

General Description

The Watchdog is a simple microprocessor watchdog timer IP core. Once enabled, a register in the watchdog timer must be written to at regular intervals. If the microprocessor fails to write to the watchdog timer's register in the set time, normally because of a software lockup, the watchdog timer asserts the system reset line and causes the system to reboot.

The watchdog timer also keeps a count of the number of times a watchdog reset has occurred.

Build-Time Options

- ◆ 1 cycle or 2 cycle read accesses.
- ◆ Clock prescale divider to allow a wide variety of clock frequencies to be supported.

Features

- ◆ Programmable reset pulse width.
- ◆ Programmable reset timeout delay.
- ◆ Immediate system reset control.
- ◆ Keeps a count of watchdog resets.
- ◆ Core uses a single clock domain.
- ◆ Industry standard PPCI compliant processor interface

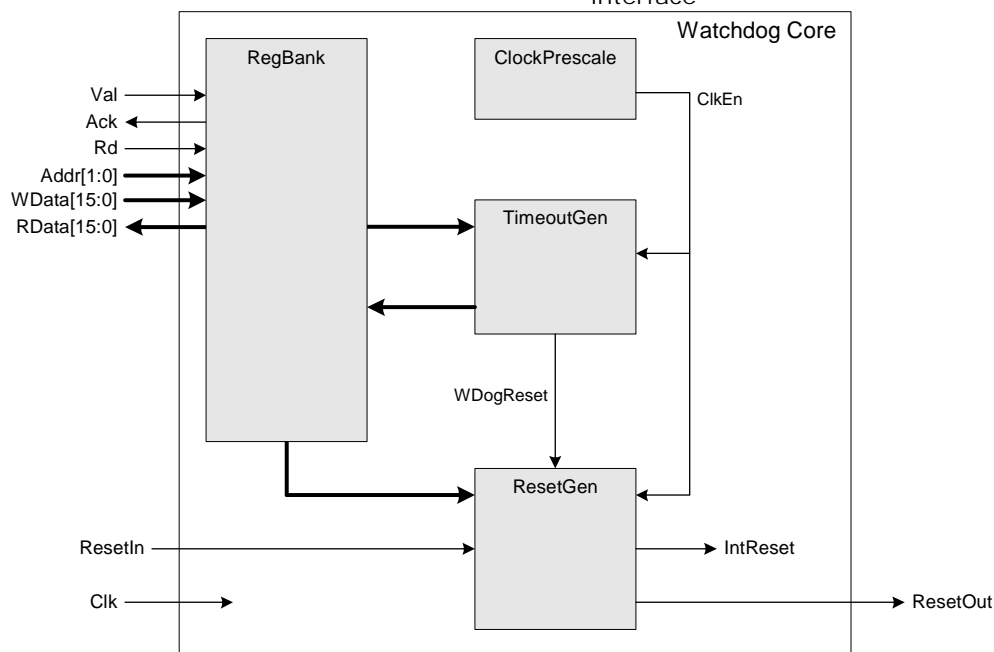


Figure 1: Watchdog Core Block Diagram

Functional Description

The Watchdog core block diagram shows the four blocks that form the Watchdog core. The function of each of these blocks is described in turn next.

RegBank

The *Register Bank* block implements the processor interface and the register bank. The values written to the control registers in the register bank are used to control the TimeoutGen and ResetGen blocks.

ClkPrescale

The *Clock Prescale* block produces a clock enable, ClkEn, effectively dividing the input clock rate by the generic, gClkDivisor, plus 1. The generic, gClkDivisor, should be set to produce a prescaled clock with a frequency as close to 1 KHz as possible. The Clock Prescale block is not affected by any register settings.

TimeoutGen

The *Timeout Generation* block monitors microprocessor writes to the RSTTIMER (Reset Timer) register, resetting the timeout count each time a 1 is written to bit 0 of this register. If the timeout timer does reach the value set in the TIMEOUT register, the WDogReset signal is asserted for one clock cycle to tell the ResetGen block that a watchdog reset is required. The TimeoutGen block also keeps a count of the number of watchdog resets that have occurred.

ResetGen

The *Reset Generation* block waits until the WDogReset signal is strobed, then it asserts the ResetOut signal for the number of prescaled clock ticks specified by the PULSELEN (Pulse Length) register. The ResetGen block also takes the main reset input, ResetIn, and produces an internal reset signal that does not get activated when the

reset received was caused by a watchdog reset.

Port & Generic Descriptions

The Watchdog core has 40 ports at its top level, which are described in Table 1 below.

