IC TRAINING CENTER VIET NAM

FUNDAMENTAL IC DESIGN AND VERIFICATION COURSE

FINAL PROJECT
TIMER IP DESIGN

Student: Nguyen Thanh Hao

Class: IC28

Instructor: Kha At Kham

Contents

LIST OF TABLES

LIST OF FIGURES

1. Overview

1.1. Introduction

In modern digital systems, timers and counters are fundamental components that play a crucial role in time measurement, event scheduling, and system synchronization. The 64-bit Timer Counter presented in this project is a versatile and robust hardware module designed to provide precise timing functionality in embedded systems, System-on-Chip (SoC) designs, and microcontroller applications.

This timer module implements a high-resolution 64-bit counter capable of operating in multiple modes, offering both free-running and periodic counting capabilities. With its configurable prescaler, the timer can adapt to various timing requirements, from high-frequency operations to long-duration measurements. The module incorporates an Advanced Peripheral Bus (APB) interface, making it compatible with industry-standard ARM-based systems and easily integrable into complex digital designs.

The timer's architecture includes essential features such as:

- 64-bit counter with programmable compare value for interrupt generation
- Multiple operating modes (free-running and periodic)
- Programmable prescaler for clock division (1-256)
- Debug-mode support with halt capability
- APB 3.0 compliant interface for easy system integration
- Configurable interrupt mechanism with enable/disable control

Applications:

1. Real-Time Systems: Task scheduling, time-stamping, and deadline monitoring in real-time operating systems
2. Embedded Systems: Peripheral control, PWM generation, and timeout management in microcontrollers
3. Communication Protocols: Baud rate generation, timeout counters, and synchronization in serial communication interfaces
4. Digital Signal Processing: Sample timing and frame synchronization in audio/video processing systems
5. Industrial Control: Process timing, motor control, and sensor data acquisition in automation systems
6. Power Management: Sleep timer and wake-up scheduling in low-power designs

7. Measurement Systems: Frequency measurement, pulse width measurement, and event counting in test equipment

The modular design of this timer counter separates the APB interface logic from the core timer functionality, enhancing maintainability and testability. The debug mode feature allows for system debugging without disrupting timer operations, making it particularly valuable during development and testing phases.

With its comprehensive feature set and standard-compliant interface, this 64-bit Timer Counter provides a reliable and flexible timing solution suitable for a wide range of digital system applications, from simple embedded controllers to complex multi-core processors.

1.2. Main features

1.2.1. Core Timer Functionality

Feature	Description	Implementation Details
64-bit Timer/Counter	Full 64-bit counter for extended timing range	Combines TDR0 (32-bit low) and TDR1 (32-bit high) registers
Dual Operating Modes	Two configurable counting modes	TCR[1]: 0=Free-running, 1=Periodic mode
Programmable Prescaler	Clock division from 1 to 256	TCR[11:8]: 4-bit prescaler value (0-8)
Timer Enable/Disable	Software-controlled timer operation	TCR[0]: 1=Enable, 0=Disable

1.2.2. Compare and Interrupt System

Feature	Description	Register Mapping
64-bit Compare Value	Programmable compare value for interrupt generation	TCMP0 (low 32-bit) + TCMP1 (high 32-bit)

Interrupt Enable Control	Configurable interrupt masking	TIER[0]: 1=Interrupt enabled
Interrupt Status	Readable interrupt pending status	TISR[0]: 1=Interrupt pending
Interrupt Clear	Software interrupt acknowledge	Write 1 to TISR[0] to clear

1.2.3. System Integration

Feature	Description	Implementation
APB 3.0 Interface	Standard AMBA APB bus interface	Full APB slave implementation
Byte-level Write Access	Flexible register updates using byte strobes	wstrb[3:0] for byte-wise writes
Error Handling	Protocol and semantic error detection	pslverr signal generation
Ready Signal	Transaction completion indication	pready signal control

1.2.4. Debug and Control Features

Feature	Description	Register/Control
Debug Mode Halt	Timer pause during debugging	THCSR[0]: Debug halt control
Register Protection	Runtime write protection for active timer	Error on mode/prescaler change when enabled
Status Readback	Real-time counter value reading	TDR0/TDR1 reflect the current count
Reset Behavior	Defined power-on/reset states	All registers have reset values

1.2.5. Technical Specifications

Parameter	Value/Specification	Notes
Counter Size	64-bit	Maximum count: 2^64-1
Address Space	8 registers × 4 bytes	32-byte memory map
Register Width	32-bit	All registers aligned
Prescaler Range	1 to 256	Power-of-2 division
Clock Domain	Single clock (sys_clk)	Synchronous design

Reset Type	Active-low asynchronous	sys_rst_n

1.2.6. Memory Map

Address Offset	Register Name	Read/Write	Description
0x000	TCR	R/W	Timer Control Register (enable, mode, prescaler)
0x004	TDR0	R/W	Timer Data Register 0 (Counter low 32-bit)
0x008	TDR1	R/W	Timer Data Register 1 (Counter high 32-bit)
0x00C	TCMP0	R/W	Timer Compare Register 0 (Compare value low 32-bit)
0x010	TCMP1	R/W	Timer Compare Register 1 (Compare value high 32-bit)
0x014	TIER	R/W	Timer Interrupt Enable Register
0x018	TISR	R/W	Timer Interrupt Status Register

0x01C	THCSR	R/W	Timer Halt Control and Status Register

1.2.7. Operational Modes

1. Free-running Mode (TCR[1]=0)
    - Counter counts continuously from 0 to (2^64) -1
    - Wraps around to 0 on overflow
    - Compare match generates an interrupt if enabled
2. Periodic Mode (TCR[1]=1)
    - Counter resets on a compare match
    - Automatic reload from 0
    - Generates periodic interrupts
3. Debug Halt Mode (THCSR[0]=1 with debug_mode=1)
    - Timer counter pauses
    - Prescaler counter pauses
    - Register access remains available

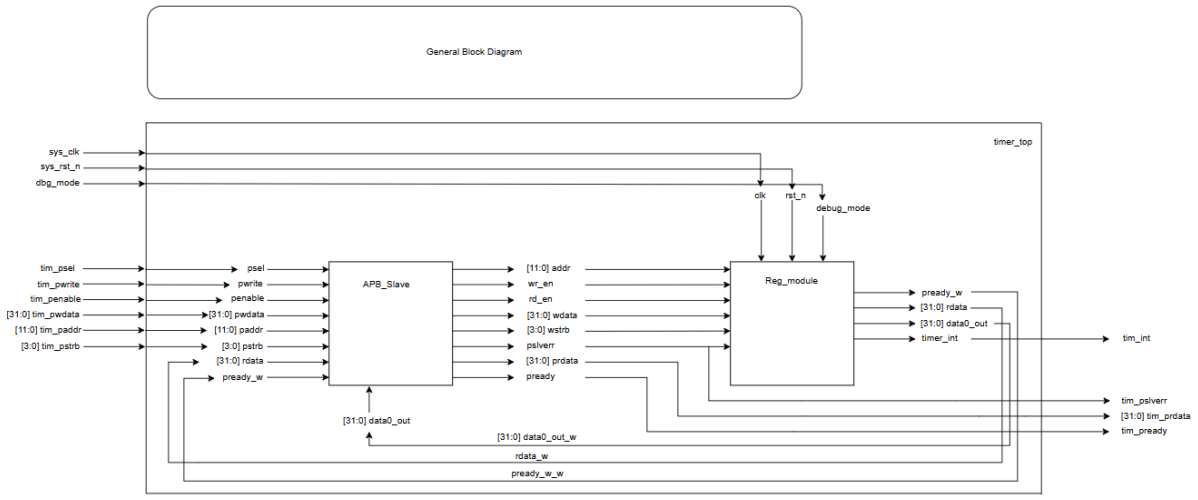
1.2.8. Error Conditions

The timer detects and reports the following error conditions through pslverr:

- Writing an invalid prescaler value (≥9)
- Changing operating mode while the timer is enabled
- Modifying the prescaler value while the timer is enabled

This comprehensive feature set makes the timer suitable for a wide range of applications, including real-time systems, communication protocols, motor control, and measurement systems requiring precise timing control.

1.3. Block diagram



Timer Module Overview

Apb_slave - APB Bus Interface

Main Functions:

- Communication bridge between APB bus and timer logic
- Protocol conversion from APB to simple control signals

Core Features:

1. **APB State Decoding**
    - Generates write enable signal: wr_en = psel && penable && pwrite && !pready_w
    - Generates read enable signal: rd_en = psel && penable && !pwrite && !pready_w
    - Creates write/read enable signals from APB transactions
2. **Error Detection**
    - Detects invalid writes to Control Register
    - Blocks changes to div_en/div_val when timer is running
    - Reports error for prescaler value ≥ 9
3. **Signal Routing**
    - Pass-through address, data, strobes

- Forwards ready signal and read data

Reg_module - Timer Core & Register Management

Main Functions:

- Manages 8 registers (DATA0-DATA7) for timer control
- Controls 64-bit timer counter
- Handles interrupt system

Key Features:

Register File with Byte-wise Access

- Supports individual byte writes via wstrb[3:0]
- Each register has specific function (control, counter, compare, interrupt)

64-bit Timer Counter

- Automatic or selectable counting speeds (from 1 to 256 times - 8-bit divider)
- Auto-reset when timer is disabled

Compare Match & Interrupt

- Compares counter with preset value (64-bit comparison)
- Interrupt flow: Compare match → Status bit set → Output when enabled
- Clear mechanism: Write-1-to-clear interrupt status

Debug Mode Support

- Halts timer when debug mode is active and halt bit is enabled
- Still allows register read/write during debug

Protection Logic

- Prevents changes to div_en/div_val when timer is running
- Automatically resets counter to 0 when disabled

PREADY Generation

- Generates ready signal for APB transactions
- Simple logic: asserts when valid read/write occurs

1.4. Interface signals

INPUTS (10 signals):

1. sys_clk (input wire)
    - Main system clock.
2. sys_rst_n (input wire)
    - Active-low reset signal.
3. tim_psel (input wire)
    - APB select signal - indicates the timer is selected on the APB bus.
4. tim_pwrite (input wire)
    - APB write/read indicator: 1 = write, 0 = read.
5. tim_penable (input wire)
    - APB enable signal - validates the APB transaction.
6. tim_paddr[11:0] (input wire [11:0])
    - APB address (12-bit) for accessing timer registers.
7. tim_pwdata[31:0] (input wire [31:0])
    - Write data from APB bus (32-bit).
8. tim_pstrb[3:0] (input wire [3:0])
    - Byte strobe for selective writing (4-bit, each bit for 1 byte).
9. debug_mode (input wire)
    - Debug mode - when high, the timer can be halted when the halt bit is set.

