ARM Watchdog Module (SP805)

Revision: r1p0

Technical Reference Manual

ARM®

ARM DDI 0270B

ARM Watchdog Module (SP805)
Technical Reference Manual

Release Information

Change history

Date	Issue	Change
31 October 2002	A	First release
14 November 2003	B	Change security to Open Access

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
ARM Watchdog Module (SP805) Technical Reference Manual

List of Tables
ARM Watchdog Module (SP805) Technical Reference Manual

List of Figures
ARM Watchdog Module (SP805) Technical Reference Manual

Preface

This preface introduces the ARM Watchdog module (SP805) and its reference documentation. It contains the following sections:

About this book

This document is the *Technical Reference Manual* (TRM) for the ARM Watchdog module (SP805).

Intended audience

This document in intended for hardware and software engineers implementing *System-on-Chip* (SoC) designs.

Using this book

This book is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an introduction to the ARM Watchdog module and its features.

Chapter 2 ***Functional Overview***

Read this chapter for an overview of the major functional blocks and the operation of the ARM Watchdog module.

Chapter 3 ***Programmer's Model***

Read this chapter for a description of the registers and for details of system initialization.

Chapter 4 ***Programmer's Model for Test***

Read this chapter for a description of the additional logic for functional verification and production testing.

Appendix A ***Signal Descriptions***

Read this appendix for a description of the ARM Watchdog module signals.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this document, where:

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

Typographical conventions

The following typographical conventions are used in this book:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes ARM processor signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
`monospace italic`	Denotes arguments to commands and functions where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.

Other conventions

This document uses other conventions. They are described in the following sections:

- *Signals*
- *Bytes, Halfwords, and Words* on page xii
- *Bits, bytes, k, and M* on page xii
- *Register fields* on page xii.

Signals

When a signal is described as being asserted, the level depends on whether the signal is active HIGH or active LOW. Asserted means HIGH for active high signals and LOW for active low signals:

Prefix n	Active LOW signals are prefixed by a lowercase n except in the case of AHB or APB reset signals. These are named **HRESETn** and **PRESETn** respectively.
Prefix H	AHB signals are prefixed by an upper case H.
Prefix P	APB signals are prefixed by an upper case P.

Bytes, Halfwords, and Words

Byte Eight bits.

Halfword Two bytes (16 bits).

Word Four bytes (32 bits).

Quadword 16 contiguous bytes (128 bits).

Bits, bytes, k, and M

Suffix b Indicates bits.

Suffix B Indicates bytes.

Suffix K When used to indicate an amount of memory means 1024. When used to indicate a frequency means 1000.

Suffix M When used to indicate an amount of memory means $1024^2 = 1\ 048\ 576$. When used to indicate a frequency means 1 000 000.

Register fields

All reserved or unused address locations must not be accessed as this can result in unpredictable behavior of the device.

All reserved or unused bits of registers must be written as zero, and ignored on read unless otherwise stated in the relevant text.

All registers bits are reset to logic 0 by a system reset unless otherwise stated in the relevant text.

Unless otherwise stated in the relevant text, all registers support read and write accesses. A write updates the contents of the register and a read returns the contents of the register.

All registers defined in this document can only be accessed using word reads and word writes, unless otherwise stated in the relevant text.

Further reading

This section lists publications from ARM Limited that provide additional information about ARM devices.

ARM periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets, addenda, and Frequently Asked Questions.

ARM publications

This document contains information that is specific to the ARM Watchdog module. Refer to the following document for other relevant information:

- *AMBA Specification (Rev 2.0)* (ARM IHI 0011).

Feedback

ARM Limited welcomes feedback on both the ARM Watchdog module (SP805), and its documentation.

Feedback on the ARM Watchdog module (SP805)

If you have any problems with the ARM Watchdog module (SP805), contact your supplier. To help us provide a rapid and useful response, give:

- details of the release you are using
- details of the platform you are running on, such as the hardware platform, operating system type and version
- a small standalone sample of code that reproduces the problem
- a clear explanation of what you expected to happen, and what actually happened
- the commands you used, including any command-line options
- sample output illustrating the problem
- the version string of the tool, including the version number and date.

Feedback on this book

If you have any comments on this book, send email to `errata@arm.com` giving:

- the document title
- the document number
- the page number(s) to which your comments apply
- a concise explanation of your comments.

General suggestions for additions and improvements are also welcome.

Chapter 1
Introduction

This chapter introduces the ARM Watchdog module (SP805). It contains the following section:

- *About the Watchdog module (SP805)* on page 1-2.

1.1 About the Watchdog module (SP805)

The Watchdog module is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral developed, tested and licensed by ARM Limited.

The Watchdog module is an AMBA slave module and connects to the *Advanced Peripheral Bus* (APB). The Watchdog module consists of a 32-bit down counter with a programmable timeout interval that has the capability to generate an interrupt and a reset signal on timing out. It is intended to be used to apply a reset to a system in the event of a software failure.

The Watchdog module is further described in:

- *Features*
- *Programmable parameters* on page 1-3.

1.1.1 Features

The features of the Watchdog module are:

- Compliance to the *AMBA Specification (Rev 2.0)* for easy integration into an SoC implementation.
- 32-bit down counter with a programmable timeout interval.
- Separate Watchdog clock with clock enable for flexible control of the timeout interval.
- Interrupt output generation on timeout.
- Reset signal generation on timeout if the interrupt from the previous timeout remains unserviced by software.
- Lock register to protect registers from being altered by runaway software.
- Identification registers that uniquely identify the Watchdog module. These can be used by software to automatically configure itself.

