Generated Specification Document

INTRODUCTION

-- Overview --

wbqspiflash is a Wishbone-controlled SPI/Quad■SPI NOR flash controller that presents a 32■bit memory■mapped data port alongside a compact control space. It bridges a 20■bit word■addressed Wishbone bus to a serial flash via an embedded llqspi shift engine, handling command sequencing, address formation (byte address derived from the word address), data byte reordering on reads, and automatic busy (WIP) polling. Reads use fast■read commands (0x0B in SPI or 0xEB in Quad■SPI) with required dummy cycles and keep chip■select asserted to stream sequential words for high throughput. Writes issue WREN followed by page■program (0x02 SPI or 0x32 Quad) and chain beats within the same 256■byte page; sector erase (0xD8) is available via control space. The core enforces safe interlocks by mirroring SR[0] (WIP), deferring conflicting operations, and optionally generating an interrupt when background program/erase completes. Management features include a software write■protect latch, tracking of the last erased sector and whether it has been written, and a brief reset■time pin override to place the flash in a known state before handing control to llqspi. Control space provides direct access to status/configuration (including quad mode via CR[1]), JEDEC ID, and consolidated status/control functions. llqspi supplies low■level 1■bit/4■bit signaling, continuous CS for bursts, and back■to■back word chaining for efficient transfers.

-- Key Features --

- Wishbone-controlled SPI/Quad-SPI flash controller with separate data/control address spaces; stall/ack handshake and burst support with CS hold for high throughput.
- 32-bit data path with 20-bit word addressing ({addr, 2'b00} byte address); bytes reordered to present natural 32-bit word order.
- Dual serial modes: SPI (1-1-1) and Quad-SPI (1-4-4), auto-tracks quad enable via Configuration Register CR[1].
- High-throughput streaming reads: Fast Read 0x0B (SPI) and 0xEB (Quad), full address + dummy/mode phases; maintains CS across sequential words.
- Page Program writes: WREN then 0x02 (SPI) or 0x32 (Quad); chains within same 256-byte page; enforces page boundary; tracks write_in_progress and dirty_sector.
- Sector Erase control: software-triggered via control address 00 (bit31=1); sector select in bits [19:14]; issues WREN + 0xD8; records erased_sector; clears dirty_sector.
- Comprehensive control/status space: read/write Status (0x05/0x01) and Configuration (0x35/0x01), optional 0x30 to clear protection; JEDEC ID (0x9F); exposes WIP, quad mode, busy, write-protect, and sector flags.
- Robust busy/WIP management: transparent RDSR polling; o_wb_stall while device is busy; o_wb_ack on completion; background o_interrupt when WIP clears with no active WB cycle.
- Software write-protect latch: safely acks program/erase requests without executing when enabled.
- Startup override sequence to place flash in a known state before llqspi operations.
- llqspi submodule: compact SPI/QSPI shifter supporting 1-bit/4-bit widths, word chaining without CS deassertion, CS-hold idle, per-word o_valid, speed selection, and quad direction control; forwards CS/SCK/data/mode pins.

- On-the-wire command set: 0x06 WREN, 0x0B/0xEB Fast Read, 0x02/0x32 Page Program, 0x05 RDSR, 0x35 Read Config, 0x01 Write Status/Config, 0x9F JEDEC ID, 0xD8 Sector Erase, 0x30 Clear Protection.

-- Supported Commands and Modes --

Supported wire-level command set (opcode → function):
- 0x06: Write Enable (WREN). Required before WRR, Page Program (0x02/0x32), and Sector Erase (0xD8).
- 0x02: Page Program over SPI (1-1-1). Used when quad mode is disabled.
- 0x32: Quad Page Program (1-4-4). Used when quad mode is enabled.
- 0x0B: Fast Read over SPI (1-1-1). 24-bit address + 1 dummy byte; streams data.
- 0xEB: Quad Fast Read (1-4-4). 24-bit address + device-specific mode/dummy cycles; streams data.
- 0x05: Read Status Register (SR1). Polled for WIP and status bits.
- 0x35: Read Configuration Register (CR). Used to detect/manage quad enable (CR[1]).
- 0x01: Write Status/Configuration (WRR). One-byte SR1 write or two-byte SR1+CR write.
- 0x9F: Read JEDEC ID (32-bit).
- 0xD8: Sector Erase. Target sector selected via control write bits [19:14].
- 0x30: Auxiliary command to clear protection bits (conditionally issued when needed).

Supported modes and behaviors:
- Bus width modes: SPI (1-bit) and Quad-SPI (4-bit). Quad enable mirrors CR[1].
- Read paths: 1-1-1 (0x0B) when quad disabled; 1-4-4 (0xEB) when quad enabled. Controller inserts required mode/dummy cycles for 0xEB.
- Write paths: 1-1-1 (0x02) when quad disabled; 1-4-4 (0x32) when quad enabled.
- Addressing: 24-bit wire address formed as {i_wb_addr[19:0], 2'b00}.
- Control vs data spaces: i_wb_ctrl_stb selects status/config/ID/erase; i_wb_data_stb selects array read/program.
- Streaming/continuous CS: Holds CS low across sequential word bursts for reads and programs; program bursts are constrained to a single 256-byte page (i_wb_addr[19:6]).
- Busy/WIP gating: Operations that require readiness stall or transparently poll SR1[0] (WIP) until clear; background polling when idle, with an interrupt pulse on completion.
- Software write-protect mode: Latch causes program/erase requests to acknowledge without executing.
- Startup override: Brief direct control of CS/SCK at reset to force the flash into a known state.
- llqspi interface features used: i_spd (SPI/Quad select), i_dir (quad I/O direction), i_hold (keep CS asserted), and variable transfer lengths via i_len.

-- Addressing and Data Alignment --

- Address space and unit size:
- Wishbone address i_wb_addr[19:0] is a 20-bit word index into the flash array.
- Each word is 32 bits (4 bytes); all data-space accesses are 32-bit word aligned and sized.
- Flash bus byte address:
- The on-wire flash address is the word address shifted left by two: {i_wb_addr[19:0], 2'b00}.
- Lower two address bits are always zero, guaranteeing 4-byte alignment on all reads/writes.
- Control-space addressing:
- When i_wb_ctrl_stb is asserted, only i_wb_addr[1:0] is decoded:
- 00: status/control
- 01: configuration
- 10: status register
- 11: JEDEC ID

- Higher address bits are ignored in control mode.
- Read/program command address phase:
- SPI (0x0B) and QSPI (0xEB) fast-read, and SPI (0x02)/QSPI (0x32) page-program commands use the aligned byte address derived from i_wb_addr << 2.
- Dummy/mode cycles in fast-read do not affect 32-bit word alignment of returned data.
- Data ordering:
- Wishbone data is always presented as natural 32-bit words.
- The controller assembles/disassembles bytes internally so SPI/QSPI bit/nibble shift order does not affect the Wishbone view.
- Burst and chip-select behavior:
- Sequential streaming is based on contiguous word addresses.
- If the next request equals the previous word address + 1, CS remains asserted and data continues without readdressing.
- Page-program alignment rule:
- Page size is 256 bytes (64 words); the page index is i_wb_addr[19:6].
- Program operations must not cross a page boundary; crossing i_wb_addr[19:6] terminates the program and requires a new transaction.
- Sector erase addressing:
- For erase via control address 00, the target sector number is provided in control write data[19:14].
- The controller maps this field to the proper flash address for the SE command and mirrors it into erased_sector[5:0].
- Partial writes:
- Sub-word write enables are not supported; only full 32-bit word writes are valid.
- Transport considerations:
- llqspi supports various transfer lengths (8/16/24/32 bits), but alignment and addressing policies are enforced by wbqspiflash as described above.

-- Performance and Burst Considerations --

- Data throughput modes: SPI fast read (1-1-1, 0x0B) vs Quad fast read (1-4-4, 0xEB). Quad provides ~4x data-phase throughput; setup overhead (command/address/dummy) is reduced or amortized in quad.
- Wishbone burst behavior: o_wb_stall=1 during setup/shift; o_wb_ack pulses once per completed 32-bit beat. Maximum throughput when requests use sequential word addresses (next == prev+1).
- Chip-select hold and chaining: When the next word is queued at the word boundary, CS remains asserted (spi_hold=1) for zero-gap chaining. If a burst pauses but i_hold remains set, CS can stay low to resume without full teardown.
- Read bursts: After initial latency (cmd+addr+dummy), data streams one 32-bit word per beat. Quad mode streams 4 bits/clk; SPI streams 1 bit/clk. Non-sequential addresses end the burst (CS released) and force a new setup.
- Write (program) bursts: Each burst requires WREN (0x06) then PP (SPI 0x02, Quad 0x32) with a 24-bit byte address. Streaming continues only within the same 256-byte page (i_wb_addr[19:6] == spif_addr[19:6]) and sequential addresses; max 64 words (256 bytes) per burst. Crossing a page boundary terminates the burst and requires a new WREN+PP sequence. Quad PP materially increases data-phase throughput.
- Busy/WIP effects: If SR[0] (WIP)=1, the controller polls RDSR (0x05), stalls WB, and breaks bursts until ready. Erase/status/config writes block data bursts; background polling clears WIP and can assert o_interrupt. A write_protect latch causes program/erase requests to ack without executing.
- Addressing and sequencing: 32-bit word bus; wire byte address is {i_wb_addr[19:0], 2'b00}. Sequential streaming condition is next addr == previous+1. Write-page boundary check uses i_wb_addr[19:6].
- Practical performance guidance: Enable quad mode (CR[1]) for long reads/programs; align write

bursts to page boundaries and keep within one page; present the next WB beat promptly to maintain CS hold and avoid gaps; avoid initiating large bursts while WIP is set; plan around long-latency operations like sector erase (0xD8) and use the interrupt to schedule follow-on traffic.

