Generated Specification Document

INTRODUCTION

-- Purpose and Scope --

Purpose: rtcclock is a Wishbone-accessible real-time clock peripheral for FPGA/SoC designs that provides calibrated time-of-day keeping, programmable timing functions (timer, stopwatch, alarm), precise timestamp capture, a four-digit seven-segment display driver, status LEDs, and a consolidated interrupt output.
Scope:
- Time base generation: derives a software-trimmable 1 Hz tick (1-PPS) from an arbitrary system clock using a programmable fractional-N accumulator (ckspeed).
- Real-time clock: 24-hour HH:MM:SS in packed BCD with software read/write access and latched readback to avoid race conditions.
- Programmable tick rate: ckspeed register enables frequency trimming/calibration of the 1-PPS.
- Countdown timer: BCD HH:MM:SS with run/expired control and an interrupt on expiry.
- Stopwatch: run/clear control with subsecond resolution and accumulation in BCD SS:MM:HH.
- Alarm: match on HH:MM:SS with enable/trip control and an interrupt on match.
- Precise timestamp (hack) capture: captures current time and accumulator state on an external strobe for cycle-accurate event timing.
- User interface outputs: four-digit seven-segment display with selectable modes (clock, timer, stopwatch) and blink indication on alarm/timer; LED activity/status indications.
- Bus interface: Wishbone register map for control/status, timekeeping, calibration, and hack data; single o_interrupt as the OR of timer/alarm events.

-- Features Summary --

- Wishbone-accessible real-time clock with 24-hour HH:MM:SS in packed BCD; latched readout; per-byte writes using sentinels (sec/min=0xFF, hour=0x3F); writing seconds=0 clears subsecond divider.
- Programmable time base via 40-bit fractional-N accumulator; software-trimmable tick rate (ckspeed, default 2814750) for precise 1PPS; generates second, minute, and hour rollovers.
- Countdown timer (0–99:59:59 BCD HH:MM:SS) with run control, expired latch, per-second decrement, auto-stop on zero, and interrupt on expiration; write semantics support load/start/reload and expired clear on run.
- Stopwatch with subsecond mirror, BCD HH:MM:SS accumulation, run/clear control, and hour field extending beyond 99.
- Alarm comparator against target HH:MM:SS with enable and tripped latch; pulses interrupt on match; cleared by write or disabling; sentinel-based writes.
- Combined interrupt output (o_interrupt) from timer and alarm events.
- High-precision "hack" timestamp capture aligned to tick carry, returning current time+subsecond and accumulator via multi-register readback.
- Integrated 4-digit seven-segment display driver with selectable sources (clock/timer/stopwatch), leading-zero suppression, and global blink when alarm/timer active; includes synchronous hex-to-7-seg decoder submodule.

- Status LED activity indicator that increments per carry, resets at top-of-minute, and blinks on alarm/timer activity.
- Wishbone-readable register map for clock, timer, stopwatch, alarm status/target, ckspeed, and hack capture values.

-- Design Overview --

rtcclock is a Wishbone-accessible real-time clock subsystem that consolidates a 24-hour BCD RTC, programmable tick generation, countdown timer, stopwatch, alarm, precise timestamp capture, a 4-digit 7-segment display driver, LED status, and a single combined interrupt. A single programmable time base drives all timing: a 40-bit fractional-N accumulator advances by software-set ckspeed each cycle; its carry (ck_carry) feeds a subsecond divider (ck_sub) that produces a 1-PPS strobe (ck_pps). This time base sources the RTC second ticks, the timer decrement, stopwatch subseconds, display blink phase, and LED activity. The RTC holds HH:MM:SS in packed BCD, advances on ck_pps with proper minute/hour carry, and is written via field-selective sentinels; seconds=0 writes also clear the subsecond divider. Reads return a latched snapshot to avoid visualization races. The countdown timer is BCD HH:MM:SS with run and expired latches; it decrements at 1 Hz when enabled, asserts an interrupt on reaching zero, and supports load/reload semantics with expired clearing on (re)start. The stopwatch accumulates subseconds from the accumulator carry and rolls seconds/minutes/hours accordingly, with run and clear control via writes. The alarm compares the RTC time against a programmed BCD target under an enable gate, latches on match, and contributes to the interrupt. A hack capture path precisely samples {RTC time, subsecond divider, accumulator} on an external i_hack event aligned to a carry boundary for cycle-accurate timestamps, exposing the capture over three read addresses. The 7-segment display presents multiple modes (RTC, timer, stopwatch) with leading-zero suppression and a global blink driven by subsecond phase and alarm/timer status; segment glyphs are produced by a synchronous one-cycle hexmap submodule with active-high segments (invert externally for common-anode). LEDs reflect activity and status, blinking with the subsecond phase. The Wishbone register map provides access to the RTC snapshot, timer, stopwatch, alarm enable/status/time, ckspeed (default programmable value), and hack capture words; writes are decoded per function with field-specific sentinels and latching behavior. A single o_interrupt output is the OR of timer-expired and alarm-match events. The top-level module instantiates the hexmap submodule; all timing domains are derived from the single accumulator/subsecond chain to ensure coherent behavior across features.

-- Interrupt Overview --

Overview
- o_interrupt is the logical OR of two independent, edge-generated sources: tm_int (countdown timer expiry) and al_int (alarm match). There is no global interrupt enable/mask or in-module acknowledge.

Countdown timer interrupt (tm_int)
- Trigger: Asserts when the countdown timer decrements to 00:00:00 with timer[24]=1 (run) and the timer not already expired.
- Form: One-cycle pulse aligned to the 1PPS-derived second boundary; synchronous to the module clock.
- Side effects: Sets timer[25] (expired latch, aka tm_alarm) and auto-clears timer[24] (run) to stop counting.
- Clear/re-arm: Any write that sets run=1 clears timer[25]. If the counter is at zero, writing zero reloads tm_start; writing a nonzero value loads a new start time. No separate interrupt mask.

Alarm interrupt (al_int)
- Trigger: When clock[21:0] (HH:MM:SS, BCD) equals alarm_time[21:0] and al_enabled (bit24) is 1.
- Form: One-cycle pulse on the equality edge (when the match becomes true), aligned to second

boundaries; synchronous to the module clock.
- Side effects: Sets al_tripped (sticky alarm-fired latch).
- Clear/gating: Clear al_tripped via a write with bit25=1 or by disabling al_enabled (bit24=0). Per-field write sentinels apply (seconds/minutes 0xFF, hours 0x3F). No separate interrupt mask.

o_interrupt behavior
- o_interrupt = tm_int | al_int. If both events occur in the same cycle, o_interrupt asserts for that cycle once; sources are otherwise independent.
- Interrupts are pulse/edge style; persistent software-visible status is via timer[25] and al_tripped latches.

Software visibility (Wishbone)
- Timer status/control at addr 1: {6'h00, timer[25:0]} where timer[25] indicates expiry.
- Alarm status/control at addr 3: {6'h00, al_tripped, al_enabled, 2'b00, alarm_time[21:0]}.

Non-interrupt UI
- Seven-segment/LED blinking driven by timer[25] (tm_alarm) or al_tripped for user feedback; does not gate or mask o_interrupt.

-- Use Cases --

- Embedded RTC: Maintain a 24-hour HH:MM:SS BCD clock accessible via Wishbone for systems needing time-of-day stamping, scheduling, and UI display.
- Partial time updates: Firmware updates only specific fields (sec/min/hour) using sentinel values (sec/min=0xFF, hour=0x3F) to avoid disturbing untouched fields; write seconds=0 to phase-align the subsecond divider.
- Alarm scheduling: Program a target HH:MM:SS and enable it to generate an interrupt on match; use the tripped latch for reliable firmware acknowledgment/clearing; blink the display for user feedback.
- Countdown workflows: Load a timer (up to 99:59:59), start/stop via a run bit, auto-stop at zero with an expired latch and interrupt; support quick reload from the saved start value for repetitive operations.
- Stopwatch measurement: Start/stop a subsecond stopwatch for profiling and process timing; clear behavior adapts to the running state; extend hours beyond 99 for long runs; display MM:SS when selected.
- Interrupt-driven control: Use o_interrupt (OR of timer and alarm events) to trigger actions (beep, log, state transitions) without polling; rely on latched status to acknowledge and clear events.
- Front-panel display: Drive four 7-segment digits to show HH:MM or MM:SS based on mode bits; blank leading zeros for readability; globally blink on alarm/trip/expire; no CPU refresh required.
- Status LEDs: Provide a heartbeat via a minute-resetting counter; blink LEDs on alarm or timer expiration to attract attention.
- Precise event capture: On an external i_hack strobe, capture aligned time and the fractional accumulator for PPS alignment, button-latency measurement, or hardware event correlation; read back via dedicated addresses.
- Clock calibration: Adjust ckspeed to trim the 1-PPS tick against an external reference (GPSDO/lab standard) or simulate faster/slower time for testing.
- Display decoder integration: Use the hexmap submodule to convert BCD to segment patterns with deterministic one-cycle latency; invert output for common-anode hardware.
- Safe readback: Read a latched snapshot of time/timer/stopwatch/alarm to avoid race conditions in UI/logging; leverage the defined Wishbone address map for structured firmware access.
- Lab/demo tooling: Build a bench-top timer/stopwatch with alarm, a system heartbeat, and precise event capture for demonstrations, validation, and instrumentation.
- Automated workflows: Combine timer/alarm and interrupts for step timing, safety timeouts, and periodic tasks without continuous CPU involvement.
- Calibration and drift monitoring: Periodically compare hack-captured timestamps to an external PPS

to measure drift and adjust ckspeed in the field.

-- Assumptions and Dependencies --

- Single synchronous clock domain for all logic; timing depends on system clock frequency and configured ckspeed and subsecond divider depth (assume 8-bit ck_sub unless otherwise stated).
- 1PPS generation and blink/blanking rates are derived from the 40-bit accumulator and subsecond divider; integrators must program ckspeed to match Fclk (ck_carry rate ≈ Fclk * ckspeed / 2^40).
- Bus interface is a simple synchronous, single-cycle Wishbone-like 32-bit register map (8 words via i_wb_addr[2:0]); no wait states, no byte enables; software must use word-aligned accesses and sentinel per-byte values to selectively update fields.
- Read side-effect: clock readback is latched to ck_last_clock to avoid display races; other registers are read without destructive side-effects.
- No asynchronous reset; power-up state is undefined. Software must initialize clock time, ckspeed, alarm/timer, and mode bits before relying on outputs; writing seconds=0 rephases the subsecond divider to realign PPS.
- Time fields are packed BCD; software is responsible for writing valid BCD encodings. Invalid BCD may yield undefined comparisons or display artifacts.
- Stopwatch hours can exceed 99 via upper nibble; consumers of the value must handle extended range.
- Countdown timer accepts BCD HH:MM:SS up to 99:59:59; while running, only the run bit is writable; write semantics clear/refresh latches as documented.
- Interrupt output is the OR of alarm-match and timer-expire pulses; upstream logic must edge-detect and clear sources via the defined register writes.
- Seven-segment output (o_sseg) expects external digit multiplexing; segments and LEDs are active-high. Systems using common-anode displays must invert as needed. Blink gating uses ck_sub[7].
- Display pipeline depends on the hexmap submodule (synchronous, 1-cycle latency, no reset); initial mapped output is undefined until at least one clock cycle after input update.
- Event capture input (i_hack) is assumed synchronous to i_clk or at least sampled without explicit metastability protection; if asynchronous, provide external synchronization; capture aligns to ck_carry boundaries.
- No hardware validation or range checking beyond increment/carry paths; correctness and bounds enforcement on direct writes is a software responsibility.
- Endianness is not explicitly specified beyond bit positions; software must follow documented bit fields.