Figure 1-1 on page 1-3 shows a simplified block diagram of the Watchdog module.

Note

Test logic is not shown in Figure 1-1 on page 1-3 for clarity.

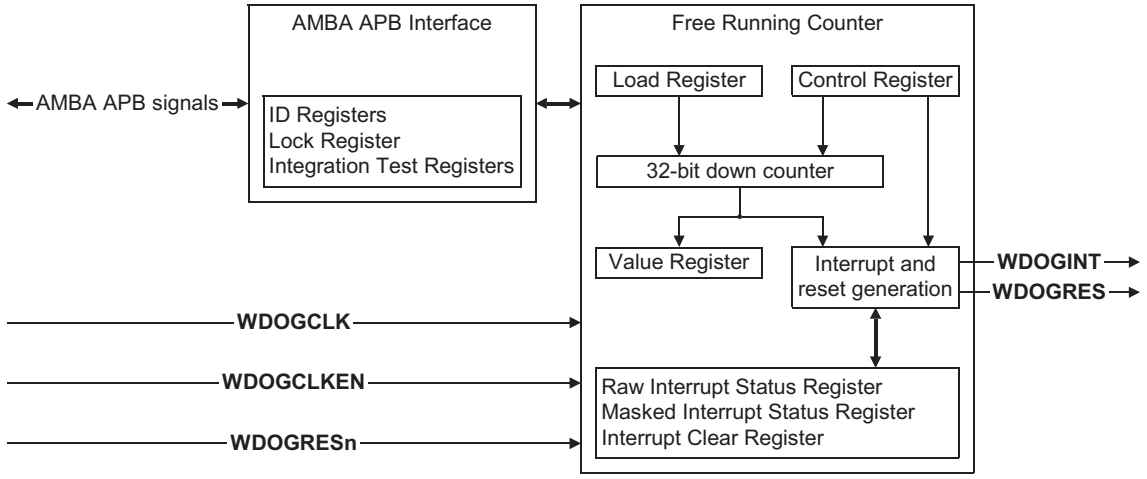

Figure 1-1 Simplified block diagram

1.1.2 Programmable parameters

The following Watchdog module parameters are programmable:

- interrupt generation enable/disable
- interrupt masking
- reset signal generation enable and/disable
- interrupt interval.

Chapter 2
Functional Overview

This chapter describes the ARM Watchdog module (SP805) operation. It contains the following sections:

2.1 Watchdog module (SP805) overview

The Watchdog module is based around a 32-bit down counter that is initialized from the Reload Register, WdogLoad. The counter decrements by one on each positive clock edge of **WDOGCLK** when the clock enable **WDOGCLKEN** is HIGH. When the counter reaches zero, an interrupt is generated. On the next enabled **WDOGCLK** clock edge the counter is reloaded from the WdogLoad Register and the count down sequence continues. If the interrupt is not cleared by the time that the counter next reaches zero then the Watchdog module asserts the reset signal, **WDOGRES**, and the counter is stopped.

WDOGCLK can be equal to or be a sub-multiple of the **PCLK** frequency. However, the positive edges of **WDOGCLK** and **PCLK** must be synchronous and balanced.

The Watchdog module interrupt and reset generation can be enabled or disabled as required by use of the Control Register, WdogControl. When the interrupt generation is disabled then the counter is stopped. When the interrupt is re-enabled then the counter starts from the value programmed in WdogLoad, and not from the last count value.

Write access to the registers in the Watchdog module can be disabled by the use of the Watchdog module Lock Register, WdogLock. Writing a value of `0x1ACCE551` to the register enables write accesses to all of the other registers. Writing any other value disables write accesses to all registers except the Lock Register. This feature protects the Watchdog module registers from being spuriously changed by runaway software that might otherwise disable the Watchdog module operation.

2.2 Functional description

The Watchdog module block diagram is shown in Figure 2-1.

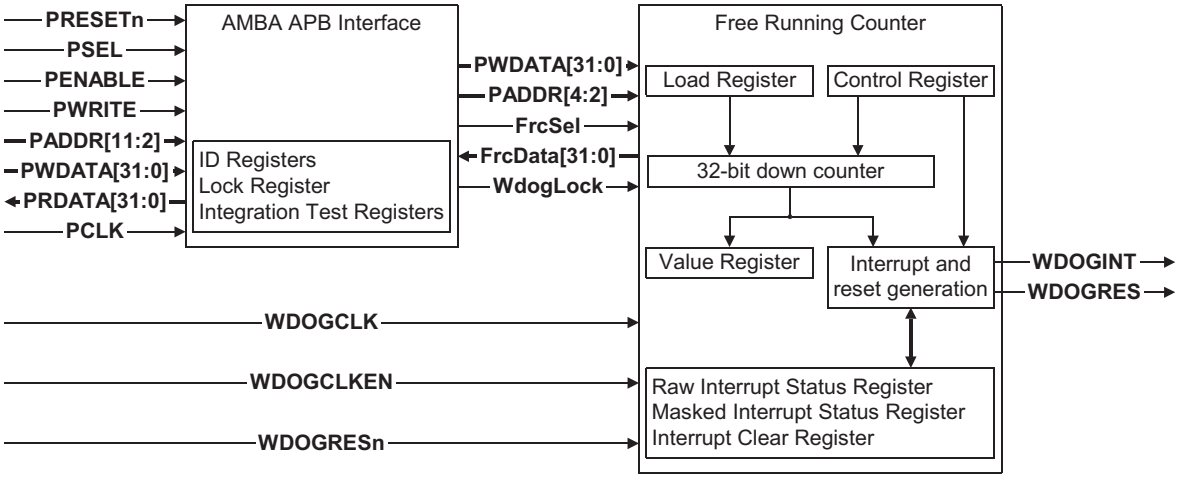

Figure 2-1 Watchdog module block diagram

The Watchdog module is described functionally in:

- *AMBA APB interface*
- *Free running counter block* on page 2-4
- *Interface resets* on page 2-4
- *Clock signals* on page 2-4.