-- Limitations and Assumptions --

- Assumes JEDEC-compliant SPI NOR flash behavior and opcode set: 0x06 WREN, 0x02 Page Program, 0x32 Quad Page Program, 0x0B Fast Read, 0xEB Fast Read (1-4-4), 0x05 Read Status, 0x35 Read Configuration, 0x01 Write Status/Configuration, 0x9F JEDEC ID, 0xD8 Sector Erase.
- Assumes Status Register SR[0] is WIP and Configuration Register CR[1] enables Quad mode; WRR (0x01) register write semantics follow standard conventions.
- Uses 24-bit addressing on the wire; internally limits accessible range to ~4 MB via a 22-bit window (i_wb_addr[19:0]<<2). No support for 4-byte (32-bit) addresses; devices >16 MB are only partially accessible.
- Page size fixed at 256 bytes; erase supported only for 64 KB sectors (0xD8). No 4 KB subsector, other block erase variants, or chip erase.
- Quad read path fixed to 1-4-4 (0xEB) with expected mode/dummy cycles; no 4-4-4 or vendor-specific fast-read variants.
- Write/program operations can be blocked by a software write-protect latch; requests are acknowledged but not executed when enabled.
- WIP handling via polling RDSR (0x05) using only SR[0]; no timeouts and limited error reporting (other status/error bits not decoded).
- Minimal state tracking: only the most recently erased sector (6-bit index) and a single dirty flag are maintained; no global wear-leveling or per-sector metadata.
- Program bursts must not cross 256-byte page boundaries (enforced by address page match); cross-page chaining is unsupported.
- CS hold during streaming occurs only for strictly contiguous word accesses; non-sequential accesses release CS.
- Wishbone interface supports a single outstanding request; o_wb_stall asserts while the SPI engine/controller is busy, with variable latency and ack on completion. External arbitration is required for multiple masters.
- Control-space writes that start long operations (e.g., sector erase) provide no progress feedback beyond write_in_progress; completion is detected via status polling or an idle-only interrupt pulse.
- JEDEC ID write access is a no-op (ack without effect).
- Data words are 32-bit; byte ordering is internally reordered so reads return natural word order (not raw wire order).
- Startup spif_override assumes toggling CS/SCK returns the device to standard SPI mode; parts requiring explicit reset/exit-QPI commands may not recover.
- Timing/IO assumptions: SCK idles high (CPOL=1), no programmable SCK divider (SPI rate derives from system clock/FSM), IO1 is MISO in 1-bit mode, quad I/O direction controlled by o_mod, transfer lengths limited to 8/16/24/32 bits, and chaining/hold requires precise i_wr timing at word boundaries.
- Board-level IO buffering and tri-state control must match these IO assumptions.

-- Terminology and Abbreviations --

WB (Wishbone): On-chip bus interface; key signals i_wb_addr, i_wb_we, i_wb_data_stb, i_wb_ctrl_stb, o_wb_stall, o_wb_ack; supports data space (array access) and control space (status/config/ID/erase).
SPI: 1-bit serial interface using CS (active-low), SCK; MOSI/MISO data lines; CPOL=1 means clock idles high.
QSPI (Quad-SPI): 4-bit SPI using IO[3:0] for higher throughput; supports mixed-width sequences.
1-1-1 / 1-4-4: Bus width per phase (Command-Address-Data); 1-1-1 is single-bit throughout; 1-4-4 is 1-bit command then quad-bit address/data.
CS / CS_n: Chip Select, active-low; asserting and holding CS keeps the device selected across bursts.

SCK: Serial clock; idles high in this design.
IO[3:0]: Quad data lines; IO1 is MISO in 1-bit SPI mode.
MISO/MOSI: SPI input/output data lines; MISO maps to IO1 in SPI mode.
CPOL: Clock polarity; CPOL=1 indicates SCK idles high.
WIP: Write-In-Progress status bit (SR[0]); set while program/erase is active.
WREN (0x06): Write Enable command required before status/config writes and program/erase.
RDSR (0x05): Read Status Register (SR1) command; returns bits including WIP.
RDCR (0x35): Read Configuration Register command; includes quad-mode enable (CR[1]).
WRR (0x01): Write Status/Configuration Register command; writes SR/CR.
SE (0xD8): Sector Erase command; erases sector selected by high address bits.
PP (0x02): Page Program in SPI mode (1-bit).
QPP (0x32): Quad Page Program in QSPI mode (4-bit).
FR (0x0B): Fast Read (SPI 1-1-1) with dummy byte.
QFR (0xEB): Quad Fast Read (1-4-4) with mode/dummy cycles.
JEDEC ID (0x9F): Read manufacturer/device ID command.
Dummy cycle/byte: Clocked gaps to meet timing (e.g., after fast read); no payload.
Mode byte: Defined pattern required during quad fast read sequences.
Page: 256-byte program granularity; writes cannot cross a page boundary.
Sector: Erase granularity; addressed via bits [19:14] in this design.
erased_sector[5:0]: Index of most recently erased sector.
dirty_sector: Flag indicating the erased sector has since been programmed.
write_protect: Software latch preventing program/erase when set.
quad_mode_enabled: Tracks CR[1]; selects SPI vs QSPI paths.
o_interrupt: Pulse generated when background polling finds WIP cleared and no active WB request.
spif_req: Latched descriptor of the current SPI/QSPI engine request.
spi_hold: Control to keep CS asserted across contiguous bursts.
spif_addr: Latched flash word address used for chaining and boundary checks.
i_len: Transfer length code to llqspi (00/01/10/11 => 8/16/24/32 bits).
i_spd: llqspi bus width select; 0=SPI, 1=QSPI.
i_dir: llqspi data direction indicator in quad mode; reflected on o_mod[0].
i_hold: llqspi request to hold CS low after a word when no immediate chain.
i_wr: llqspi strobe to start or chain a transfer.
o_busy: llqspi busy flag; high while shifting; low when idle or in hold.
o_valid: One-cycle pulse when a 32-bit receive word is ready.
o_word: 32-bit received word from llqspi.
o_dat[3:0]: llqspi driven data on quad bus lines.
o_mod[1:0]: llqspi bus mode; bit1=1 in quad mode; bit0=i_dir in quad; 00 when idle.
Chaining: Starting the next word immediately without releasing CS.
Hold: Keeping CS asserted after a word to allow later chaining.
spif_override: Temporary control of SPI pins at reset to force/synchronize device state.
Byte reordering: Internal reordering so host sees natural 32-bit word order.
Word addressing: i_wb_addr[19:0] index 32-bit words; byte address is {i_wb_addr, 2'b00}.
Data streaming: Sequential word bursts (addr == spif_addr+1) maintain CS for throughput.
Page boundary rule: Program bursts enforced to remain within i_wb_addr[19:6] == spif_addr[19:6].
Status path: Controller mirrors SR[0] into write_in_progress and may auto-poll RDSR.
Control addr map: i_wb_addr[1:0] selects 00=status/control, 01=config, 10=status register, 11=JEDEC ID.
Erase trigger: Control write at addr 00 with bit31=1 and sector in [19:14], gated by write_protect and WIP.
Quad enable: quad_mode_enabled updated via RDCR reads and WRR writes (CR[1]).
Protection clear: Optional 0x30 command to clear certain status-protect bits after WRR.
llqspi timing: SCK idles high; input sampled per bit (SPI) or nibble (QSPI); o_valid at word end.

llqspi length: spi_len = (i_len<<3)+8; decrements by 1 per SPI bit or 4 per QSPI nibble.
MISO mapping: In SPI mode, input is IO1.
Override at reset: spif_override drives CS/SCK pattern to synchronize flash before normal operation.
Dummy/mode cycles: Inserted automatically to meet fast read requirements.
Bus handshake: o_wb_stall asserts while operations run; o_wb_ack pulses on completion.
Byte order: Host-facing data uses natural 32-bit endianness despite serial shift order.

IO PORTS

-- Clock and Reset --

A single system clock drives both wbqspiflash and its llqspi submodule; all controller state machines, Wishbone handshakes (stall/ack), and SPI/QSPI transfers are synchronous to this clock. llqspi derives SPI SCK from the system clock via an internal generator; SCK idles high (CPOL=1) and all SPI pins transition only on system-clock edges. There are no internal clock-domain crossings.

Reset places the design into a known idle state. Reset aborts any in-flight Wishbone or SPI transaction, clears controller and llqspi state, deasserts busy/valid/ack and stall, and returns external SPI pins to CS=high, SCK=high. Immediately after reset, a brief spif_override sequence directly toggles CS/SCK to wake/synchronize the flash, then control returns to llqspi. All status/configuration tracking and request latches are cleared to safe defaults (e.g., write_in_progress=0, write_protect=0, quad_mode_enabled=0, erased_sector=0, dirty_sector=0) and are refreshed/updated by subsequent control reads/writes. The module comes up idle with no outstanding bus or SPI activity.

-- Host Bus Interface (Wishbone) Signals --

Wishbone-like host interface (word-wide, custom strobes; no CYC/STB/SEL/ERR):

Inputs
- i_wb_addr[19:0]: 32-bit word address. For array access, the flash byte address is {i_wb_addr, 2'b00}. In control space, i_wb_addr[1:0] selects the sub-register.
- i_wb_data[31:0]: Write data for data-space and control-space writes.
- i_wb_we: Write enable. 0 = read, 1 = write.
- i_wb_data_stb: Data-space request strobe (array reads/programs). Assert exactly one strobe per request.
- i_wb_ctrl_stb: Control/status-space request strobe (status/config/ID/erase). Sub-addressed by i_wb_addr[1:0].

Outputs
- o_wb_data[31:0]: Read data. Bytes are reordered to present natural 32-bit word order regardless of on-wire shift order. Valid only when o_wb_ack=1 on a read.
- o_wb_ack: One-cycle pulse acknowledging the current request at the bus level. For reads, pulses when the returned word/control data is available. For writes (program/erase/control), pulses when the command/data for the beat has been issued—final flash completion may occur later.
- o_wb_stall: Back-pressure. 1 indicates the core cannot accept/finish the present request (engine busy, polling WIP, or chaining constraints); hold inputs and the strobe steady until it returns 0. 0 indicates the request can be accepted/serviced.

- o_interrupt: Background completion pulse when an in-progress program/erase finishes (WIP 1->0) and no active WB cycle is being serviced.