Port	Direction	Description
System Ports		
Clk	Input	Master clock input
Processor Interface Ports		
Val	Input	Valid strobe input.
Ack	Output	Acknowledge strobe output.
Rd	Input	Read/Not Write strobe input.
Addr[1:0]	Input	2-bit address bus for register read/writes.
WData[15:0]	Input	16-bit Write Data bus for register writes.
RData[15:0]	Output	16-bit Read Data bus for register reads.
Reset Ports		
ResetIn	Input	Main Reset input.
ResetOut	Output	Watchdog reset output.

Table 1: Watchdog core ports

The Watchdog core has 2 generics than can be modified when the component is instantiated to control the build-time options. These are listed in Table 2 below.

Generic	Range	Description
gExtndRd	0 to 1	0 = Normal read accesses. 1 = Extended read accesses. (0 by default)
gClkDivisor	0 to 262143	Amount (less 1) to divide the main clock to produce the prescale clock.

Table 2: Watchdog core generics

The value of the generic gClkDivisor should be chosen to produce a prescaled clock of 1 KHz where possible. The formula for calculating gClkDivisor is:

$$gClkDivisor = (\text{Input-Clock-Freq} / 1 \times 10^3) - 1$$

Thus, for a 40 MHz input clock the calculation will be:

$$gClkDivisor = (40 \times 10^6 / 1 \times 10^3) - 1 = 39999.$$

This is not actually essential, but does allow the values in the TIMEOUT and PULSELEN registers to always refer to prescale clock ticks 1 ms long (1 KHz frequency).

Processor Interface

The processor interface is a synchronous interface, with all signals being generated on the rising edge of Clk and sampled on the subsequent rising edge of Clk.

Processor Read Cycle

The processor initiates a read cycle by driving the Addr[1:0] lines with the address or the register to be read, the Rd signal high to indicate it is a read access and the Val signal high to indicate the other signals are valid.

The processor then waits in this state until the device drives the RData[15:0] lines with the data and drives the Ack line high. Once the processor has seen the Ack signal high and latched the data

on the RData lines, the read cycle is complete.

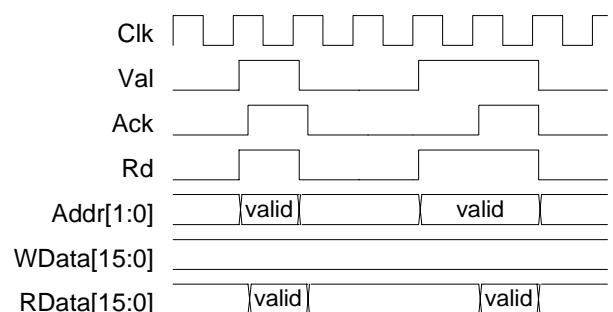


Figure 2: Example processor read cycles

The waveforms in Figure 2 show two separate read accesses. In the first access the peripheral asserts the Ack signal soon after the Val signal has gone high (using combinatorial logic) and the

read cycle completes in 1 Clk cycle.

In the second read access, the peripheral does not assert the Ack signal until 1 Clk cycle after the Val signal has been asserted. Because of this the read cycle is extended and takes 2 Clk cycles.

The Watchdog core behaves like the first read access in Figure 2 when the generic gExtndRd is set to 0, with the Ack signal effectively being the Val signal fed back to the processor.

When gExtndRd is set to 1 the Watchdog core behaves like the second read access in Figure 2, with the Ack signal being asserted 1 Clk cycle after the Val signal is asserted. This option is designed to be used when the frequency of Clk is too high to complete a read access in one Clk cycle.

Processor Write Cycle

The processor initiates a write cycle by driving the required register address onto the Addr[1:0] lines, the data to be written onto the WData lines, the Rd signal low and the Val signal high to indicate the other signals are valid.

The processor then waits until the peripheral asserts the Ack signal to indicate the write cycle is complete.

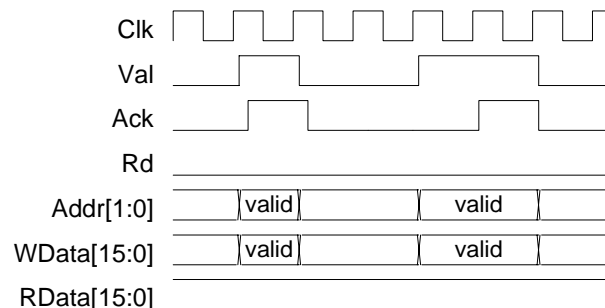


Figure 3: Example processor write cycles

The waveforms in Figure 3 show two separate write accesses. Again, in the first access the peripheral asserts the Ack signal soon after the Val signal has gone high (using combinatorial logic) and the write cycle is complete in 1 Clk cycle.

In the second write access, the peripheral does not assert the Ack signal until 1 cycle after the Val signal has gone high. Again this causes the processor to extend the cycle until the Ack signal is seen, and thus the write cycle takes 2 Clk cycles.

The Watchdog core always behaves like the first write access in Figure 3, with the Ack signal just being the Val signal fed back to the processor.