IO PORTS

-- Signal Summary --

- Top module: rtcclock
- i_clk (input, 1b): System clock driving all synchronous logic.
- i_hack (input, 1b): Timestamp capture request; latches {clock[21:0], ck_sub} and ck_counter on a carry-aligned boundary.
- i_wb_we (input, 1b): Wishbone write enable; gates writes to clock/timer/stopwatch/alarm/ckspeed via internal address selects.
- i_wb_addr[2:0] (input, 3b): Lower address bits selecting readback map/regions (0–7).
- o_data[31:0] (output, 32b): Wishbone read data bus; muxed per i_wb_addr to expose clock, timer,

stopwatch, alarm, ckspeed, and hack captures.
- o_interrupt (output, 1b): OR of timer-expire and alarm-match interrupt (pulse/level per RTL).
- o_sseg[31:0] (output, 32b): Four 8-bit seven-seg drive bytes (per digit: [dp,g,f,e,d,c,b,a]); active-high; supports leading-zero blanking and blink on timer/alarm.
- o_led[W-1:0] (output, Wb): Status LEDs indicating activity and alarm/timer state; blinks on alarm/timer (W unspecified).
- Child module: hexmap
- i_clk (input, 1b): Clock for registered hex-to-7seg decoding.
- i_hex[3:0] (input, 4b): Hex nibble to convert.
- o_map[6:0] (output, 7b): Active-high seven-seg pattern [a,b,c,d,e,f,g]; registered (1-cycle latency).
- Not present/unspecified: Wishbone cyc/stb/ack and write-data signals, reset port, precise o_led width.

-- Wishbone Bus Interface --

Bus type and clocking
- 32-bit Wishbone B4 classic slave, synchronous to i_clk.

Ports
- i_clk: Wishbone clock.
- i_wb_cyc: Cycle valid.
- i_wb_stb: Strobe; qualifies a valid transfer when asserted with i_wb_cyc.
- i_wb_we: Write enable; 1=write, 0=read.
- i_wb_addr[2:0]: Register word address (0–7).
- i_wb_data[31:0]: Write data bus.
- i_wb_sel[3:0] (optional): Byte lane selects; not used—see Byte enables.
- o_wb_ack: Acknowledge; asserts once per accepted transfer.
- o_data[31:0]: Read data bus.
- o_wb_err (optional): Error; not used (driven 0).

Addressing and data width
- Eight 32-bit word registers decoded by i_wb_addr[2:0].
- If the system uses byte addressing, connect the word index bits A[4:2] to i_wb_addr[2:0]; base address/region assignment is external to this module.
- All accesses are 32-bit word-aligned.

Handshake and timing
- A legal bus request is i_wb_cyc && i_wb_stb.
- o_wb_ack asserts once per request when the transfer completes; read data on o_data is valid in the same cycle as o_wb_ack.
- Writes take effect on the cycle o_wb_ack is asserted.
- No retry; o_wb_err is not used (0).
- Stall/inserted wait states are not used in the reference design; typical response is a single-cycle acknowledge (ack one clock after strobe). If system integration requires additional latency, ack is asserted when data is ready.

Byte enables and endianness
- i_wb_sel[3:0] is ignored; partial writes are not supported. Masters should drive sel=4'b1111 for all writes.
- Data is treated as a 32-bit word with byte lanes [31:24], [23:16], [15:8], [7:0]. BCD fields map per the register descriptions; selective field updates use in-band sentinels (0xFF for sec/min bytes, 0x3F for hour field) rather than byte enables.

Access semantics

- Read register map and write side effects are defined per address (0–7) in the module specification. Address 0 returns a latched time snapshot; reads do not trigger updates.
- Addresses 5–7 are read-only; writes are ignored (ack returned, no side effects, no error).

Reset behavior
- On module reset, bus-facing outputs deassert (o_wb_ack=0, o_wb_err=0). Internal register defaults follow the module reset rules (ckspeed resets to 32'd2814750; other register defaults not specified here).

Illegal access/addresses
- Only addresses 0–7 are implemented. Accesses outside this range should not occur when i_wb_addr[2:0] is correctly wired; if they do, the decode returns o_wb_ack with benign data and no error.

Interrupt
- o_interrupt is not part of the Wishbone interface; it asserts based on timer/alarm conditions as specified elsewhere.

-- Clock and Reset --

- Clocking
- Single synchronous clock domain driven by the system/Wishbone clock; all rtcclock logic (timebase, RTC, timer, stopwatch, alarm, hack capture, LEDs, seven-segment path) and the hexmap submodule share this clock.
- No internally generated or gated clocks are used. All timekeeping and event sequencing use clock-enable strobes in the same domain.
- Timebase: a 40-bit fractional-N accumulator increments by ckspeed each cycle; the accumulator carry (ck_carry) is a one-clock pulse used as the fine tick. A subsecond divider derives a 1-PPS strobe (ck_pps) on rollover. ckspeed defaults to 32'd2814750 and is software programmable to match the system clock frequency.
- Derived strobes (all one system-clock cycle wide):
- ck_carry: base fine tick; drives subsecond counting and stopwatch fine resolution.
- ck_pps: 1 Hz tick; advances the real-time clock and sources per-second enables.
- ck_ppm/ck_pph: top-of-minute/hour detections for carries/LED behavior.
- tm_pps: per-second enable for the countdown timer (decrements while enabled and not expired), derived from the same timebase.
- Functional alignment:
- RTC increments HH:MM:SS on ck_pps.
- Stopwatch uses ck_carry for subsecond and rolls seconds/minutes/hours on per-second boundaries.
- Timer decrements on tm_pps when enabled.
- Display/LED blinking uses a subsecond bit (e.g., ck_sub[7]) for blanking.
- Hexmap is synchronous with one-cycle latency from input to output on the same clock.
- External capture timing: i_hack is sampled in the system clock domain; capture is aligned to a ck_carry boundary (on i_hack high or the subsequent carry) to improve timestamp precision. If i_hack is off-chip, it should be properly synchronized to avoid metastability.

- Reset
- No explicit reset input or reset polarity/type is specified in this description for rtcclock; initial values of RTC time, timer, stopwatch, alarm, accumulator, subsecond divider, and status latches are not defined here.
- hexmap has no reset; its output may be undefined until the first registered update after the clock starts.

- Software-visible reinitializations:
- Writing seconds=0 clears/re-phases the subsecond divider (software re-sync of 1-PPS).
- Status and control latches (e.g., alarm tripped, timer expired, stopwatch clear) are controlled/cleared via register writes.
- Guidance: All internal strobes are enables (not clocks) and are single-cycle in the system clock domain. If an implementation includes a reset, it should be synchronous to the system clock and cover all stateful elements to ensure coherent strobe generation. Software must program ckspeed appropriately if the system clock frequency changes to maintain 1-PPS accuracy.

-- Hack Capture Input --

Hack Capture Input (i_hack) provides a precise, software-readable timestamp of an external event. When i_hack is asserted high, the module defers the snapshot to the next ck_carry boundary so hack_time and hack_counter reflect the same exact tick, eliminating fractional ambiguity. Captured fields: hack_time is a 30-bit composite {clock[21:0], ck_sub[7:0]}, where clock[21:0] is the current 24-hour BCD HH:MM:SS and ck_sub[7:0] is the subsecond divider state; hack_counter is the full 40-bit fractional-N accumulator (ck_counter) at the aligned tick. Readback over the bus: addr 5 returns {2'b00, hack_time}; addr 6 returns hack_counter[39:8]; addr 7 returns {hack_counter[7:0], 24'h0}. Reading does not clear or arm the capture. Each assertion of i_hack overwrites the previous snapshot; the last captured values persist until i_hack is asserted again. There are no status/ready bits, interrupts, or handshakes for this feature. The effective resolution and phase of the capture alignment depend on ckspeed configuration; writing seconds=0 clears the subsecond divider and can affect ck_sub at capture. Input synchronization for i_hack is not specified; if i_hack is asynchronous to the internal clock, ensure the RTL or upstream logic provides proper synchronization to avoid metastability. Software can detect a new capture by comparing hack_time and/or hack_counter to previously read values.

-- Seven-Segment Display Outputs --

o_sseg is a 32-bit output bus carrying four 8-bit seven-segment glyphs (4 digits × 8 bits). Each 8-bit field encodes 7 segments (a–g) plus a decimal point (dp), driven active-high; invert externally for common-anode displays. Digit multiplexing/selection is external to this module. The per-digit bit order and digit ordering on o_sseg are implementation details; confirm in RTL and match the board driver. DP/colon behavior is not defined here; assume dp off unless separately implemented.
Display source selection (clock[27:24]):
- 0 or unspecified: HH:MM from ck_last_clock; leading hour tens digit is blanked when zero.
- 1: countdown timer MM:SS; while running and not expired, suppress leading zeros on upper digits; when expired, show all digits.
- 2: stopwatch MM:SS; suppress upper leading zeros.
- 3: current clock MM:SS.
Glyph generation and masking: Each 4-bit BCD nibble is converted to a 7-segment glyph by hexmap (one clock cycle latency); outputs are undefined until the first clocked update. A dmask function blanks leading zeros per mode rules.
Blink/attention: When tm_alarm (timer alarm/expired) or al_tripped (time-of-day alarm) is active, the entire display blinks by blanking all digits when ck_sub[7] = 1. With the default tick, this yields ~1 Hz blink at 50% duty; frequency scales with the programmable tick.
Update coherency: HH:MM mode uses ck_last_clock to avoid tearing/glitches around second boundaries; other modes display live values and may change at second boundaries.

-- Status LED Output --

o_led drives the front-panel/status LEDs to convey both ongoing activity and alert conditions. Activity is produced by an internal counter (ledreg) that increments on each base tick (ck_carry) from the

programmable fractional-N timebase and is synchronously reset at every top-of-minute event (ck_ppm), yielding a minute-synchronous pattern. Alert indication is applied as a blink overlay whenever the alarm latch (al_tripped) is set by a HH:MM:SS match or the countdown timer's expired latch (timer[25]) is set; during alerts, the LED output is gated by ck_sub[7] to create an approximately 1 Hz on/off cadence. o_led is not directly software-writable; its behavior derives entirely from the timebase, alarm latch, and timer state. Clearing al_tripped or timer[25] via their Wishbone controls stops the blink overlay and returns the LEDs to the activity-only pattern. The ckspeed setting affects the ck_carry rate (thus the activity increment cadence) but does not alter the 1 Hz blink, which is tied to ck_sub[7]. All sources (ck_carry, ck_ppm, ck_sub[7], al_tripped, timer[25]) are generated synchronously within rtcclock relative to i_clk, ensuring glitch-free LED output. The width and external mapping of o_led are top-level integration details and are not specified here.

