
- EPx_CSR: direction/type/maxpkt/DMA enable/stall/toggle and UC fields (uc_bsel/uc_dpd).
- EPx_INT: detailed per-EP events (read-to-clear).
- EPx_BUF0/BUF1: buffer descriptors (base address and size), used to correlate memory traffic in sim.

External stimulus hooks (useful in testbenches)
- resume_req_i pin to model external remote-wakeup requests.
- Software-injected rf_resume_req via MAIN_CSR write (din[5]=1).
- UTMI vendor control strobes to exercise PHY vendor paths.
- Wishbone access to MEM region for seeding/inspecting payloads; RF region for full configuration.

Observability signals (top-level)
- DMA handshakes per endpoint: dma_req_o[15:0] (from core) and dma_ack_i[15:0] (to core) for transfer pacing analysis.
- UTMI LineState and vendor status are latched each phy clock (LineState_r, VStatus_r) for stable CPU/sim readback.
- External SSRAM bus: sram_adr_o, sram_data_i/o, sram_re_o, sram_we_o to validate memory cycles.

Behavioral notes impacting simulation
- RF/system and per-EP INT registers are read-to-clear; tests should read them to acknowledge events.
- System bank reads are aliased across adr[6:2]==0 and 1; writes must target the documented row.
- W-side MEM accesses may be delayed/preempted by USB M-side (priority arbiter); expect wait states.
- Wishbone ACK is a one-shot pulse in wb_clk; hold CYC/STB/address/data until ACK.

Build-time knobs useful for simulation sweeps
- USBF_UFC_HADR: selects RF/MEM split bit (e.g., 17 in test, 12 in non-test).
- SSRAM_HADR: sets external buffer size and address widths (e.g., 14 $\rightarrow$ 128 KiB).
- USBF_HAVE_EPn: compile-time enable per endpoint; non-enabled EPs are stubbed with a dummy.
- USBF_ASYNC_RESET: choose async-assert/sync-deassert vs fully synchronous resets in each clock domain.

Limitations
- No built-in HS certification test modes (Test_J, Test_K, SE0_NAK, Test_Packet); not available as simulation toggles.
- No internal waveform/dump controls provided; use external simulator facilities.

-- Configuration Constraints and Dependencies --

Configuration Constraints and Dependencies

Build-time parameters/macros
- USBF_UFC_HADR (addr split): Must be consistently decoded by firmware. Changing this bit moves the RF/MEM boundary (RF when wb_addr_i[HADR]=0; MEM when =1). usbf_wb and software must agree.
- SSRAM_HADR (buffer memory size): Must match external 32-bit SSRAM depth. Drives sram_adr_o width, MEM address slicing (ma_adr=wb_addr_i[SSRAM_HADR+2:2]), and descriptor address field width (idin[SSRAM_HADR+2:4]).
- USBF_HAVE_EPn (endpoint enable set): EP0 is mandatory; EP1..EP15 must form a contiguous

range from EP0 upward. Gaps are unsupported (checked at simulation start). Disabled EPs become dummies.
- USBF_ASYNC_RESET (reset style): If defined, resets are async-assert/sync-deassert in each domain; otherwise fully synchronous. Must be applied consistently across top and submodules.

Clocking and CDC
- Two primary clocks: phy_clk_pad_i (UTMI/link/protocol/memory arbiter) and clk_i (Wishbone/RF). All core timing-critical logic runs on phy_clk.
- usbf_mem_arb uses only phy_clk; its wclk port is unused. W-side memory requests are bridged/synchronized by usbf_wb; do not drive mem_arb from any other clock domain.
- Wishbone CDC: wb_req level-sampled into phy_clk; wb_ack returned as a one-shot pulse in clk_i. The Wishbone master must hold address/data/REQ stable until ACK.

Wishbone/bus assumptions
- Classic WB B3 single-beat only: no byte enables (SEL), no ERR/RTY, no bursts, no STALL/pipeline. 32-bit, word-aligned addressing (ADR[n:2]).
- usbf_wb generates a single-cycle ACK; MEM accesses may incur wait states due to arbiter priority and are preemptible by the USB M-side.

UTMI/PHY dependencies
- Standard UTMI pins required; SuspendM polarity/behavior must match the PHY (module asserts per internal expression). VBUS is used as an additional POR source in the link-state controller.
- Vendor sideband: VControl[3:0]/VControl_Load and VStatus[7:0] must be supported by the PHY if used; writes are one-shot crossed into phy_clk.

Memory/buffer path
- External memory must be 32-bit wide; all MEM accesses are word-aligned (drop [1:0]). sram_re_o may be asserted continuously; external SSRAM must tolerate always-on read enable if applicable.
- Buffer descriptors: Base addresses must point within the MEM region and be word-aligned; size fields must fit descriptor/counter widths (TX size ≤ 16 KiB; RX byte count ≤ 2 KiB typical from IDMA counters).

Endpoint configuration/dependencies
- EP number used for match is csr[21:18]; ensure unique mapping and consistency with software's EP addressing.
- DMA enable (csr[15]) and direction/type fields must be consistent with intended use; for control EP, special sequencing rules apply inside the core.
- Per-EP interrupts are read-to-clear; top-level RF/system events are also read-to-clear. RF masks (word 2) gate only RF/system events, not per-EP INT masks.

Software/firmware requirements
- Align firmware address decoding to USBF_UFC_HADR and word addressing for RF. Program EPx_CSR and EPx_BUFx before enabling transfers; keep BUF addresses within SSRAM space.
- Observe read-to-clear semantics for INT registers; acknowledge events by reading the appropriate words.
- Expect CPU MEM accesses to be stalled/preempted by USB side; poll or wait for WB ACK properly.

Compliance and feature caveats
- HS certification test modes (Test_J/K, SE0_NAK, Test_Packet, Force_Enable) are not implemented. Remote-wakeup feature gating is not enforced in hardware (must be managed by firmware).
- Function address is not forced to 0 on USB reset in hardware; firmware must ensure Default state behavior.

Integration notes

- Changing USBF_UFC_HADR or SSRAM_HADR alters external interface widths/decoding; update SoC interconnect, memory maps, and software accordingly.
- Ensure resume_req_i and software-triggered rf_resume_req are used only when device is suspended; link FSM clears requests via suspend_clr.

MODULE HIERARCHY AND INTEGRATION NOTES

-- Submodule Overview and Responsibilities --

Submodule Overview and Responsibilities

Top wrapper: usbf_top
- Role: Integrates the USB device core. Bridges UTMI PHY, Wishbone host, and external 32-bit SSRAM. Owns simple status latches (susp_o, LineState_r, VStatus_r) and a latched resume request (resume_req_r). Ties global reset to phy_rst_pad_o.
- Key paths: RF region (registers/endpoint windows) vs MEM region (buffer memory) split by USBF_UFC_HADR. Exposes aggregated interrupts (inta_o, intb_o) and per-EP DMA handshakes.