Usage and handshake
- Only one of i_wb_data_stb or i_wb_ctrl_stb must be asserted per request. Present i_wb_addr, i_wb_we, i_wb_data (if write), and the chosen strobe. If o_wb_stall=0, the request is accepted that cycle; otherwise, keep the request stable until o_wb_stall deasserts. The core latches the request and later pulses o_wb_ack when the beat completes.
- Sequential data-space reads acknowledge per word and may sustain back-to-back acks with CS held if the engine keeps pace. Sequential data-space writes may chain within the same 256-byte page only (i_wb_addr[19:6] equal); crossing a page boundary ends the burst.

Address map via i_wb_ctrl_stb (i_wb_addr[1:0])
- 00: Controller status/control. Read returns consolidated status (e.g., WIP, quad enable, write_protect). Write updates software write_protect and can request sector erase via bit31 with sector in data[19:14].
- 01: Configuration register (read via 0x35; write via WRR with status+config).
- 10: Status register (read via 0x05; write via WRR; optional 0x30 clear).
- 11: JEDEC ID (read via 0x9F; write is a no-op and is acknowledged immediately).

Operation-dependent behavior
- Reads (i_wb_we=0): Data-space reads start a fast read (SPI 1-1-1 or Quad 1-4-4 when enabled). If flash is busy (WIP=1), the core transparently polls RDSR (0x05) and stalls until ready. Control-space reads issue the respective command and stall as needed until data is valid.
- Writes (i_wb_we=1): Data-space programs require write_protect=0 and WIP=0. The core issues WREN then programs one word (0x02 SPI or 0x32 Quad). o_wb_ack follows command/data launch; completion is reported later via control-space status or o_interrupt. Control-space writes update status/config or trigger erase; if write_protect=1, these are acknowledged without effect. If the device is busy, requests stall until launch is possible.

Restrictions and assumptions
- No byte enables; all accesses are 32-bit word-aligned. Do not assert both strobes simultaneously.
- The interface does not provide o_wb_err; error conditions are not signaled via the bus.
- o_wb_data is meaningful only on read cycles when o_wb_ack=1.

-- Data vs Control Space Selection --

Accesses are explicitly partitioned into two spaces, selected per-cycle by mutually exclusive Wishbone strobes: i_wb_data_stb selects the flash array data space (word reads/programs) and i_wb_ctrl_stb selects the control space (status/config/SR1/JEDEC). Exactly one strobe must be asserted per bus cycle; i_wb_we then chooses read (0) vs write (1) semantics within the selected space. Data space addressing uses i_wb_addr[19:0] as a 20-bit word index, mapped to the flash byte address by left-shifting two bits; sequential bursts are detected by i_wb_addr == previous+1, maintaining CS across reads, while program bursts are constrained to a single 256-byte page. Control space uses i_wb_addr[1:0] to select sub-registers: 00=Status/Control, 01=Configuration (CR), 10=Status (SR1), 11=JEDEC ID; higher address bits are ignored, with parameters (e.g., sector index for erase in data[19:14]) carried in the write data word when applicable. Handshake behavior applies uniformly: o_wb_stall=1 whenever the engine is busy or the flash indicates WIP; o_wb_ack pulses on completion. Data-space reads always perform a flash read (fast or quad based on quad_mode_enabled) and reorder bytes to present natural 32-bit words; data-space writes issue WREN + Page Program (single/quad) only if not write_protect and not WIP, otherwise they are acked without execution. Control-space accesses vary by sub-register: status/control reads may transparently poll RDSR until ready; configuration/status writes issue WREN/WRR sequences; JEDEC writes are no-ops that ack

immediately. The software write_protect flag suppresses program/erase requests in both spaces while allowing reads; quad_mode_enabled is maintained via control-space configuration and affects only data-space command selection.

-- Read/Write and Handshake Signals (WE/ACK/STALL) --

Request qualification: i_wb_data_stb selects the flash data space; i_wb_ctrl_stb selects the control/status space. Only one strobe should be asserted per request. i_wb_we=0 denotes a read; i_wb_we=1 denotes a write.
Back■pressure (o_wb_stall): o_wb_stall=1 means the controller cannot accept a new request. Causes include an active SPI/QSPI transfer (llqspi o_busy), device write■in■progress (WIP), and required status polling (e.g., RDSR gating reads/writes). o_wb_stall=0 means a request can be accepted in the current cycle.
Acceptance rule: A request is accepted on a cycle when its strobe is high and o_wb_stall=0; the address, i_wb_we, and data are latched. No new request is accepted while o_wb_stall=1. In streaming paths (sequential addresses), additional beats are accepted back■to■back when o_wb_stall remains 0.
Completion (o_wb_ack): Each accepted request generates exactly one o_wb_ack pulse, asserted for a single cycle. o_wb_ack never occurs while o_wb_stall=1.
Read timing: For reads (i_wb_we=0), o_wb_ack coincides with the cycle that returns o_wb_data (32■bit). Reads may hold o_wb_stall high until data is available (including while polling status for a current snapshot), then deassert stall and pulse ack with valid data. In streaming reads, sequential words return one ack per beat; stall remains low when throughput can be sustained.
Write/program/erase timing: For writes (i_wb_we=1), the controller acks after issuing the necessary SPI sequence (e.g., WREN + Program/Erase + address + data). The flash may remain busy afterward; WIP completion is tracked separately and does not delay o_wb_ack. Burst programming accepts sequential words within a page with one ack per beat; crossing a page boundary or any busy condition reasserts stall.
Protection and no■ops: If software write_protect is set, program/erase requests are acked as no■ops. JEDEC■ID writes are defined as no■ops and ack immediately. Control■space writes that simply update a latch may ack immediately; WRR (config/status) acks after the command/data are sent; sector erase acks after WREN+SE are issued or as a no■op if protection/busy prevents execution.
General rule: One ack per accepted request; stall enforces back■pressure until the controller can start or resume the operation; reads present data only in the ack cycle. No error signal is defined—disallowed operations are either stalled until valid or acked as no■ops.

-- Interrupts --

Interrupt output o_interrupt is an active-high, single i_clk–cycle pulse generated only by wbqspiflash when a flash write/program/erase operation completes. The pulse is emitted upon detection that the status register WIP bit (SR[0]) has transitioned from 1 to 0 via background RDSR (0x05) polling. Background polling and interrupt generation occur only while the Wishbone bus is idle; if a Wishbone transaction is active when WIP clears, no interrupt is produced and completion must be observed via normal WB acknowledgments/status reads. Operations that can assert WIP (and thus later cause an interrupt) include page program (0x02/0x32), sector erase (0xD8), status/configuration writes (WRR 0x01, one or two bytes), and protection-clear commands (e.g., 0x30). No interrupts are generated for any read operations. There is no interrupt enable/mask/clear register; the pulse is edge-only and not latched—missing the edge is not retained. If the software write_protect latch is set, such operations are blocked, WIP never asserts, and no interrupt will occur. The pulse is synchronous to i_clk; systems requiring longer hold must externally capture/extend it. llqspi provides no interrupt mechanism.

-- External SPI/QSPI Interface Pins --

- Purpose: Presents a standard SPI (1-bit) or QSPI (4-bit) flash interface driven by llqspi under wbqspiflash control.
- Pins (logical):
- CS_N (output): Active-low chip select. May remain low across back-to-back words for streaming; otherwise deasserts between operations. May be directly driven during a brief reset/startup override.
- SCK (output): SPI/QSPI clock. Idles high (CPOL=1). Toggles once per bit in SPI mode and once per nibble in QSPI mode. Data is presented/sampled on the low half-cycle.
- IO[3:0] (bidirectional): Quad data bus. When the controller transmits, lines are driven by o_dat[3:0]; when the flash responds, lines are sampled as i_dat[3:0].
- MOD[1:0] (output, pad-direction hint):
- MOD[1]=0: SPI (1-bit) mode; MOD[1]=1: QSPI (4-bit) mode.
- MOD[0]=0: Controller drives IO[3:0] (write/command). MOD[0]=1: Flash drives IO[3:0] (read) in quad mode.
- SPI (1-bit) mode:
- Input is sampled from IO1 (MISO).
- Transmit bit is encoded into o_dat[3:0]; the pad wrapper must map this to the physical MOSI (typically IO0) and hold IO1 as input. Unused IOs should be high-Z/safe.
- QSPI (4-bit) mode:
- All IO[3:0] active per SCK unit (nibble transfers). Use MOD[0] to control line direction (tri-state controller outputs for reads; drive for writes).
- Reset/idle behavior:
- On reset/idle: CS_N=1, SCK=1 (idle high), o_dat=4'hD (benign), MOD=2'b00.
- Startup override may temporarily bypass llqspi to toggle CS_N/SCK to return the flash to a known state; pad logic must not contend with this.
- Integration/timing notes:
- wbqspiflash forwards llqspi's o_cs_n, o_sck, o_dat[3:0], o_mod[1:0], and receives i_dat[3:0], except during the reset override.
- Pads/I/O buffers should implement direction per MOD and allow continuous CS_N assertion during bursts without forcing deassertion.
- Assume CPOL=1 timing: present and sample data on the low half of SCK, with SCK high when idle.

-- Mode/Direction and Quad Enable Signals --

- Purpose: Define how bus width (SPI vs Quad) and IO direction are selected and exposed, based on the flash Quad Enable (QE) bit.
- Quad Enable tracking (QE):
- quad_mode_enabled mirrors the flash Configuration Register bit CR[1] (QE).
- Updated on control-space operations:
- 0x35 Read Configuration latches CR and sets quad_mode_enabled = CR[1].
- 0x01 Write Status+Configuration (after 0x06 WREN) updates quad_mode_enabled = Config[1] once the write is accepted by the flash.
- If write_protect=1 or the flash is busy (SR[0] WIP=1), QE cannot be changed; controller remains in 1-bit SPI until QE is observed set.
- Mode (bus width) selection:
- i_spd = 0 selects classic 1-bit SPI; i_spd = 1 selects quad-wide transfers for eligible phases.
- Controller automatically sets i_spd per operation and phase:
- Reads: 0x0B fast read uses i_spd=0 (1-1-1). 0xEB fast read uses i_spd=1 during quad phases (1-4-4 style).
- Programs: 0x02 page program uses i_spd=0. 0x32 quad page program uses i_spd=1 during the data phase.
- quad_mode_enabled is the sole gate: when 0, all operations use i_spd=0; when 1, quad-capable operations use i_spd=1 where applicable.