-- Interrupt Output --

o_interrupt is a single-bit, synchronous pulse output asserted as the logical OR of tm_int (countdown timer expiration) and al_int (alarm match). tm_int issues a one-clock pulse when the countdown timer transitions from 00:00:01 to 00:00:00 with timer[24] (run) = 1; at this edge timer[25] (expired) latches for status and run auto-clears to prevent re-trigger until software restarts the timer. Writing the timer with run=1 clears the expired latch. al_int issues a one-clock pulse when the real-time clock HH:MM:SS equals alarm_time[21:0] and al_enabled (bit24) = 1; at this match al_tripped latches for status and is cleared by writing the alarm with bit25=1 or by disabling al_enabled. Disabled alarms do not generate al_int. Status latches do not hold o_interrupt. No other features (stopwatch, hack capture, 1PPS, display/LED logic) drive o_interrupt. All pulses are synchronous to the rtcclock system clock; o_interrupt = tm_int | al_int.

-- Signal Polarity and Active Levels --

Default polarity
- All control/status bits and event strobes are active-high unless explicitly noted.

External interface
- i_wb_we: active-high write enable.
- i_hack: active-high timestamp capture request; capture occurs when i_hack=1 or on the next ck_carry edge.
- o_interrupt: active-high; logical OR of tm_int and al_int.

Time-base strobes (single-cycle pulses)
- ck_carry: active-high base tick (fractional-N accumulator carry).
- ck_pps: active-high 1PPS strobe (subsecond divider rollover).
- ck_ppm: active-high top-of-minute strobe.
- ck_pph: active-high top-of-hour strobe.
- ck_sub[7]: active-high blink blanking enable for display/LEDs when asserted.

Countdown timer
- timer[24] (run/enable): active-high to start/enable countdown.
- timer[25] (expired/alarm latch, aka tm_alarm): active-high when expiration latched; cleared by a write that sets run=1 or by explicit clear logic.
- tm_int: active-high pulse on transition to zero while running.

Stopwatch
- stopwatch[0] (run/enable): active-high to run.
- stopwatch[1] (clear on write): active-high command bit to clear time/subsecond flags (effective when not running or when run=0 in the same write).

Alarm
- al_enabled (bit24): active-high to enable alarm compare.
- al_tripped (latch): active-high when time match occurs; cleared by disabling or by a write with bit25=1.
- Alarm clear (bit25 on write): active-high command to clear alarm latch.
- al_int: active-high pulse asserted on match edge.

Display and LEDs
- hexmap.o_map[6:0]: active-high segment drives (a,b,c,d,e,f,g). Note: common-anode hardware requires external inversion.
- o_sseg (4×8-bit digits): segments driven active-high; entire display blanked when ck_sub[7]=1.
- o_led: active-high LED drive; blink gating active when ck_sub[7]=1.

Miscellaneous
- Subsecond divider clear: writing seconds=0 performs an active-high (value-driven) clear; no dedicated active-low reset is used.
- Event strobes (ck_carry, ck_ppS/PPM/PPH) are single-cycle, active-high pulses.
- Other Wishbone handshake signal polarities are not specified here and should be confirmed.

-- Clock Domains and Timing --

Clock domains: single synchronous domain driven by i_clk. All state machines, counters, bus interface, display logic, rtcclock, hexmap, and timing strobes operate on the rising edge of i_clk. No generated or asynchronous clock domains are used; all timing derives from i_clk and internal tick/pps generation.

Base tick and PPS: a 40-bit fractional-N accumulator updates each i_clk: {ck_carry, ck_counter} <= ck_counter + ckspeed. ck_carry is a one-cycle pulse on overflow. Average carry rate = (i_clk_freq * ckspeed) / 2^40. A subsecond divider ck_sub counts ck_carry pulses; on its rollover, ck_pps asserts for one i_clk cycle. Writing seconds=0 clears ck_sub to re-align PPS epoch. ckspeed is software-programmable (addr 4); default 32'd2814750 yields nominal 1 PPS via ck_sub rollover. ck_sub[7] provides a subsecond blink cadence.

Event timing and strobes: all named strobes (ck_carry, ck_pps, ck_ppm, ck_pph, tm_int, al_int) are single-cycle pulses synchronous to i_clk. RTC HH:MM:SS advances on ck_pps; ck_ppm and ck_pph pulse at top-of-minute and top-of-hour to manage BCD carries. Countdown timer decrements on a PPS-derived strobe (tm_pps) and asserts tm_int for one cycle when reaching 00:00:00. Stopwatch subsecond increments on each ck_carry; seconds roll each 1-second boundary, cascading to minutes and hours. Alarm compares clock to alarm_time each i_clk; al_int pulses on match and al_tripped latches until cleared/disabled. o_interrupt is the OR of tm_int and al_int (single-cycle).

Display/LED timing: seven-segment digits are driven from a latched snapshot (ck_last_clock) captured around PPS to avoid races. The hexmap submodule is synchronous to i_clk and adds one-cycle latency from nibble input to segment output. Blink gating blanks display and status LEDs when ck_sub[7]=1 while tm_alarm or al_tripped are active. ledreg increments on ck_carry and resets at ck_ppm.

Bus and capture timing: Wishbone reads/writes are sampled on i_clk rising edges. Readback uses a latched snapshot ({clock[31:22], ck_last_clock}) for coherence across PPS boundaries. i_hack is sampled in the i_clk domain; capture aligns to a ck_carry boundary for subsecond precision, storing {clock[21:0], ck_sub} and the 40-bit ck_counter.

ARCHITECTURE

-- Time Base and PPS Generation --

Time base is derived from a 40-bit fractional-N accumulator updated each system clock: {ck_carry, ck_counter} <= ck_counter + zero-extended ckspeed. The accumulator carry (ck_carry) is the base tick feeding a subsecond divider (ck_sub). ck_sub increments on each ck_carry; when it rolls over, a one-cycle 1-PPS pulse (ck_pps) is emitted and used to advance the HH:MM:SS BCD clock. Top-of-minute (ck_ppm) and top-of-hour (ck_pph) events are detected at second boundaries for cascade carries. ckspeed is RW at bus address 4 (default 32'd2814750) and trims the effective PPS rate; increasing/decreasing ckspeed proportionally speeds/slows ck_pps. Writing seconds=0 clears ck_sub to realign PPS boundaries (resync). Derived strobes: tm_pps drives the countdown timer once per second; stopwatch subsecond follows ck_sub (advanced by ck_carry) and stopwatch seconds advance on ck_pps. Blink gating uses ck_sub[7]; LED activity increments on ck_carry and resets at top-of-minute. Timestamp captures align to ck_carry and record both {clock[21:0], ck_sub} and ck_counter for subsecond and accumulator-resolution time stamping.

-- Real-Time Clock Datapath --

Time base and tick generation:
- 40-bit fractional-N accumulator (ck_counter) advances by ckspeed each i_clk cycle; overflow generates ck_carry pulses (sub-ticks).
- Subsecond divider (ck_sub) counts ck_carry; its rollover asserts ck_pps (1 pulse per second). Minute/hour boundary strobes (ck_ppm, ck_pph) are derived from RTC field transitions at rollovers.

RTC registers and increment path (24-hour BCD):
- clock[21:0] packs HH:MM:SS in BCD: hours [21:16] 00–23, minutes [15:8] 00–59, seconds [7:0] 00–59; clock[31:22] are user/mode bits (not used by timekeeping).
- On each ck_pps: seconds increment with BCD carry into minutes at 59→00; minutes increment with BCD carry into hours at 59→00 (ck_ppm flags the minute rollover); hours increment at 23→00 (ck_pph flags the hour rollover when MM=00 and HH wraps).

Software interface (Wishbone):
- ckspeed (RW at addr 4) sets the accumulator step; default 32'd2814750; software-tunable to trim PPS rate.
- Time writes (ck_sel && i_wb_we) use per-byte semantics with sentinels: 0xFF for seconds/minutes and 0x3F for hours mean "no change"; any other value loads that field. Writing seconds=0 additionally clears ck_sub to re-align PPS phase.
- Readback (addr 0) returns {clock[31:22], ck_last_clock}, where ck_last_clock is a latched snapshot of clock[21:0] to ensure stable reads/display.

Derived/latch signals:
- ck_last_clock captures clock[21:0] for read/display stability.
- ck_pps is the single source of second-based advances; ck_ppm and ck_pph indicate minute/hour boundaries.

Separation from display/timestamp:
- Display logic (e.g., seven-segment) uses ck_last_clock and is outside the RTC datapath.
- A parallel "hack capture" path snapshots {clock[21:0], ck_sub} (and ck_counter) aligned to ck_carry

for precise timestamps; it is orthogonal to the RTC increment logic.

-- Countdown Timer Engine --

Countdown Timer Engine provides a Wishbone-controlled 1 Hz countdown with run/enable control, an expired/alarm latch, and an interrupt on expiration. The timer register is 26 bits: timer[23:0] holds BCD HH:MM:SS (max 99:59:59), timer[24] is the run/enable bit, and timer[25] is the expired/alarm latch. A shadow register tm_start captures the last nonzero programmed duration for quick reloads.

Operation: While run=1 and expired=0, the counter decrements once per second on tm_pps. BCD borrow rolls across SS→MM→HH with valid ranges (SS/MM 00–59, HH 00–99). When reaching 00:00:00 while running, timer[25] is set, tm_int pulses, and run auto-clears to halt the countdown and prevent underflow.

Programming model (Wishbone writes at tm_sel): If running (timer[24]=1), only the run bit is writable; HH:MM:SS remains unchanged. Writing run=0 stops the countdown; writing run=1 also clears the expired latch. If stopped (timer[24]=0), software may load HH:MM:SS. Writing a nonzero value loads timer[23:0] and copies it to tm_start. Writing zero when the counter is already zero reloads timer[23:0] from tm_start; writing zero when nonzero sets the counter to zero. Any write with run=1, or occurring while the prior run=1, clears timer[25].

Readback: Bus address 1 returns {6'h00, timer[25:0]} for status and current value.

Interrupts: tm_int is a pulse when the counter expires; the top-level o_interrupt is tm_int OR'd with the alarm interrupt. The expired latch (timer[25]) provides a sticky status for polling and user feedback.

User interface effects: Seven-segment mode 1 displays the timer as MM:SS, suppressing leading zeros unless expired. When expired (timer[25]=1), display and LEDs blink under ck_sub[7] gating.

Guards and constraints: Countdown halts on expiration (run auto-clears). BCD limits are enforced to maintain valid time ranges. Writes while running avoid changing HH:MM:SS to prevent race conditions. Max duration is 99:59:59.

-- Stopwatch Engine --

Purpose: Free-running, software-controlled stopwatch using the same fractional-N time base as the RTC. Accumulates subsecond, seconds, minutes, and hours; presents time in packed BCD with an hour-extension for runs beyond 99 hours.

Register/Bus: Readback at bus address 2 as stopwatch[31:0]. Bit 0 = run (R/W). Bits [7:1] = subsecond mirror of internal sw_sub[7:1] (R). Remaining bits contain packed BCD SS, MM, HH nibbles; hours extend beyond 99 via upper nibble [31:28]. Write decode via sw_sel.

Counting: When run=1, subsecond increments on each ck_carry from the fractional-N accumulator. On the one-second rollover, seconds (SS) increment in BCD, cascading to minutes (MM), then hours (HH). Hours use packed BCD in the lower two nibbles and continue past 99 by advancing the upper nibble [31:28] (may exceed BCD 9 when >99).

Software control: Writes with sw_sel && i_wb_we update run from bit0. Bit1 acts as a write-only clear request: if set, and the stopwatch is not running prior to the write or run=0 is written in the same cycle, the engine clears HH:MM:SS and all subsecond/pps edge flags. Writes do not set time fields; only run and conditional clear are acted upon.