2.2.1 AMBA APB interface

The AMBA APB slave interface generates read and write decodes for accesses to all registers in the Watchdog module.

The Lock Register, WdogLock, is used to control the enabling of write accesses to all the other registers in order to ensure software cannot unintentionally disable the Watchdog module operation.

2.2.2 Free running counter block

The free running counter block contains the 32-bit down counter functionality and generates the interrupt and reset signal outputs. The counter and interrupt/reset logic is clocked independently of **PCLK** by **WDOGCLK** in conjunction with a clock enable, **WDOGCLKEN**, although there are constraints on the relationship between **PCLK** and **WDOGCLK**. See *Clock signals* for details of these constraints.

2.2.3 Interface resets

The Watchdog module is reset by:

- the global reset signal, **PRESETn**
- a block specific reset signal, **WDOGRESn**.

PRESETn can be asserted asynchronously to **PCLK** but must be deasserted synchronously to the rising edge of **PCLK**. **PRESETn** is used to reset the state of the Watchdog module registers. The Watchdog module requires **PRESETn** to be asserted LOW for at least one period of **PCLK**. The values of the registers after reset are defined in Chapter 3 *Programmer's Model*.

WDOGRESn can be asserted asynchronously to **WDOGCLK** but must be deasserted synchronously to the rising edge of **WDOGCLK**. **WDOGRESn** is used to reset the state of registers in the **WDOGCLK** domain. The Watchdog module requires **WDOGRESn** to be asserted LOW for at least one period of **WDOGCLK**.

2.2.4 Clock signals

The Watchdog uses two input clocks:

PCLK This is used to time all APB accesses to the Watchdog module registers.

WDOGCLK

This clock, in conjunction with its clock enable, **WDOGCLKEN**, is used to clock the Watchdog module counter and its associated interrupt and reset generation logic. The Watchdog counter only decrements on a rising edge of **WDOGCLK** when **WDOGCLKEN** is HIGH. The relationship between **WDOGCLK** and **PCLK** must observe the following constraints:

- the rising edges of **WDOGCLK** must be synchronous and balanced with a rising edge of **PCLK**
- the **WDOGCLK** frequency cannot be greater than the **PCLK** frequency.

WDOGCLK and **WDOGCLKEN** can be used in a variety of ways as illustrated by the following examples:

- *WDOGCLK equals PCLK and WDOGCLKEN equals 1*
- *WDOGCLK equals PCLK and WDOGCLKEN is pulsed*
- *WDOGCLK is less than PCLK and WDOGCLKEN is 1* on page 2-6
- *WDOGCLK is less than PCLK and WDOGCLKEN is pulsed* on page 2-6.

WDOGCLK equals PCLK and WDOGCLKEN equals 1

Figure 2-2 shows the case where **WDOGCLK** is identical to **PCLK** and **WDOGCLKEN** is permanently enabled. In this case, the Watchdog module counter is decremented on every **WDOGCLK** edge.

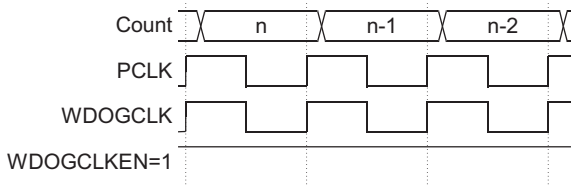

Figure 2-2 Clock example where WDOGCLK equals PCLK and WDOGCLKEN equals 1

WDOGCLK equals PCLK and WDOGCLKEN is pulsed

Figure 2-3 shows the case where **WDOGCLK** is identical to **PCLK** but **WDOGCLKEN** only enables every second **WDOGCLK** edge. In this case, the Watchdog module counter is decremented on every second **WDOGCLK** rising edge.

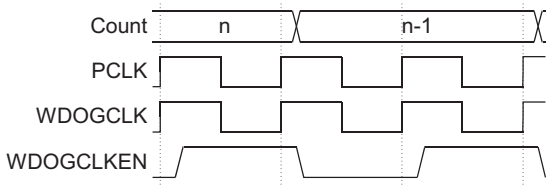

Figure 2-3 Clock example where WDOGCLK equals PCLK and WDOGCLKEN is pulsed

WDOGCLK is less than PCLK and WDOGCLKEN is 1

Figure 2-4 shows the case where the **WDOGCLK** frequency is a submultiple of the **PCLK** frequency but the rising edges of **WDOGCLK** are synchronous and balanced with **PCLK** edges. **WDOGCLKEN** is permanently enabled. In this case, the Watchdog module counter is decremented on every **WDOGCLK** rising edge.