- Direction (quad mode only):
- i_dir indicates quad IO direction (valid only when i_spd=1):
- i_dir = 0: controller drives the IO bus (command/address/mode/dummy, and program data phases).
- i_dir = 1: controller samples from the IO bus (read data phase).
- In 1-bit SPI (i_spd=0), direction is implicit: controller drives MOSI and samples MISO on IO1.
- Mode/Direction outputs to the IO wrapper:
- o_mod[1:0] encodes current width and direction for external IO buffers:
- o_mod[1] = 1 when quad width is active (i_spd=1); 0 in 1-bit SPI or idle.
- o_mod[0] = i_dir when in quad; 0 otherwise.
- External wrapper should use o_mod to switch IOBUF direction: {1,0} = drive on all four IOs; {1,1} = receive on all four IOs; {0,0} = 1-bit SPI (IO1=MISO).
- In 1-bit SPI, o_dat is formatted so the single valid input bit is on IO1 (MISO); other lanes carry a known pattern.
- Automatic fallback and gating:
- If quad_mode_enabled=0, or writes are blocked (write_protect=1) or pending (WIP=1), controller performs all accesses via 1-bit SPI until QE is confirmed set.
- Idle and phase transition behavior:
- o_mod returns to 2'b00 when idle or upon CS deassertion; o_cs_n goes high and o_sck idles high.
- During continuous CS-low transactions, o_mod may change per phase (e.g., {1,0} while driving cmd/addr, then {1,1} during read data).
- Operation examples (for clarity):
- Read: 0x0B => i_spd=0 throughout. 0xEB => command possibly on 1-bit, quad phases with i_spd=1; i_dir=1 only during data-in.
- Program: 0x02 => i_spd=0, drive only. 0x32 => i_spd=1 during data, i_dir=0.

-- Startup Override/Bypass Signals --

On every reset, an internal, non-software-controlled signal (spif_override) briefly takes ownership of the external flash pins to guarantee a clean startup state. While spif_override is asserted, wbqspiflash bypasses the llqspi engine and directly drives o_cs_n, o_sck, and, as needed, o_dat[3:0] and o_mod to generate a controlled CS/SCK pattern that exits any lingering continuous-read or quad-mode contexts and ensures the device is in standard 1-bit SPI idle. During this window, all llqspi outputs are ignored/not forwarded; external pins reflect only the override pattern. The sequence is implemented in the controller’s initial reset states, runs for a short, fixed duration, and has no register interface to trigger, extend, or disable it. After completion, spif_override deasserts and control cleanly returns to llqspi with benign defaults (CS high, SCK high, o_mod=00, data lines at safe idle), enabling normal Wishbone-driven operation. The override does not attempt to enable quad mode or perform configuration beyond ensuring a known idle state.

-- Signal Polarity and Levels --

Wishbone: i_wb_data_stb and i_wb_ctrl_stb are active-high strobes; i_wb_we high selects write (low=read); o_wb_stall high means not ready to accept a request; o_wb_ack is an active■high pulse signaling completion (o_wb_data valid when o_wb_ack=1); o_interrupt is an active■high pulse when a background write/erase completes. SPI/QSPI physical: o_cs_n is active■low (low=selected, high=idle); o_sck idles high (CPOL=1) and toggles low/high during transfers, with data presentation/shifting on the low half-cycle; o_dat[3:0] actively drives high onto IOs (SPI 1■bit uses a coded pattern carrying one bit per unit; quad uses a nibble per unit); i_dat[3:0] is sampled active■high (SPI uses i_dat[1] as MISO; quad samples all four bits per unit); o_mod[1:0] encodes bus width/direction: bit1=1 for quad, 0 for SPI; in quad, bit0 mirrors i_dir (IO direction), otherwise bit0=0; o_mod returns to 2'b00 when idle or CS released. llqspi local: i_wr is an active■high strobe; i_spd=0 selects SPI, i_spd=1 selects Quad; i_dir is active (meaningful) only in quad and forwarded via o_mod[0]; i_hold active■high keeps CS asserted

between words; o_busy is active■high while a transfer or teardown is in progress; o_valid is an active■high one■clock pulse when o_word updates. Controller/status flags: write_in_progress high mirrors SR[0]=1 (flash busy); spi_busy high means SPI engine active; quad_mode_enabled high means CR[1]=1; write_protect high blocks program/erase (status read reports ~write_protect so 1=unprotected); dirty_sector high indicates a post■erase sector has been written. Reset/idle defaults: o_cs_n=1 (deasserted), o_sck=1 (idle high), o_mod=2'b00, o_busy=0, o_valid=0, and o_dat=0xD; o_cs_n remains low during streaming when i_hold=1; an internal spif_override is active■high briefly after reset to directly drive CS/SCK and produce a known clock pattern before handing control to llqspi.

ARCHITECTURE

-- Top-Level Block Diagram and Data Flow --

Top-Level Blocks and Connections
- Wishbone Interface: Accepts 32-bit word operations on two spaces (data, control) via i_wb_data_stb and i_wb_ctrl_stb, with i_wb_addr[19:0], i_wb_we, and returns o_wb_ack, o_wb_stall, o_wb_data, o_interrupt.
- Request Arbiter and Control FSM: Single-issue front-end that arbitrates data vs control requests, sequences flash command flows, tracks operation progress, and enforces stall/ack policy.
- Address and Status Registers: Latches request address, maps to 24-bit byte address {addr[19:0], 2'b00}; holds software write_protect, write_in_progress, quad_mode_enabled, erased_sector, dirty_sector, last_status.
- Sequencer and Command Generator: Builds opcode streams (WREN, READ, PP/QPP, RDSR, RDCR, WRR, SE, RDID), selects SPI vs QSPI width, dummy cycles, and controls chip-select hold.
- Data Path and Word Repacker: Serial-to-32-bit and 32-to-serial adapters that assemble read words and serialize write words; ensures natural 32-bit word ordering to/from o_wb_data.
- WIP Poller and Interrupt Logic: Issues background/foreground RDSR reads; mirrors SR[0] into write_in_progress; generates o_interrupt when WIP clears and bus is idle.
- Startup Override: Briefly drives CS/SCK to place the flash into a known state after reset, then releases control to the SPI engine.
- Low-Level SPI/Quad-SPI Engine (llqspi): Executes length- and width-programmed transfers with chip-select auto-hold; exposes o_busy, o_valid, o_word; drives o_cs_n, o_sck, o_dat[3:0], o_mod to the flash pins.

Primary Data Flow (Reads)
- WB master raises i_wb_data_stb with i_wb_we=0 and address.
- Arbiter latches address; if write_in_progress or engine busy, enters a polling or wait state and asserts o_wb_stall.
- When ready, Sequencer issues fast-read opcode and 24-bit address (0x0B SPI 1-1-1 with 1 dummy byte or 0xEB 1-4-4 with required dummy/mode cycles) to llqspi.
- llqspi streams return data; Data Path repacks bytes into 32-bit words. For each word received, FSM drives o_wb_data and pulses o_wb_ack.
- Sequential access (next address == last+1) keeps CS asserted (hold) and continues streaming; a gap or end breaks hold and deasserts CS.

Primary Data Flow (Programs/Writes)
- WB master raises i_wb_data_stb with i_wb_we=1 and address/data.
- If write_protect is set or WIP asserted, the FSM stalls or (for protected writes by policy) acks without

flash-side change.
- Otherwise Sequencer issues WREN (0x06), then PP/QPP (0x02 SPI or 0x32 QSPI) and 24-bit address, followed by the 32-bit data payload.
- Page chaining: Additional sequential words are accepted while i_wb_addr[19:6] is constant (within the 256-byte page). CS is held across the burst. Crossing the page boundary terminates the chain.
- FSM sets write_in_progress; background poller later detects WIP clear and may pulse o_interrupt.

Control-Space Flow (i_wb_ctrl_stb, addr[1:0])
- 00 Status/Control: Read returns a packed 32-bit status (software write_protect, quad_mode_enabled, write_in_progress, spi_busy, erased_sector, dirty_sector). Write can set write_protect and optionally request sector erase: WREN $\rightarrow$ SE (0xD8) for sector=data[19:14]; records erased_sector and clears dirty_sector.
- 01 Configuration Register: Read RDCR (0x35) and refresh quad_mode_enabled from CR[1]. Write WRR (0x01) with {last_status, Config}; sets write_in_progress and updates quad_mode_enabled from Config[1].
- 10 Status Register: Read RDSR (0x05); mirrors SR[0] into write_in_progress and updates last_status. Write WRR with a single status byte; may conditionally issue 0x30 to clear protection bits.
- 11 JEDEC ID: Read RDID (0x9F) returns 32-bit ID. Writes are NOP (immediate ack).

Flow Control and Timing
- o_wb_stall=1 while sequencing commands, during llqspi activity, or while polling for WIP clear; requests are single-issue.
- o_wb_ack pulses once per completed operation; on reads it coincides with valid o_wb_data. Some disallowed control/data writes may ack without executing on the flash side.
- CS Hold Policy: Enabled across sequential reads and page-confined writes to maximize throughput; disabled on discontinuities or required re-commanding.

External Interfaces
- Wishbone: i_wb_data_stb, i_wb_ctrl_stb, i_wb_we, i_wb_addr[19:0], i_wb_data (writes), o_wb_data (reads), o_wb_ack, o_wb_stall, o_interrupt.
- Flash: o_cs_n, o_sck, o_dat[3:0], o_mod from llqspi; supports SPI 1-1-1 and QSPI 1-4-4 transactions with programmable lengths and dummy cycles.

-- Request Latching and Control FSM --

Scope
- Defines how Wishbone requests are captured (latching) and how the centralized control FSM sequences, overlaps, and completes flash operations via the llqspi engine.