Integration: No interrupt generation; independent of timer/alarm. Display mode 2 shows stopwatch MM:SS with leading-zero suppression on upper digits; subsecond is not displayed but remains visible in bits [7:1].

Reset/Initialization: No explicit reset behavior; software clears via the gated clear semantics when the engine is stopped. Downstream consumers must handle hours >99 via the extension nibble.

-- Alarm Comparator and Latches --

Compares the live 24-hour BCD clock value clock[21:0] (HH:MM:SS) against a programmable alarm_time[21:0], gated by the enable bit (al_enabled, write bit24). On each 1-PPS tick (ck_pps), if al_enabled=1 and clock==alarm_time, a single-cycle pulse al_int is generated and the status latch al_tripped is set. The pulse al_int contributes to the shared interrupt output: o_interrupt = tm_int | al_int. The compare naturally deasserts on the next second as the clock advances; al_int only pulses on the match edge. The al_tripped latch remains asserted until software clears it by writing bit25=1 (clear) or by disabling the alarm (writing al_enabled=0). Either action clears al_tripped and any pending/tripped state. Writes to the alarm register support per-field no-change sentinels: seconds and minutes fields interpret 0xFF as no update; hours interpret 0x3F as no update; bit24 updates al_enabled; bit25 requests a clear. Readback at bus address 3 provides {al_tripped, al_enabled, alarm_time} (upper bits zero-filled). While al_tripped=1, user interface indicators blink under ck_sub[7] (seven-segment display and LEDs). The comparator uses the live clock, not the display-latched value; changes to the base clock (e.g., writing clock seconds or ckspeed) affect when the next compare may occur.

-- Hack Timestamp Capture Path --

Captures an aligned snapshot of the internal time base when an external hack event occurs. Trigger: When i_hack is asserted high, the snapshot is taken on a ck_carry boundary—on the same ck_carry edge if coincident, otherwise on the immediately following ck_carry edge—bounding capture latency to ≤1 ck_carry period and aligning all fields to the same tick. Captured fields: (1) hack_time[29:0] = {clock[21:0], ck_sub[7:0]} where clock[21:0] is the BCD 24-hour HH:MM:SS and ck_sub[7:0] is the subsecond divider; (2) hack_counter[39:0] = the 40-bit fractional-N accumulator state (fine phase of the 1-PPS generator). Persistence: A new i_hack event overwrites the prior snapshot; values remain stable until the next capture. Readback (Wishbone): addr 5 returns {2'b00, hack_time[29:0]}; addr 6 returns hack_counter[39:8]; addr 7 returns {hack_counter[7:0], 24'h000000} (low 8 bits in [31:24]). Coherency: hack_time and the full 40-bit hack_counter are latched atomically at the capture instant; a subsequent capture between reads can change the values—software should avoid hack events during multiword reads or verify by rereading. Relation to time base: ckspeed (programmable at addr 4) sets the accumulator step and thus the meaning of hack_counter; hack_time carries BCD seconds plus subsecond divider for full subsecond correlation. Writing seconds = 0 resets the subsecond divider; captures at top-of-second will reflect ck_sub near zero. No separate clear/enable registers exist; i_hack solely governs capture.

-- Display Pipeline and hexmap Decoder --

Display Pipeline
- Source select: mode selects one of four packed-BCD sources to form four nibbles (MS digit to LS digit) for output:
- 0/other: HH:MM from ck_last_clock (blank leading zero on hour tens).
- 1: Timer MM:SS; suppress leading zeros on upper digits while running; when expired (timer[25]==1), show all digits.
- 2: Stopwatch MM:SS; suppress leading zeros on upper digits.
- 3: Clock MM:SS from the latched display/readback image.

- Digit extraction: each of the four digits takes its BCD [3:0] nibble from the chosen source.
- Decode stage: each nibble feeds hexmap; outputs registered 7-bit active-high [a b c d e f g].
- Gating: after decode, apply dmask for leading-zero blanking per mode; apply global blink gate when tm_alarm or al_tripped is set, blanking all digits when ck_sub[7]==1 (~1 Hz, ~50% duty).
- Pack/output: combine each 7-bit pattern with an optional DP bit (default 0) to form four 8-bit words; drive o_sseg as a static 4×8-bit bus.

hexmap Decoder
- Function: synchronous 16×7 ROM + output register mapping 0–F to segment bits [a..g], active-high.
- Latency: 1 clock cycle from nibble input to registered segment output.
- Polarity: for common-anode displays, invert externally.

Timing and coherency
- Use ck_last_clock for clock-sourced modes to avoid tearing at second boundaries; aligned with bus readback at address 0.
- Place dmask/blink gating after the registered hexmap output; register gating as needed so all four digits remain cycle-aligned with the 1-cycle decode latency.
- o_sseg is static; external hardware may handle multiplexing/common-anode/cathode specifics.
- DP/colon behavior is unspecified; keep DP=0 unless higher-level logic drives it.
- Blink gating is derived from the same subsecond divider as 1 PPS; changing ckspeed affects both PPS and blink rate.

-- LED Activity Generator --

Generates a continuously evolving LED activity pattern from the RTC time base and overlays a visible blink to indicate pending alarm or timer events. Inputs: ck_carry (fractional-N accumulator carry tick), ck_sub[7] (subsecond bit for blink gating), ck_ppm (top-of-minute pulse), tm_alarm (timer expired latch), al_tripped (alarm match latch). Internal state: ledreg, a free-running register incremented on each ck_carry and synchronously cleared on ck_ppm to align the activity pattern every minute; no explicit asynchronous reset—state becomes well-defined after initial timebase activity. Output behavior: o_led is an LED vector driven by a function of ledreg; with no alerts active (tm_alarm=0 and al_tripped=0), o_led presents a continuously changing activity pattern; with any alert active (tm_alarm=1 or al_tripped=1), o_led is blink-gated by ck_sub[7] such that LEDs are blanked when ck_sub[7]=1 and otherwise show the activity/status pattern; clearing the latches halts the blink and returns to normal activity. Interactions: writing the RTC seconds field to 0 resets the subsecond divider and phase-resets blink timing; the programmable tick rate (ckspeed) affects both the ledreg increment rate and blink cadence because both derive from the same timebase. Notes: o_led width and the exact mapping from ledreg and status to individual LEDs are implementation-defined; blink behavior mirrors the seven-segment display gating via ck_sub[7].

-- Bus Address Decode and Readback --

32-bit bus; eight word locations are selected by i_wb_addr[2:0]. Reads return the following o_data values; unless stated otherwise, reads have no side effects and do not clear latches:
- Addr 0 (Clock snapshot and user/mode): o_data = {clock[31:22], ck_last_clock[21:0]}. clock[31:22] hold user/mode (incl. display mode). ck_last_clock[21:0] is a latched BCD HH:MM:SS (24-hour) snapshot taken to avoid races at second boundaries.
- Addr 1 (Countdown timer): o_data = {6'h00, timer[25:0]}. timer[25] = expired/alarm latch; timer[24] = run/enable; timer[23:0] = BCD HH:MM:SS (max 99:59:59).
- Addr 2 (Stopwatch): o_data = stopwatch[31:0]. stopwatch[0] = run; stopwatch[7:1] = subsecond bits; remaining nibbles are BCD SS, MM, HH (hours may extend via [31:28]).
- Addr 3 (Alarm status/target): o_data = {6'h00, al_tripped, al_enabled, 2'b00, alarm_time[21:0]}.

alarm_time is BCD HH:MM:SS (24-hour).
- Addr 4 (Tick-rate control): o_data = ckspeed[31:0] (fractional-N accumulator step; read returns current programmed value).
- Addr 5 (Hack capture – time stamp): o_data = {2'b00, hack_time[29:0]}, where hack_time = {clock[21:0], ck_sub[7:0]} captured on i_hack (or next carry). ck_sub is 8 bits.
- Addr 6 (Hack capture – counter upper): o_data = hack_counter[39:8].
- Addr 7 (Hack capture – counter lower): o_data = {hack_counter[7:0], 24'h000000}. The 40-bit hack_counter can be reconstructed as {word6, word7[31:24]}.
Atomicity/consistency: Addr 0 returns a latched time image (ck_last_clock) ensuring a stable HH:MM:SS during multi-word reads. Hack capture words (5–7) form a coherent snapshot taken on the i_hack event; readback does not clear or re-arm the capture. Status latches such as al_tripped and the timer expired bit clear only via writes, not reads. Higher-level address mapping beyond i_wb_addr[2:0] and any bus handshake/ack timing are outside this module's scope. hexmap is unrelated to bus access.

-- Parameterization and Constants --

- Design is static (no compile-time parameters); all field widths, bit positions, and the Wishbone address map are fixed.
- Clock/time base constants:
- Fractional-N accumulator: 40 bits (ck_counter[39:0]).
- Tick step (ckspeed): 32 bits; reset default = 32'd2814750; upper 8 bits of the 40-bit addend are implicitly zero.
- Subsecond divider: 8 bits (ck_sub[7:0]); 1PPS (ck_pps) asserted on ck_sub rollover; base tick strobe from accumulator carry (ck_carry).
- Real-time clock register (clock[31:0]):
- Time: clock[21:0] = {HH[7:0], MM[7:0], SS[7:0]} packed BCD; HH = 00–23, MM/SS = 00–59.
- Mode/user bits: clock[31:22]; display mode at clock[27:24].
- Write sentinels (no-change): SS=8'hFF, MM=8'hFF, HH=8'h3F; writing SS=0 clears subsecond divider.
- Countdown timer (timer[25:0]):
- timer[23:0] = BCD HH:MM:SS (00:00:00–99:59:59), timer[24] = run/enable, timer[25] = expired/alarm latch.
- Stopwatch (stopwatch[31:0]):
- stopwatch[0] = run; stopwatch[1] = clear; stopwatch[7:1] = subsecond mirror; remaining bits = BCD SS, MM, HH (hours can extend past 99 using [31:28]).
- Alarm (alarm_time and control):
- alarm_time[21:0] = BCD HH:MM:SS; al_enabled at bit24; clear/trip reset at bit25.
- Write sentinels: SS=8'hFF, MM=8'hFF, HH=8'h3F.
- Display/LED constants:
- Display mode codes (clock[27:24]): 0=show HH:MM (blank leading zero), 1=timer MM:SS (blank leading zeros unless expired), 2=stopwatch MM:SS (blank upper leading zeros), 3=clock MM:SS.
- Blink source: ck_sub[7] (active = blank when alarm/timer active).
- o_sseg: 4×8-bit bus (four digits). Segments are active-high; invert externally for common-anode.
- Interrupts: o_interrupt = al_int | tm_int.
- Hack capture:
- hack_time: 30 bits = {clock[21:0], ck_sub[7:0]}; read at addr 5 as {2'b00, hack_time}.
- hack_counter: 40 bits; read addr 6 as hack_counter[39:8]; addr 7 as {hack_counter[7:0], 24'h000000}.
- Wishbone bus map (i_wb_addr[2:0]):
- 0: {clock[31:22], ck_last_clock[21:0]}
- 1: {6'h00, timer[25:0]}
- 2: stopwatch[31:0]
- 3: {6'h00, al_tripped, al_enabled, 2'b00, alarm_time[21:0]}

- 4: ckspeed[31:0]
- 5: {2'b00, hack_time[29:0]}
- 6: hack_counter[39:8]
- 7: {hack_counter[7:0], 24'h000000}
- hexmap submodule constants:
- Input: 4-bit (0–F); output: 7-bit segments [a b c d e f g], active-high; synchronous 1-cycle latency; fixed 16-entry LUT.
- Notes and parameterization candidates:
- Subsecond width confirmed as 8 bits by hack_time packing; hours sentinel HH=0x3F is a 6-bit pattern within an 8-bit lane (masking behavior in RTL should be verified).
- No explicit reset defaults beyond ckspeed; avoid assuming others.
- Potential future parameters: accumulator width, subsecond width, number of 7-seg digits, segment polarity.