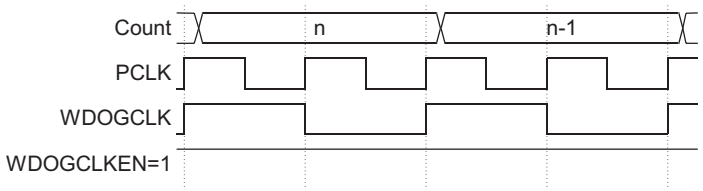

Figure 2-4 Clock example where WDOGCLK is less than PCLK and WDOGCLKEN=1

WDOGCLK is less than PCLK and WDOGCLKEN is pulsed

Figure 2-5 shows the case where **WDOGCLK** frequency is a sub-multiple of the **PCLK** frequency but the rising edges of **WDOGCLK** are synchronous and balanced with **PCLK** edges. **WDOGCLKEN** only enables every second **WDOGCLK** edge. In this case, the Watchdog module counter is decremented on every second **WDOGCLK** rising edge.

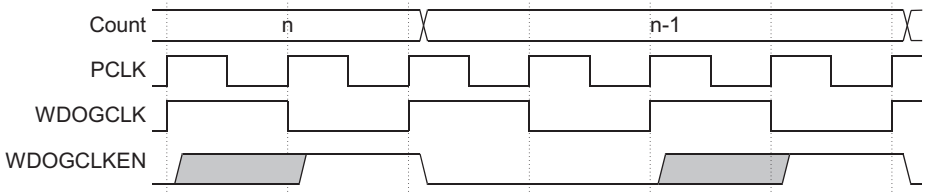

Figure 2-5 Clock example where WDOGCLK is less than PCLK and WDOGCLKEN is pulsed

2.3 Operation

After the initial application and release of **PRESETn** and **WDOGRESn**, the Control Register is reset and interrupt and reset generation is disabled. The Lock Register, WdogLock, is initialized in the unlocked state so that write access to all Watchdog module registers is enabled. The Watchdog counter remains at its initial value (`0xFFFFFFFF`) until the interrupt generation is enabled by setting the INTEN bit in the WdogControl Register.

The WdogLoad Register must be programmed with the desired timeout interval before the Watchdog module is enabled. After the INTEN bit is set, the counter is loaded with the value in the WdogLoad Register on the next rising edge of **WDOGCLK** enabled by **WDOGCLKEN**. On each subsequent enabled **WDOGCLK** rising edge the counter decrements by one. When the counter reaches zero an interrupt is generated and the Watchdog interrupt signal, **WDOGINT**, is asserted. The counter is then reloaded from the value in the WdogLoad Register and starts another count down sequence.

The interrupt is cleared by a write of any data value to the WdogIntClr Register. This causes the counter to reload with the value held in the WdogLoad Register and another count down sequence starts. If the interrupt is not cleared before the counter next reaches zero then the **WDOGRES** signal is asserted if the reset enable bit, RESEN, in the WdogControl Register is set. After the **WDOGRES** signal is asserted, the counter stops.

In a SoC, the **WDOGRES** signal is used to reset a system that has got into an unpredictable state. Therefore, the Watchdog module expects to be reset by **PRESETn** and **WDOGRESn** and the initialization procedure starts again.

To protect the Watchdog module registers from being changed unintentionally, the Lock Register, WdogLock, must be used to disable the write access to the Watchdog module registers after registers have been modified. To enable write access to all registers, write `0x1ACCE551` to the Lock Register, WdogLock. After writing to the required Watchdog registers, disable write access to all registers except the Lock Register by writing any value other than `0x1ACCE551` to the Lock Register. Reading the Lock Register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the lock register returns `0x00000001` (locked) otherwise the return value is `0x00000000` (unlocked).

If the Load Register, WdogLoad, is written to with a new value while the Watchdog counter is decrementing then the counter is reloaded immediately with the new load value and continues decrementing from the new value. Writing to WdogLoad does not clear an active interrupt. An interrupt must be specifically cleared by writing to the Interrupt Clear Register, WdogIntClr.

If the interrupt generation is disabled by clearing the INTEN bit in the Control Register, WdogControl, the counter stops at its current value. When the interrupt generation is enabled again the counter reloads from the Load Register, WdogLoad, and starts to decrement.

2.4 Interrupt behavior

When the Watchdog raises an interrupt by asserting **WDOGINT**, the timing of this signal is generated from a rising clock edge of **WDOGCLK** enabled by **WDOGCLKEN**. When the interrupt is cleared by a write to the Interrupt Clear Register, WdogIntClr, the **WDOGINT** signal is deasserted immediately in the **PCLK** domain rather than waiting for the next enabled **WDOGCLK** rising edge.

Figure 2-6 shows an example of the timing for an interrupt being raised and cleared.

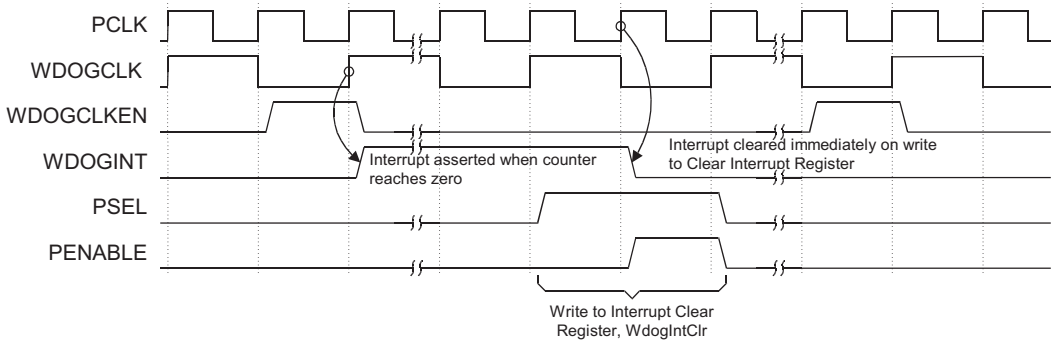

Figure 2-6 Example interrupt signal timing

2.5 Programming the timeout interval

The Watchdog module counter is clocked by the rising edge of **WDOGCLK** when **WDOGCLKEN** is HIGH. In the case where **WDOGCLKEN** is permanently HIGH, the count rate is equal to the **WDOGCLK** frequency. When **WDOGCLKEN** is periodically pulsed HIGH for one **WDOGCLK** rising edge then the count rate is equal to the frequency of the **WDOGCLKEN** pulses. The frequency of enabled clock edges is referred to as the effective watchdog clock frequency and the period is referred to as the effective watchdog clock period.