UTMI front-end: usbf_utmi_if
- Responsibility: UTMI PHY adaptation and link/power control front-end.
- RX: Register and clean RxValid/RxActive/RxError/DataIn into core clock (phy_clk) domain.
- TX: Adapt core tx_data/tx_valid/tx_first/tx_valid_last to UTMI TxValid/TxReady/DataOut; hold semantics for last byte and resume/chirp (drive_k).
- Link/power: Interface to the link-state controller, export mode_hs, usb_reset, usb_suspend, usb_attached, and generate suspend_clr pulse.
- UTMI control pins: Drive XcvSelect, TermSel, OpMode, SuspendM per link state.
- Submodule: usbf_utmi_ls (link state machine)
- Role: Implements attach debounce, reset detect, suspend/resume, and HS chirp negotiation with timing windows; selects FS/HS transceiver and termination; produces drive_k and status signals.

Protocol layer: usbf_pl #(SSRAM_HADR)
- Responsibility: USB packet processing and endpoint policy engine; master of M-side memory requests.
- RX path: Decode tokens/data, validate CRCs, stream payload bytes.
- TX path: Assemble handshake/data packets with DATA PIDs and CRC16; handle ZLP.
- Policy: Per-endpoint accept/ignore decisions, handshake selection (ACK/NAK/STALL/NYET), data PID/toggle, sizing, HS/FS timeouts.
- Timing/diag: Maintain frm_nat (frame/microframe/time-since-SOF snapshot).
- RF coupling: Consume csr/buf0/buf1 and flow flags; broadcast descriptor/status updates and event strobes back to RF/EPs.
- Submodules inside pl:
- usbf_pd: RX packet decoder (PID classify, token ADDR/ENDP/CRC5, DATA stream, CRC16 check, token_valid/seq_err).
- usbf_pa: TX packet assembler (handshake/Data PID + payload + CRC16, tx_first/tx_valid_last, ZLP support).
- usbf_idma: Byte↔32-bit DMA bridge for RX (pack/write) and TX (read/unpack); unaligned starts, partial words, wrap, size/count, simple req/ack.

- usbf_pe: Endpoint policy engine (control/bulk/iso/int behaviors, timeouts, data PID/toggle management, descriptor updates, per-event interrupt strobes).

Memory arbiter: usbf_mem_arb #(SSRAM_HADR)
- Responsibility: Single-port 32-bit SRAM arbiter with fixed priority to USB core (M side) over Wishbone (W side).
- M side: Immediate combinational ack (mack=mreq); can preempt W at any time.
- W side: Toggles/pulses wack when granted while M is idle; maintains bus for ack phase; address/data/we muxed accordingly.
- Read data: Broadcast to both masters; each samples on its ack.

Register file and endpoint mux: usbf_rf
- Responsibility: CPU-visible RF/system registers, per-endpoint windows, interrupt aggregation, DMA plumbing, and UTMI vendor/resume crossings.
- System bank: MAIN_CSR (status + SW resume one-shot), FUNCTION_ADDRESS, INT masks, INT sources (read-to-clear), FRM_NAT, UTMI vendor interface.
- Endpoint windows EP0..EP15: CSR, INT (read-to-clear), BUF0, BUF1; only selected EP captures broadcast updates.
- Aggregation: OR per-EP inta/intb with masked RF/system events to drive inta_o/intb_o; mirror per-EP any-pending in int_srca.
- Matched EP mux (core side): Present selected EP’s csr/buf0/buf1 and flow flags back to protocol layer.
- CDC: One-shots for UTMI vendor write and rf_resume_req; edge/latch RF/system events into wclk.
- Submodules per endpoint:
- usbf_ep_rf: Per-EP CSR/INT/BUF registers, endpoint match, per-EP DMA request generation and counters, INT masking/level outputs.
- usbf_ep_rf_dummy: Tie-off stub for disabled endpoints (no DMA/INT/match; fixed outputs).

Wishbone bridge: usbf_wb
- Responsibility: Minimal Wishbone slave and RF/MEM router with CDC.
- Address decode: Top address bit (USBF_UFC_HADR) selects RF vs MEM space.
- Control FSM (phy_clk): Issue RF strobes (single-cycle) or forward MEM requests to arbiter; generate wb_ack_d on completion; fixed W1/W2 dead cycles.
- WB ack: One-shot pulse in wb_clk domain; read data muxed from RF/MEM accordingly; assumes classic single-beat WB.

Data/control flow (top-level integration)
- PHY ↔ Protocol: usbf_utmi_if adapts signals to/from usbf_pl; usbf_utmi_ls drives link control/status.
- Protocol ↔ Memory: usbf_pl masters M-side of usbf_mem_arb for buffer moves via usbf_idma; preempts W-side when active.
- Protocol ↔ RF/EPs: Broadcast update bus (idin + strobes) to all EPs; only the matched EP latches. RF mux returns matched CSR/BUF/flags to pl.
- CPU/Wishbone ↔ RF/MEM: usbf_wb decodes region and sequences RF vs MEM transactions; usbf_rf provides CSR/INT/BUF access; per-EP DMA handshakes exposed at top.

Clock/reset domains and crossings
- phy_clk domain: UTMI/link/protocol/mem_arb; time-critical data paths and FSMs.
- clk_i domain: Wishbone interface and CPU-visible RF registers/interrupts.
- CDC: Level + one-shot pulses (wb_req→phy, wb_ack→wb; UTMI vendor write and resume req wclk→clk; RF/system event latching into wclk).