OPERATION

-- Initialization and Reset Behavior --

- Reset source/polarity
- Not specified in the RTL description. Treat all registers as undefined at reset unless noted; verify actual reset type/polarity in code.
- Time base and tick generation
- ckspeed has a defined reset/default of 32'd2814750 and is software-programmable thereafter.
- Fractional-N state (e.g., ck_counter, ck_sub, carry/ck_pps strobes) is unspecified at reset; assume cleared/idle behavior until accumulation begins under the default/programmed ckspeed.
- First 1-PPS (ck_pps) will not occur until sufficient accumulation; subseconds can be realigned by writing clock seconds=00, which clears the subsecond divider.
- Real-time clock (clock[31:0])
- Reset value not defined; software must write HH:MM:SS after reset. For precise subsecond alignment, write seconds=00 (other fields may use sentinel/no-change rules per design).
- ck_ppm/ck_pph (top-of-minute/hour) occur only after time advances via ck_pps.
- Latched display/readback (ck_last_clock) is undefined at reset; becomes valid after the first latch/update cycle.
- Countdown timer (timer[25:0])
- Counter, run bit (bit24), and expired/alarm latch (bit25) reset values are unspecified; software must initialize.
- Any write with run=1 (or a prior run=1) clears the expired latch. If stopped, writing a nonzero value loads the counter and tm_start; writing zero when the counter is zero reloads tm_start.
- Stopwatch (stopwatch[31:0])
- Reset state unspecified; assume run=0 and initialize explicitly.
- Writing with bit1=1 (clear) while not running (or with run=0 in the same write) clears time and subsecond/pps flags.
- Alarm and interrupts
- Alarm enable (bit24) and tripped (bit25) reset states are unspecified; software must set desired HH:MM:SS, enable/disable, and clear tripped on init.
- o_interrupt stays low except on timer expiry (tm_int) or alarm match edge (al_int). With alarm disabled by default (recommended), interrupts are quiescent after reset.

- Hack (timestamp) capture
- hack_time and hack_counter are undefined until the first i_hack capture; do not use until a capture occurs.
- Seven-segment display, LEDs, and hexmap
- Display mode from clock[27:24] has no defined reset value; if zero, default behavior is to show HH:MM from ck_last_clock. Before the first valid latch/update, digits may be blank/indeterminate.
- Blink behavior (e.g., on timer/alarm) depends on ck_sub[7]; behavior before subsecond ticking starts is undefined.
- LED activity counter (ledreg) increments with ck_carry and resets at top-of-minute; initial value is unspecified.
- hexmap submodule has no reset; o_map is unknown until the first clocked update after i_hex is presented (one-cycle latency). Gate display enable or provide valid nibbles during startup.
- Bus readback after reset
- Address 4 (ckspeed) reads the default 32'd2814750 until reprogrammed.
- Other readable registers may return zeros or undefined values until initialized or updated internally (e.g., hack_* before capture, stopwatch/timer/clock before software write).
- Recommended software initialization sequence
- Program ckspeed if needed.
- Write clock HH:MM:SS with seconds=00 to align subseconds; set display mode in clock[27:24] if used.
- Load and (optionally) start the timer; starting clears expired latch.
- Set/clear and enable the alarm as desired.
- Clear and/or start the stopwatch as needed.
- Ignore hack_* readbacks until at least one i_hack capture has occurred.

-- Programming Model and Access Conventions --

Bus and addressing: Wishbone, 32-bit data words; i_wb_addr[2:0] selects one of 8 registers (0–7). Standard single-word accesses are assumed; endianness not specified. All time fields are packed BCD per byte (tens nibble [7:4], ones nibble [3:0]). Hours ranges: clock 00–23, timer 00–99, stopwatch hours can exceed 99 using bits [31:28]. Display control: clock[27:24] selects seven-segment display source/mode. Read atomicity: RTC read returns a latched snapshot (ck_last_clock) of HH:MM:SS to avoid second-boundary races; hack capture produces a coherent snapshot across addrs 5–7. Register access semantics:
- Addr 0 (RTC, RW): Read returns {clock[31:22], ck_last_clock[21:0]} where ck_last_clock is a stable HH:MM:SS snapshot. Write per-field updates using sentinels: seconds/minutes 0xFF = no change, hours 0x3F = no change; write packed BCD values otherwise. Writing seconds=0 also resets the subsecond divider to re-align PPS. Bits [31:22] include user/mode (display mode in [27:24]).
- Addr 1 (Countdown timer, RW): Read {6'h00, timer[25:0]} with timer[23:0]=BCD HH:MM:SS (00:00:00–99:59:59), timer[24]=run, timer[25]=expired latch. Write: if running (run=1), only the run bit can be changed; HH:MM:SS writes are ignored. If stopped (run=0), writing nonzero HH:MM:SS loads the counter and captures tm_start; writing 00:00:00 when the counter is zero reloads tm_start. Clearing expired latch: any write with run=1 (or if prior run was 1) clears timer[25]. Interrupt tm_int asserts when the counter hits zero while running.
- Addr 2 (Stopwatch, RW): Read returns [0]=run, [7:1]=subsecond mirror, [15:8]=SS (BCD), [23:16]=MM (BCD), [31:24]=HH (BCD; [31:28] extend beyond 99). Write: set/clear run via bit0. Clear via bit1: if bit1=1 and stopwatch is not running, or run=0 in the same write, clear time and all subsecond/PPS flags; clearing while run=1 is inhibited.
- Addr 3 (Alarm, RW): Read {6'h00, al_tripped (bit25), al_enabled (bit24), 2'b00, alarm_time[21:0]} (HH:MM:SS BCD). Write per-field sentinels as for RTC (sec/min 0xFF, hours 0x3F). bit24 enables/disables comparison; bit25=1 clears al_tripped and related latches. On HH:MM:SS match with al_enabled=1, al_tripped latches and al_int pulses.

- Addr 4 (ckspeed, RW): 32-bit accumulator step controlling base tick and PPS rate; default 32'd2814750. Program to trim 1-PPS accuracy.
- Addr 5 (Hack time, RO): {2'b00, hack_time[23:0]} = {clock[21:0], ck_sub[...]}; captured when i_hack asserts (aligned to a ck_carry boundary).
- Addr 6 (Hack counter high, RO): hack_counter[39:8].
- Addr 7 (Hack counter low, RO): {hack_counter[7:0], 24'h000000}.
Interrupts: o_interrupt = tm_int | al_int. Timer expired status is latched in timer[25] until cleared via write with run=1. Alarm match pulses al_int and latches al_tripped until cleared via bit25=1 or by disabling (bit24=0).
Recommended access sequences: Set RTC time by writing HH:MM:SS to addr 0 using 0xFF/0x3F sentinels; write seconds=0 to re-align PPS if needed. Configure seven-seg display by writing clock[27:24] at addr 0. Start a countdown: ensure timer run=0, write nonzero HH:MM:SS to addr 1, then set run=1; to reload initial value after expiry, write HH:MM:SS=0 when counter is zero, then set run=1. Clear timer expired by writing with run=1 at addr 1. Set an alarm by writing alarm_time and setting al_enabled=1 at addr 3; clear al_tripped by writing bit25=1 or disabling al_enabled. Use the stopwatch by writing bit0 at addr 2 to start/stop; to clear, write with run=0 and bit1=1. For precise timestamps, assert i_hack externally and then read addrs 5–7 for the captured snapshot.

-- Wishbone Read/Write Transactions --

Bus conforms to a 32-bit Wishbone classic single-word access model. i_wb_addr[2:0] select one of eight 32-bit registers; upper address bits are ignored by the local decode. Reads use i_wb_we=0; writes use i_wb_we=1.

Handshake and response
- Transactions are accepted when CYC and STB are asserted. A single o_wb_ack pulse is returned per accepted access. o_wb_stall is not expected to be asserted (back-to-back accesses are supported); o_wb_err is not used. Verify exact ack/stall timing in RTL.
- Read data on o_wb_data is valid when o_wb_ack is asserted.

Read side-effects
- Reads do not clear latches or modify state. Address 0 returns a display-safe time snapshot by including ck_last_clock to avoid HH:MM:SS race conditions; all other registers are side-effect free on read.

Write semantics
- Writes complete on o_wb_ack and update only the targeted register per address-specific rules:
• Addr 0 (clock): Per-byte updates with in-band no-change sentinels (seconds/minutes byte=0xFF; hours byte=0x3F). Writing seconds byte=0x00 also clears the subsecond divider to re-align PPS. Bits [31:22] are user/mode RW.
• Addr 1 (timer): If running (bit24 was 1 before write), only bit24 (run) is writable; HH:MM:SS bytes are ignored. If stopped, writing HH:MM:SS≠0 loads new BCD HH:MM:SS and captures tm_start; writing 0 when the counter is already 00:00:00 reloads tm_start. Expired latch (bit25) is cleared by any write that sets run=1 or any write performed while prior run was 1; otherwise RO.
• Addr 2 (stopwatch): Bit0=run (1 run, 0 stop). Bit1=clear request; effective only if stopwatch was not running prior to the write or if the same write sets run=0. Upper bits ignored on write; time advances only via hardware.
• Addr 3 (alarm): Time bytes use the same per-field no-change sentinels as the clock. Bit24 (al_enabled) gates comparison; writing 0 also clears al_tripped. Bit25=1 clears al_tripped (write-1-to-clear). al_tripped is RO on read.
• Addr 4 (ckspeed): 32-bit RW fractional-N step for 1-PPS.
• Addrs 5–7 (hack capture): RO; writes are ignored without side-effects.

Byte enables and lane ordering
- If i_wb_sel is implemented and honored, unselected byte lanes must not change state on write. Independently of i_wb_sel, clock and alarm time fields support in-band no-change sentinels as described above. Byte lane ordering within the 32-bit word is little-endian: seconds [7:0], minutes [15:8], hours [21:16], user/mode bits [31:22].

Error and undefined accesses
- Only addresses 0–7 are defined. Accesses outside this decode range are implementation-defined; no bus error is expected. Verify behavior in RTL.

Side-effect policy
- No read clears latches or acknowledges events. All clears and state changes occur only via the documented write bits.

-- Clock Setting and No-Change Sentinels --

- Write format: Packed BCD HH:MM:SS across a 32-bit clock register.
- seconds: clock[7:0]
- minutes: clock[15:8]
- hours: clock[21:16]
- user/mode: clock[31:22] (not subject to sentinel logic)
- No-change sentinels (per byte):
- seconds 0xFF, minutes 0xFF, hours 0x3F.
- On write, any field byte equal to its sentinel leaves the stored field unchanged; any non-sentinel byte replaces the stored field with the written BCD value.
- PPS alignment on seconds write:
- Writing seconds = 0x00 updates seconds to 00 and also clears the subsecond divider (ck_sub) to realign to the 1-PPS boundary.
- Writing seconds to any nonzero value does not clear the subsecond divider.
- Atomic partial updates:
- A single 32-bit write can modify any subset of HH/MM/SS by using sentinels on the bytes to be preserved.
- If all three time bytes are sentinels, only clock[31:22] are affected by the write.
- Valid BCD ranges (expected): seconds and minutes 00–59, hours 00–23. Behavior for values outside these ranges is unspecified by this spec (no range enforcement implied).
- Running-time behavior:
- Time fields advance on ck_pps when not being written.
- Writes do not otherwise affect the running counters; only the seconds=0 case resets the subsecond divider.
- Alarm programming (if present) mirrors this sentinel scheme on alarm_time[21:0]:
- seconds/minutes 0xFF, hours 0x3F.
- Allows updating control bits (e.g., al_enabled at bit24, clear at bit25) without changing stored HH:MM:SS when all three time bytes are sentinels.