The Watchdog counter is reloaded from the Load Register, WdogLoad, whenever:

- the counter reaches zero
- the interrupt generation is enabled by setting the INTEN bit in the Control Register, WdogControl, when it was previously disabled
- an interrupt is cleared by writing to the Interrupt Clear register, WdogIntClr
- a new value is written to the Load Register, WdogLoad.

The time interval between the counter load occurring, and the counter reaching zero and generating an interrupt is given by the following expression:

Interrupt interval = (WdogLoad+1) x effective watchdog clock period

The initial reset value for WdogLoad is `0xFFFFFFFF` and for an example effective watchdog frequency of 1MHz (period of 1ms) the interrupt interval is 4295 seconds.

The minimum valid value for WdogLoad is `0x00000001`. If WdogLoad is set to `0x00000000`, an interrupt is always generated immediately.

Table 2-1 on page 2-11 shows examples of WdogLoad values required for a variety of interrupt intervals when the effective watchdog clock frequency is 1MHz.

Table 2-1 Example values for a 1MHz clock

Interrupt interval (ms)	WdogLoad	
	Hex	Decimal
1	0x000003E7	999
50	0x00001387	4999
100	0x0001869F	99999
500	0x0007A11F	499999
1000	0x000F423F	999999

2.6 Identification registers

The Watchdog module contains a set of read-only identification registers that can be used by software to identify the peripheral type and revision. Software can use this information to automatically configure itself.

See Chapter 3 *Programmer's Model* for details of the identification registers.

Chapter 3
Programmer's Model

This chapter describes the registers of the ARM Watchdog module (SP805). It contains the following sections:

3.1 Summery of registers

A summary of the registers is provided in Table 3-1.

Table 3-1 Summary of Watchdog module registers

Address	Type	Width	Reset value	Name	Description
Base + 0x00	Read/write	32	0xFFFFFFFF	WdogLoad	See *Load Register, WdogLoad* on page 3-4
Base + 0x04	Read-only	32	0xFFFFFFFF	WdogValue	See *Value Register, WdogValue* on page 3-4
Base + 0x08	Read/write	2	0x0	WdogControl	See *Control register, WdogControl* on page 3-4
Base + 0x0C	Write-only	-	-	WdogIntClr	See *Interrupt Clear Register, WdogIntClr* on page 3-5
Base + 0x10	Read-only	1	0x0	WdogRIS	See *Raw Interrupt Status Register, WdogRIS* on page 3-5
Base + 0x14	Read-only	1	0x0	WdogMIS	See *Masked Interrupt Status Register, WdogMIS* on page 3-5
Base + 0x18-0xBFC	-	-	-	-	Reserved
Base + 0xC00	Read/write	32	0x0	WdogLock	See *Lock Register, WdogLock* on page 3-5
Base + 0xC04-0xEFC	-	-	-	-	Reserved
Base + 0xF00	Read/write	1	0x0	WdogITCR	See *Integration Test Control Register, WdogITCR* on page 4-4
Base + 0xF04	Write-only	2	0x0	WdogITOP	See *Integration Test Output Set Register, WdogITOP* on page 4-4
Base + 0xF08-0xFDC	-	-	-	-	Reserved
Base + 0xFE0	Read-only	8	0x05	WdogPeriphID0	See *Peripheral Identification Register 0, WdogPeriphID0* on page 3-8
Base + 0xFE4	Read-only	8	0x18	WdogPeriphID1	See *Peripheral Identification Register 1, WdogPeriphID1* on page 3-8
Base + 0xFE8	Read-only	8	0x14	WdogPeriphID2	See *Peripheral Identification Register 2, WdogPeriphID2* on page 3-8

Table 3-1 Summary of Watchdog module registers (continued)

Address	Type	Width	Reset value	Name	Description
Base + 0xFEC	Read-only	8	0x00	WdogPeriphID3	See *Peripheral Identification Register 3, WdogPeriphID3* on page 3-9
Base + 0xFF0	Read-only	8	0x0D	WdogPCellID0	See *PrimeCell Identification Register 0, WdogPCellID0* on page 3-10
Base + 0xFF4	Read-only	8	0xF0	WdogPCellID1	See *PrimeCell Identification Register 1, WdogPCellID1* on page 3-10
Base + 0xFF8	Read-only	8	0x05	WdogPCellID2	See *PrimeCell Identification Register 2, WdogPCellID2* on page 3-10
Base + 0xFFC	Read-only	8	0xB1	WdogPCellID3	See *PrimeCell Identification Register 3,WdogPCellID3* on page 3-11

3.2 Register descriptions

This section describes the Watchdog module registers:

- *Load Register, WdogLoad*
- *Value Register, WdogValue*
- *Control register, WdogControl*
- *Interrupt Clear Register, WdogIntClr* on page 3-5
- *Raw Interrupt Status Register, WdogRIS* on page 3-5
- *Masked Interrupt Status Register, WdogMIS* on page 3-5
- *Lock Register, WdogLock* on page 3-5
- *Peripheral identification registers, WdogPeriphID0-3* on page 3-6
- *PrimeCell Identification Registers, WdogPCellID0-3* on page 3-9.