OUTPUTS (4 signals):

1. tim_prdata[31:0] (output wire [31:0])
    - Read data returned to APB bus (32-bit).
2. tim_pready (output wire)
    - APB ready signal - indicates transaction completion.
3. tim_pslverr (output wire)
    - APB slave error - signals an invalid access attempt.
4. tim_int (output wire)
    - Timer interrupt signal - activated when a compare event occurs and interrupts are enabled.

Table 1: Interface signals of timer IP

Signal name	Width	Direction	Description
sys_clk	1	input	System clock input
sys_rst_n	1	input	System reset, active low
tim_psel	1	input	APB select signal

tim_pwrite	1	input	APB write enable
tim_penable	1	input	APB enable signal
tim_paddr	12	input	APB address bus
tim_pwdata	32	input	APB write data bus
tim_pstrb	4	input	APB write strobes
debug_mode	1	input	Debug mode control input
tim_prdata	32	output	APB read data bus
tim_pready	1	output	APB ready signal
tim_pslverr	1	output	APB error response
tim_int	1	output	Timer interrupt output

2.Register Specification

2.1. Register Summary

Table 2: Register Summary

Offset	**Abbreviation**	**Register name**
0x00	TCR	Timer Control Register
0x04	TDR0	Timer Data Register 0
0x08	TDR1	Timer Data Register 1
0x0C	TCMP0	Timer Compare Register 0
0x10	TCMP1	Timer Compare Register 1
0x14	TIER	Timer Interrupt Enable Register
0x18	TISR	Timer Interrupt Status Register
0x1C	THCSR	Timer Halt Control Status Register
Others	Reserved	Reserved

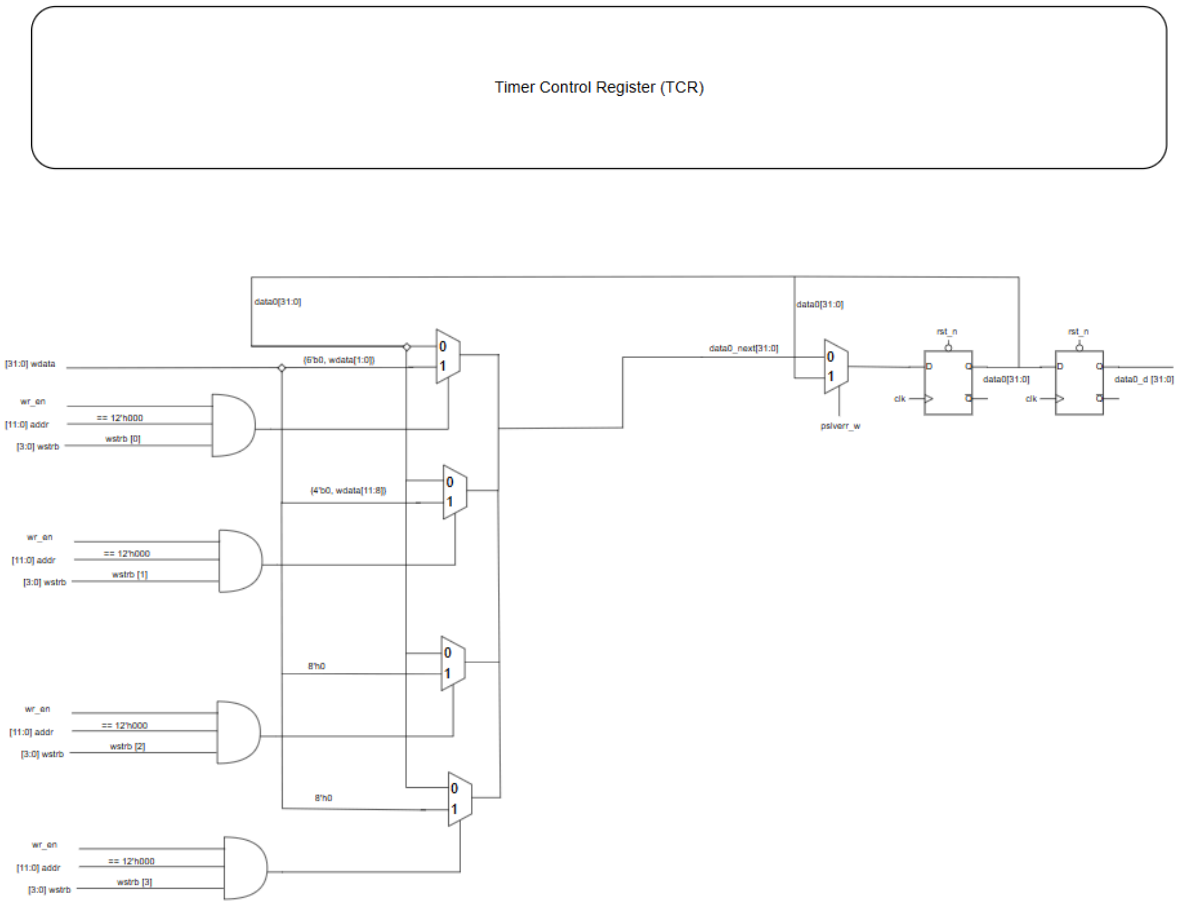
2.2. Timer Control Register (TCR)

Offset address: 0x0

Reset value: 0x0000_0000

Table 3. Timer Control Register (TCR)

Bit	Name	Type	Default value	Description
31:12	Reserved	-	20’h0	Reserved
11:8	DIV_VAL[3:0]	RW	4’b0001	Counter control mode setting: • 4’b0000: Counting speed is not divided • 4’b0001: Counting speed is divided by 2 (default) • 4’b0010: Counting speed is divided by 4 • 4’b0011: Counting speed is divided by 8 • 4’b0100: Counting speed is divided by 16 • 4’b0101: Counting speed is divided by 32 • 4’b0110: Counting speed is divided by 64 • 4’b0111: Counting speed is divided by 128 • 4’b1000: Counting speed is divided by 256 • Others: reserved, (*)prohibit settings. When setting the prohibit value, div_val is not changed. Note: user must not change div_val while timer_en is High (*): add hardware logic to ensure div_val is prohibited to change when timer_en is High. Access is error response in this case. (*)access is “error response” when setting prohibit value to div_val
7:2	Reserved	RO	6’b0	Reserved
1	DIV_EN	RW	1’b0	Counter control mode enable. • 0: Disabled. Counter counts with normal speed based on system clock • 1: Enabled. The coutingspeed of counter is controlled based on div_val Note: user must not change div_en while timer_en is High (*): add hardware logic to ensure div_en is prohibited to change when timer_en is High. Access is error response in this case.
0	TIM_EN	RW	1’b0	Timer enable • 0: Disabled. Counter does not count. • 1: Enabled. Counter starts counting. (*) timer_en changes from H->L will initialize the TDR0/1 to their initial value



Timer Control Register (TCR) Functional Description

The Timer Control Register (TCR) controls all operations of the 64-bit timer, including enabling/disabling, mode selection, and clock prescaling.

Register Bits:

- Bit 0: ENABLE - 1: Timer enabled, 0: Timer disabled
- Bit 1: MODE - 0: Free-running mode (continuous count), 1: Periodic mode (reset on compare match)
- Bits 11:8: PRESCALER - Clock divider value (0-8). Division ratio = 2^PRESCALER

Operational Rules:

1. The timer must be disabled before changing MODE or PRESCALER values
2. Attempting to modify the configuration while the timer is active triggers a PSLVERR error

3. When disabled, both the main counter and the internal prescaler reset to zero
4. Configuration values are preserved during the disabled state

Initial State (after reset):

- ENABLE = 0 (disabled)
- MODE = 0 (free-running)
- PRESCALER = 1 (divide by 2)
- All other bits = 0

Usage Procedure:

1. Write 0 to ENABLE to disable the timer
2. Configure MODE and PRESCALER as needed
3. Write 1 to ENABLE to start counting

Register Bit Field Mapping

Bit Position	Field Name	Access Type	Reset Value	Description
31:24	Reserved	RO	8'h0	Not used
23:16	Reserved	RO	8'h0	Not used
15:12	Reserved	RO	4'b0	Not used
11:8	PRESCALER	R/W	4'b0001	Prescaler value for clock division

7:2	Reserved	RO	6'b0	Not used
1	MODE	R/W	1'b0	Operating mode
0	ENABLE	R/W	1'b0	Timer enable/disable

2.3. Timer Data Register 0 (TDR0) & Timer Data Register 1 (TDR1)

Timer Data Register 0 (TDR0)

Offset address: 0x04

Reset value: 0x0000_0000

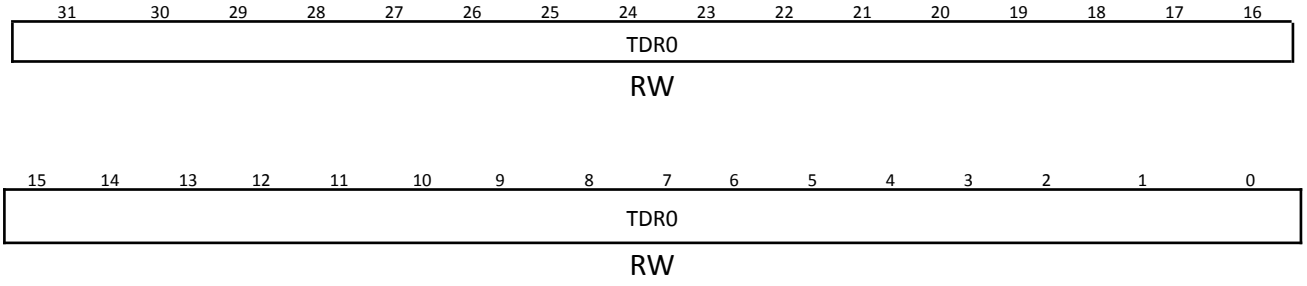


Table 4. Timer Data Register 0 (TDR0)

Bit	Name	Type	Default value	Description
31:0	TDR0	RW	32'h0000_0000	Lower 32-bit of 64-bit counter. Value of this register is cleared to initial value when timer_en changes from H->L.