-- Hierarchical Connectivity and Interfaces --

Hierarchical Connectivity and Interfaces

External interfaces at usbf_top
- Wishbone host (clk_i domain)
- Inputs: wb_adr_i, wb_dat_i, wb_we_i, wb_stb_i, wb_cyc_i
- Outputs: wb_dat_o, wb_ack_o
- Interrupts: inta_o, intb_o
- UTMI PHY (phy_clk_pad_i domain)
- Inputs: DataIn[7:0], RxValid, RxActive, RxError, TxReady, LineState[1:0], usb_vbus_pad_i
- Outputs: DataOut[7:0], TxValid, XcvSelect, TermSel, SuspendM, OpMode[1:0], phy_rst_pad_o
- UTMI vendor sideband
- Inputs: VStatus_pad_i[7:0]
- Outputs: VControl_pad_o[3:0], VControl_Load_pad_o
- External 32-bit SSRAM
- Outputs: sram_adr_o[SSRAM_HADR:0], sram_we_o, sram_re_o, sram_data_o[31:0]
- Inputs: sram_data_i[31:0]
- Power/PM
- Output: susp_o
- Input: resume_req_i
- Per-endpoint DMA handshakes
- Outputs: dma_req_o[15:0]
- Inputs: dma_ack_i[15:0]

Top-level partitioning (internal block connections)
- usbf_utmi_if (PHY front-end)
- External UTMI pins ↔ usbf_utmi_if
- To protocol layer (phy_clk): rx_data[7:0], rx_valid, rx_active, rx_err → usbf_pl; tx_ready → usbf_pl
- From protocol layer: tx_data[7:0], tx_valid, tx_valid_last, tx_first ← usbf_pl
- Link status/control (to core/regs): mode_hs, usb_reset, usb_suspend, usb_attached → usbf_rf/main CSR readback; suspend_clr → top resume latch clear
- UTMI control pins: XcvSelect, TermSel, SuspendM, OpMode[1:0] → PHY
- Vendor write handshake from usbf_rf (clk_i→phy_clk crossed)

- usbf_pl (protocol layer)
- From usbf_utmi_if (PHY RX/TX handshake as above)
- From usbf_rf (selected endpoint image, via matched-EP mux and CDC): csr[31:0], buf0[31:0], buf1[31:0], dma_in_buf_sz1, dma_out_buf_avail; function address (funct_adr)
- To usbf_rf (broadcast update bus): idin[31:0], buf0_set, buf1_set, buf0_rl, uc_bsel_set, uc_dpd_set, int_*_set strobes, out_to_small
- M-side memory master to usbf_mem_arb (phy_clk): madr[SSRAM_HADR:0], mdout[31:0], mwe, mreq → arbiter; mdin[31:0], mack ← arbiter
- Diagnostics to usbf_rf: frm_nat[31:0] → RF Word 4

- usbf_mem_arb (SRAM arbiter)
- To external SSRAM: sram_adr_o, sram_data_o, sram_we_o, sram_re_o; from SSRAM: sram_data_i
- M side (priority) from usbf_pl: madr, mdout, mwe, mreq; to usbf_pl: mdin, mack
- W side (CPU) from usbf_wb: wadr, wdin, wwe, wreq; to usbf_wb: wdout, wack
- Read data (sram_data_i) broadcast to both M and W; each side samples on its ack

- usbf_rf (register file, endpoint mux, interrupts)
- Wishbone RF access from usbf_wb (clk_i): adr[8:2], re, we, din[31:0] → rf; dout[31:0] → wb

- System/status inputs (phy_clk→clk_i crossed where needed): LineState_r[1:0], VStatus_r[7:0], mode_hs, usb_attached, usb_suspend, usb_reset
- UTMI vendor interface: utmi_vend_ctrl[3:0] → usbf_utmi_if (via clk crossing), VControl_Load_pad_o pulse; VStatus_pad_i[7:0] → latched for RF readback
- Broadcast update bus from usbf_pl (phy_clk) to all EP sub-blocks; only matched EP captures via ep_match
- Matched EP mux back to usbf_pl: selected csr/buf0/buf1/dma flags (with safe CDC)
- Interrupt aggregation: per-EP inta/intb ORed with masked RF/system events → inta_o, intb_o; int_srca mirror exposed in RF
- DMA handshakes: per-EP dma_req → dma_req_o[15:0] (top-level out); dma_ack_i[15:0] (top-level in) → respective EP blocks
- SW resume: rf_resume_req (clk crossing) → usbf_utmi_if/link path

- usbf_wb (Wishbone bridge)
- Wishbone slave (clk_i): wb_* ↔ usbf_wb; one-shot wb_ack_o generation
- Region decode by USBF_UFC_HADR: RF_SEL routes to usbf_rf; MEM_SEL routes to usbf_mem_arb W side
- RF path (clk_i): adr[8:2], we/re, din → usbf_rf; dout ← usbf_rf
- MEM W-side path (phy_clk FSM): ma_adr=wb_addr_i[SSRAM_HADR+2:2], ma_dout/ma_we/ma_req → usbf_mem_arb; ma_din/ma_ack ← arbiter; completion reflected as wb_ack_o in clk_i domain

Addressing/connectivity conventions
- RF region (wb_addr_i[HADR]=0): 32-bit word addressing into usbf_rf via adr[8:2]; system bank aliased on adr[6:2]==0/1 for reads
- MEM region (wb_addr_i[HADR]=1): byte address down-converted to word index ma_adr=wb_addr_i[SSRAM_HADR+2:2] before reaching usbf_mem_arb/SSRAM
- Buffer descriptors (in EP windows) hold word-aligned base addresses within external MEM region

Clock and CDC boundaries
- phy_clk_pad_i domain: usbf_utmi_if, usbf_pl (including pd/pa/idma/pe), usbf_mem_arb
- clk_i domain: usbf_wb, CPU-visible usbf_rf registers and interrupt logic
- Crossings:
- Wishbone req/ack: wb_req (clk_i→phy_clk level-sampled), wb_ack_d (phy_clk→clk_i one-shot)
- RF/system status and vendor control: UTMI vendor write (clk_i→phy_clk one-shot), VStatus/LineState sampled in phy_clk and read in clk_i
- Protocol↔RF endpoint image and events: matched EP mux outputs and broadcast strobes crossed safely between domains

Power/management connectivity
- suspend: usb_suspend (from usbf_utmi_if) reflected to susp_o; status readable in RF MAIN_CSR
- resume: external resume_req_i latched to resume_req_r and consumed by link FSM; software-triggered rf_resume_req from RF MAIN_CSR write also feeds link FSM (one-shot)

Interrupt visibility
- inta_o/intb_o: OR of per-EP levels and masked RF/system events from usbf_rf
- Detailed per-EP events: readable via EPx_INT (read-to-clear)
- RF/system events: readable via RF Word 3 (read-to-clear); per-EP any-pending mirrored in int_srca[15:0]

-- CDC Boundaries and Synchronization --

CDC Boundaries and Synchronization

Clock domains
- phy_clk_pad_i (PHY/core domain): UTMI front-end (usbf_utmi_if, usbf_utmi_ls), protocol layer (usbf_pl: pd/pa/idma/pe), memory arbiter (usbf_mem_arb)
- clk_i (CPU/bus domain): Wishbone bridge (usbf_wb), CPU-visible register file and interrupt logic (usbf_rf wclk side)

Primary CDC interfaces and techniques
1) Wishbone ↔ Core sequencing (clk_i ↔ phy_clk)
- Request path (clk_i→phy_clk): usbf_wb samples wb_stb_i & wb_cyc_i as a level (wb_req_s1) into phy_clk to start an RF or MEM access.
- Completion path (phy_clk→clk_i): usbf_wb generates wb_ack_d in phy_clk; a one-shot edge-detected pulse produces wb_ack_o in clk_i. Note: this uses a single sampling stage before edge detection; acceptable for related clocks but marginal for fully asynchronous clocks.