Request latching
- Acceptance: One request is accepted only when the FSM is idle or at defined stream-accept points; all others see o_wb_stall=1. If multiple strobes are asserted simultaneously, only one is latched; tie-breaking is implementation-defined but deterministic.
- Sources: i_wb_data_stb (memory space) and i_wb_ctrl_stb (control space). i_wb_we selects read vs write.
- Latched fields (spif_req):
- req_space: data vs control
- req_we: read(0)/write(1)
- req_addr_w: i_wb_addr[19:0] (word address); a 24-bit byte address is formed as {i_wb_addr, 2'b00} when issuing flash cycles
- req_wdata: i_wb_data (write payload)
- req_ctrl_subaddr: i_wb_addr[1:0] selects control function (00 status/control, 01 configuration, 10 status register, 11 JEDEC ID)

- Sequential recognition: For data-space operations, the FSM tracks the last serviced word address spif_addr. If the next accepted request targets spif_addr+1, it is treated as a sequential burst candidate.

Stall/ack protocol
- o_wb_stall=1 while the FSM is busy executing an operation or polling WIP (RDSR). It deasserts only when the FSM is able to accept a new request (idle or approved stream-accept point).
- o_wb_ack pulses once per accepted request when its operation completes:
- Data read: when a 32-bit word is available in natural order
- Data write (program): after issuing WREN and the required program sequence for the beat
- Control: when the command sequence ends; some writes that are blocked by software write-protect may ack immediately without issuing flash commands

Control FSM: high-level states and flow
- Reset/override: On reset, asserts a brief spif_override to initialize/clear the flash interface, then relinquishes to normal llqspi operation.
- Idle: o_wb_stall=0; waits for a request. If write_in_progress=1 and no WB activity, may transition to background WIP polling.
- Decode: Classifies the latched request by space, direction, and (for control) subaddress, checks gating conditions (write_protect, write_in_progress), and selects SPI vs Quad based on quad_mode_enabled.
- Execute (control space):
- 00 Status/control: Read returns packed status; if WIP, transparently issues RDSR until SR[0]==0. Write: bit28 sets software write_protect; bit31 requests sector erase—issues WREN then SE (0xD8) if permitted, records erased_sector, clears dirty_sector.
- 01 Configuration: Read issues 0x35 and updates quad_mode_enabled<=CR[1]. Write issues WREN then WRR (0x01) with {Status, Config}; sets write_in_progress=1; updates quad_mode_enabled<=Config[1].
- 10 Status register: Read issues RDSR (0x05), updates last_status and write_in_progress. Write issues WREN then WRR (status byte); device-specific protection clearing may follow.
- 11 JEDEC ID: Read issues 0x9F and returns 32-bit ID; write is a no-op and acks immediately.
- Execute (data read): If not WIP, starts a fast-read sequence:
- SPI 1-1-1: 0x0B + 24b addr + 1 dummy byte
- Quad 1-4-4: 0xEB + 24b addr + mode/dummy; i_spd=1
Enters stream loop to deliver one 32-bit word per beat.
- Execute (data write): Requires not write_protect and not WIP. Issues WREN then Page Program (0x02 SPI or 0x32 Quad), sends 24b addr, then the 32-bit data. Sets write_in_progress=1.
- Stream/burst handling: During data reads and page programs, if the next accepted data request is sequential and (for program) remains within the current 256-byte page (i_wb_addr[19:6]==spif_addr[19:6]), the FSM asserts spi_hold/i_hold to keep CS low and chains transfers without teardown; otherwise, it releases CS and ends the stream.
- Polling/teardown: For any operation that sets or observes WIP, the FSM polls RDSR (0x05) as needed. Upon operation completion, it updates internal mirrors (last_status, write_in_progress) and asserts o_wb_ack. CS is released at the end of a non-streamed sequence or when a stream breaks.
- Background completion/interrupt: When idle and write_in_progress=1, the FSM enters a periodic RDSR poll state; on detecting WIP clear (SR[0]==0), it pulses o_interrupt and returns to Idle.

llqspi coordination
- The FSM sequences each high-level command into chained llqspi segments by driving:
- i_word: opcode, address, and data words in order
- i_len: per-segment bit lengths (8/16/24/32)
- i_spd: 0 for SPI, 1 for Quad
- i_dir: data direction in quad phase (read/write)

- i_hold: keeps CS asserted across chained segments and during qualified stream bursts
- Progression is synchronized to llqspi handshakes: advance on o_busy/o_valid boundaries; release CS when a transaction ends or when a stream cannot continue.

State-dependent updates and gating
- write_in_progress mirrors SR[0]; blocks program/erase while set and governs polling.
- quad_mode_enabled mirrors CR[1]; selects fast-read/program mode.
- spif_addr is updated on each serviced data beat; used for burst recognition and page-boundary enforcement.
- write_protect (software) blocks program/erase; such writes ack without issuing flash commands.

-- Transmit/Receive Path and Byte Reordering --

Wishbone presents 32-bit words in natural little-endian order, while SPI/QSPI flashes transmit and receive bytes MSB-first on the wire. wbqspiflash bridges this by deterministically reordering byte lanes at every 32-bit data beat boundary, so software always sees ascending memory bytes in little-endian order on the Wishbone side.

Transmit path (Wishbone to flash):
- Data writes (page program):
- The 24-bit byte address on the wire is {i_wb_addr[19:0], 2'b00}.
- The controller selects 0x02 (SPI) or 0x32 (QSPI) after WREN, then sends the address MSB-first per flash protocol.
- For every 32-bit Wishbone word, byte lanes are reordered so the wire emits bytes in ascending byte address order: b0=i_wb_data[7:0], b1=i_wb_data[15:8], b2=i_wb_data[23:16], b3=i_wb_data[31:24]; each byte is shifted MSB-first on the wire.
- When bursting sequential words within the same 256-byte page, CS may be held low (spi_hold) and words are chained; CS is released or a new command issued if the next word would cross the page boundary or is non-sequential.
- Control/command transmissions (opcodes, status/config writes, erase):
- Opcode, address, mode, and dummy fields are packetized into one or more llqspi transfers with i_len=8/16/24/32, i_spd indicating SPI(0)/Quad(1), i_dir for quad write/read, and optional i_hold. These non-data fields follow the flash’s MSB-first convention directly with no byte-lane reordering.

Receive path (flash to Wishbone):
- Data reads:
- Fast read uses 0x0B (SPI) or 0xEB (QSPI) with a 24-bit address followed by required mode/dummy cycles. For 1-4-4 reads, the sequence is: opcode on SPI output, address/mode/dummy on quad output, then data on quad input.
- llqspi returns one 32-bit word per beat with o_valid. Because the wire delivers bytes MSB-first, each received beat is reordered so o_wb_data[7:0] is the lowest byte address and o_wb_data[31:24] is the highest, preserving natural little-endian and ascending-address semantics for software.
- Streaming reads may hold CS low across sequential words; reordering is performed independently on each 32-bit beat.
- Control/status reads (e.g., RDSR 0x05, Read Config 0x35, JEDEC ID 0x9F):
- Returns are assembled from 8/16/32-bit wire transfers. Where a 32-bit value is returned (e.g., JEDEC ID), bytes are presented to software in natural order; no additional data-lane reordering is applied beyond the protocol’s MSB-first assembly.

llqspi interaction:
- Each wire transfer is initiated with i_wr and configured by: i_word (32-bit payload), i_len (8/16/24/32), i_spd (SPI=0, Quad=1), i_dir (quad write/read), and i_hold (keep CS asserted across transfers). llqspi

shifts MSB-first per unit (bit in SPI, nibble in quad) and asserts o_valid for one cycle when a 32-bit receive word is ready. wbqspiflash sequences these transfers to form higher-level commands (opcode, address, dummy/mode, data) and applies byte-lane reordering only at the 32-bit data beat boundary.

Byte reordering rules:
- Scope: Applied to every 32-bit data beat for both reads and writes; never applied to 8/16/24-bit control, address, mode, or dummy fields.
- Transmit (Wishbone -> wire): For each 32-bit i_wb_data, emit bytes in ascending address order b0=i_wb_data[7:0], b1=i_wb_data[15:8], b2=i_wb_data[23:16], b3=i_wb_data[31:24], each shifted MSB-first.
- Receive (wire -> Wishbone): Map the lowest-address byte of the beat to o_wb_data[7:0] and the highest-address byte to o_wb_data[31:24], yielding little-endian, ascending-address words to software.
- Streaming: Reordering is performed per beat; it does not modify page-program constraints. Sequential Wishbone word addresses map to contiguous 4-byte regions at byte address {i_wb_addr, 2'b00}.

-- Address Mapping and Word-to-Byte Translation --

- Wishbone address model: i_wb_addr is a 20-bit word address (32-bit words). Each +1 in i_wb_addr advances +4 bytes in flash, yielding a 4 MB byte-addressable array.
- Byte address computation: For array read/program/erase commands, the flash byte address B is B = i_wb_addr << 2. The 24-bit on-wire address field is A[23:0] = {2'b00, i_wb_addr[19:0], 2'b00}. Upper two bits are zero since only 4 MB are addressed.
- Data vs control space: i_wb_data_stb selects the flash array path and uses the above translation. i_wb_ctrl_stb selects control registers via i_wb_addr[1:0] and does not use array address translation (non-addressed commands send no address field).
- Page mapping and limits: Pages are 256 bytes (64 words). Page index = i_wb_addr[19:6]; word-in-page = i_wb_addr[5:0]. Equivalently, B[23:8] encode the page, B[7:2] the word offset within the page, and B[1:0]=2'b00. Read streaming may cross pages; program streaming is constrained to a single page and only chains when the next request keeps i_wb_addr[19:6] equal to the current page.
- Sequential streaming: The controller treats a request as sequential when the next i_wb_addr equals the internally tracked spif_addr + 1; in that case CS can be held and the burst continued (subject to page limits for program operations).
- Sector erase addressing: A sector erase is initiated via the control space (address 00) with wdata[31]=1. The 6-bit sector index is taken from wdata[19:14]. The erase base address is aligned to the 64 KB boundary of that sector, sent on wire as A = {sector_index[5:0], 16'h0000}. There are 64 sectors across 4 MB.
- Byte ordering: Serial transfers are MSB-first on the SPI bus, but the controller reorders received data so o_wb_data presents 32-bit words in natural software order. During program, the four constituent bytes of the word are emitted in the order expected by the flash. No software byte-swapping is required.
- Mode independence: Quad/serial mode selection and dummy/mode cycles do not affect the address mapping or word-to-byte translation described above.
- Non-addressed commands: JEDEC ID, status (RDSR), and configuration commands carry no address field and are unaffected by the translation.