Timer Data Register 1 (TDR1)

Offset address: 0x08

Reset value: 0x0000_0000

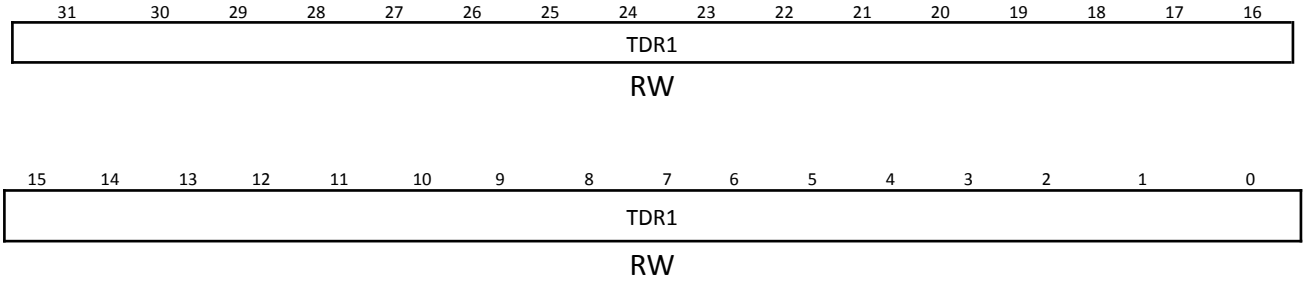
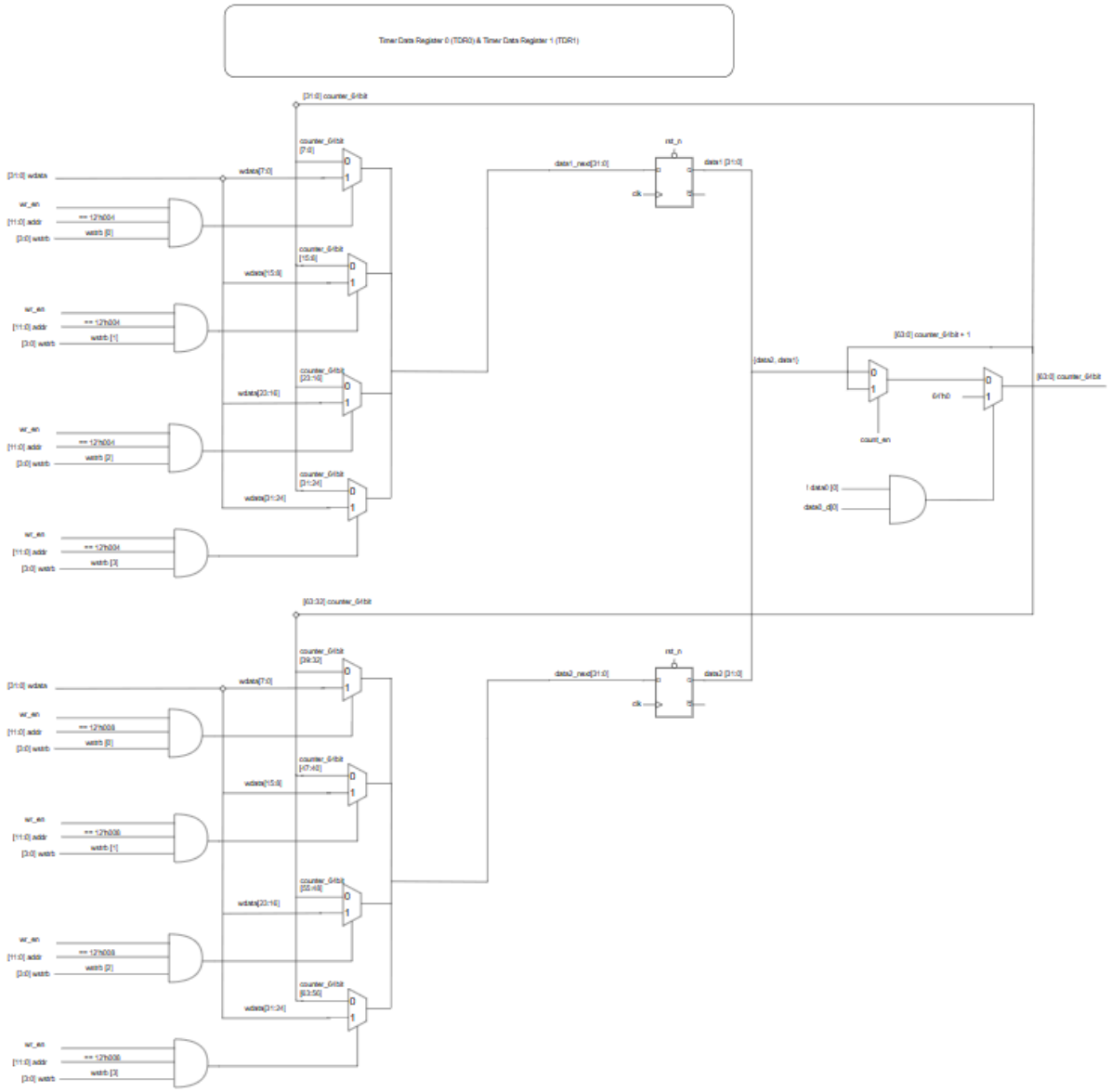


Table 5. Timer Data Register 1 (TDR1)

Bit	Name	Type	Default value	Description
31:0	TDR1	RW	32'h0000_0000	Upper 32-bit of 64-bit counter. Value of this register is cleared to initial value when timer_en changes from H->L.



TDR0 and TDR1 Registers in Timer Module

TDR0 (Timer Data Register 0)

TDR0 is a 32-bit register located at offset address 0x004 in the APB address space. It contains the lower 32 bits of the 64-bit timer counter.

1. Read Function:

- When reading from address 0x004, the register returns the current lower 32-bit value of the timer counter.
- The read value reflects the actual state of the counter at the moment of reading.

2. Write Function:

- When writing to address 0x004, the value is written directly to the lower 32 bits of the timer counter.
- Supports byte-wise writing through the wstrb signal.
- Writing takes effect immediately and affects the counting operation.

3. Special Operations:

- When the timer is disabled (enable bit = 0), the counter automatically resets to 0, including TDR0.
- In debug mode with halt enabled, the TDR0 value is frozen when the timer pauses.

TDR1 (Timer Data Register 1)

TDR1 is a 32-bit register located at offset address 0x008 in the APB address space. It contains the upper 32 bits of the 64-bit timer counter.

1. Read Function:

- When reading from address 0x008, the register returns the current upper 32-bit value of the timer counter.
- Combined with TDR0, it forms the complete 64-bit counter value.

2. Write Function:

- When writing to address 0x008, the value is written directly to the upper 32 bits of the timer counter.
- Similar to TDR0, supports byte-wise writing.
- Writing takes immediate effect.

3. Special Operations:

- Together with TDR0, it resets to 0 when the timer is disabled.
- Frozen in debug halt mode.

Combined Functions of TDR0 and TDR1

1. Formation of Complete 64-bit Counter:

- TDR1: Contains upper 32-bit [63:32]

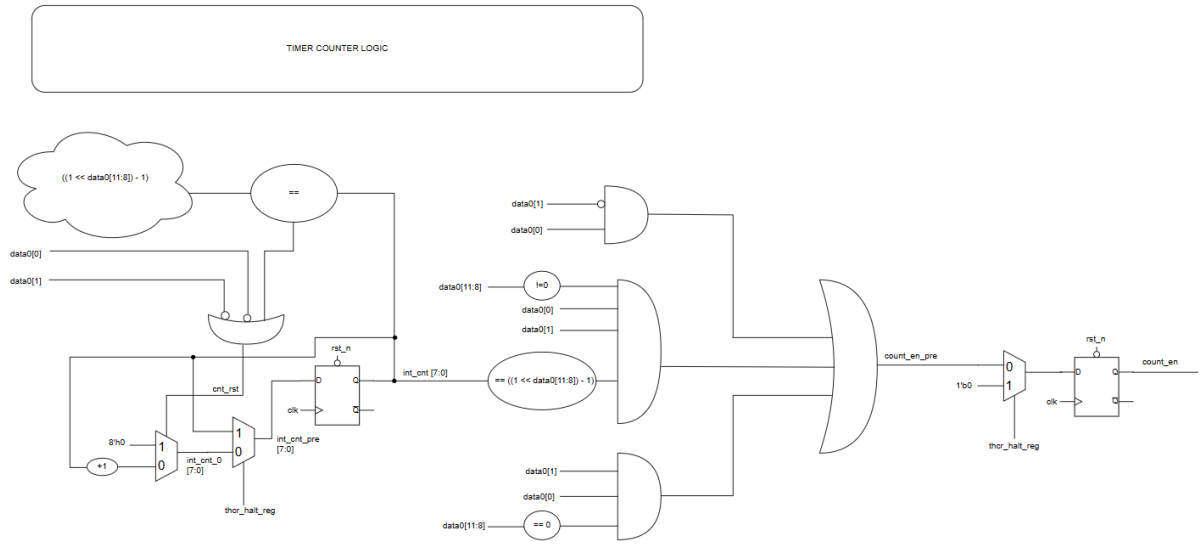
- TDR0: Contains lower 32-bit [31:0]
- Together, they create a 64-bit timer counter with an extensive counting range.

2. Counting Operation:

- The counter increments from the current value in TDR1:TDR0.
- In free-running mode: Counts from 0 to 2^64-1 then wraps around to 0.
- In period mode: Counts from 0 to the compare value (TCMP0:TCMP1) then resets to 0.
- Counting speed is divided by the prescaler configured in TCR[11:8].

3. Interaction with Other Registers:

- TCR: Timer Control Register - controls enable/prescaler mode.
- TCMP0/TCMP1: Timer Compare registers - compare values against TDR1:TDR0.
- TISR: Interrupt status register - sets when TDR1:TDR0 matches TCMP0:TCMP1.
- TIER: Interrupt enable register - enables the interrupt when the comparison matches.



Timer Counter Logic

1. Timer Counter Logic

1.1. Counter Structure

64-bit Counter Architecture:
The timer counter system uses a 64-bit structure consisting of two 32-bit registers: TDR0

(data1) representing the lower 32 bits and TDR1 (data2) representing the upper 32 bits. These two registers combine to form a complete 64-bit counter.

Counter Update Logic:
The counter is updated through the logic: counter_64bit = count_en ? {TDR1, TDR0} + 1 : {TDR1, TDR0}. This means the counter increments by 1 when the count_en signal is at logic level 1, and maintains its current value when count_en is at logic level 0.

1.2. Prescaler Control System

Prescaler Counter (int_cnt):
The system uses an 8-bit prescaler counter (int_cnt) with values from 0 to 255. This counter is controlled by the cnt_rst signal for reset and increments when conditions permit.

Prescaler Reset Condition:
The cnt_rst signal is determined by the formula: cnt_rst = !TCR[0] || !TCR[1] || (int_cnt == ((1 << TCR[11:8]) - 1)). This means the prescaler resets when: the timer transitions from enabled to disabled (TCR[0] = 1 -> TCR[0] = 0), in free-running mode (TCR[0] = 1 -> TCR[1] = 0), or when the prescaler counter reaches its maximum value.