3.2.1 Load Register, WdogLoad

This is a 32-bit read/write register that contains the value from which the counter is to decrement. When this register is written to, the count is immediately restarted from the new value. The minimum valid value for WdogLoad is 1. If WdogLoad is set to 0 then an interrupt is generated immediately.

3.2.2 Value Register, WdogValue

This read-only 32-bit register gives the current value of the decrementing counter.

3.2.3 Control register, WdogControl

This is a read/write register that enables the software to control the Watchdog module. Table 3-2 shows the bit assignment of the WdogControl Register.

Table 3-2 Control register bit assignment

Bit	Name	Type	Function
[31:2]	-	-	Reserved.
[1]	RESEN	Read/write	Enable Watchdog module reset output, **WDOGRES**. Acts as a mask for the reset output. Set HIGH to enable the reset, and LOW to disable the reset.
[0]	INTEN	Read/write	Enable the interrupt event, **WDOGINT**. Set HIGH to enable the counter and the interrupt, and set LOW to disable the counter and interrupt. Reloads the counter from the value in WdogLoad when the interrupt is enabled, and was previously disabled.

3.2.4 Interrupt Clear Register, WdogIntClr

A write of any value to this location clears the Watchdog module interrupt, and reloads the counter from the value in the WdogLoad Register.

3.2.5 Raw Interrupt Status Register, WdogRIS

This register indicates the raw interrupt status from the counter. The Raw Interrupt Status Register indicates that an interrupt has been raised by the Watchdog counter reaching zero. Table 3-3 shows the bit assignment of the WdogRIS Register.

Table 3-3 Raw Interrupt Status Register bit assignment

Bit	Name	Type	Function
[31:1]	-	-	Reserved
[0]	WDOGRIS	Read	Raw interrupt status from the counter

3.2.6 Masked Interrupt Status Register, WdogMIS

This register indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupt status with the INTEN bit from the Control Register, and is the same value that is passed to the interrupt output pin **WDOGINT**. Table 3-4 shows the bit assignment of the WdogMIS Register.

Table 3-4 Interrupt Status Register bit assignment

Bit	Name	Type	Function
[31:1]	-	-	Reserved
[0]	WDOGMIS	Read	Enabled interrupt status from the counter

3.2.7 Lock Register, WdogLock

This register allows write-access to all other registers to be disabled. This is to prevent rogue software from disabling the Watchdog module operation. Writing a value of `0x1ACCE551` enables write access to all other registers. Writing any other value disables write accesses. A read from this register returns the lock status rather than the value written:

- 0 indicates that write access is enabled (not locked)
- 1 indicates that write access is disabled (locked).

Table 3-5 shows the bit assignment of the WdogLock Register.

Table 3-5 Lock Register bit assignment

Bit	Name	Type	Function
[31:0]	WDOGLOCK	Read/write	Writing 0x1ACCE551 to this register enables write access to all other registers. Writing any other value disables write access to all other registers. A read returns the lock status: 0x00000000 = write access to all other registers is enabled 0x00000001 = write access to all other registers is disabled.

3.2.8 Peripheral identification registers, WdogPeriphID0-3

The WdogPeriphID0-3 registers are four 8-bit registers, that span address locations 0xFE0-0xFEC. The registers can conceptually be treated as a 32-bit register. The read-only registers provide the peripheral options listed in Table 3-6.

Table 3-6 Peripheral Identification Register options

Bits	Function
PartNumber[11:0]	This is used to identify the peripheral. The three digits product code 0x805 is used.
Designer ID[19:12]	This is the identification of the designer. ARM Limited is 0x41 (ASCII A).
Revision[23:20]	This is the revision number of the peripheral. The revision number starts from 0.
Configuration[31:24]	This is the configuration option of the peripheral. The configuration value is 0.

Figure 3-1 on page 3-7 shows the bit assignment for the WdogPeriphID0-3 registers.

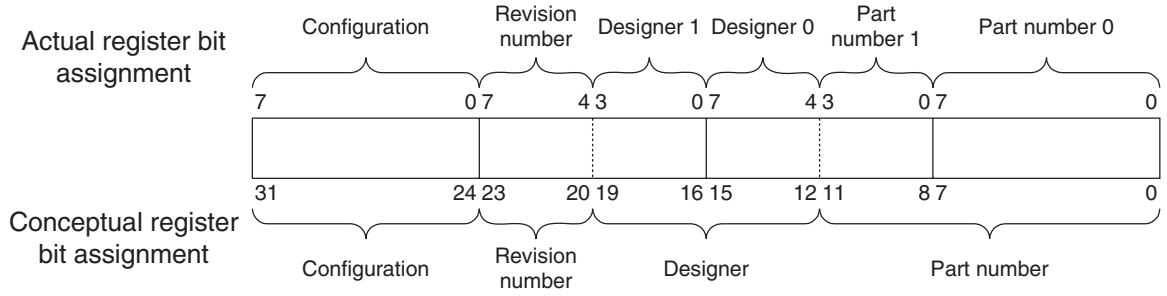

Figure 3-1 Peripheral identification register bit assignment

Note

When you design a system memory map you must remember that the Watchdog module has a 4KB-memory footprint. The 4-bit revision number is implemented by instantiating a component called RevAnd four times with its inputs tied off as appropriate, and the output sent to the read multiplexor. All memory accesses to the Peripheral Identification Registers must be 32-bit, using the `LDR` instruction.