-- SPI Engine (llqspi) Integration --

SPI Engine (llqspi) Integration
- Role: A single llqspi instance serializes all SPI/QSPI transactions. wbqspiflash decomposes each high-level command into llqspi "segments" (opcode, address, dummy/mode, data) and programs width, length, direction, and chip-select behavior per segment.
- Control inputs to llqspi: i_word[31:0] (payload/opcode/address), i_len[1:0] (00=8b, 01=16b, 10=24b, 11=32b; internally expands to (i_len<<3)+8 bits), i_spd (0=SPI 1-bit, 1=QSPI 4-bit), i_dir (0=drive/tx,

1=receive/rx), i_wr (issue/arm segment), i_hold (keep CS asserted between segments/words when idle between beats).
- Status/outputs from llqspi: o_busy (shift engine active), o_valid (segment complete; o_word valid), o_word[31:0] (returned data aligned to segment size), o_cs_n, o_sck, o_dat[3:0] (outgoing IO), o_mod[1:0] (mode: [1]=quad enable, [0]=dir in quad for IO direction control).
- Pin forwarding: In normal operation, wbqspiflash routes o_cs_n/o_sck/o_dat/o_mod directly to pads and returns pad inputs to llqspi.i_dat[3:0] (IO1 is MISO in SPI). During spif_override, pins are driven by a small startup sequencer and llqspi is held idle.
- Width and direction: Opcodes are always sent in SPI (i_spd=0). Subsequent phases may switch to quad (i_spd=1) for 1-4-4 reads or quad program. i_dir=0 for opcode/address/program payload; i_dir=1 for read data and for dummy/mode phases that precede a read in quad. In SPI mode, direction has no electrical effect but remains driven consistently.
- Chip-select management: wbqspiflash uses chaining (issuing i_wr at the llqspi word boundary) to avoid any CS glitch between adjacent segments, and i_hold to keep CS low when inserting gaps (e.g., between Wishbone beats or while polling). CS remains low across multi-phase operations and contiguous bursts when permitted by the flash protocol.
- Handshake and sequencing: For each segment, wbqspiflash writes i_word/i_len/i_spd/i_dir, asserts i_wr, and waits for o_busy/o_valid. It advances its FSM on o_valid and avoids reasserting i_wr while o_busy is active unless chaining at a boundary. Wishbone o_wb_stall is held while required segments are in flight; o_wb_ack aligns to the relevant o_valid(s) or end-of-command.
- Data handling: llqspi returns 32-bit o_word on every o_valid. wbqspiflash extracts the low 8/16/24 bits as needed and reorders bytes so software sees natural 32-bit little-endian words.
- Common flows (segment programming):
- 1-1-1 Fast Read (0x0B): SPI opcode (8b, tx) → SPI address (24b, tx) → SPI dummy (8b, rx) → SPI data beats (32b, rx). CS held throughout.
- 1-4-4 Quad Read (0xEB): SPI opcode (8b, tx) → Quad address (24b, tx) → Quad mode/dummy (device-specific length, rx) → Quad data beats (32b, rx). CS held throughout.
- Page Program (0x02 SPI or 0x32 Quad): WREN (SPI, 8b, tx) → Program opcode (8b, selected width, tx) → Address (24b, selected width, tx) → Data beats (32b, tx). Stream with CS held while addresses remain sequential within a 256-byte page; terminate at page boundary.
- Status/Config/ID: RDSR (0x05): 8b opcode (tx) → 8b read (rx). RCFR (0x35): 8b opcode → 8b read. WRR (0x01): WREN → 0x01 → 8/16b payload (tx only). JEDEC ID (0x9F): 8b opcode → 24–32b read (SPI).
- WIP polling: wbqspiflash issues repeated 0x05 reads using llqspi, optionally holding CS across polls to reduce overhead, and gates Wishbone completion until WIP clears; background polling can raise o_interrupt when complete.
- Electrical/PHY notes: llqspi shifts MSB-first (bit or nibble). o_sck idles high. o_mod[1] selects quad drive; o_mod[0] mirrors i_dir in quad to control IO direction. In SPI mode, IO1 is sampled as MISO via i_dat[1].
- Performance considerations: Maintain sequential addresses to allow chaining and i_hold across beats for maximum throughput in both read and program operations.

-- CS Hold, Chaining, and Burst Support --

Leverages llqspi's CS-hold and word-chaining to sustain high-throughput bursts with minimal reselection overhead. Read bursts: under one CS, issue 0x0B (SPI) or 0xEB (Quad) + 24-bit address + dummy/mode, then stream 32-bit beats. If the next WB read targets the sequential word (addr == last_addr + 1) and i_wr arrives exactly at the end-of-word boundary, the next beat chains immediately (no idle, CS stays low). If the next request is slightly late, spi_hold keeps CS low (llqspi hold mode, o_busy deasserted) until i_wr is presented, then resumes under the same CS. CS is released at a boundary when neither chaining nor hold is requested, or on address discontinuity. Read bursts are effectively unbounded and end on address or access-type change. Program bursts: WREN (0x06) is issued separately; Page Program starts with 0x02 (SPI) or 0x32 (Quad) + 24-bit address, then the first

32-bit word is sent. Additional words chain only while addresses are sequential and remain within the same 256-byte page (i_wb_addr[19:6] == last_addr[19:6]); crossing a page boundary or any discontinuity forces CS release and requires a new PP (and prior WREN). Control/status: single-command transactions complete under one CS; RDSR (0x05) polling may chain back-to-back status reads or hold CS between reads until SR[0] clears. Wishbone timing: o_wb_stall stays asserted while shifting; each streamed beat generates one o_wb_ack; chaining at boundaries and CS-hold across short gaps minimize inter-word latency. CS release conditions: address discontinuity, write page boundary, completion of single-command operations, transitions into/out of WIP polling, or absence of a chaining request with hold deasserted at a word boundary.

-- Busy/WIP Polling and Status Tracking --

Definitions
- spi_busy: True while the llqspi shifter is active (command/address/data transfer or teardown). Derived from llqspi o_busy; reflects wire-level activity only.
- write_in_progress (WIP): Mirrors flash SR1[0] (RDSR 0x05). Indicates flash internal program/erase/status-write is ongoing.

Sources and tracking
- WIP is set immediately when WREN + {Page Program 0x02/0x32, Sector Erase 0xD8, WRR 0x01} is issued (unless write_protect blocks the operation) and is cleared only after an RDSR shows SR[0]==0.
- last_status holds the most recently read status byte from RDSR (ctrl 10 updates it).
- quad_mode_enabled reflects CR[1], updated on 0x35 reads and WRR writes.
- write_protect is a software latch; when set, program/erase/WRR requests are acknowledged without execution and WIP is not set. The controller status bit reports ~write_protect.
- erased_sector records the last erased sector index; dirty_sector flags whether that sector has since been written.

Polling policy (RDSR 0x05)
- New data/control operations requested while WIP=1 are deferred; the controller polls RDSR until SR[0]==0, then proceeds. o_wb_stall remains asserted during this pre-operation polling.
- Controller status read (ctrl 00) transparently issues RDSR if needed, then returns a coherent composite status snapshot.
- Explicit status read (ctrl 10) always performs RDSR, updates last_status and WIP, and returns {24'h0, SR}.
- CS may be held across back-to-back status polls using llqspi hold/chaining to reduce latency.

Wishbone interaction
- o_wb_stall=1 while spi_busy=1 or when mandatory WIP polling is in progress before starting a requested operation.
- o_wb_ack pulses when the SPI-level operation completes (i.e., all required bytes shifted). For program/erase/WRR, this does not imply flash internal completion; WIP may remain 1.
- If write_protect=1, program/erase/WRR are acknowledged without execution; no WIP set.

Operation gating
- Data reads (0x0B/0xEB) are deferred when WIP=1; the controller polls until WIP clears, then performs the read (supporting burst streams with CS held).
- Page program (0x02/0x32) and sector erase (0xD8) start only if write_protect=0 and WIP=0; upon issuance, WIP is set to 1. Page-program bursts respect 256-byte page boundaries; no new burst begins while WIP=1.
- Status/config writes (WRR 0x01) are issued only when WIP=0; upon issuance, WIP is set to 1. Any follow-up commands (e.g., 0x30) occur under the same WIP tracking.

Background completion
- When WIP=1 and the bus is idle, the controller periodically polls RDSR in the background. On detecting SR[0] transition from 1 to 0, it clears WIP and pulses o_interrupt for one cycle.

Status reporting
- Controller status (ctrl 00) returns: {write_in_progress, dirty_sector, spi_busy, ~write_protect, quad_mode_enabled, 7'h0, erased_sector[5:0], 14'h0} after ensuring a valid SR sample when needed.

Implementation notes
- RDSR polling occurs in dedicated wait states; background checks run in the idle state.
- WIP is only mutated by: (a) synchronous set on issuing WREN+{Program/Erase/WRR} (if not write-protected), and (b) clear on successful RDSR showing SR[0]==0.
- llqspi hold/chaining is used to minimize CS toggles during polling and burst transfers.