Prescaler Next Value Calculation:
The next value of the prescaler is calculated through two signals: int_cnt_0 = cnt_rst ? 8'h0 : int_cnt + 1 and int_cnt_pre = thcr_halt_reg ? int_cnt : int_cnt_0. When cnt_rst is active, the prescaler resets to 0; when not reset and not halted, the prescaler increments by 1; when halted, the prescaler maintains its current value.

1.3. Counter Enable Logic

Enable Conditions:
The count_en_pre signal is determined by the formula: count_en_pre = thcr_halt_reg ? 1'b0 : ((!TCR[1] && TCR[0]) || (TCR[11:8] != 0 && TCR[1] && TCR[0] && (int_cnt == ((1 << TCR[11:8]) - 1))) || (TCR[11:8] == 0 && TCR[1] && TCR[0])).

Operating Modes:

- Free-running mode: Activated when TCR[0]=1 and TCR[1]=0
- Periodic mode with prescaler: Activated when TCR[0]=1, TCR[1]=1, TCR[11:8] ≠ 0 and int_cnt reaches maximum value
- Periodic mode without prescaler: Activated when TCR[1]=1, TCR[1]=1 and TCR[11:8] = 0

Halt Control:
When thcr_halt_reg = 1, the entire counter logic is disabled. This allows the system to pause the timer in debug mode.

1.4. Register Update Behavior

Normal Operation:
During normal operation, TDR0 and TDR1 are updated with the new counter value, while other registers are updated from their next values.

Timer Disable Transition:
When the timer transitions from enabled to disabled state, the system resets TDR0 and TDR1 to 0, ensuring the counter starts from 0 when re-enabled.

2.4. Timer Compare Register 0 (TCMP0) & Timer Compare Register 1 (TCMP1)

Timer Compare Register 0 (TCMP0)

Offset address: 0x0C

Reset value: 0xFFFF_FFFF

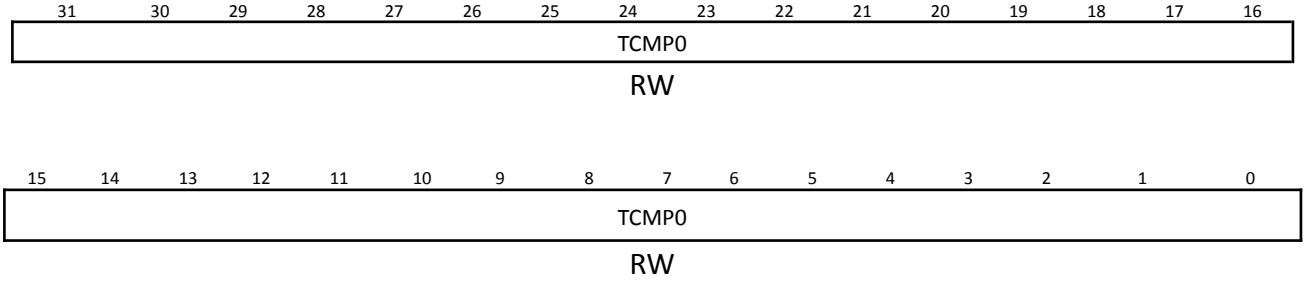


Table 6. Timer Compare Register 0 (TCMP0)

Bit	Name	Type	Default value	Description
31:0	TCMP0	RW	32'hFFFF_FFFF	Lower 32-bit of 64-bit compare value. Interrupt pending bit is asserted when counter value is qual to compare value.

Timer Data Register 1 (TDR1)

Offset address: 0x10

Reset value: 0xFFFF_FFFF

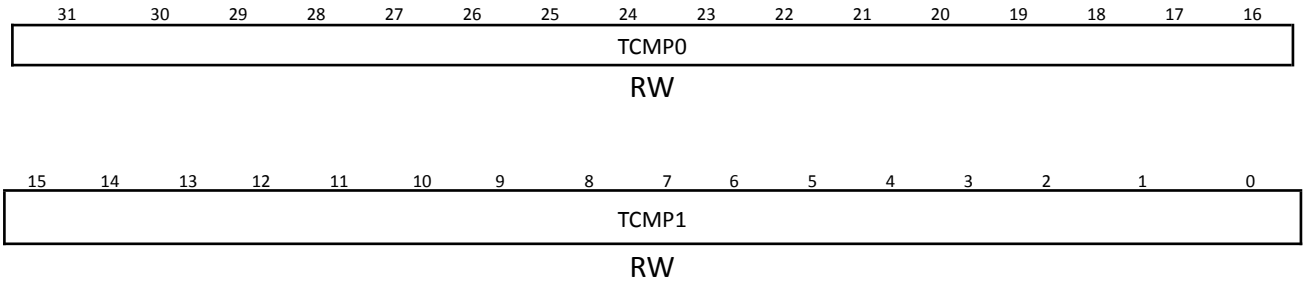
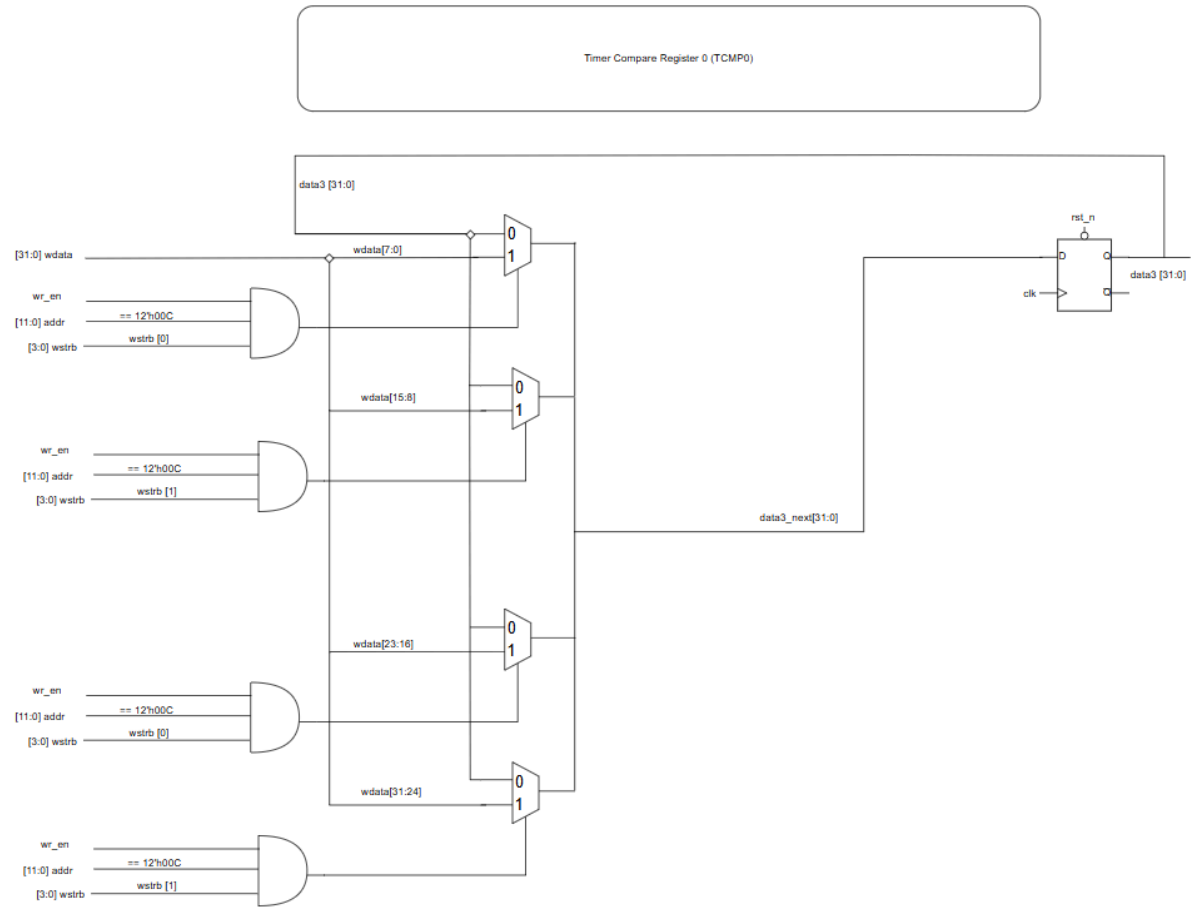
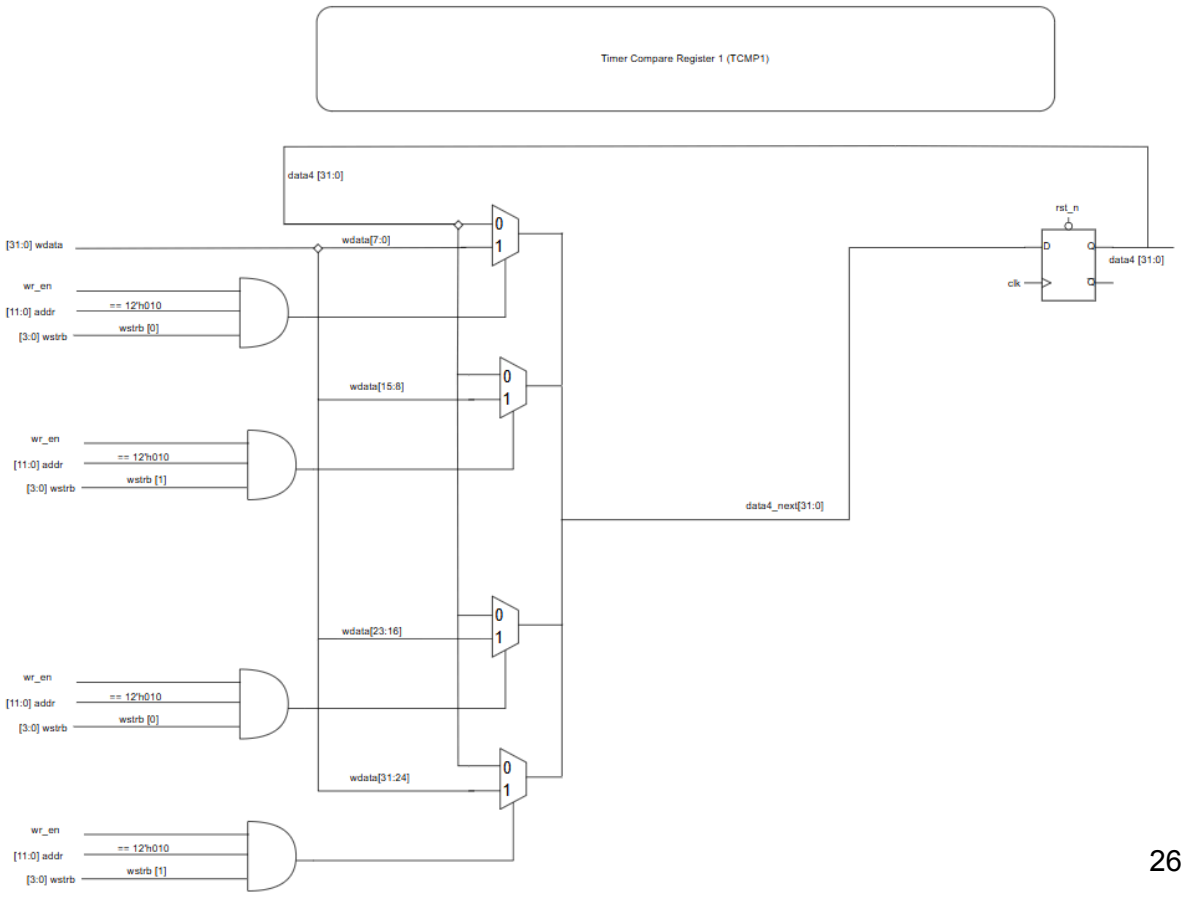


Table 7. Timer Compare Register 1 (TCMP1)

Bit	Name	Type	Default value	Description
31:0	TCMP1	RW	32'hFFFF_FFFF	Upper 32-bit of 64-bit compare value. Interrupt pending bit is asserted when counter value is qual to compare value.