2) Protocol layer ↔ Register file/Endpoints (phy_clk ↔ clk_i)
- pl→rf broadcast bus (phy_clk→clk_i): idin[31:0] with buf0_set, buf1_set, buf0_rl, uc_bsel_set, uc_dpd_set, and int_*_set event strobes are crossed into the rf wclk side; only the matched endpoint captures (ep_match in clk_i domain).
- rf→pl matched EP image (clk_i→phy_clk): Selected csr, buf0, buf1, and flow flags (dma_in_buf_sz1, dma_out_buf_avail) are presented from rf (clk_i) and safely synchronized into pl (phy_clk) for policy decisions.
- frm_nat (phy_clk→clk_i): pl provides frm_nat[31:0] to rf; rf exposes it in the system bank for software reads.

3) UTMI vendor interface and status (clk_i ↔ phy_clk)
- Vendor write (clk_i→phy_clk): Writing RF Word 5 asserts utmi_vend_wr and transfers utmi_vend_ctrl[3:0] across as a one-shot; usbf_utmi_if drives VControl and VControl_Load to the PHY.
- Vendor/line-state status (phy_clk→clk_i): LineState_r[1:0] and VStatus_r[7:0] are latched in phy_clk and read back via rf in clk_i.

4) Power management and remote wake (clk_i ↔ phy_clk)
- Suspend mirror (phy_clk→clk_i): usb_suspend (phy) is sampled to produce susp_o and MAIN_CSR[0].
- External resume (clk_i→phy_clk): resume_req_i is latched at top (clk_i) as resume_req_r and consumed by the link FSM; cleared by suspend_clr (phy) creating a cross-domain one-shot handshake.
- SW resume (clk_i→phy_clk): Writing MAIN_CSR with din[5]=1 generates rf_resume_req (one-shot) that crosses to the link FSM in phy_clk; suspend_clr returns to clear it.

5) Per-endpoint DMA handshake (clk_i ↔ phy_clk inside rf)
- Request generation (clk_i): rf computes dma_req_d from counters (clk_i) and asserts a wclk-domain dma_req state machine.
- Ack return (clk_i←phy_clk path as documented in rf): dma_ack (wclk) is pipelined and re-crossed back into clk (rf's core side) as dma_ack_i to update counters; hold policies avoid chatter between acks.

6) Memory subsystem boundaries
- usbf_mem_arb runs entirely in phy_clk; its W side is driven by usbf_wb's phy_clk FSM. The wclk input of mem_arb is unused; no CDC within the arbiter. Read data (SSRAM) is broadcast in phy_clk and qualified by each side's ack within its domain.

Reset strategy across domains

- Default: fully synchronous resets per local domain.
- Optional: if USBF_ASYNC_RESET is defined, resets are async-assert/sync-deassert within each clock domain (phy_clk and clk_i) across all submodules.

CDC quality notes and caveats
- The wb_ack crossing uses minimal sampling before one-shot generation; robust for related clocks but may be marginal under fully asynchronous clk_i/phy_clk.
- Broadcast update strobes and matched EP mux employ level/one-shot style crossings; only the addressed endpoint latches updates, reducing metastability risk windows.
- LineState/VStatus snapshots are registered in phy_clk before CPU-side access, avoiding mid-cycle sampling in clk_i.
- All PHY Rx/Tx indicators are registered in phy_clk by usbf_utmi_if; no CDC on the byte stream.

Summary
- Two-domain design (phy_clk, clk_i) with CDC handled via level sampling plus one-shot pulse synchronizers for control paths (WB ack, vendor write, resume), registered snapshots for status (LineState/VStatus/frm_nat), and explicit request/ack pipelines for DMA and Wishbone sequencing. The memory arbiter and protocol datapaths remain entirely in phy_clk to avoid data CDC.

-- Reset Distribution and Power-Up --

Reset Distribution and Power-Up

Global reset sources and distribution
- Top-level system reset: rst_i (active as defined by build) fans out to both clock domains and is also driven to the PHY as phy_rst_pad_o.
- Domain-local style: By default, resets are synchronous to the local clock in each domain; if USBF_ASYNC_RESET is defined, resets are async-assert/sync-deassert per domain (phy_clk_pad_i and clk_i).
- Additional POR in link: usb_vbus_pad_i is treated inside the UTMI link-state controller (usbf_utmi_ls) as an additional POR source, forcing the link FSM to its POR state when asserted.

Per-block reset behavior (post-assertion state)
- usbf_utmi_if (phy_clk domain)
- Clears RX indicators (rx_valid, rx_active, rx_err) and initializes Tx path with TxValid=0; DataOut driven only when re-enabled.
- UTMI control is managed by usbf_utmi_ls after reset release.
- usbf_utmi_ls (phy_clk domain)
- Enters POR: defaults to FS interface (XcvSelect=FS, TermSel on, OpMode non-driving until sequenced), mode_hs=0, suspend cleared, attached cleared.
- After attach debounce (~100 ms with VBUS present), sets usb_attached and transitions to NORMAL.
- usbf_pl (phy_clk domain)
- All internal FSMs (pd/pa/idma/pe) return to IDLE; CRC accumulators cleared; no DMA requests issued.
- usbf_idma (phy_clk domain)
- TX/RX engines idle; size/byte counters cleared; no outstanding mreq; write enables deasserted.
- usbf_pe (phy_clk domain)
- Endpoint policy resets to idle; no handshakes scheduled; data PID/toggle state will be recomputed from descriptor/UC fields on first use.
- usbf_mem_arb (phy_clk domain)
- Grants reset; wack_r=0; no requester selected; sram_we_o deasserted.
- usbf_rf (clk_i domain)

- Function address set to 0; interrupt masks cleared; RF/system event latches (int_srcb) cleared; rf_resume_req deasserted; UTMI vendor write pulse deasserted.
- Per-endpoint register files: BUF0/BUF1 initialize to 0xFFFF_FFFF (sentinel invalid), INT latched status cleared, masks reset; CSR fields come up in reset defaults (disabled until programmed).
- usbf_wb (clk_i/phy_clk domains)
- Control FSM reset to IDLE (phy_clk); wb_ack_o generation flops cleared (clk_i). Only the FSM state is explicitly reset; other paths rely on defaulted logic.