-- Write-Protection and Safety Mechanisms --

- Software write-protect latch
- Control-space address 00, write bit 28 (inverted semantics): write_protect <= ~data[28].
- When write_protect is set, page program, sector erase, and other mutating requests are acknowledged without execution to prevent unintended changes.
- Status readback exposes writes_allowed = ~write_protect for host-side decisions.
- Busy/WIP gating and polling
- All mutating operations (program, erase, WRR) and data reads are blocked while the flash Write-In-Progress (WIP) bit is set.
- The controller automatically polls RDSR (0x05) and stalls the Wishbone request until SR[0]==0; completion generates an interrupt pulse to signal write completion.
- Sector erase safety and tracking
- Erase permitted only when not write_protect and not WIP.
- Requested via control-space address 00, write bit 31, with target sector encoded in data[19:14].
- Sequence: WREN (0x06) then SE (0xD8). erased_sector is tracked; dirty_sector is cleared upon successful erase.
- Page program safety
- Program permitted only when not write_protect and not WIP; WREN (0x06) precedes 0x02 (SPI) or 0x32 (QSPI).
- Chained bursts must remain within a single 256-byte page; enforcement requires i_wb_addr[19:6] == spif_addr[19:6] to prevent wrap-around corruption.
- dirty_sector is set when programming within erased_sector to indicate the sector is no longer clean.
- Status/config register write safety
- WRR (0x01) is guarded by WREN and tracked via write_in_progress; last_status is maintained.
- If last_status[6:5] indicate protection/error bits requiring clearing, the controller conditionally issues 0x30 to restore a safe state.
- Quad-mode enable is tracked; 0x35 read updates quad_mode_enabled.
- Read-path safety
- Fast reads (0x0B/0xEB) defer while WIP is set; the controller polls and only returns data when safe.
- Sequential bursts keep CS asserted for transaction integrity; address changes are honored to terminate cleanly.
- Reset and hardware safety
- On reset, spif_override briefly drives CS/SCK to place the flash in a benign state before llqspi takes control.
- llqspi enforces atomic word boundaries, proper CS timing, and a busy/valid handshake to preserve integrity across chained transfers.
- Wishbone-level safeguards

- o_wb_stall asserts while the SPI engine or controller is busy; o_wb_ack pulses only upon safe completion.
- Control-space writes that would be unsafe under write_protect are acknowledged without side effects to avoid bus deadlock.
- Non-mutating operation safeguards
- JEDEC ID (0x9F): writes are treated as no-ops (immediate ack) to prevent accidental configuration changes; reads return a 32-bit ID.
- Status readback contents (addr 00)
- {write_in_progress, dirty_sector, spi_busy, ~write_protect, quad_mode_enabled, 7'h0, erased_sector[5:0], 14'h0} for comprehensive host visibility.

-- Quad-Mode Control and Configuration --

Quad-Mode is controlled by the flash Configuration Register (CR) bit1. The controller mirrors this in quad_mode_enabled, which is updated on CR reads (0x35) and Configuration writes (WRR, 0x01). After reset, quad mode is not assumed until CR is read or written; spif_override places the device in a known, safe state. Control-space: at address 01 (Configuration), a read issues 0x35 and returns {24'h0, CR}, updating quad_mode_enabled<=CR[1]; a write issues 0x06 (WREN) then 0x01 (WRR) with two bytes {Status=last_status, Config=spif_data[7:0]}, setting quad_mode_enabled<=Config[1], and is blocked if write_protect is set or SR[0] (WIP) is active. At address 10 (Status), a read issues 0x05 (RDSR), returning {24'h0, SR} and updating write_in_progress and last_status to prepare the WRR status byte; optional sequences may clear protect bits per device policy. Enabling quad: read SR (0x05) to capture last_status; write WRR (0x01) with Status=last_status and Config having bit1=1; the controller handles WREN, WIP gating, and completion. Verify by reading CR (0x35) and checking CR[1]. Operational effect: all data-space transfers automatically select width based on quad_mode_enabled—reads use 0xEB (1-4-4 fast read) when enabled or 0x0B (1-1-1) otherwise; page programs use 0x32 (Quad Page Program) or 0x02 (SPI Page Program), respecting 256-byte page boundaries and supporting CS hold within a page. Commands are always sent in single-bit SPI (no 4-4-4 opcode phase); width applies to address/data. llqspi signaling: i_spd=1 selects quad, o_mod[1]=1 indicates quad address/data phases, o_mod[0]=i_dir controls bus direction, and i_hold maintains CS across chained beats. Constraints: configuration/status writes require prior WREN and are held off while WIP is set; the controller polls RDSR and can raise an interrupt when background WIP clears. write_protect blocks WRR/program/erase operations; reads (0x35/0x05/0x9F) remain allowed.

-- Erase/Dirty Sector Tracking --

wbqspiflash maintains informational tracking of the most recently erased sector and whether it has since been programmed (dirty); this tracking does not gate or block any operation. A control-space read at addr[1:0]=00 returns a 32-bit status where: bit31=WIP (write-in-progress), bit30=dirty_sector, bit29=spi_busy, bit28=~write_protect (1 means writes/erase allowed), bits[19:14]=erased_sector[5:0]; all other bits are zero. To request a sector erase, write to addr[1:0]=00 with bit31=1 and the target sector in bits[19:14]; the request is accepted only if write_protect=0 and WIP=0. On acceptance, the controller issues WREN (0x06) then SE (0xD8) using a 24-bit byte address derived from sector bits[19:14] with a zero offset (64 kB alignment), sets erased_sector to the requested value, and clears dirty_sector to 0. If the preconditions are not met (WIP=1 or write_protect=1), no SPI command is issued and tracking registers are unchanged. After an accepted erase, any subsequent page program (0x02) or quad page program (0x32) whose address falls within the tracked erased_sector sets dirty_sector to 1; programs to other sectors do not affect the flag. Only one sector is tracked at a time; a new accepted sector erase overwrites erased_sector and clears dirty_sector regardless of which sector is erased. A control-space read at addr 00 may internally poll RDSR (0x05) while the device is busy; the returned fields (including WIP/dirty/erased_sector) reflect the latest valid state. Sector numbering uses address bits[19:14] of the controller’s 20-bit word address model, implying 64 kB sector

granularity. Page-boundary rules are orthogonal to dirty tracking; only sector match is considered. On reset, tracking registers are cleared unless the RTL specifies otherwise. The low-level SPI engine (llqspi) is unaware of this feature; all tracking is implemented in wbqspiflash.

-- Reset/Startup Override Sequencer --

On reset, a two-state startup sequencer asserts spif_override to bypass llqspi and directly drive the external SPI/QSPI pins. While asserted, the sequencer (not llqspi) generates a short, deterministic clocking pattern on o_cs_n and o_sck without issuing any opcodes; o_dat[3:0] and o_mod are held at benign values. The pattern is intended to clear any lingering continuous-read/XIP/QPI modes and reestablish a known SPI 1-1-1 idle with CPOL=1 (SCK idle high) and CS deasserted. During this window the Wishbone interface is held off (no new requests accepted). Upon completion and verification of an idle link (CS high, SCK high), spif_override deasserts, pin control returns to llqspi, and normal command traffic and Wishbone servicing begin. Scope is reset-only; duration is fixed and ends before any transaction-level activity; quad/status discovery is deferred to normal reads (e.g., 0x35/0x05).

OPERATION

-- Bus Transaction Timing and Handshake --

- Request presentation: Assert exactly one strobe (i_wb_data_stb for array space or i_wb_ctrl_stb for control/status) with i_wb_we, address, and write data (if applicable). Do not assert both strobes in the same cycle.
- Acceptance: A request is accepted in any cycle where the chosen strobe is high and o_wb_stall=0. On acceptance, the controller latches address, i_wb_we, and write data (for writes) and starts the SPI/QSPI operation. Requests presented while o_wb_stall=1 are not accepted.
- Back-pressure: o_wb_stall=1 whenever a prior request is being serviced, the SPI/QSPI shifter is busy, the flash indicates WIP=1 and requires polling, or during reset/initialization. Masters must only present new requests when o_wb_stall=0.
- Completion/ack: Each accepted request produces exactly one single-cycle o_wb_ack when the operation completes at the bus level. For reads, o_wb_data is valid in the same cycle as o_wb_ack and must be sampled then.
- Read timing: The first read ack occurs when the first 32-bit word is received from the SPI engine (aligned to llqspi o_valid) after opcode+address+dummy latency (SPI 0x0B or QSPI 0xEB). Sequential reads (next word address) stream under CS hold; each word yields its own single-cycle ack, with o_wb_stall asserted only while the shifter advances between words. Non-sequential accesses break the stream and re-incur startup latency.
- Write timing: If write_protect=1, program/erase/control writes that would change flash state are acked immediately with no SPI side effects. Otherwise, writes stall until WIP=0 as needed; the controller issues WREN then Page Program (SPI 0x02/QSPI 0x32) or control write, and o_wb_ack pulses at the end of the 32-bit word shift (llqspi word boundary). Sequential program words stream under CS hold provided the address remains within the same 256-byte page; each word produces its own ack. Crossing a page boundary or non-sequential addressing terminates the stream and requires a new sequence.
- Control-space timing: Status/control reads may stall while polling (e.g., RDSR 0x05) and ack when fresh data is received. Control writes ack after command launch (e.g., WREN+WRR 0x01, sector erase 0xD8), not after internal completion; background WIP may remain set.

- Reset/initialization: During reset, o_wb_stall=1 and no requests are accepted. Normal handshake resumes once initialization completes.
- llqspi mapping: The controller asserts llqspi i_wr to launch commands/words; llqspi o_busy directly drives o_wb_stall. Read acks coincide with llqspi o_valid; write acks coincide with the end-of-word shift. spi_hold maintains CS across chained words.
- Interrupts: o_interrupt may pulse when WIP clears in the background without an active bus cycle; it is independent of the Wishbone handshake.

-- Read Operations (SPI and Quad Fast Read) --

The controller supports high-throughput array reads from SPI flash via two protocols, selected automatically by quad_mode_enabled (reflecting CR[1]): SPI Fast Read (1-1-1, opcode 0x0B) and Quad Fast Read (1-4-4, opcode 0xEB). A read operation is triggered by a Wishbone data-space read (i_wb_data_stb=1 with i_wb_we=0). The 24-bit byte address placed on the flash is formed from the word address as {i_wb_addr[19:0], 2'b00}, and data are returned as 32-bit words with byte-order corrected to natural word order on o_wb_data.

If the device indicates write-in-progress (WIP=1), the controller defers the read and transparently polls the Status Register (opcode 0x05) until WIP clears; o_wb_stall remains asserted during this wait and the read is acknowledged only when data are ready.

SPI Fast Read sequence (quad_mode_enabled=0): send 0x0B on the 1-bit bus, then the 24-bit address on 1-bit, followed by 8 dummy bits, then stream data bytes on the 1-bit bus. Quad Fast Read sequence (quad_mode_enabled=1): send 0xEB on the 1-bit bus per 1-4-4 protocol, then the 24-bit address on the 4-bit bus, device-required mode/dummy cycles, and stream data on the 4-bit bus.