Timer Compare Register 0 (TCMP0)
data3 [31:0]
rst_n
[31:0] wdata
wdata[7:0]
wr_en
[11:0] addr
== 12'h00C
[3:0] wstrb
wstrb [0]
clk
data3 [31:0]
wdata[15:8]
wr_en
[11:0] addr
== 12'h00C
[3:0] wstrb
wstrb [1]
data3_next[31:0]
wdata[23:16]
wr_en
[11:0] addr
== 12'h00C
[3:0] wstrb
wstrb [0]
wdata[31:24]
wr_en
[11:0] addr
== 12'h00C
[3:0] wstrb
wstrb [1]


Timer Compare Register 1 (TCMP1)
data4 [31:0]
rst_n
[31:0] wdata
wdata[7:0]
wr_en
[11:0] addr
== 12'h010
[3:0] wstrb
wstrb [0]
clk
data4 [31:0]
wdata[15:8]
wr_en
[11:0] addr
== 12'h010
[3:0] wstrb
wstrb [1]
data4_next[31:0]
wdata[23:16]
wr_en
[11:0] addr
== 12'h010
[3:0] wstrb
wstrb [0]
wdata[31:24]
wr_en
[11:0] addr
== 12'h010
[3:0] wstrb
wstrb [1]

TCMP0 and TCMP1 Registers in Timer Module

TCMP0 (Timer Compare Register 0)

TCMP0 is a 32-bit register located at offset address 0x00C in the APB address space. This register contains the lower 32 bits of the 64-bit compare value for the timer counter.

1. Read Function:

- When reading from address 0x00C, the register returns the value of the lower 32 bits

2. Write Function:

- When writing to address 0x00C, the value is written directly to the lower 32 bits of the compare register
- Supports byte-wise writing through the wstrb signal
- Writing takes effect immediately and affects the comparison operation

3. Special Operations:

- The default value after reset is 0xFFFF_FFFF

TCMP1 (Timer Compare Register 1)

TCMP1 is a 32-bit register located at offset address 0x010 in the APB address space. This register contains the upper 32 bits of the 64-bit compare value for the timer counter.

1. Read Function:

- When reading from address 0x010, the register returns the value of the upper 32 bits
- Combined with TCMP0, it forms the complete 64-bit compare value

2. Write Function:

- When writing to address 0x010, the value is written directly to the upper 32 bits of the compare register
- Similar to TCMP0, supports byte-wise writing
- Writing takes effect immediately

3. Special Operations:

- The default value after reset is 0xFFFF_FFFF
- Combined with TCMP0, it forms the complete 64-bit compare value

Combined Functions of TCMP0 and TCMP1

1. Formation of Complete 64-bit Compare Value:

- TCMP1: Contains the upper 32-bit [63:32] of the compare value
- TCMP0: Contains the lower 32-bit [31:0] of the compare value
- Together, they create a 64-bit timer compare value with an extensive range

2. Compare Operation:

- Comparison is performed between the counter value (TDR1:TDR0) and the compare value (TCMP1:TCMP0)
- When {TDR1, TDR0} == {TCMP1, TCMP0}, the timer_int signal will be activated
- The comparison only triggers an interrupt and does not affect the counting operation of the timer

3. Interaction with Other Registers:

- TCR (Timer Control Register): Controls enable and prescaler
- TDR0/TDR1: Timer counter - compared with TCMP0/TCMP1
- TISR (data6): Interrupt status register - bit 0 is set when compare matches
- TIER (data5): Interrupt enable register - bit 0 enables interrupt when compare matches

4. Default Configuration:

- TCMP0: Default value after reset is 0xFFFF_FFFF
- TCMP1: Default value after reset is 0xFFFF_FFFF
- Default values are set very high to avoid unwanted interrupt triggering after reset
- Users need to configure appropriate values according to timing requirements

5. Interrupt Handling:

- When comparing matches: bit 0 of data6 (TISR) is set to 1
- If bit 0 of data5 (TIER) is enabled, the interrupt signal timer_int will be activated
- The interrupt must be cleared by writing 1 to bit 0 of address 0x018 (data6)

2.5. Timer Interrupt Enable Register (TIER)

Offset address: 0x14

Reset value: 0x0000_0000

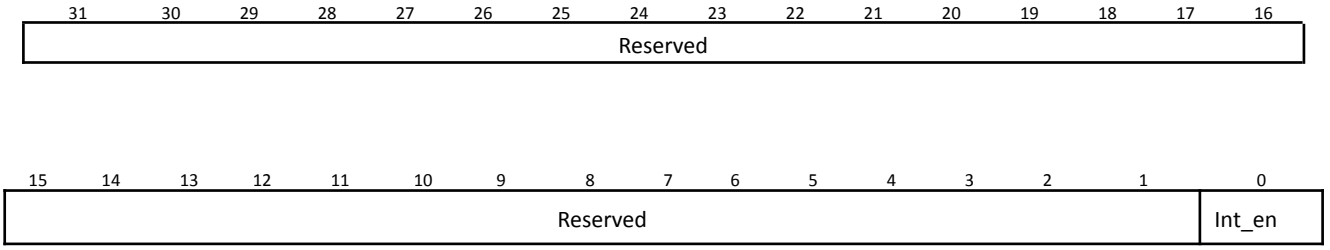
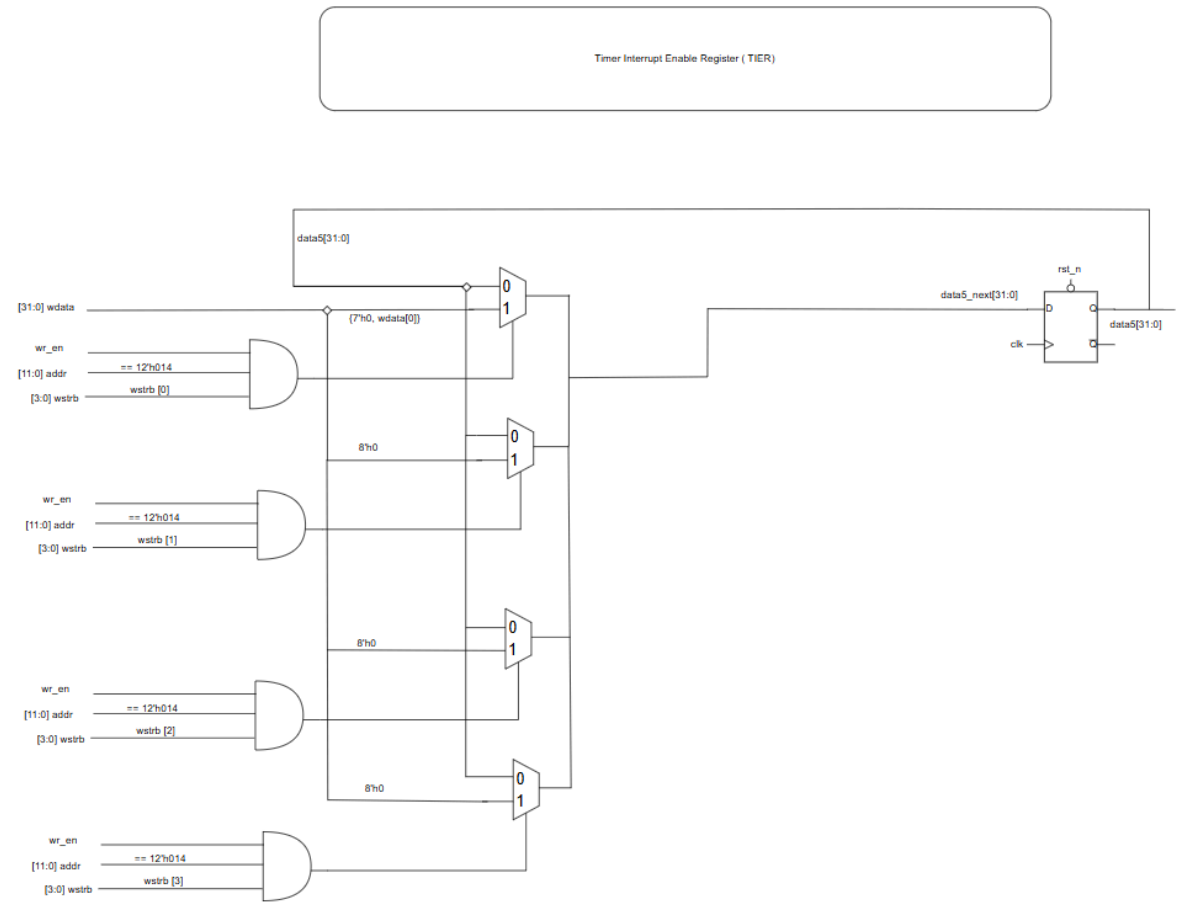


Table 8. Timer Interrupt Enable Register (TIER)

Bit	Name	Type	Default value	Description
31:1	Reserved	RO	31'h0	Reserved
0	Int_en	RW	1'b0	Timer interrupt enable 0: Timer interrupt is disabled. 1: Timer interrupt is enabled. When this bit is 0, no timer interrupt is output. When this bit is 1, timer interrupt pending bit can be output (interrupt pending bit is set when counter value is equals to compare value). Clearing this bit to 0 while interrupt is asserting will mask the interrupt to 0 but does not affect the interrupt pending bit (TISR.int_st bit)



Timer Interrupt Enable Register (TIER) - DATA5

1. Introduction

The Timer Interrupt Enable Register (TIER), internally referenced as data5, is a 32-bit register dedicated to controlling the interrupt generation capability of the timer module. It provides a simple mechanism to enable or disable timer interrupt generation.