Run-time USB bus reset (link-level)
- usbf_utmi_ls detects SE0 > 2.5 µs and asserts usb_reset during RESET state, then performs HS negotiation (FS/HS chirp) before returning to NORMAL.
- RF/System visibility: usb_reset is latched into RF int_srcb[8] (read-to-clear) and can be routed to inta_o/intb_o via masks.
- Note: The FUNCTION_ADDRESS register is not auto-cleared by hardware on USB bus reset; firmware must write 0 to return the device to the Default state per USB 2.0.

Power-up and attach sequence
- With rst_i asserted: all domains held in reset; PHY held via phy_rst_pad_o.
- After rst_i deasserts and VBUS present: link FSM debounces attach (~100 ms), then enters NORMAL in FS, and later negotiates HS if host chirps are observed.
- Software initialization sequence (typical):
1) Program global masks (RF Word 2) and verify MAIN_CSR status.
2) Configure EPx_CSR for each enabled endpoint (direction/type/max packet/DMA enable).
3) Initialize EPx_BUFx descriptors (base/size/flags) and seed external MEM payloads as needed.
4) Enable system DMA handshakes if used and unmask interrupts.

Suspend/resume interactions (not a reset but affects power state)
- Suspend indication (usb_suspend) appears in MAIN_CSR[0] and is mirrored on susp_o; UTMI SuspendM drives the PHY appropriately.
- Remote-wakeup requests can be asserted externally (resume_req_i) or by software (MAIN_CSR din[5]=1 $\rightarrow$ rf_resume_req one-shot). The link FSM issues suspend_clr (phy) to clear these requests after resume sequencing.

Reset style and clock-domain considerations
- Two-domain reset application: rst_i is applied in both clk_i and phy_clk_pad_i domains with the chosen style (sync by default; async-assert/sync-deassert if USBF_ASYNC_RESET).
- CDC during/after reset: Status snapshots (LineState_r, VStatus_r) are registered in phy_clk before CPU reads; Wishbone ack one-shot generation crosses from phy_clk to clk_i only after FSM restart; RF/system event latches are cleared in clk_i domain on reset.

External interface expectations at power-up
- UTMI PHY: phy_rst_pad_o asserts with rst_i; after release, PHY must meet UTMI spec for ready/TxReady behavior.
- External SSRAM: contents undefined at power-up; software must initialize buffer regions prior to use.
- Interrupt lines: inta_o/intb_o deasserted until events/masks enable them.

-- Memory Arbiter Integration --

Memory Arbiter Integration

Role and topology
- usbf_mem_arb is a single-port, 32-bit SSRAM arbiter that shares the external buffer memory

between:
- M side (USB core master): usbf_pl (protocol layer + idma) in phy_clk domain, highest/preemptive priority.
- W side (CPU/Wishbone master): usbf_wb's memory path in phy_clk domain, lower priority.
- Entire arbiter runs in phy_clk; wclk input is unused.

External memory interface
- Ports: sram_adr_o[SSRAM_HADR:0], sram_data_o[31:0], sram_data_i[31:0], sram_we_o, sram_re_o.
- Read enable sram_re_o is asserted continuously (always-on read); external SSRAM must tolerate this.
- Data width is fixed at 32 bits; address bus width parameterized by SSRAM_HADR.

Master-side interfaces and mapping
- M side (to usbf_pl/idma): madr[SSRAM_HADR:0], mdin[31:0] (write), mdout[31:0] (read), mwe, mreq, mack.
- Arbitration/ack: mack = mreq (combinational). M writes when (mreq & mwe).
- W side (to usbf_wb phy_clk FSM): wadr[SSRAM_HADR:0], wdin[31:0], wdout[31:0], wwe, wreq, wack.
- Addressing: usbf_wb converts Wishbone byte address to word index: ma_adr = wb_addr_i[SSRAM_HADR+2:2].
- Arbitration/ack: wack is a one-clock pulse/toggle generated by wack_r when M is idle.

Arbitration policy and muxing
- Fixed, immediate priority to M:
- Selection: wsel = (wreq | wack) & !mreq.
- Address/data/we mux: sram_* driven from W when wsel, else from M.
- Read data broadcast: sram_data_i is fed to both mdout and wdout; each master must sample only on its ack (mack/wack).
- Preemption: Any assertion of mreq immediately de-selects W (wsel=0) and forces wack low; W must retry when M releases the bus.

Behavioral timing and throughput
- M transfers are zero-wait from arbiter perspective (mack=mreq); overall latency set by external SSRAM.
- W transfers when M idle:
- If W holds wreq high, arbiter produces a transfer opportunity every other phy_clk (wack toggling), yielding ~0.5× phy_clk throughput in steady state.
- For single transfers, W should deassert wreq after wack to avoid unintended repeats.

Reset and initialization
- On reset: wack_r=0, no requester selected, sram_we_o deasserted.
- M and W side acks are quiescent until valid requests arrive.

Clock/CDC considerations
- Arbiter is entirely in phy_clk; no internal CDC.
- W side signals must already be synchronous to phy_clk. usbf_wb performs clk_i↔phy_clk bridging (wb_req level sample into phy, wb_ack one-shot back to clk_i).

Addressing/alignment and capacity
- All memory accesses are 32-bit word-aligned; low address bits [1:0] of Wishbone addresses are dropped before reaching the arbiter.
- Effective buffer size = 2^(SSRAM_HADR+1) words × 4 bytes; descriptors (in RF) must point within

this range and be word-aligned.

Integration endpoints
- Upstream M master: usbf_pl/usbf_idma drives madr/mdin/mwe/mreq, consumes mdout/mack.
- Upstream W master: usbf_wb drives wadr/wdin/wwe/wreq, consumes wdout/wack.
- Downstream: external SSRAM via sram_adr_o/sram_data_o/sram_data_i/sram_we_o/sram_re_o.

System/firmware implications
- CPU/Wishbone accesses to MEM region may be delayed or preempted by USB activity; software must honor wb_ack_o handshake and avoid timing assumptions.
- Use word-wide, word-aligned accesses; plan bulk transfers during USB idle periods for best throughput.
- External SSRAM interface timing must meet continuous-read-enable constraint.