Notes:

- Read/write accesses can occur back-to-back.
- Although the Watchdog core's write access always behaves like the first access in Figure 3, it is advisable that any bridge logic is able to handle the extended version of the accesses also, as they may be used in future versions of the core.

Register Descriptions

The Watchdog core has 4 internal 16-bit registers. Which register is accessed depends on address lines, Addr[1:0]. Table 3 below summarise these registers.

Addr[1:0]	R/W	Register
00	R/W	TIMEOUT: Timeout length.
01	R/W	PULSELEN: Reset pulse length.
10	R/W	RSTTIMER: Reset the timeout timer.
11	R	RSTCOUNT: Reset Count.

Table 3: Watchdog registers

TIMEOUT: Timeout Length Reg (0x0)

This register specifies the length of time the watchdog will wait without a 1 being written to bit 0 of its RSTTIMER register before it will do a watchdog reset.

Bit	Default	Description
15:12	0x0	Not Used.
11:0	0xFFF	Timeout Length.

The Timeout Length is specified in prescale clock periods (1 ms each if generic gClkDivisor is set up as recommended). If a minimum timeout value of X prescale clock ticks is required, the TIMEOUT register should be set to X + 1.

PULSELEN: Reset Pulse Length Reg (0x1)

The Reset Pulse Length register specifies the length of time that the ResetOut output is asserted for in the event of a watchdog timeout.

Bit	Default	Description
15:4	0x000	Not Used.
3:0	0x5	Reset Pulse Length.

The Reset Pulse Length is specified in prescale clock periods (1 ms each if generic gClkDivisor is set up as recommended).

RSTTIMER: Reset Timeout Timer Reg (0x2)

The Reset Timeout Timer register is the watchdog register that, in normal operation, the processor must write 0x0001 to regularly to stop the Watchdog core from asserting its ResetOut output.

Bit	Default	Description
15:2	0x0000	Not Used.
1	0	Immediate System reset.
0	0	Restart Timeout Timer.

Writing a 1 to bit 1 of this register causes an immediate reset, irrespective of the value written to bit 0. This may be used by the software to perform a hard reset of the system.

Writing a 1 to bit 0 of this register causes a Timeout period to begin, to prevent a watchdog reset from occurring, the processor must write to this register again before the timeout period has elapsed.

Writing a 0 to bit 0 of this register cancels any timeout periods that are in process and leaves the watchdog core disabled.

RSTCOUNT: Reset Count Reg (0x3)

The read only Reset Count register provides a count of the number the times the Watchdog core has reset the system since the last reset that was not initiated by the watchdog core.

Bit	Default	Description
15:0	0x0000	Count of Watchdog Resets

Software Operation

This section can be ignored if the Watchdog device driver for eCos is going to be used.

Initialisation of the Watchdog Core

To initialise the watchdog core the following procedure is recommended.

Read the RSTTIMER register (optional):

- If the RSTTIMER register value is 0x0000, then the last reset was a real (non-watchdog) system reset.
- If bit 1 is set then the previous reset was an immediate system reset initiated by software.
- If bit 0 is set (and not bit 1) then the previous reset was a watchdog timeout reset.

- If the RSTTIMER register has a non-zero value, then the RSTCOUNT register may be read to find out how many times the watchdog has reset the system since the last real system reset.

Set-up the PULSELEN register:

- The PULSELEN register should be set to specify the length of reset pulse required. A value of zero should not be used.

Set-up the TIMEOUT register:

- The TIMEOUT register should be set to specify the timeout delay for the watchdog timer. The TIMEOUT register should always be programmed with a value greater than 1.

The watchdog core is now initialised and ready to begin normal operation.

Normal Operation of the Watchdog Core

Write 0x0001 to the RSTTIMER register:

- During normal operation the software must write 0x0001 to the RSTTIMER register at regular intervals to prevent the watchdog core from resetting the system. This first write to the RSTTIMER register with 0x0001 actually starts the watchdog timer running.

Disabling the Watchdog Core

Write 0x0000 to the RSTTIMER register:

- Writing 0x0000 to the RSTTIMER register causes the watchdog operation to be disabled, with no more writes to the RSTTIMER being required. The watchdog operation may be re-enabled by writing 0x0001 to the RSTTIMER register.

Initiating an Immediate System Reset

Write 0x0002 to the RSTTIMER register:

- Writing 0x0002 to the RSTTIMER register causes the watchdog to immediately reset the system by asserting its ResetOut output.

Sweeney Design Ltd

40 Hopkins Close
Cambridge
CB4 1FD
United Kingdom

Tel: +44 (0) 7977 129406
Fax: +44 (0) 7977 045842
E-mail: cores@sweeneydesign.co.uk
URL: www.sweeneydesign.co.uk