2. Register Structure

2.1. Register Bit Field Mapping

Bit Position	Field Name	Access Type	Reset Value	Description
31:1	Reserved	Read-Only	31'h0	Not used

0	INT_EN	Read/Write	1'b0	Interrupt enable control

2.2. Detailed Bit Field Description

Bit 0 - INT_EN (Interrupt Enable):

- Value 0: Interrupt output disabled - the timer_int signal remains low.
- Value 1: Interrupt output enabled - the timer_int signal is activated when an interrupt is pending.

3. Register Implementation Details

3.1. Write Access

Only byte 0 (bits 7:0) can be written via the wstrb[0] signal. Only bit 0 is implemented; bits 7:1 always have a value of 0. The upper bytes (31:8) always read as 0.

3.2. Reset State

The default reset value is 32'h0000_0000. Interrupts are disabled after system reset.

4. Operation

4.1. Interrupt Generation Logic

Interrupt output is generated through two steps:

Step 1: Setting interrupt status (TISR[0])

- Interrupt status TISR[0] is set to 1 when the 64-bit counter value {TDR1, TDR0} equals the 64-bit compare value {TCMP1, TCMP0}.

Step 2: Generating output interrupt signal

- The timer_int interrupt signal is activated when both of the following conditions are satisfied:
- Interrupt enable bit (TIER[0]) has value 1
- Pending interrupt status (TISR[0]) has value 1

4.2. Usage Examples

To use the timer interrupt function:

Enabling interrupts:

- Configure the 64-bit compare value in TCMP1 and TCMP0
- Enable the interrupt enable bit by writing value 0x00000001 to the TIER register
- When the counter reaches the compare value, the interrupt status TISR[0] is automatically set and the interrupt signal is activated

Disabling interrupts:

- Write value 0x00000000 to the TIER register to disable interrupts
- Or clear the interrupt status by writing 1 to bit 0 of the TISR register

5. Key Features

Key features of TIER:

- Simple interrupt enable/disable control
- Software-controlled interrupt masking
- Default disabled state
- Minimal hardware overhead

2.6. Timer Interrupt Status Register (TISR)

Offset address: 0x18

Reset value: 0x0000_0000

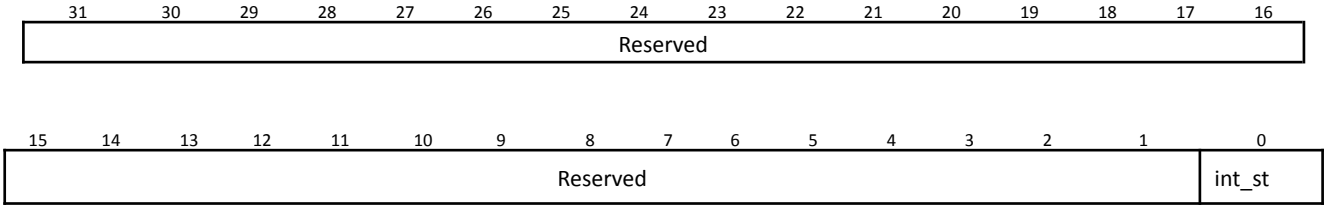
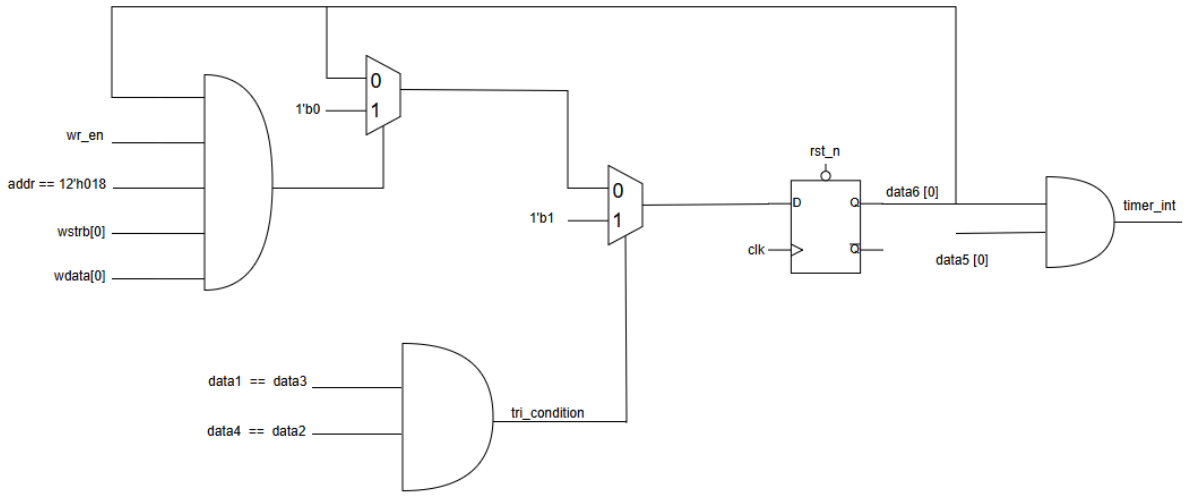


Table 9. Timer Interrupt Status Register (TISR)

Bit	Name	Type	Default value	Description
31:1	Reserved	RO	31'h0	Reserved
0	Int_st	RW	1'b0	Timer interrupt trigger condition status bit (interrupt pending bit) 0: the interrupt trigger condition does not occur. 1: the interrupt trigger condition occurred. Write 1 when this bit is 1 to clear it to 0. Write 0 when this bit is 1 has no effect Write to this bit when it is 0 has no effect. Note1: When interrupt trigger condition occurred (counter reached compare value), counter continues to count normally. Note2: the clearance condition has highest priority



Interrupt Logic & Timer Interrupt Status Register (TISR)

1. Compare Condition Detection

64-bit Compare Logic:
The comparison system uses the formula: tri_condition = (TCMP0 == TDR0) && (TCMP1 == TDR1). This performs a simultaneous 64-bit comparison of the counter with the compare value, where TCMP0 (data3) is the lower compare value and TCMP1 (data4) is the upper compare value.

2. Interrupt Status Register (TISR)

Set Logic:
When tri_condition = 1, the system automatically sets TISR[0] = 1 to indicate an interrupt is pending. Interrupt storage feature: Once TISR[0] has been set to 1, it maintains this value even if the tri_condition subsequently returns to 0. The interrupt pending state is maintained until explicitly cleared by software.

Clear Logic:
The interrupt is cleared through the condition: wr_en && (addr == 12'h018) && wstrb[0] && wdata[0] && TISR[0]. This occurs when there is a write operation to address 0x018 (TISR) with byte strobe[0] active, write data[0] = 1, and interrupt pending. The interrupt state is only cleared when value 1 is written to TISR[0].

Reset State:
After system reset, TISR[0] is initialized to 0, indicating no interrupts are pending.

3. Interrupt Output Generation

Interrupt Enable Control:
Bit TIER[0] serves as interrupt enable control. When this bit is at level 1, interrupt output is enabled; when at level 0, interrupt output is disabled.

Final Interrupt Signal:
The output interrupt signal is generated by the formula: timer_int = TIER[0] && TISR[0]. This signal is only activated when both interrupt enabled and interrupt pending are at logic level 1.

2.7. Timer Halt Control Status Register (THCSR)

Offset address: 0x1C

Reset value: 0x0000_0000

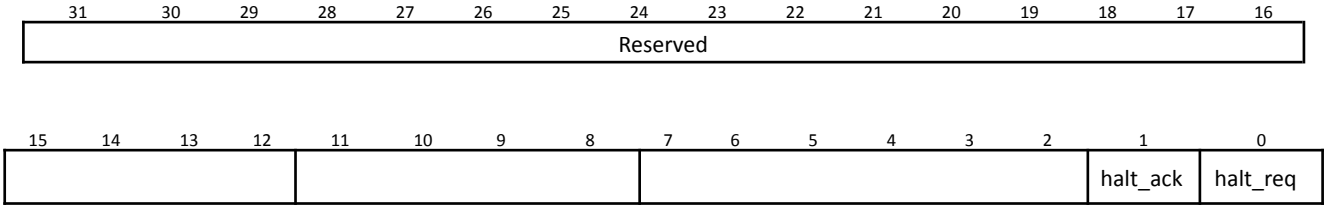
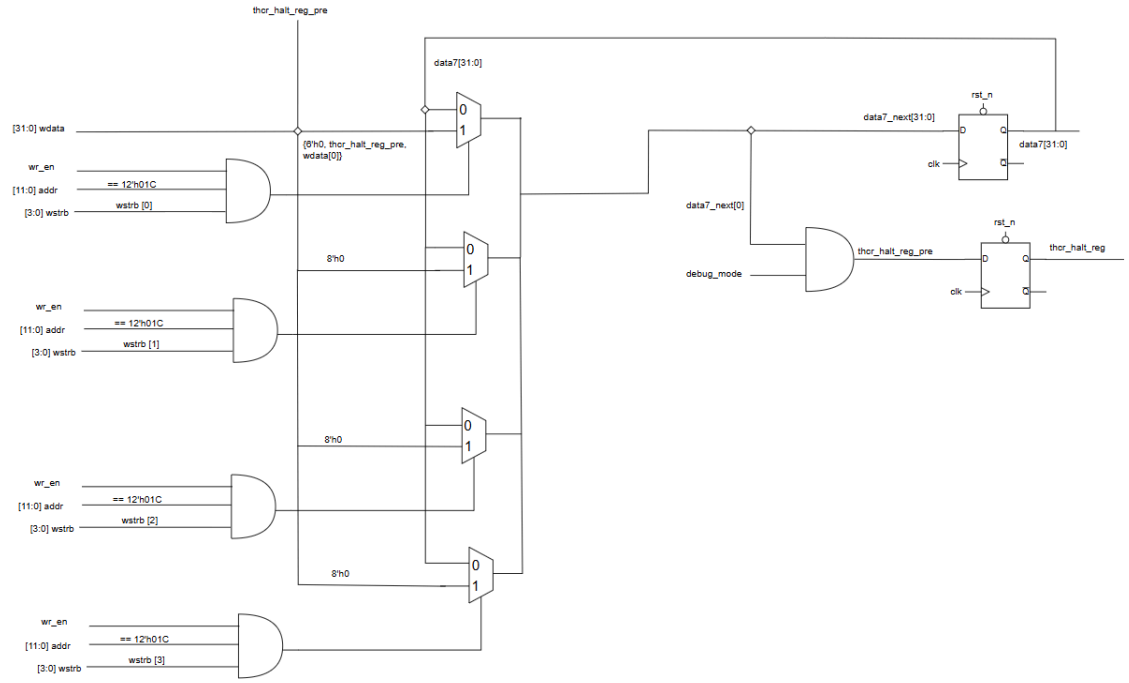


Table 10. Timer Halt Control Status Register (THCSR)

Bit	Name	Type	Default value	Description
31:12	Reserved	RO	30'h0	Reserved
1	halt_ack	RO	1'b0	Timer halt acknowledge 0: timer is NOT halted 1: timer is halted Timer accepts the halt request only in debug mode, indicates by debug_mode input signal
0	halt_req	RW	1'b0	Timer halt request 0: no halt req. 1: timer is requested to halt.