The four, 8-bit peripheral identification registers are described in the following subsections:

- *Peripheral Identification Register 0, WdogPeriphID0* on page 3-8
- *Peripheral Identification Register 1, WdogPeriphID1* on page 3-8
- *Peripheral Identification Register 2, WdogPeriphID2* on page 3-8
- *Peripheral Identification Register 3, WdogPeriphID3* on page 3-9.

Peripheral Identification Register 0, WdogPeriphID0

The WdogPeriphID0 Register is hard-coded and the fields in the register determine the reset value. Table 3-7 shows the bit assignment of the WdogPeriphID0 Register.

Table 3-7 WdogPeriphID0 Register bit assignment

Bit	Name	Description
[31:8]	-	Reserved, read undefined must be written as zeros
[7:0]	PartNumber0	These bits read back as `0x05`

Peripheral Identification Register 1, WdogPeriphID1

The WdogPeriphID1 Register is hard-coded and the fields in the register determine the reset value. Table 3-8 shows the bit assignment of the WdogPeriphID1 Register.

Table 3-8 WdogPeriphID1 Register bit assignment

Bit	Name	Description
[31:8]	-	Reserved, read undefined, must be written as zeros
[7:4]	Designer0	These bits read back as `0x1`
[3:0]	PartNumber1	These bits read back as `0x8`

Peripheral Identification Register 2, WdogPeriphID2

The WdogPeriphID2 Register is hard-coded and the fields in the register determine the reset value. Table 3-9 shows the bit assignment of the WdogPeriphID2 Register.

Table 3-9 WdogPeriphID2 Register bit assignment

Bit	Name	Description
[31:8]	-	Reserved, read undefined, must be written as zeros
[7:4]	Revision	These bits read back as `0x1`
[3:0]	Designer1	These bits read back as `0x4`

Peripheral Identification Register 3, WdogPeriphID3

The WdogPeriphID3 register is hard-coded and the fields in the register determine the reset value. Table 3-10 shows the bit assignment of the WdogPeriphID3 register.

Table 3-10 WdogPeriphID3 register bit assignment

Bit	Name	Description
[31:8]	-	Reserved, read undefined, must be written as zeros
[7:0]	Configuration	These bits read back as `0x00`

3.2.9 PrimeCell Identification Registers, WdogPCellID0-3

The WdogPCellID0-3 registers are four 8-bit registers, that span address locations `0xFF0-0xFFC`. The read-only registers can conceptually be treated as a 32-bit register. The register is used as a standard cross-peripheral identification system. The WdogPCellID register is set to `0xB105F00D`. Figure 3-2 shows the bit assignment for the WdogPCellID0-3 registers.

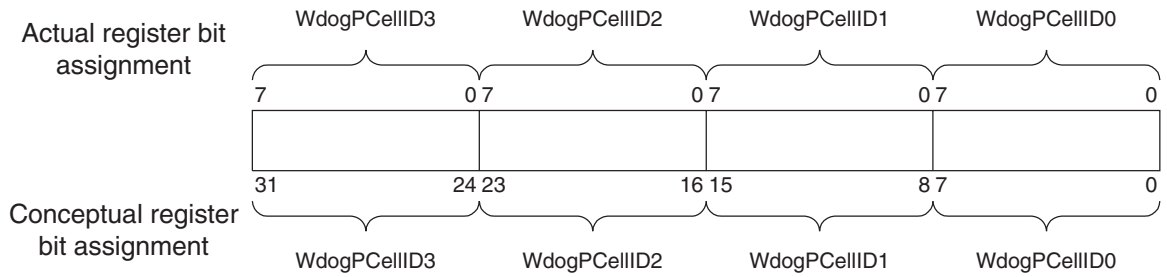

Figure 3-2 Watchdog module identification register bit assignment

The four, 8-bit PrimeCell identification registers are described in the following subsections:

- *PrimeCell Identification Register 0, WdogPCellID0* on page 3-10
- *PrimeCell Identification Register 1, WdogPCellID1* on page 3-10
- *PrimeCell Identification Register 2, WdogPCellID2* on page 3-10
- *PrimeCell Identification Register 3,WdogPCellID3* on page 3-11.

PrimeCell Identification Register 0, WdogPCellID0

The WdogPCellID0 register is hard-coded and the fields in the register determine the reset value. Table 3-11 shows the bit assignment of the WdogPCellID0 register.

Table 3-11 WdogPCellID0 register bit assignment

Bit	Name	Description
[31:8]	-	Reserved, read undefined, must be written as zeros
[7:0]	WdogPCellID0	These bits read back as `0x0D`

PrimeCell Identification Register 1, WdogPCellID1

The WdogPCellID1 register is hard-coded and the fields in the register determine the reset value. Table 3-12 shows the bit assignment of the WdogPCellID1 register.

Table 3-12 WdogPCellID1 register bit assignment

Bit	Name	Description
[31:8]	-	Reserved, read undefined, must be written as zeros
[7:0]	WdogPCellID1	These bits read back as `0xF0`

PrimeCell Identification Register 2, WdogPCellID2

The WdogPCellID2 register is hard-coded and the fields in the register determine the reset value. Table 3-13 shows the bit assignment of the WdogPCellID2 register.

Table 3-13 WdogPCellID2 register bit assignment

Bit	Name	Description
[31:8]	-	Reserved, read undefined, must be written as zeros
[7:0]	WdogPCellID2	These bits read back as `0x05`

PrimeCell Identification Register 3,WdogPCellID3

The WdogPCellID3 register is hard-coded and the fields in the register determine the reset value. Table 3-14 shows the bit assignment of the WdogPCellID3 register.