-- Software Integration Guidelines --

Software Integration Guidelines

Address map and bus usage
- Regions (compile-time HADR): RF region when wb_addr[HADR]==0; MEM region when wb_addr[HADR]==1. HADR=17 (test) or 12 (non-test).
- RF addressing: 32-bit word addresses via adr[8:2]; reads of the system bank are aliased on adr[6:2]==0 or 1; writes must target adr[6:2]==0.
- MEM addressing: 32-bit word-aligned; Wishbone byte address is converted to word index (drop [1:0]). SSRAM size = 2^(SSRAM_HADR+3) bytes.
- Wishbone protocol: classic single-beat. Hold CYC/STB until wb_ack_o; no byte enables; one ack per access. Do not assume throughput—USB side can preempt MEM accesses.

Initialization sequence (typical)
1) Apply rst_i; wait for clocks stable. After deassertion, poll MAIN_CSR for attach/mode/suspend.
2) Program interrupt masks (RF Word 2) for desired RF/system events; clear any latched sources by reading Word 3.
3) Configure EPx_CSR for each used endpoint (direction/type/max packet/DMA enable/STALL state/UC fields), EP0 at minimum.
4) Initialize EPx_BUFx descriptors (base=word-aligned MEM addr, size in bytes, flags). Preload MEM payloads for IN endpoints.
5) Unmask per-EP interrupts inside EPx_INT as needed and enable system DMA handshakes if using external DMA (dma_req_o/dma_ack_i).
6) After SET_ADDRESS on EP0, write FUNCTION_ADDRESS (RF Word 1).

Runtime operation
- Interrupt handling
- Top-level: inta_o/intb_o are OR(per-EP) with masked RF/system events.
- Service order: (a) Read RF Word 3 to snapshot/clear RF/system events (read-to-clear), then (b) read EPx_INT of affected endpoints (read-to-clear) and act.
- Mirrors: int_srca[15:0] exposes per-EP any-pending to help triage ISRs.
- Buffer management
- Only write EPx_BUFx/CSR when the endpoint is idle (no ongoing transfer) to avoid racing the core. For DMA-enabled flows, prefer updating after buffer-done interrupts.
- Descriptors must be word-aligned addresses within MEM region; sizes in bytes. UC fields (toggle/buffer select) are updated by the core; software should treat them as read-mostly unless intentionally overriding.
- MEM region access (CPU)

- Plan bulk MEM reads/writes when USB is idle: the arbiter gives absolute priority to USB transfers; CPU sees wb_ack gaps/preemption. Hold CYC/STB until ack and do not assume back-to-back acks.

Suspend/Resume and power
- Status: MAIN_CSR[0]=suspend, [1]=mode_hs, [2]=attached, [4:3]=LineState snapshot.
- Remote wake (device-initiated): write MAIN_CSR with din[5]=1 to assert a one-shot resume request. Ensure USB spec prerequisites (remote feature enabled, bus idle >5 ms) are enforced by firmware.
- Host-initiated resume/suspend edges appear as RF/system events (Word 3) and can be interrupt-masked.

USB bus reset behavior
- A bus reset is reported in RF Word 3 int_srcb[8]. After handling, firmware must set FUNCTION_ADDRESS back to 0 (hardware does not auto-clear) and reinitialize any EP state as required by the USB 2.0 state machine.

PHY vendor sideband (optional)
- Write RF Word 5 to drive VControl[3:0] (one-shot VControl_Load); read VStatus[7:0] for PHY/vendor diagnostics.

Timing/diagnostics
- FRM_NAT (RF Word 4) provides {microframe index, frame number, time since SOF}; useful for scheduling IN data or isochronous timing.

DMA options
- Internal IDMA (protocol core) moves bytes between USB and SSRAM automatically when endpoints are armed.
- External/system DMA (optional): per-EP dma_req_o[15:0] indicate work pending; assert dma_ack_i[x] when your DMA completes a word transfer per the integration of usbf_ep_rf counters. If unused, tie dma_ack_i low and manage buffers via CPU.

Error and exception handling
- System-level RF Word 3 bits (pid_cs_err, crc5_err, rx_err, nse_err) reflect core-wide conditions; EPx_INT captures endpoint-local events (CRC16 error, timeout, unexpected PID, out_to_small, BUF0/1_DONE). All are read-to-clear—read once per ISR.
- For OUT size-policy: if sml_ok/lrg_ok are not set in EPx_CSR, the core may NACK or flag out_to_small/too_large; software should adjust max packet/descriptor sizes accordingly.

Build-time considerations for software
- HADR and SSRAM_HADR determine address decode and MEM size; align your driver’s address math to the synthesized configuration.
- Endpoint presence is compile-time (macros). Software should be built against the same configuration; probing disabled EPs will typically read zeros/dummy values.

Compliance caveats (firmware duties)
- EP0 policy and some certification behaviors are not hard-enforced in RTL. Firmware must ensure default-control-pipe rules (e.g., SETUP acceptance while halted, correct status ZLP handling) and clear FUNCTION_ADDRESS on bus reset.

Best practices
- Gate descriptor updates by endpoint-idle checks (no active DMA, no match in progress).
- Use interrupts over polling; fall back to polling EPx_INT/RF Word 3 if inta_o/intb_o aren’t wired.
- Keep all RF/MEM accesses word-aligned, full 32-bit words; avoid assumptions on ack cadence; insert

timeouts in software loops around Wishbone acks.
- Log FRM_NAT snapshots for elusive timing issues and to correlate with host microframe scheduling.

-- Addressing and Alignment Guidelines --

Addressing and Alignment Guidelines

Global regions and decode
- Two top-level regions selected by a single high address bit (HADR):
- RF region: wb_addr[HADR]==0 → usbf_rf (register file and EP windows)
- MEM region: wb_addr[HADR]==1 → external SSRAM via usbf_mem_arb
- HADR is compile-time: 17 (test) or 12 (non-test). Software must match the synthesized value.

RF region (register file) addressing
- Access width: 32-bit words; all RF addresses are word addresses (increment by 1 per 4 bytes).
- usbf_wb presents adr[8:2] to usbf_rf; byte lanes/byte writes are not supported.
- System bank (aliased rows): reads at adr[6:2]==0 or 1 return the same contents; writes must target adr[6:2]==0 exactly.
- System bank word map (adr[2:0]):
- 0 MAIN_CSR, 1 FUNCTION_ADDRESS, 2 INTERRUPT_MASKS, 3 INTERRUPT_SOURCE (read-to-clear), 4 FRM_NAT, 5 UTMI_VENDOR
- Endpoint windows (present or dummy for disabled EPs):
- EPn base word address = 4 + 4*n in adr[6:2] space (n=0..15)
- Within each EP window (adr[1:0]): 0=CSR, 1=INT (read-to-clear), 2=BUF0, 3=BUF1

MEM region (external SSRAM) addressing
- Access width: 32-bit words only. Wishbone byte address is down-converted: ma_adr = wb_addr[SSRAM_HADR+2:2].
- Effective buffer size = 2^(SSRAM_HADR+3) bytes (parameterized by SSRAM_HADR).
- The arbiter always enables read (sram_re_o=1); external memory must tolerate continuous read-enable.