-- Timer Load/Run/Reload/Clear --

Timer supports load, run, reload, and clear via a single 26-bit register: timer[25]=expired (hardware latch, read-only to software), timer[24]=run (enable, RW), timer[23:0]=HH:MM:SS in packed BCD (00–99:59:59). Operation: while run=1 and expired=0, the counter decrements once per second; on reaching 00:00:00, hardware sets expired=1, clears run=0, and asserts the timer interrupt. Writes while running (prior run=1): only the run bit is honored; HH:MM:SS is ignored; any such write clears expired.

Writes while stopped (run=0): HH:MM:SS is writable; writing a nonzero value loads the counter and captures it into the reload shadow tm_start; writing 00:00:00 when the counter already reads 00:00:00 performs a reload from tm_start; run may be changed in the same write, and if set to 1, expired is cleared. Expired is set only by hardware on timeout and is cleared by any write that sets run=1 or by any write performed while the timer was previously running. Reload model: tm_start holds the last nonzero HH:MM:SS written while stopped; reload occurs only via the 00:00:00-at-00:00:00 write while stopped. BCD validity: HH, MM, SS nibbles must be valid BCD; HH range is 00–99. Ambiguity: behavior for writing 00:00:00 while stopped when the counter is nonzero is unspecified.

-- Stopwatch Run/Clear and Subsecond Handling --

Bus address 2 exposes a 32-bit stopwatch register: [0]=run/enable, [1]=clear (write-only control), [7:1]=mirror of internal subsecond counter sw_sub[7:1], followed by packed-BCD SS, MM, and HH (hours can extend via [31:28]). Run behavior: when run=1, sw_sub increments on each ck_carry; rollover generates ck_pps (1PPS) that advances BCD seconds, then minutes, then hours with proper carry. Stop behavior: when run=0, subsecond and time values hold. Clear behavior: a write with clear=1 zeroes HH:MM:SS and resets sw_sub plus PPS/edge/latch state only if the stopwatch was not running at the time of the write or the write sets run=0; if already running and run remains 1, clear is ignored. If clear=1 is written while stopped and the write sets run=1, the effect is clear-then-start from zero, ensuring the next second boundary occurs after a full subsecond interval. Subsecond exposure: bits [7:1] provide a live 7-bit view of subsecond progress; a successful clear zeroes these bits and related latches. Rate coupling: ckspeed scales ck_carry and ck_pps; a successful clear realigns subsecond/PPS state to the current rate. Only run and clear are writable; time fields are read-only.

-- Alarm Enable/Clear/Trip --

Alarm has three controls: Enable, Clear, and Trip (latched status).\n- Enable (write bit24 to alarm register): Sets al_enabled. If written 0, disables comparison and immediately clears al_tripped. If written 1, enables comparison against the real-time HH:MM:SS.\n- Clear (write bit25=1 to alarm register): Clears the al_tripped latch without affecting al_enabled or alarm_time. Writing 0 to bit25 has no effect.\n- Trip latch (al_tripped): When al_enabled=1 and the clock HH:MM:SS equals alarm_time, al_tripped sets (latches) and remains set until cleared by write bit25=1 or by disabling (al_enabled=0).\n- Match/interrupt behavior: Trip detection is edge-based; al_int pulses once on the transition to a match. While al_tripped is set, no retrigger occurs. Re-enabling after a disable does not retro-trigger if the time is already equal; a fresh match edge is required. o_interrupt = tm_int | al_int.\n- Readback (addr 3): bit25 reports al_tripped status; bit24 reports al_enabled; bits [21:0] report alarm_time; other bits read as zero. Note: write bit25 is a clear command, not a latched control—do not confuse write-clear with read-status.\n- Write/update semantics: When writing the alarm register, packed BCD seconds/minutes fields use 0xFF and hours uses 0x3F as no-change sentinels, allowing enable/clear updates without altering alarm_time.\n- UI/LED effects: When al_tripped=1, seven-segment display is blanked in a blink pattern and LEDs blink; blinking is gated by ck_sub[7] and shared with timer expiry.

-- Interrupt Handling --

- Sources: tm_int (countdown timer expiry) and al_int (alarm time match). No other features (stopwatch, hack capture) generate interrupts.
- Output: o_interrupt is the logical OR of tm_int and al_int. It is a short pulse, synchronous to the module clock and aligned with the 1 PPS second boundary.
- Timer interrupt (tm_int): Generated when the countdown decrements to 00:00:00 while timer[24] (run) = 1. The pulse is non-sticky, but the timer expired latch timer[25] sets and persists. On expiry, run auto-clears.
- Clearing timer status: timer[25] (expired) is cleared by a software write that sets run=1 (and typically

reloads the timer). This action restarts the timer and clears the latch.
- Alarm interrupt (al_int): Generated on the edge when the current HH:MM:SS equals the programmed alarm time and al_enabled = 1. The pulse is non-sticky; the latch al_tripped sets and persists.
- Clearing alarm status: al_tripped is cleared by a write with bit25=1 (alarm clear) or by disabling al_enabled.
- Masking/enable: Alarm comparisons are gated by al_enabled. Timer events only occur when run=1. There is no global interrupt mask or acknowledge handshake.
- Simultaneous events: If timer expiry and alarm match occur in the same second, o_interrupt asserts once (OR), and both timer[25] and al_tripped may be set. Software must read status to determine the cause(s).
- Software handling: On o_interrupt, read timer/status at addr 1 (includes timer[25] expired and timer[24] run) and alarm/status at addr 3 (includes al_tripped, al_enabled, and alarm time). Clear per-feature latches as described.
- Timing domain: All events are synchronous and aligned to the internal 1 PPS. Writes that adjust time (e.g., seconds=0) reset the subsecond divider and may shift subsequent PPS/interrupt alignment.

-- Tick-Rate Programming --

- Mechanism: A 40-bit fractional-N accumulator advances each clock: {ck_carry, ck_counter} <= ck_counter + ckspeed. ck_carry pulses are sub-second ticks. An 8-bit divider (ck_sub) counts ck_carry; its overflow asserts ck_pps (1-PPS target).
- Register: ckspeed is a 32-bit RW register at Wishbone read address index 4; reset/default is 32'd2814750 (for 100 MHz). Writes take effect immediately.
- Frequency: f_carry = f_clk * ckspeed / 2^40; f_pps = f_carry / 256 = f_clk * ckspeed / (2^40 * 256).
- 1-PPS programming: ckspeed = round(2^40 * 256 / f_clk). Example: f_clk=100 MHz -> ckspeed≈2,814,750.
- Resolution: One LSB of ckspeed changes f_pps by ≈ f_clk / (2^40 * 256). At 100 MHz, ~3.55e-7 Hz (~0.355 ppm per LSB).
- Calibration: If measured error is Δppm (positive if fast), set ckspeed_new ≈ ckspeed_nominal * (1 – Δppm/1e6), then round to integer.
- Operational notes: Writing seconds=0 clears ck_sub, allowing second-boundary realignment (e.g., after ckspeed changes or syncing to external 1-PPS). ckspeed=0 halts ck_carry/ck_pps; larger values speed time proportionally.
- System effects: RTC seconds tick on ck_pps; countdown timer decrements on ck_pps; stopwatch subsecond uses ck_carry and seconds use ck_pps; display/LED blink (ck_sub[7]) is ~0.5 s only when ckspeed yields 1-PPS.

-- Display Mode Selection and Blinking --

- Source and selection
- Display mode is selected by clock[27:24]. Any unrecognized value falls back to Mode 0.
- Modes
- Mode 0 (Clock HH:MM): Shows real-time clock hours and minutes from the latched snapshot ck_last_clock to avoid second-boundary glitches. The leading hour digit is blanked if it is zero.
- Mode 1 (Timer MM:SS): Shows countdown timer minutes and seconds. While the timer is running, suppress leading zeros on the left two digits. When the timer has expired (timer[25]=1), show all four digits (no suppression).
- Mode 2 (Stopwatch MM:SS): Shows stopwatch minutes and seconds; suppress leading zeros on the left two digits.
- Mode 3 (Clock MM:SS): Shows real-time clock minutes and seconds.
- Zero suppression
- Suppression is applied only to the upper (left) digits per the mode rules above and is disabled in Timer mode when expired. Digit masking is applied before any blink masking.

- Blinking behavior
- Trigger: Blink whenever the countdown timer has expired (timer[25]=1) or the time-of-day alarm is tripped (al_tripped=1).
- Mechanism and rate: The entire 4-digit display is blanked when ck_sub[7]=1 and shown when ck_sub[7]=0, yielding approximately 1 Hz blink (about 0.5 s on, 0.5 s off). The blink phase is tied to the RTC second (1-PPS) and tracks any ckspeed trimming.
- Priority/order: Glyph generation and zero suppression occur first; blink masking is applied last and overrides all content. During the blank phase, all segments are off regardless of mode or value.
- Notes
- Only Mode 0 explicitly uses ck_last_clock; other modes use live values. The o_sseg bus is 4×8 bits; only 7 segment bits are defined here (decimal point/colon usage is unspecified).

-- LED Behavior --

The LED output bus (o_led) provides a visual heartbeat and status tied to the real-time clock core. In normal operation, o_led reflects an activity pattern derived from a free-running LED register (ledreg): ledreg increments on each fractional-N carry (ck_carry) and is synchronously cleared at the top-of-minute event (ck_ppm), producing a monotonic, human-visible progression across the minute. When either the alarm has tripped (al_tripped=1) or the countdown timer has expired (timer[25]=1), this steady display is overridden and the LEDs blink at approximately 1 Hz by gating with ck_sub[7]; the blink is synchronized with the seven-segment display behavior and its exact frequency tracks PPS timing and any ckspeed (tick-trim) adjustments. The mapping of ledreg bits to o_led, the bus width and polarity, and whether blink manifests as blanking or toggling are defined in the RTL; upper ledreg bits are expected to be used for visibility. ledreg resets on each ck_ppm; no additional LED reset behavior is specified. Confirm if running timer status (e.g., timer[24]) or alarm enable (al_enabled) affects o_led when not tripped.