Timer Halt Control and Status Register (THCSR) - DATA7 Description

1. Introduction

The Timer Halt Control and Status Register (THCSR), internally referenced as data7, is a 32-bit register controlling the timer's debug mode functionality. It provides the ability to pause timer operation during system debugging.

2. Register Structure

2.1. Register Bit Field Mapping

Bit Position	Field Name	Access Type	Reset Value	Description
31:2	Reserved	Read-Only	30'h0	Not used
1	HALT_STATUS	Read-Only	1'b0	Current halt status
0	THCSR[0]	Read/Write	1'b0	Halt enable control

2.2. Detailed Bit Field Description

Bit 0 - THCSR[0] (Halt Enable):

- Value 0: Timer runs normally.
- Value 1: Can only halt when in debug mode.

Bit 1 - HALT_STATUS (Halt Status):

- Value 0: Timer is running.
- Value 1: Timer is halted.

3. Register Implementation Details

3.1. Write Access

Only byte 0 (bits 7:0) can be written via the wstrb[0] signal. Bit 1 (HALT_STATUS) is read-only and displays the current halt status. The upper bytes (31:8) always read as 0.

3.2. Reset State

The default reset value is 32'h0000_0000. The halt function is disabled after system reset.

4. Operation

4.1. Halt Control Logic

The halt function has two control levels: software control via bit THCSR[0] and system control via the debug_mode input signal.

4.2. Halt Conditions

The timer halts when all of the following conditions are met:

- Bit THCSR[0] has value 1
- The system is in debug mode

4.3. Effects When Halted

When halted:

- The main 64-bit counter stops incrementing
- The internal prescaler counter stops counting
- Register access remains available

4.4. Usage Examples

To enable debug halt mode:

1. Software configuration:
    - Write value 1 to bit THCSR[0] to enable halt request mode
2. Hardware activation:
    - Set the debug_mode signal to 1 to put the system into debug mode
3. Result:

    - The status bit THCSR[1] is automatically set to 1
    - The timer counter stops counting and preserves the current value
    - Registers remain accessible for debugging purposes

To resume normal operation:

1. Disable debug mode:
    - Set the debug_mode signal back to 0
2. Or disable halt request:
    - Write value 0 to bit THCSR[0]
    - Bit THCSR[1] will automatically return to 0
    - The timer resumes counting from the preserved value

Reading current status:

- Read the THCSR register to know the halt status
- THCSR[1] = 0: Timer is running normally
- THCSR[1] = 1: Timer is halted in debug mode

3. Functional Description

3.1. APB slave

APB Slave Interface Description

1. Purpose of the APB Slave Module

The APB Slave interface module serves as the primary communication bridge between the timer module and the system bus (AMBA APB). Its main purposes are:

- Protocol Translation: Convert APB bus transactions into internal register access signals for the timer.
- Signal Conditioning: Decode APB signals and generate appropriate control signals for the register module.
- Error Detection: Validate incoming transactions and generate error responses when necessary.
- Timing Control: Manage handshake signals (pready, pslverr) for proper bus timing.

2. Operation Principle

Signal Interface

The APB Slave module receives input signals from the APB bus and the register module, then processes them to generate appropriate control and response signals.

Inputs from APB Bus:

- psel: Peripheral select signal.
- pwrite: Write/read indicator (1 = write, 0 = read).
- penable: Enable phase indicator.
- paddr[11:0]: 12-bit address bus.
- pwdata[31:0]: 32-bit write data bus.
- pstrb[3:0]: Byte strobe signals for write operations.

Inputs from Register Module:

- pready_w: Internal ready signal from registers.
- rdata[31:0]: Read data from registers.
- data0_out[31:0]: Current value of control register (for error checking).

Outputs to Register Module:

- addr[11:0]: Decoded address.
- wr_en: Write enable pulse.
- rd_en: Read enable pulse.
- wdata[31:0]: Write data.
- wstrb[3:0]: Write byte strobes.

Outputs to APB Bus:

- pready: Transfer ready indicator.
- prdata[31:0]: Read data output.
- pslverr: Slave error indicator.

Operation Flow

The APB Slave operation is described through the following steps:

- Transaction Detection:
  - Waits for both psel and penable signals to be asserted (psel && penable == 1).
  - Checks the pwrite signal to determine operation type (write or read).
- Control Signal Generation:
  - Generates wr_en signal for write operations when: psel && penable && pwrite && !pready_w.
  - Generates rd_en signal for read operations when: psel && penable && !pwrite && !pready_w.
  - Address (addr), write data (wdata), and byte strobe (wstrb) signals are passed directly from the APB bus to the register module.
- Error Checking:
  - Monitors write operations to the control register (TCR at address 0x000).
  - Validates the prescaler value (must be less than 9).
  - Prevents modification of the operating mode or prescaler value when the timer is active.
  - Sets pslverr = 1 when error conditions are detected.
- Response Management:
  - Forwards the internal pready_w signal from the register module to the APB bus as pready.
  - Forwards read data rdata from the register module to the APB bus as prdata.
  - Propagates any detected errors through the pslverr signal.

Error Conditions Detected

The APB Slave performs semantic validation on write transactions:

When writing to the control register TCR (address 0x000):

- If the timer is currently enabled (TCR[0] == 1):
    - ERROR if attempting to change the operating mode (bit TCR[1]).
    - ERROR if the new prescaler value (TCR[11:8]) is invalid (≥ 9).
    - ERROR if the new prescaler value differs from the current prescaler value (even if the new value is valid).
- If the timer is currently disabled (TCR[0] == 0):
    - ERROR if the new prescaler value (TCR[11:8]) is invalid (≥ 9).

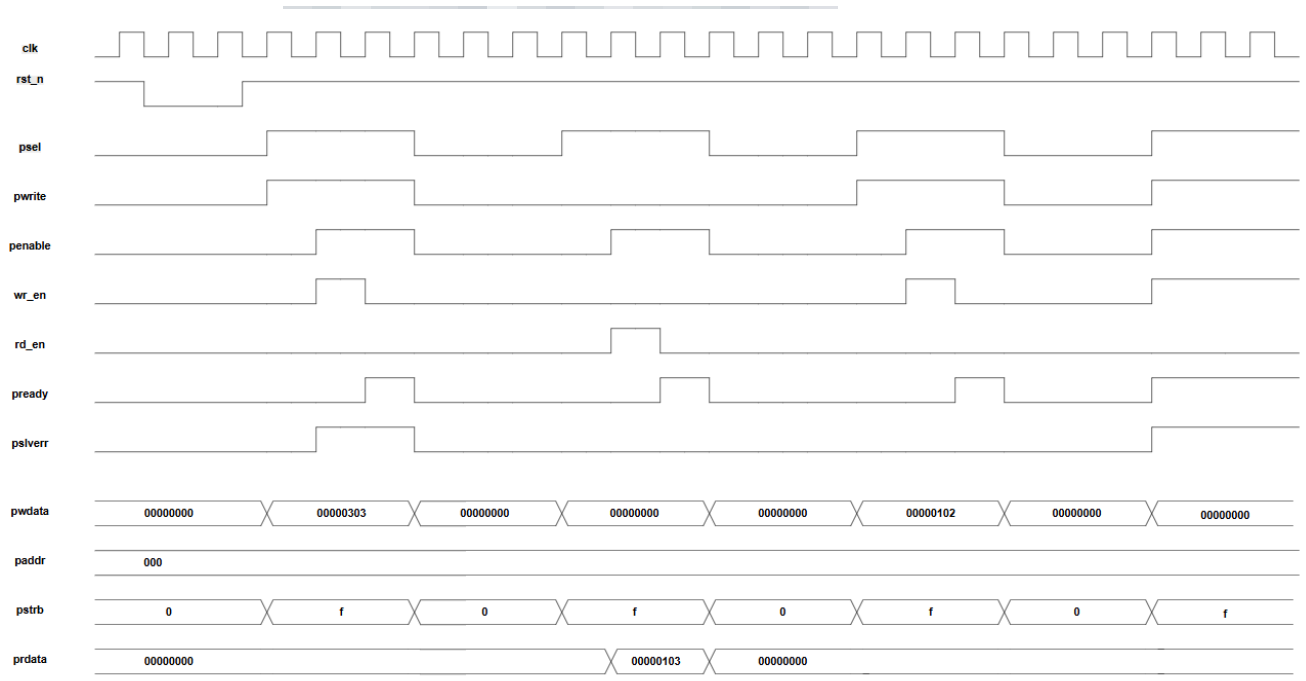
Key Design Features

- Combinational Logic: The entire APB decoding process is implemented using combinational logic to achieve minimum latency.
- Byte-Level Access Support: Full support for APB byte strobes (pstrb), allowing writes to individual bytes of 32-bit registers.
- Protection Mechanism: Prevents runtime configuration changes that could cause unpredictable timer behavior.
- Minimal Overhead: Simple pass-through design with an error-checking layer on top.

- APB 3.0 Compliance: Fully implements the slave interface as required by the APB 3.0 standard.

Timing Characteristics

- Setup Phase: When psel = 1 and penable = 0 (not used in this design).
- Access Phase: When psel = 1 and penable = 1 (generates wr_en/rd_en pulses).
- Response: pready is asserted one cycle after wr_en/rd_en (based on the signal from the register module).
- Error Reporting: pslverr is generated immediately for invalid write operations.



APB Protocol Waveform Signals

wr_en: A pulse that is activated high when psel, penable, and pwrite are all high and pready_w is low. This signal indicates a valid write command from the master and is sent to the register module to store data into the registers.

rd_en: A pulse that is activated high when psel and penable are both high, pwrite is low, and pready_w is low. This signal indicates a valid read command from the master and is used to trigger reading data from the registers.

addr, wdata, wstrb: These are pass-through signals transmitted directly from the input APB signals (paddr, pwdata, pstrb) without processing. They provide the address, write data, and byte strobe for the register module.

pready: Directly passed through from pready_w, it signals to the master that the slave is ready for the current transaction. When high, it allows the master to complete the transaction.

prdata: Directly passed through from rdata, it contains the data read from the registers to be returned to the master during a read operation.

pslverr: Generated by the pslverr_w logic, it indicates an error in communication. It is activated high when violations are detected in the rules for writing to the DATA0 register (address 0x000), such as changing the mode while the timer is enabled or an invalid prescaler value (≥9).