Endpoint buffer descriptor alignment and sizing
- BUF0/BUF1 base addresses must be 32-bit word-aligned within the MEM region.
- Size fields are in bytes; software should set sizes consistent with endpoint max packet size and transfer policy.
- UC fields (buffer select/data PID) are primarily maintained by hardware; treat as read-mostly unless intentionally overriding policy.

DMA/address granularity and unaligned handling
- CPU path is word-aligned only (no SEL_I/byte enables). All RF/MEM writes must be full 32-bit words.
- Internal IDMA can handle unaligned packet boundaries via read-modify-write and partial-word updates; this does not relax software’s requirement to program word-aligned base addresses.

Wishbone usage and acks
- Classic single-beat protocol: hold CYC/STB until wb_ack_o; one ack per access; no bursts, no byte enables.
- MEM accesses may be preempted by USB activity (M-master priority). Do not assume back-to-back acks.

Address validity and disabled endpoints
- Accesses to disabled EP windows return dummy values (e.g., CSR=0, BUF0/BUF1=0xFFFF_FFFF) and never assert DMA/interrupts.

- Endpoint numbering must be contiguous from 0 at build-time; software should rely on the compiled configuration.

Recommended software practices
- Keep all RF and MEM accesses 32-bit word-aligned; never program byte or halfword addresses.
- Validate that descriptor base + size lies entirely within the MEM region capacity defined by SSRAM_HADR.
- Update descriptors/CSR only when the endpoint is idle (no in-flight DMA or token match) to avoid races.

-- Remote Wakeup Integration --

Remote Wakeup Integration

Sources of a resume (device-initiated)
- External pin: resume_req_i (clk_i domain). Top-level latches it as resume_req_r and holds it until cleared by suspend_clr from the link.
- Software write: MAIN_CSR write with din[5]=1 in RF (clk_i) generates rf_resume_req (one-shot), safely crossed into the core (phy_clk) domain.
- Either source can request remote wake; both are cleared by the link's suspend_clr pulse after the resume sequence begins.

CDC and handshaking
- clk_i→phy_clk: resume_req_i path latched at top; rf_resume_req generated in usbf_rf; both cross as one-shot/level+clear into the link FSM.
- phy_clk→clk_i: suspend_clr is produced by usbf_utmi_if/usbf_utmi_ls and sampled to clear resume_req_r and the RF one-shot.

Link/PHY behavior (in usbf_utmi_ls)
- Device-initiated resume is honored only from SUSPEND after bus idle >5 ms (T2_wakeup).
- Sequence: RESUME_REQUEST → prepare PHY (FS term on, OpMode=non-driving) → RESUME_SIG with drive_k asserted for ≥1 ms → RESUME (clear suspend) → RESUME_WAIT (>100 µs) → NORMAL.
- SuspendM is deasserted while a resume request is pending (and asserted during suspend otherwise); Tx path keeps UTMI active while drive_k is asserted.

Firmware interface and status
- MAIN_CSR bits (read): [0]=suspend, [1]=mode_hs, [2]=attached, [4:3]=LineState snapshot.
- RF/system events (Word 3, read-to-clear): suspend_start, suspend_end, attach/detach, usb_reset, etc., can be mapped to inta_o/intb_o via masks.
- Software must enforce USB policy: only request remote wake when the host has enabled DEVICE_REMOTE_WAKEUP and bus idle timing requirements are met (hardware enforces >5 ms idle, not the feature enable).

Typical SW flow
1) Detect suspend (MAIN_CSR[0]=1 or interrupt suspend_start). Optionally gate on host feature enable.
2) When appropriate, request resume via MAIN_CSR write din[5]=1 (or assert resume_req_i).
3) Wait for suspend_end event and/or MAIN_CSR[0] to clear; service any resulting status/interrupts.

Notes and caveats
- Requests asserted while not in SUSPEND are ignored by the link FSM.

- Bus reset supersedes resume handling; firmware should reinitialize state after usb_reset.
- Both request sources produce a single resume sequence per suspend entry; additional writes/pulses are masked until suspend_clr returns.

-- Verification and Bring-Up Notes --

Verification and Bring-Up Notes

Pre-silicon simulation checklist
- Clocks and reset
- Drive both clk_i (Wishbone) and phy_clk_pad_i (UTMI/core). Hold rst_i active; verify phy_rst_pad_o mirrors rst_i.
- Check reset style variants if enabled (USBF_ASYNC_RESET): async assert/sync deassert in each domain.
- UTMI/link bring-up
- Apply usb_vbus_pad_i to trigger ATTACH; expect ~100 ms debounce before usb_attached=1 then NORMAL state (mode_hs=0 until HS negotiation).
- Issue SE0 reset on LineState for >2.5 µs and verify usb_reset assertion, then HS chirp/negotiation and mode_hs set accordingly.
- Wishbone smoke tests
- Confirm HADR decode (test=17, non-test=12). Reads of RF system bank alias at adr[6:2]==0 and 1; writes must target adr[6:2]==0.
- Exercise MEM region reads/writes; observe wb_ack_o one-shot pulses and possible preemption (acks may gap) when the USB M-master is active.
- RF/system and interrupts
- Program Word 2 masks; stimulate attach/detach, suspend_start/end, usb_reset and confirm int_srcb[8:0] set and clear on read (read-to-clear of Word 3).
- Validate int_srca[15:0] mirrors per-EP any-pending.
- Endpoint and DMA paths
- Program EPx_CSR (dir/type/maxpkt/DMA enable) and EPx_BUFx with word-aligned MEM base and size.
- For OUT: send DATA packets; verify RX DMA writes with read-modify-write on unaligned starts and partial last word handling.
- For IN: preload MEM payload; verify TX DMA prefetch/double-buffer and ZLP handling when size=0.
- Confirm per-EP INT (read-to-clear) events: BUF0/1_DONE, CRC16 error, timeout, unexpected PID, out_to_small.
- Memory arbiter behavior
- While holding W-side wreq high with M idle, expect wack to toggle every other phy_clk (approx. 0.5× throughput). Assert that any M-side mreq immediately preempts W (wack low, wsel=0).
- CDC/handshake robustness
- Validate wb_req level sampling into phy_clk and wb_ack one-shot back into clk_i; ensure the WB master holds address/data until ack.
- Verify LineState_r/VStatus_r snapshots are stable for CPU reads.
- Error injection
- PID nibble mismatch → pid_cs_err; bad token CRC → crc5_err; bad data CRC → per-EP CRC16 error; verify latching and read-to-clear semantics.
- Remote wakeup
- From SUSPEND, write MAIN_CSR din[5]=1 (or assert resume_req_i); expect RESUME_SIG (drive_k ≥1 ms) then suspend_clr and suspend_end interrupt.