-- Hack Capture Triggering and Readback --

Hack capture provides a coherent snapshot of the RTC time and the fractional-N accumulator, triggered by i_hack and aligned to the ck_carry boundary for maximum precision. Triggering: When i_hack is observed high, the snapshot is taken on the next ck_carry edge; if i_hack is high exactly at a ck_carry edge, capture occurs on that same edge. Each new capture overwrites the previous; reads do not clear or acknowledge the snapshot. Captured fields: hack_time (30 bits) = {clock[21:0], ck_sub[7:0]} where clock[21:0] is BCD HH:MM:SS (hours [21:16], minutes [15:8], seconds [7:0]) and ck_sub[7:0] is the subsecond divider state (0–255) at the boundary; hack_counter (40 bits) is the fractional-N accumulator aligned to the same boundary for fine calibration against ckspeed. Readback (Wishbone, 32-bit words): Addr 5 returns {2'b00, hack_time}; Addr 6 returns hack_counter[39:8]; Addr 7 returns {hack_counter[7:0], 24'h000000}. Usage and coherency: Assert i_hack as a pulse, then read addr 6, addr 7, and addr 5 in one critical section. Prevent another hack event between these reads to avoid tearing across multiword values; no queuing or valid/ready status is provided. Notes: ck_sub rolls over each second (drives the 1 PPS increment); writing seconds=0 in RTC write path clears the subsecond divider, affecting subsequent captures. The combination of {HH:MM:SS, ck_sub} uniquely identifies the fractional second at the boundary, while hack_counter enables reconstruction of precise timing relative to ckspeed.

-- Corner Cases and Error Handling --

- Tick rate and PPS
- Writing ckspeed=0 halts all time progression (RTC, timer, stopwatch), LED activity, and display blink; no error flag is raised.
- Very large ckspeed values saturate at at most one carry per i_clk; multi-carry per cycle does not occur

and no error is indicated.
- Writing RTC seconds=0 clears the subsecond divider and realigns PPS; the first interval may be short/long without error reporting.
- Writes coincident with a PPS boundary can race with carry/compare; behavior is defined only for seconds=0 subsecond clear. Software should avoid boundary writes when relying on alarm/timer compares.

- Range/format validation
- No BCD or range checking is enforced for RTC, timer, stopwatch, or alarm fields; out-of-range digits are accepted and may cause undefined rollover/decrement. Software must validate inputs; no error flags are provided.

- Read atomicity
- RTC readback uses a latched snapshot (ck_last_clock) to avoid mixed-digit reads near PPS; timer/stopwatch/hack are not latched and can change between multiword reads. To ensure atomicity, stop the counter or avoid triggering events during reads.

- Countdown timer corner cases
- While running, HH:MM:SS writes are ignored; only run bit is writable (no error feedback).
- Writing zero when the counter is zero reloads from tm_start; writing zero when nonzero sets the counter to 00:00:00 (no explicit reload error indication).
- On expiration, run auto-clears and the expired latch sets; starting the timer clears the expired latch implicitly.

- Stopwatch corner cases
- Clear is ignored if the stopwatch is running and remains run=1 in the same write; to clear, ensure prior run=0 or write run=0 with clear.
- Hours can extend beyond 99 into upper bits without overflow indication; display modes may hide long durations.
- If ckspeed=0, stopwatch does not advance; no error is reported.

- Alarm corner cases
- Strict equality compare only; jumping RTC time across the target can miss the match without error. al_int is a pulse; rely on al_tripped for sticky status. Disabling the alarm gates compare and clears the latch.

- Hack (timestamp) capture
- i_hack is level-sampled on ck_carry boundaries; if held high, repeated captures may occur (no one-shot guarantee).
- i_hack should be synchronized to i_clk externally; no internal debouncing/synchronization, so metastability is possible if asynchronous.
- Multiword hack_time/counter are captured together; a new capture mid-read updates both words—avoid triggering during read to prevent perceived inconsistency.

- Display and LEDs
- Seven-segment outputs have one-cycle latency and may be undefined immediately after reset; no error flag.
- Leading-zero suppression can blank upper digits; blink depends on ck_sub and ckspeed. With ckspeed=0, blinking and LED activity stop.
- LED activity counter resets only at natural top-of-minute detection, not on writes or ckspeed changes.

- Interrupts and pulses
- o_interrupt is a pulse (OR of alarm/timer pulses); pulses may be missed. Use sticky latches (timer

expired, al_tripped) for reliable event detection.

- Write semantics
- Byte-level writes with per-field no-change sentinels (RTC/alarm sec/min=0xFF, hour=0x3F); omitted fields remain unchanged. No error on partial writes.

- Accumulator limits
- ckspeed accumulation is effectively 40-bit; values intending >1 carry/clk are ineffective beyond saturation. No error or warning is provided.

REGISTERS

-- Address Map --

Wishbone 32-bit address map (i_wb_addr[2:0] selects word offsets 0–7; base address defined by integration).
- 0x0 — RTC Time/Mode (R/W): Read returns {clock[31:22], ck_last_clock[21:0]} where ck_last_clock is a read latch of clock[21:0] (packed BCD HH:MM:SS). Write updates clock[21:0] with sentinels (seconds/minutes byte 0xFF=no change; hours nibble 0x3F=no change) and writes clock[31:22] (mode/user). Writing seconds=0 resets the subsecond divider.
- 0x4 — Countdown Timer (R/W): Read returns {6'h00, timer[25:0]} with timer[23:0]=BCD HH:MM:SS, timer[24]=run, timer[25]=expired latch. Write: if running, only bit24 (run) is writable; if stopped, load timer[23:0] (nonzero loads and captures to tm_start; writing zero when the counter is zero reloads tm_start). Any write with run=1, or if run was previously 1, clears the expired latch.
- 0x8 — Stopwatch (R/W): Read returns stopwatch[31:0] with bit0=run, bits[7:1]=subsecond mirror, remaining nibbles are BCD SS, MM, HH (HH extends beyond 99 via bits[31:28]). Write sets run; if bit1 (clear) is set and either it wasn’t running or run=0 in the same write, clears time and subsecond/pps flags.
- 0xC — Alarm (R/W): Read returns {6'h00, al_tripped (bit25), al_enabled (bit24), 2'b00, alarm_time[21:0]} where alarm_time is packed BCD HH:MM:SS. Write updates alarm_time with sentinels (sec/min 0xFF, hours 0x3F), sets al_enabled (bit24), and clears al_tripped by writing bit25=1; disabling also clears al_tripped.
- 0x10 — ckspeed (R/W): 32-bit accumulator step controlling 1-PPS trim/speed (default 32'd2814750).
- 0x14 — Hack Time (RO): Read returns {2'b00, hack_time[29:0]} where hack_time={clock[21:0], ck_sub[7:0]} captured on i_hack or the next carry edge.
- 0x18 — Hack Counter MSW (RO): Read returns hack_counter[39:8], the upper 32 bits of the 40-bit counter captured with hack_time.
- 0x1C — Hack Counter LSW (RO): Read returns {hack_counter[7:0], 24'h00}, placing the lower 8 bits in [31:24] and zeros in [23:0].
Notes: Hack captures are aligned to a carry boundary for precision. The hexmap submodule has no bus-visible registers. Interrupt outputs (tm_int, al_int, o_interrupt) are signals only and are not memory-mapped.

-- Register Summary --

Bus/register map (i_wb_addr[2:0], 32-bit data). Narrow payloads pack in low bits; unused high bits read as zero. All time fields are packed BCD; writes must use valid BCD nibbles.
- [0] CLOCK/Mode (RW)

- Read: {clock[31:22], ck_last_clock[21:0]} returns a latched snapshot of time to avoid race with live updates.
- Write fields:
- clock[21:0] = BCD HH:MM:SS (hours [21:16], minutes [15:8], seconds [7:0]); per-byte no-change sentinels: seconds/minutes 0xFF, hours 0x3F. Writing seconds=0 also clears the subsecond divider.
- clock[27:24] = display/mode (0=HH:MM, 1=Timer MM:SS, 2=Stopwatch MM:SS, 3=Clock MM:SS; others default to HH:MM).
- clock[31:28], clock[23], clock[22] = user bits.
- [1] TIMER (RW)
- Read: {6'h00, timer[25:0]} where timer[23:0]=BCD HH:MM:SS (up to 99:59:59), timer[24]=run, timer[25]=expired latch.
- Write semantics:
- If run=1: only run bit may be changed; any write with run=1 (or if previously run=1) clears expired latch.
- If run=0: load timer[23:0]; nonzero loads and captures tm_start; writing zero when counter is zero reloads tm_start.
- Decrements each second while run=1; on 00:00:00 sets expired, auto-clears run, asserts timer interrupt (tm_int).
- [2] STOPWATCH (RW control; counters auto-update)
- Read: [31:24]=BCD hours (extends beyond 99 via [31:28]), [23:16]=BCD minutes, [15:8]=BCD seconds, [7:1]=subsecond mirror, [0]=run.
- Write: bit0 start/stop; bit1 clear—if not running (either previously or in same write with run=0), clears time and subsecond/pps flags.
- [3] ALARM (RW)
- Read: {6'h00, al_tripped(25), al_enabled(24), 2'b00, alarm_time[21:0]=BCD HH:MM:SS}.
- Write: update alarm_time with per-field sentinels (seconds/minutes 0xFF, hours 0x3F); set al_enabled (bit24); writing bit25=1 clears al_tripped. Disabling also clears al_tripped. al_tripped latches on time match; al_int pulses on match edge.
- [4] CKSPEED (RW)
- ckspeed[31:0]: fractional-N accumulator step controlling 1-PPS rate (default 32'd2814750). Changing affects ck_carry/ck_pps.
- [5] HACK_TIME (RO)
- Read: {2'b00, hack_time[29:0]} where hack_time={clock[21:0], ck_sub[7:0]} for precise timestamp capture.
- [6] HACK_COUNTER_HI (RO)
- Read: hack_counter[39:8].
- [7] HACK_COUNTER_LO (RO)
- Read: {hack_counter[7:0], 24'h000000}.
- Interrupts (not memory-mapped): o_interrupt = tm_int | al_int.
- Addressing/width: 8 x 32-bit words at i_wb_addr[2:0]; read-only fields return zero in unused bits.

-- Field Definitions and Encoding --

Field definitions and encodings (bit positions, formats, and sentinel values):

General BCD packing
- Each time byte is packed BCD: high nibble=tens (0–9), low nibble=ones (0–9).
- Hours tens for the real-time clock are limited to 0–2 (24-hour format).
- Per-byte no-change sentinels on writes: seconds/minutes=8'hFF, hours=8'h3F. Writing seconds=8'h00 also resets the subsecond divider (ck_sub).

Real-time clock (clock[31:0])

- seconds (clock[7:0]): {SS_tens[7:4], SS_ones[3:0]} valid 00–59.
- minutes (clock[15:8]): {MM_tens[15:12], MM_ones[11:8]} valid 00–59.
- hours (clock[21:16]): {HH_tens[21:20], HH_ones[19:16]} valid 00–23.
- user/mode (clock[31:22]): status/mode/user-defined.
- display mode (clock[27:24]):
- 0/other: show HH:MM (from ck_last_clock), blank leading zero on hours.
- 1: show timer MM:SS (suppress leading zeros until expired; then show all digits).
- 2: show stopwatch MM:SS (suppress leading zeros on upper digits).
- 3: show clock MM:SS.
- Read (addr 0): {clock[31:22], ck_last_clock[21:0]} where ck_last_clock is a latched snapshot of clock[21:0].

Countdown timer (timer[25:0]; read at addr 1 as {6'h00, timer})
- timer[23:0]: packed BCD HH:MM:SS; hours supports 00–99 (two BCD nibbles for HH).
- seconds (timer[7:0]), minutes (timer[15:8]), hours (timer[23:16]).
- timer[24]: run/enable (1=running).
- timer[25]: expired/alarm latch (1=expired).
- Interrupt: tm_int asserted when counter reaches 00:00:00 while run=1.