Table 3-14 WdogPCellID3 register bit assignment

Bit	Name	Description
[31:8]	-	Reserved, read undefined, must be written as zeros
[7:0]	WdogPCellID3	These bits read back as `0xB1`

Chapter 4
Programmer's Model for Test

This chapter describes the additional logic for functional verification and production testing. It contains the following section:

4.1 Integration test harness overview

The Watchdog contains an integration test harness to enable the direct control of the non-AMBA module outputs for test purposes. The test harness is controlled by integration test registers, WDOGITCR and WDOGITOP. This allows the connectivity of the **WDOGINT** and **WDOGRES** output signals to other modules in a SoC device to be easily verified using only transfers from the APB.

Figure 3-1 shows a block diagram of the output integration test harness and how the **WDOGINT** and **WDOGRES** output signals are controlled in integration test mode.

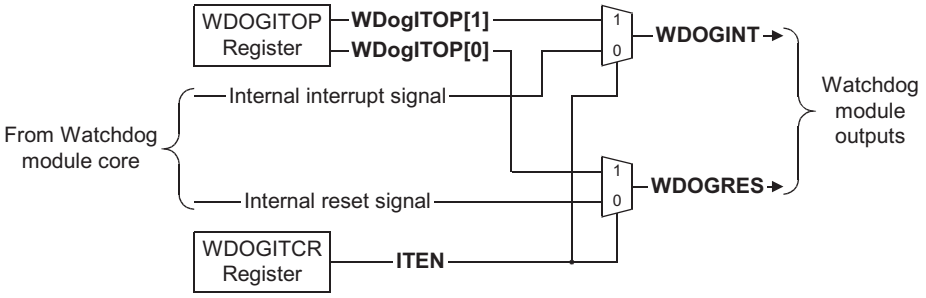

Figure 4-1 Output integration test harness

4.2 Scan testing

The Watchdog has been designed to simplify:

- insertion of scan test cells
- use of *Automatic Test Pattern Generation* (ATPG).

This is the recommended method of manufacturing test.

The Watchdog module includes placeholder signals to aid the scan insertion process:

- **SCANENABLE**
- **SCANINPCLK**
- **SCANOUTPCLK**.

4.3 Test registers

The following registers are described:

- *Integration Test Control Register, WdogITCR*
- *Integration Test Output Set Register, WdogITOP*.

4.3.1 Integration Test Control Register, WdogITCR

This is a single-bit register used to enable integration test mode. When in this mode, the masked interrupt output, **WDOGINT**, and reset output, **WDOGRES**, are directly controlled by the test output set register. Table 4-1 shows the bit assignment of the WdogITCR Register.

Table 4-1 Integration Test Control Register bit assignment

Bit	Name	Type	Function
[31:1]	-	-	Reserved.
[0]	ITEN	Read/write	Integration test enable. When this bit is 1 the Watchdog is placed in integration test mode, otherwise it is in normal mode.

4.3.2 Integration Test Output Set Register, WdogITOP

When in integration test mode, the enabled interrupt output and reset output are driven directly from the values in this register. Table 4-2 shows the bit assignment of the WdogITOP register.

Table 4-2 Integration test output set register bit assignment

Bit	Name	Type	Function
[31:2]	-	-	Reserved
[1]	WDOGINT	Write	Value output on **WDOGINT** when in integration test mode
[0]	WDOGRES	Write	Value output on **WDOGRES** when in integration test mode

Appendix A
Signal Descriptions

This chapter describes the Watchdog module signals. It contains the following sections:

A.1 AMBA APB signals

The Watchdog module is connected to the AMBA APB as a bus slave. Table A-1 describes the APB interface signals.

Table A-1 AMBA APB signal descriptions

Name	Type	Source/Destination	Description
PRESETn	Input	Reset controller	APB bus reset signal, active LOW.
PCLK	Input	Clock generator	AMBA APB clock.
PENABLE	Input	APB bridge	AMBA APB enable signal. **PENABLE** is asserted HIGH for one cycle of **PCLK** to enable a bus transfer.
PSEL	Input	APB bridge	Watchdog module select signal from the decoder within the APB bridge. When HIGH this signal indicates the slave device is selected by the APB bridge, and that a data transfer is required.
PWRITE	Input	APB bridge	AMBA APB transfer direction signal, indicates a write access when HIGH, read access when LOW.
PADDR[11:2]	Input	APB bridge	Subset of AMBA APB address bus.
PWDATA[31:0]	Input	APB bridge	Unidirectional AMBA APB write data bus.
PRDATA[31:0]	Output	APB bridge	Unidirectional AMBA APB read data bus.

A.2 Non-AMBA signals

Table A-2 describes the Watchdog module non-AMBA signals.

Table A-2 Non-AMBA signals

Name	Type	Source/Destination	Description
WDOGCLK	Input	Clock generator	Watchdog module clock
WDOGCLKEN	Input	Clock generator	Watchdog module clock enable
WDOGRESn	Input	Reset generator	Watchdog module reset signal, active LOW
WDOGINT	Output	Interrupt controller	Watchdog module interrupt, active HIGH
WDOGRES	Output	Reset controller	Watchdog module timeout reset, active HIGH
SCANENABLE	Input	Test controller	Placeholder for Watchdog module scan enable signal
SCANINPCLK	Input	Test controller	Placeholder for Watchdog module input scan signal
SCANOUTPCLK	Output	Test controller	Placeholder Watchdog module output scan signal

Index

O

P

R

S

T

V

W