Bring-up on hardware
- UTMI PHY hookup

- Confirm SuspendM polarity matches the PHY; ensure XcvSelect/TermSel/OpMode reach the PHY and that drive_k produces the expected K state.
- External SSRAM
- Memory must tolerate sram_re_o permanently high (continuous read-enable). Check timing/IO direction for sram_data_i/o and tri-state handling.
- Verify word-aligned addressing; misaligned base addresses in descriptors will corrupt data.
- Wishbone integration
- Respect classic handshake: hold CYC/STB until wb_ack_o; no byte enables supported; use 32-bit word accesses only.
- Software/firmware sequencing
- After bus reset, hardware does not clear FUNCTION_ADDRESS; firmware must write 0 to return to Default state before enumeration proceeds.
- Avoid updating EP descriptors/CSR while transfers are active; gate updates on endpoint idle or after BUF*_DONE.

Debug aids and observability
- Use FRM_NAT (Word 4) to correlate microframe index, frame number, and time-since-SOF for timing issues.
- Monitor mode_hs, usb_reset, usb_suspend, usb_attached, LineState via MAIN_CSR and/or exported status.
- Use UTMI vendor interface (Word 5) to poke VControl[3:0] and read VStatus[7:0] for PHY diagnostics.

Common pitfalls
- Forgetting word alignment for MEM accesses and EPx_BUFx base addresses.
- Assuming back-to-back MEM acks under USB load; the arbiter prioritizes the USB M-master.
- Writing RF system bank to the aliased read row (adr[6:2]==1); only adr[6:2]==0 accepts writes.
- Misusing remote wake: request while not in SUSPEND or without host enabling DEVICE_REMOTE_WAKEUP (hardware enforces idle>5 ms; feature gating is a firmware responsibility).

Known limitations impacting verification
- No HS test mode entry (Test_J/Test_K/SE0_NAK/Test_Packet/Force_Enable) implemented.
- Default control pipe (EP0) corner-case behaviors are not fully enforced in hardware; verify overall system behavior in conjunction with firmware.
- Wishbone CDC uses a simple level sample and one-flop edge detect; with fully asynchronous clocks, metastability risks exist—prefer related clocks or add synchronizers in the system environment.

-- Known Limitations and Best Practices --

Known Limitations and Best Practices

Known limitations
- USB 2.0 compliance gaps
- No HS test mode entry: Test_J, Test_K, SE0_NAK, Test_Packet, Force_Enable not implemented.
- Default control pipe (EP0) corner cases not fully enforced in hardware (e.g., guaranteed SETUP acceptance when halted, length-mismatch STALL policy, explicit Status ZLP handling). Firmware must ensure spec-compliant behavior.
- Function address is not auto-cleared on bus reset; firmware must write 0 to return to Default state.
- Wishbone interface
- Classic B3-style only: no bursts, no byte enables (SEL_I), no PIPE/STALL, no ERR/RTY. One-shot ack per access; master must hold CYC/STB until ack.
- CDC is minimal (level sample into phy_clk and single-flop edge detect back to clk_i). With unrelated

clocks, metastability risk exists; system should use related clocks or add synchronizers.
- Memory subsystem
- External SSRAM must tolerate continuous read-enable (sram_re_o=1).
- Arbiter gives fixed, immediate priority to USB (M side). CPU/Wishbone MEM accesses can be preempted at any time; no guaranteed back-to-back acks.
- No internal pipelining on the SRAM path; upstream masters must align address/data to ack timing.
- Addressing/alignment
- MEM and RF accesses are 32-bit word-only; no byte/halfword writes. Descriptor base addresses (BUF0/BUF1) must be 32-bit word-aligned.
- RF system bank reads are aliased (two rows); only adr[6:2]==0 accepts writes.
- Remote wakeup
- Hardware enforces idle>5 ms and resume signaling timing, but does not enforce host-enabled DEVICE_REMOTE_WAKEUP; firmware must gate requests.
- Endpoint configuration
- Endpoint presence is compile-time; disabled EPs read as zeros/dummy and never request DMA/interrupts.
- DMA/size counters
- Internal counters sized for typical buffers (e.g., TX up to 16 KB, RX count 2 KB per provided descriptions); plan software descriptors accordingly.

Best practices
- Firmware/driver
- After any bus reset, write FUNCTION_ADDRESS=0 before re-enumeration; set to assigned address after SET_ADDRESS.
- Gate EP descriptor/CSR updates to idle endpoints (no active DMA or match); prefer updating right after BUF*_DONE events.
- Use interrupts (inta_o/intb_o) and read-to-clear EPx_INT and RF Word 3; avoid polling in tight loops.
- Implement timeouts around Wishbone transactions; do not assume contiguous acks under USB traffic.
- Validate all descriptor base+size ranges against MEM capacity: 2^(SSRAM_HADR+3) bytes.
- Treat UC fields (toggle/buffer select) as read-mostly; only override intentionally.
- Request remote wake only in SUSPEND and only if host enabled the feature; use MAIN_CSR din[5]=1, then wait for suspend_end.
- Bus/CDC integration
- Keep WB and PHY clocks related where possible; otherwise add synchronizers around wb_ack crossing.
- Honor classic WB handshake: hold CYC/STB and address/data stable until wb_ack_o.
- Memory/PHY hardware
- Choose SSRAM that supports permanent read-enable; verify IO direction and timing for sram_data_i/o.
- Align all MEM accesses and descriptors to 32-bit words; misalignment will corrupt data.
- Throughput and scheduling
- Schedule bulk CPU MEM operations when USB is idle; expect W-side wack to toggle every other phy_clk at best when M is idle.
- Use FRM_NAT for timing diagnostics and to align IN scheduling with (micro)frame boundaries.
- UTMI linkage
- Verify SuspendM polarity and control pins (XcvSelect/TermSel/OpMode) with the chosen PHY. drive_k should yield K signaling during resume/chirp.

Debug/observability tips
- MAIN_CSR exposes suspend/mode_hs/attached/LineState; RF Word 3 logs system events (read-to-clear).
- UTMI vendor interface (Word 5) allows simple sideband pokes/reads for PHY diagnostics.

- Error injection validation: pid_cs_err, crc5_err (RF), per-EP CRC16/timeout/upid/out_to_small in EPx_INT.