Operation flow:

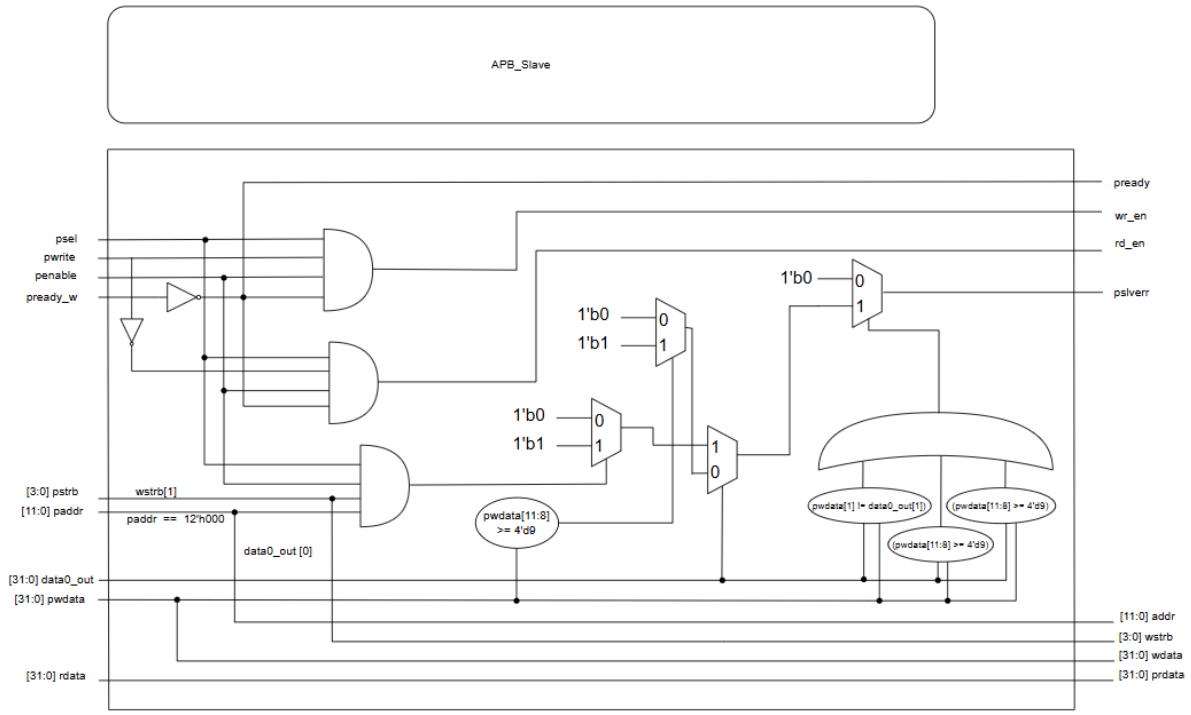
When the master wants to communicate, it activates psel and sets the address (paddr), then activates penable.

The slave module checks pwrite to determine the transaction type (write/read) and generates the corresponding wr_en or rd_en signal.

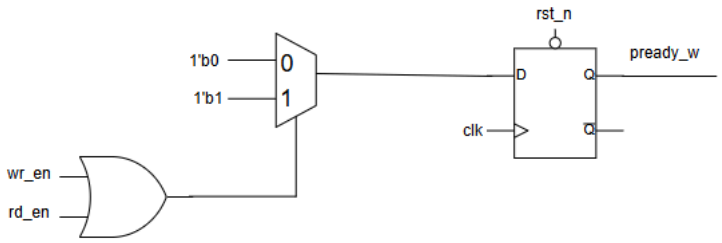
Data is transmitted between the master and slave via the pwdata/prdata buses.

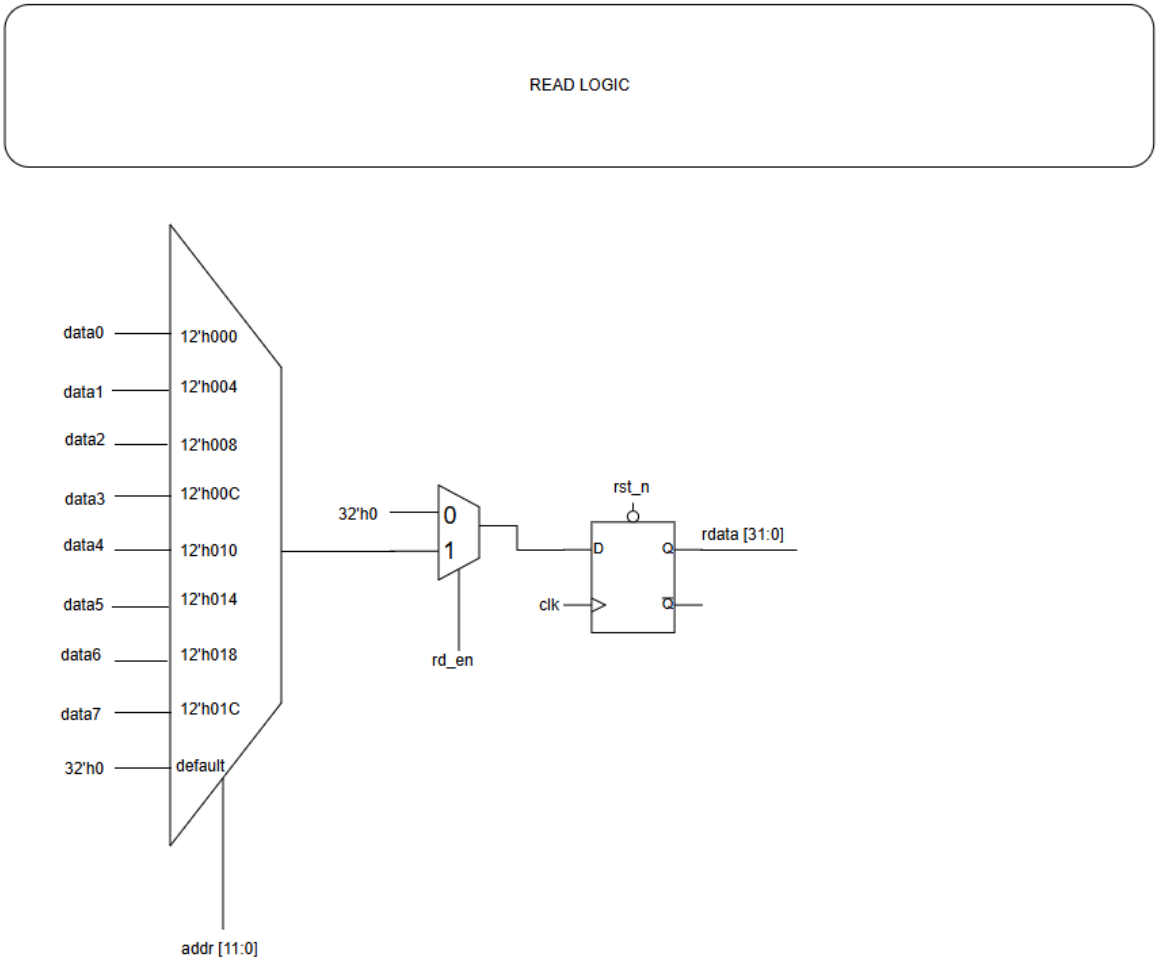
The module checks the validity of write transactions to address 0x000 and generates pslverr if an error is detected.

The transaction only completes when pready is high, allowing the master to proceed to the next state.



Wait state (1cycle)





3.2.PREADY Signal, Wait State, and Read Logic Description

1. Introduction

The PREADY signal, wait state mechanism, and read logic form the core data transfer control system of the timer module's APB interface. These components manage the timing and flow of data transactions between the APB bus and the timer registers, ensuring synchronized and reliable communication.

2. PREADY Signal and Wait State Mechanism

2.1. PREADY Signal Function

Purpose:
The PREADY (Peripheral Ready) signal is an essential component of the APB 3.0 protocol that

indicates when a data transfer transaction is complete and the bus is ready for the next operation.

2.2. Wait State Implementation

Control Logic:
The wait state control operates on a simple sequential principle where pready_w is asserted one clock cycle after detecting a read or write enable signal.

Timing Characteristics:

- Initial State: pready_w = 0 after system reset
- Activation: When wr_en or rd_en becomes 1, pready_w is set to 1 at the next clock edge
- Deactivation: Returns to 0 when no transaction is active
- Fixed Delay: Always introduces one wait state per transaction

2.3. Timing Diagram

The timing sequence follows a consistent pattern where the ready signal is delayed by exactly one clock cycle from the transaction initiation, ensuring proper setup and hold times for all register operations.

3. Read Logic Implementation

3.1. Read Operation Structure

Synchronous Design:
All read operations are synchronized to the system clock (sys_clk), with read data (rdata) updated on the rising clock edge. This synchronous approach guarantees timing consistency throughout the system.

Address Decoding:
The read logic implements a complete address decoding system that maps 12-bit APB addresses to corresponding internal registers:

- 0x000: Control Register (TCR)
- 0x004: Timer Counter Low (TDR0)
- 0x008: Timer Counter High (TDR1)
- 0x00C: Compare Value Low (TCMP0)
- 0x010: Compare Value High (TCMP1)
- 0x014: Interrupt Enable Register (TIER)

- 0x018: Interrupt Status Register (TISR)
- 0x01C: Halt Control Register (THCSR)

Default Behavior:

- Invalid addresses return 0x00000000
- Inactive read cycles (rd_en = 0) maintain rdata = 0

3.2. Read Operation Sequence

Complete Transaction Flow:

1. Setup Phase: APB master asserts psel with target address
2. Enable Phase: penable is asserted, activating the transaction
3. Processing Phase: Internal logic decodes address and retrieves data
4. Response Phase: pready is asserted and data is presented on prdata

Clock Cycle Allocation:

- Cycle 0: Address setup and select assertion
- Cycle 1: Internal processing and data retrieval
- Cycle 2: Ready signal assertion and data transfer completion

4. History

Date	Version	Description	Author
23/10/2025	1.0	Newly created	Thanh Hao
30/10/2025	1.1	Added APB slave interface logic	Thanh Hao
06/11/2025	1.2	Implemented register module with timer control	Thanh Hao
13/11/2025	1.3	Added 64-bit counter and prescaler logic	Thanh Hao
20/11/2025	1.4	Integrated interrupt generation logic	Thanh Hao
27/11/2025	1.5	Added debug mode halt control (THCR)	Thanh Hao
04/12/2025	1.6	Fixed PSLVERR generation and timing issues	Thanh Hao
11/12/2025	1.7	Final verification and clean-up	Thanh Hao