Stopwatch (stopwatch[31:0]; addr 2)
- stopwatch[0]: run (1=run).
- stopwatch[7:1]: subsecond bits (mirror sw_sub[7:1]).
- stopwatch[31:8]: packed BCD HH:MM:SS; HH may extend beyond 99, with upper HH nibbles growing into [31:28].
- Write-time clear encoding: writing bit1=1 acts as a clear command if the stopwatch wasn't running or run=0 in the same write; on read, bit1 is part of subsecond field.

Alarm (addr 3 readback: {6'h00, al_tripped, al_enabled, 2'b00, alarm_time[21:0]})
- alarm_time[21:0]: packed BCD HH:MM:SS (same packing as clock).
- al_enabled (bit24): 1=enables comparison.
- al_tripped (bit25): latch set on match.
- Write sentinels: seconds/minutes=8'hFF, hours=8'h3F (no-change). Writing bit25=1 (clear) or al_enabled=0 clears al_tripped; disabling also clears al_enabled.
- Interrupt: al_int pulses on match edge.

Programmable tick rate (ckspeed, addr 4)
- ckspeed[31:0]: fractional-N step size for ck_counter accumulation; default 32'd2814750.

Precise timestamp capture (hack)
- hack_time[29:0] = {clock[21:0], ck_sub[7:0]} (30 bits: clock BCD plus subsecond divider).
- Read (addr 5): {2'b00, hack_time[29:0]}.
- hack_counter[39:0]: snapshot of 40-bit ck_counter accumulator.
- Read (addr 6): hack_counter[39:8].
- Read (addr 7): {hack_counter[7:0], 24'h00_0000}.

Bus read map (i_wb_addr[2:0] → o_data[31:0])
- 0: {clock[31:22], ck_last_clock[21:0]}.
- 1: {6'h00, timer[25:0]}.
- 2: stopwatch[31:0].
- 3: {6'h00, al_tripped, al_enabled, 2'b00, alarm_time[21:0]}.
- 4: ckspeed[31:0].
- 5: {2'b00, hack_time[29:0]}.

- 6: hack_counter[39:8].
- 7: {hack_counter[7:0], 24'h00_0000}.

Seven-segment display encoding
- hexmap submodule: i_hex[3:0] (0–F) → o_map[6:0] (active-high segments a,b,c,d,e,f,g), registered (one-cycle latency).
- Examples: 0→7'b1111110, 1→7'b0110000, 8→7'b1111111, F→7'b1000111.
- o_sseg: 4×8-bit bus (four digits); 7 bits from hexmap drive segments; the extra bit is implementation-specific (often DP). Display may blank when tm_alarm (timer[25]=1) or al_tripped=1 and ck_sub[7]=1; dmask gates leading zeros per mode.

Interrupt output
- o_interrupt = tm_int | al_int.

-- Access Types and Side Effects --

- Bus overview: 32-bit Wishbone, 8 word addresses (0–7). Writes require i_wb_we and block selects (ck_sel, tm_sel, sw_sel, al_sel). Byte-enabled writes use sentinel values to express "no change" in packed BCD fields. Reads are non-destructive unless noted.

- Addr 0 (Clock + mode) [RW]:
- Read: returns a latched copy of time (ck_last_clock) for stability; no side effects.
- Write: per-byte with sentinels (seconds/minutes 0xFF, hours 0x3F). Writing seconds=0 resets the subsecond divider (ck_sub). Mode bits [27:24] change display/LED behavior immediately.

- Addr 1 (Countdown timer) [RW]:
- Read: status/control {6'h00, timer[25:0]}; no side effects.
- Write: if running (timer[24]=1), only run bit is writable; any write with run=1 or if the timer was previously running clears the expired latch (timer[25]). If stopped (run=0), BCD HH:MM:SS [23:0] is loadable: nonzero writes load and snapshot to tm_start; writing zero when the counter is zero reloads tm_start. Runtime: decrements once per second; on 00:00:00 sets expired (timer[25]), pulses tm_int, and auto-clears run.

- Addr 2 (Stopwatch) [RW]:
- Read: full state, including run bit, subsecond mirror, and BCD SS/MM/HH; no side effects.
- Write: can set/clear run. If clear bit (bit1) is set and either the stopwatch was not running or the write sets run=0, all time, subsecond, and pps flags are cleared. Runtime: when running, subsecond increments on ck_carry; rolls SS/MM/HH with carries.

- Addr 3 (Alarm) [RW]:
- Read: {status, al_tripped, al_enabled, alarm_time}; no side effects.
- Write: time fields use sentinels (sec/min 0xFF, hours 0x3F). Bit24 enables/disables comparison; disabling also clears al_tripped. Bit25 is write-one-to-clear for al_tripped. Runtime: while enabled, on time match, al_tripped latches and al_int pulses.

- Addr 4 (ckspeed) [RW]:
- Read: no side effects.
- Write: immediately adjusts the fractional-N accumulator step, trimming the 1-PPS rate.

- Addr 5–7 (Hack capture) [RO]:
- Captures precise timestamp on i_hack or the next carry edge. Reads do not clear or advance capture. No bus write path.

- Interrupts and outputs:
- o_interrupt is tm_int OR al_int (timer expiry or alarm match). Writes that start the timer or enable alarm comparison may lead to future interrupts. Display mode bits [27:24] update seven-seg/LED output immediately; alarm/timer status may blink with ck_sub[7].

- General constraints:
- Per-byte write sentinels guard packed BCD time fields against unintended changes.
- No read-to-clear behavior; all clears occur via explicit writes (e.g., timer expired via run-write side effect; alarm via write-one-to-clear or disabling).
- Concurrency: timer writes while running are restricted; stopwatch clear is gated by run state. Adjusting ckspeed and writing clock seconds=0 affect timing and subsecond phase.

-- Reset and Default Values --

- Global/power-on reset: Not explicitly defined. Unless otherwise noted, internal state should be assumed undefined after reset; software must initialize all user-visible registers.
- Explicit default value:
- ckspeed (addr 4): 32'd2814750 (fractional-N accumulator step for nominal 1 PPS).
- Fractional-N time base:
- ck_counter, ck_carry, ck_sub: No explicit reset values. Writing the real-time clock seconds field to 0 clears the subsecond divider (ck_sub) for re-alignment.
- Real-time clock:
- clock[21:0] (HH:MM:SS) and clock[31:22] (user/mode bits): No explicit reset/default values. Per-byte write sentinels apply at runtime (seconds/minutes use 0xFF; hours use 0x3F). Writing seconds=0 also clears ck_sub. ck_last_clock (latched readback) has no defined reset value.
- Countdown timer:
- timer[25:0] (value/run/expired): No explicit reset/default values. Runtime behavior: expired/alarm latch (timer[25]) clears on any write that sets run=1 (or if run was previously 1); run auto-clears on expire. When stopped, writing a nonzero value loads timer[23:0] and tm_start; writing zero when the counter is zero reloads tm_start.
- Stopwatch:
- stopwatch[31:0]: No explicit reset/default value. If a write sets bit1 (clear) while not running (either previously not running or run=0 in the same write), the time and all subsecond/PPS flags are cleared. Run bit (bit0) is set by write.
- Alarm:
- alarm_time[21:0], al_enabled: No explicit reset/default values. al_tripped is cleared by a write with clear bit set (bit25=1) or by disabling al_enabled.
- Hack capture:
- hack_time, hack_counter: Undefined until the first capture event; read at addr 7 always pads lower 24 bits with zeros by design.
- LED activity:
- ledreg: No explicit power-on default; reset each top-of-minute (ck_ppM) at runtime.
- Seven-segment display:
- clock[27:24] display mode: No explicit reset/default; software should initialize.
- hexmap submodule:
- No reset; o_map undefined until the first registered update after a clock edge.
- Guidance for deterministic startup:
- Firmware should program: clock time (clock[21:0]), display mode (clock[27:24]), timer value/run, stopwatch run/clear, and alarm time/enable before use. Initial values of ck_counter, ck_sub, ck_last_clock, hack_time, and hack_counter are unspecified; first valid values appear after normal operation.

-- Readback and Latching Behavior --

Wishbone reads are side■effect■free; no field is modified or cleared by read operations. Read coherence and latching per address: Address 0 returns {clock[31:22] live, ck_last_clock[21:0] snapshot}. ck_last_clock is latched on ck_pps to avoid HH:MM:SS rollover races, providing a coherent time-in-seconds view within a single read; upper user/mode bits [31:22] are always live. Address 1 returns {6'h00, timer}. timer[25] (expired) is a sticky latch set when the countdown reaches zero and is not cleared by read; timer[24] (run) and timer[23:0] (HH:MM:SS BCD) are live and may change between or during reads while running; no read latch. Address 2 returns stopwatch[31:0] live, including subsecond bits [7:1] that update on ck_carry; values may change between or during reads; no read latch. Address 3 returns {6'h00, al_tripped, al_enabled, 2'b00, alarm_time}. al_tripped is sticky on match and is not cleared by read; al_enabled and alarm_time are live. Address 4 returns ckspeed live (current programmed accumulator step). Addresses 5–7 return a hack capture snapshot: on i_hack (or the following carry edge), hack_time <= {clock[21:0], ck_sub} and hack_counter <= ck_counter are latched; reads of addr 5 = {2'b00, hack_time}, addr 6 = hack_counter[39:8], addr 7 = {hack_counter[7:0], 24'h00} return that snapshot until the next capture, enabling an atomic multiword read across 5–7. No read-to-clear semantics are implemented; sticky latches (timer expired, alarm tripped) are only cleared by the corresponding write/clear logic. Fields marked as zero padding (in addrs 1, 3, 5, 7) always read as zero. Per-field no-change write sentinels affect writes only; readback always returns the currently stored/live value. Only addr 0 (ck_last_clock) and addrs 5–7 (hack capture) provide read-stable snapshots; all other addresses are live and may be non-atomic across multiword software reads.

-- Sentinel and No-Change Values --

Sentinel and no-change semantics apply only to software writes; reads always return actual state. Illegal packed-BCD codes are used as per-field no-change indicators on specific modules.
- Real-time clock (HH:MM:SS): On write, seconds=0xFF and minutes=0xFF mean "leave unchanged"; hours=0x3F means "leave unchanged." Writing seconds=0 additionally resets the subsecond divider (ck_sub) to resynchronize the 1-PPS; this is a special action, not a sentinel. User/mode bits clock[31:22] are ordinary RW with no sentinel semantics.
- Alarm time/control: Alarm seconds/minutes=0xFF and hours=0x3F act as per-field no-change on write, same as the clock. Control: bit24 (al_enabled) enables/disables the alarm; bit25 is a clear sentinel that resets the alarm latch (al_tripped). Disabling al_enabled also clears the latch.
- Countdown timer: No explicit sentinel values for HH:MM:SS. Behavior is state-dependent: when stopped, writing a nonzero HH:MM:SS loads that value and captures it into tm_start; writing zero when the counter is already zero reloads from tm_start. When running, HH:MM:SS fields are ignored (implicit no-change) and only the run/enable bit timer[24] is applied. Any write with run=1, or when run was already 1, clears the expired latch (timer[25]).
- Stopwatch: Time fields are not writable (implicit no-change). Bit0 (run) controls execution. Bit1 acts as a clear sentinel: if set while the stopwatch is not running (either previously or in the same write with run=0), it clears the displayed time and all subsecond/pps internal flags.
- Seven-segment/hexmap display: No sentinel or no-change write values. Leading-zero suppression is handled internally (dmask) and is not software-controlled via special values.
- Other: No sentinel/no-change semantics for ckspeed, hack capture registers, LEDs, or hexmap outputs.