After the first word is returned, the controller keeps chip-select asserted and streams subsequent 32-bit words to maximize throughput. Streaming continues only if the next Wishbone read targets the sequential word address; non-sequential requests or no follow-on read terminate the transaction and release CS. Wishbone handshaking guarantees o_wb_stall=1 while the SPI engine sets up or shifts, and a single o_wb_ack pulse per returned word; best throughput is achieved with back-to-back sequential reads. Mode selection is dynamic: when CR[1] changes, the read path switches automatically between 0x0B and 0xEB. First-word latency includes command, address, and dummy phases; subsequent words stream with minimal inter-word overhead under CS hold.

-- Write Operations (Page Program, Chaining, Page Boundaries) --

- Trigger: A data-space Wishbone write (i_wb_data_stb && i_wb_we) initiates a page-program sequence when write_protect=0 and SR[0] (WIP)=0. If write_protect=1, the beat is acknowledged but no flash operation occurs. If WIP=1, the controller stalls, polls SR (0x05) until WIP clears, then starts.
- Command setup: On burst start, issue WREN (0x06) once. Use Page Program 0x02 in SPI mode, or Quad Page Program 0x32 when quad_mode_enabled=1 (i_spd set accordingly). Send a 24-bit byte address formed as {i_wb_addr[19:0], 2'b00}.
- Data write: Each Wishbone beat programs one 32-bit word. The controller manages wire byte order; software writes natural 32-bit words. o_wb_stall=1 while command/address/data are being shifted; o_wb_ack pulses per beat after its data is fully shifted to the flash.
- WIP handling: write_in_progress is set after launching the first word and mirrors SR[0]. The controller does not wait for on-flash programming to finish; completion is detected in background by SR polling, and o_interrupt is pulsed when WIP clears.
- Chaining within a single PP/QPP command: Additional 32-bit words are streamed in the same command if (1) the next Wishbone request targets the next sequential word address (addr == spif_addr+1) and (2) it remains within the same 256-byte page (i_wb_addr[19:6] == spif_addr[19:6]).

Chip-select remains asserted; no extra command/address phases between chained words.
- Termination of chaining: If the next beat is not immediately available, is non-sequential, or would cross a page boundary, the controller ends the PP/QPP command (CS released). A subsequent write starts with a fresh WREN + PP/QPP.
- Page boundaries: Page size is 256 bytes; maximum chained burst is 64 words (256/4). Hardware enforces the page-boundary check using word-address bits [19:6]; write bursts cannot cross pages.
- Sector dirtiness: If a program operation targets the sector recorded in erased_sector, dirty_sector is set to indicate the sector now contains programmed data.

-- Erase Operations (Sector Erase Flow) --

Purpose: Initiate a sector erase on the SPI flash via the wbqspiflash control space.

Request format (control addr 00):
- Wishbone write with bit31=1 to trigger erase.
- bits[19:14]=sector index (6 bits) of target sector.
- bit28 updates the software write-protect latch (write_protect <= ~data[28]). If write_protect=1, the erase is blocked.

Preconditions and gating:
- Erase only proceeds if write_protect=0.
- If the flash is busy (SR[0] WIP=1) or the SPI engine is active, the controller asserts o_wb_stall and holds the request, polling RDSR (0x05) until WIP clears.

Dispatch sequence:
- Issue WREN (0x06).
- Issue Sector Erase (0xD8) with a 24-bit address derived from bits[19:14] (controller computes the sector base address).
- Tracking on dispatch: erased_sector <= requested index; dirty_sector <= 0.
- Wishbone acknowledge: o_wb_ack pulses after WREN and SE have been sent; the flash completes the erase internally afterward.

During erase:
- write_in_progress mirrors SR[0] (WIP) and is maintained via background RDSR polls.
- While WIP=1, further program/erase requests are gated; other flash accesses stall or transparently poll until ready.
- On WIP clearing (1→0), o_interrupt pulses to signal completion.
- Subsequent page programs in the same sector set dirty_sector=1.

Status visibility (control read addr 00):
- Returns composite status including write_in_progress, dirty_sector, spi_busy, ~write_protect, quad_mode_enabled, and erased_sector[5:0].

Arbitration and edge cases:
- Only one erase can be in flight; incoming erase requests while busy are stalled, not dropped.
- If write_protect=1 at request time, the controller immediately acks without issuing SPI commands.

Implementation notes:
- All command/address shifting (WREN, SE, RDSR) is performed by the llqspi submodule; no special quad-mode handling is required for erase.

-- Status and Configuration Access (RDSR/RDCR/WRR) --

Control-space access is selected with i_wb_ctrl_stb and i_wb_addr[1:0]; addresses: 01=Configuration, 10=Status. All operations use standard 1-1-1 SPI at i_spd=0.
- RDSR (Read Status Register, 0x05) at ctrl addr 10 (read): Issues 0x05 and reads one status byte. o_wb_data returns {24'h0, SR[7:0]}. Side effects: last_status <= SR; write_in_progress <= SR[0]. Handshake: o_wb_stall asserted during SPI; o_wb_ack pulses when the read completes.
- RDCR (Read Configuration Register, 0x35) at ctrl addr 01 (read): Issues 0x35 and reads one configuration byte. o_wb_data returns {24'h0, CR[7:0]}. Side effect: quad_mode_enabled <= CR[1]. Handshake: o_wb_stall asserted during SPI; o_wb_ack pulses at completion.
- WRR (Write Status/Configuration Register, 0x01) at ctrl addr 10/01 (write): Requires flash ready (SR[0]==0). If busy, the controller auto-polls RDSR and defers WRR until ready. If the software write_protect latch is set, the request is acknowledged without issuing any flash-side write.
- Sequence: WREN (0x06) followed by WRR (0x01).
- Addr 10 (status-only): writes one status byte from the WB payload; updates last_status. If protection bits need clearing, the controller may issue 0x30 afterward.
- Addr 01 (status+config): writes two bytes {Status=last_status, Config=WB payload byte}; quad_mode_enabled <= Config[1].
- Side effects: write_in_progress <= 1 when WRR is launched. Handshake: o_wb_stall remains high throughout WREN/WRR (and any optional 0x30); o_wb_ack pulses once the command sequence has been fully issued to the flash (not waiting for internal write completion).
- Busy handling and completion: Any operation gated by SR[0]==0 uses RDSR polling. Independently of WB traffic, the core polls RDSR in the background; when WIP clears it pulses o_interrupt.
- Data formatting: Status/configuration reads return the byte in o_wb_data[7:0] with o_wb_data[31:8]=24'h0.

-- JEDEC Identification Read --

Access: Read via control space (i_wb_ctrl_stb, i_wb_we=0) at i_wb_addr[1:0]==2'b11; a write to this address is a no-op and is acked immediately without SPI activity. Operation: Issues JEDEC Read ID (0x9F) using the llqspi engine; chip-select is held (i_hold=1) across an 8-bit command write followed by a 32-bit read; command is sent in standard SPI (quad mode not used). Data: o_wb_data returns four ID bytes in natural word order {Manufacturer ID, Memory Type, Capacity, vendor/extension}; the core reorders received bytes if needed to present natural word order. Handshake: o_wb_stall remains asserted while the transaction is active; o_wb_ack pulses when the 32-bit value has been received (aligned with llqspi o_valid); if the SPI engine is busy, the request is queued and serviced when available. Constraints: No WREN required; independent of the write-protect latch; does not poll WIP; single-shot control transaction with CS released after completion.

-- Handling Flash Busy/Ready and Transparent Polling --

Busy/ready tracking and polling are driven by the flash Status Register (SR). The controller mirrors SR[0] (WIP) into an internal write_in_progress flag, set immediately when issuing program/erase/WRR (0x01) sequences and cleared only after a subsequent Read Status (RDSR, 0x05) shows WIP=0. The most recently read SR value is kept in last_status.

Preconditions and gating:
- No data reads, page programs, sector erases, or configuration/status writes are started while write_in_progress=1. If busy, the controller stalls the request and enters a polling loop until ready.
- A software write_protect latch, when set, blocks program/erase: such requests are acknowledged but suppressed, and no busy/polling is entered.

Transparent polling policy:
- Control space read at addr[1:0]==2'b00 (controller status/control) transparently waits for the flash to

be ready. If WIP=1, the controller polls RDSR until WIP clears, then returns the status/control word. The Wishbone master does not need to poll.
- Data space accesses (i_wb_data_stb) stall when WIP=1 and transparently poll until the device is ready, then proceed with the read or program sequence.
- Explicit status reads at control addr[1:0]==2'b10 issue a single RDSR (0x05) and return {24'h0, SR} immediately with no transparent waiting; software can observe instantaneous WIP.

Polling mechanism and efficiency:
- Polling is performed by repeatedly issuing RDSR (0x05) and sampling SR until SR[0]==0.
- The llqspi engine is used to minimize latency: CS may be held low via i_hold, and successive SR reads are chained via back-to-back i_wr without releasing CS.
- last_status is updated on every RDSR.

Wishbone handshake during busy/polling:
- o_wb_stall is asserted while polling for ready or while prerequisite SPI command/address/data phases are being shifted.
- o_wb_ack pulses once the controller-owned SPI sequence for the requested bus operation completes:
- For reads: when the requested data word is available.
- For program/erase/WRR: after the command/address/data phase has been fully issued to the flash. Flash-internal completion occurs later and is reflected by write_in_progress clearing (and optional interrupt), not by the original bus ack.
- llqspi o_busy is also considered when deciding when the next SPI word can be issued; this is separate from flash WIP and only affects intra-transaction timing.

Background readiness detection and interrupt:
- When idle on the Wishbone bus and write_in_progress=1, the controller autonomously polls RDSR. Upon detecting WIP transitioning from 1 to 0, it generates a one-cycle o_interrupt pulse to signal completion of the prior program/erase/WRR.

Interaction with streaming transfers:
- Sequential data reads keep CS asserted and stream words only after confirming the device is not busy; the controller will not start or extend a stream if WIP=1.
- For page program bursts, write_in_progress is set when the sequence begins; no new operation is accepted until WIP clears. Burst chaining remains within the same 256-byte page, enforced by a page boundary check.

Observability:
- The control/status read at addr[1:0]==2'b00 returns combined controller status bits only after WIP clears, whereas the explicit SR read at addr[1:0]==2'b10 returns the immediate SR snapshot without waiting.

REGISTERS

CLOCKS