FRONTGRADE

FUNCTIONAL MANUAL

UT32M0R500

32-Bit Arm® Cortex®-M0+ Microcontroller

3/4/2021

Version #: 1.2.0

Table of Contents

1 Introduction

1.1 Scope

This document provides information for application and system-level software development for the UT32M0R500 32-bit ARM Cortex M0+ microcontroller.

Chapter 2-8 give an introduction and overview of the Cortex M0+ processor, which the UT32M0R500 microcontroller is based on. Chapter 9 and 10 deal with SRAM and FLASH memory respectively. Chapter 11 provides information about the ARM Keil MDK. Chapter 12 deals with the bootloader configuration. Chapter 13 deals with GPIO's. Chapter 14-16 deal with analog signals. Chapter 17-20 deal with analog signals. Chapter 21-24 deal with serial communication, and Appendix A: Errata has the known bugs.

1.2 Related Resources

Arm Cortex M0+ Technical Reference Manual, available at www.arm.com

ARMv6-M Architecture Reference Manual, available at www.arm.com

ARM Cortex M0+ Generic User Guide, available at www.arm.com

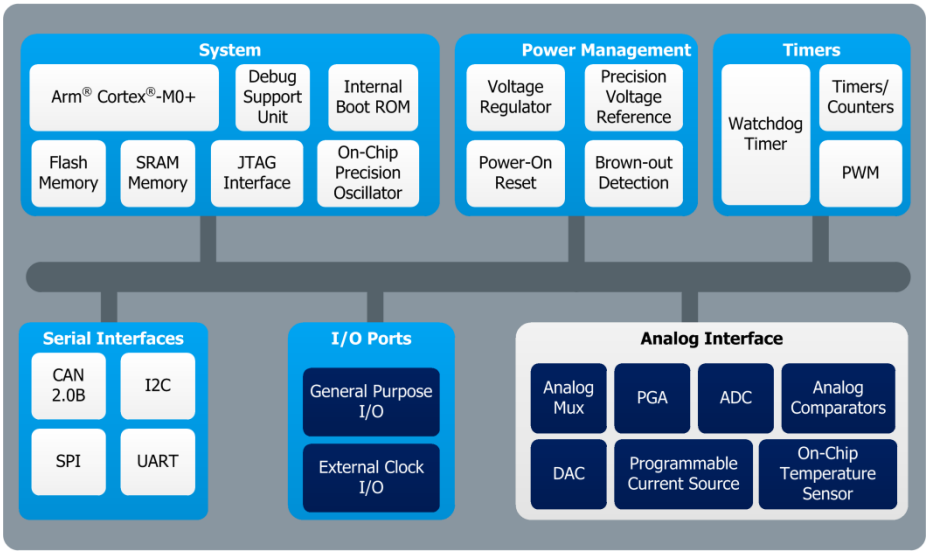
UT32M0R500 Datasheet, available at www.frontgrade.com/HiRel

1.3 Architecture

The UT32M0R500 contains the following components:

- ARM Cortex M0+
- Two CAN 2.0B Interfaces
- JTAG Debug Interface
- Forty-Eight General Purpose I/O Ports
- One 16-channel Analog-to-Digital Converters
- Two Digital-to-Analog Converters
- Watchdog Timer
- SPI Master Interface
- Two UART Interfaces
- Nor Flash Controller
- Two High-Speed Comparators
- Two I2C Interfaces
- Three Pulse Width Modulators, supporting 3 single-ended or 2 differential PWM outputs
- Four 32-bit Timers
- Real Time Clock
- AHB-to-SRAM Module
- Precision Current Source
- On-Chip Temperature Sensor

The following is a block diagram of the UT32M0R500:



1.4 Features

Table 1.1: UT32M0R500 Features

Features	ARM M0+ Configurable Option	UT32M0R500 Configuration
Interrupts	0 – 32	32
Data Endianness	Little-endian or big-endian	Little-endian
SysTick Timer	Present or absent	Present
Number of Watchpoint Comparators	0, 1, 2	2
Number of Breakpoint Comparators	0 - 4	4
Multiplier	Fast or small	Fast (Single Cycle)
Wakeup Interrupt Controller	Supported or not supported	Supported
Vector Table offset Register	Present or absent	Present
Unprivileged/Privileged Support	Present or absent	Present
Memory Protection Unit	Present or absent	Present
Reset All Registers	Present or absent	Present
Debug Configuration	Present or absent	Present
Micro Trace Buffer	Present or absent	Present

1.5 Generic Register Format

This manual uses the following register format.

Register Name					Offset = 0x0000											
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R					PARTIAL_REG [10:7]				READ_ONLY_REG							
W																
Reset	[00…0]				0xD				[00…0]							
Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					READ_WRITE_REG				AR	PARTIAL_REG [6:0]						
W	WRITE_ONLY_REG															
Reset	[11…1]				0x3				1	0x4F						

Legend: Bit# = Register bit number; R = Read; W = Write; Reset = Value after reset

Table 1.2: Description of an Example Register

Bit Number(S)	Bit Name	Reset State	Description
31-24	RESERVED	[00…0]	RESERVED bits. Must be kept at reset value
27-24	PARTIAL_REG [10:7]	0xD	Partial Register. Relevant bits are specified in the register name.
23-16	READ_ONLY _REG	[00…0]	Read-Only Register. Software can only read these bits. [00…0] = 0x00 = 0b0000_0000
15-12	WRITE_ONLY _REG	[11…1]	Write-Only Register. Software can only write to these bits. [11…1] = 0xF = 0b1111
11-8	READ_WRITE _REG	0x3	Read and Write Register. Software can read and write to these bits.
7	ABBREVIATED _REG (AR)	1	Abbreviated Register (Abbreviated name in parenthesis)
6-0	PARTIAL_REG [6:0]	0x4F	Partial Register. Relevant bits are specified in the register name. 0x0F = 0b100_1111 (hex values should be compared to the number of bits available)

2 ARM Cortex M0+ Processor

2.1 Overview

The UT32M0R500 microcontroller is based on the Arm Cortex M0+ processor. The Arm Cortex M0+ is developed based on the ARM v6-M architecture, which describes the details related to programming, including memory map, registers, instruction set architecture, and exception model. In addition to the architecture, the processor includes timing information and hardware structure. The UT32M0R500 microcontroller consists of the ARM processor core and peripheral components.

UT32M0R500 microcontroller contains the following components:

- Single core CPU optimized for power.
- On chip RAM and ROM memory components.
- Peripherals.
- Internal bus system to communicate with all microcontroller components.

Arm Cortex M0+ processor is a von Neumann architecture containing a single bus for accessing code and data. It implements a version of the Thumb instruction set based on Thumb-2 technology, ensuring high code density and reduced program memory requirements--Thumb stands for variable length execution sets with a length of 16 or 32 bit. It implements a 2-stage pipeline. It supports up to 32 external Interrupts, each with up to 4 levels of priority that can be changed dynamically. The processor can be put into a very low-power sleep mode, leaving the Wakeup Interrupt Controller (WIC) to identify and prioritize Interrupts. It has a hardware single-cycle (32x32) multiplier and Memory Protection Unit (MPU). Figure 2.1 shows a simplify block diagram of the Arm Cortex M0+.

For ARM Cortex M0+ Instruction Set, see Chapter 4 for more information.

The Arm Cortex-M0+ Memory Map describes the address space organization, see Chapter 5 for more information.

For more detailed information on Arm Cortex M0+ architecture, refer to *ARMv6-M Architecture Reference Manual*.

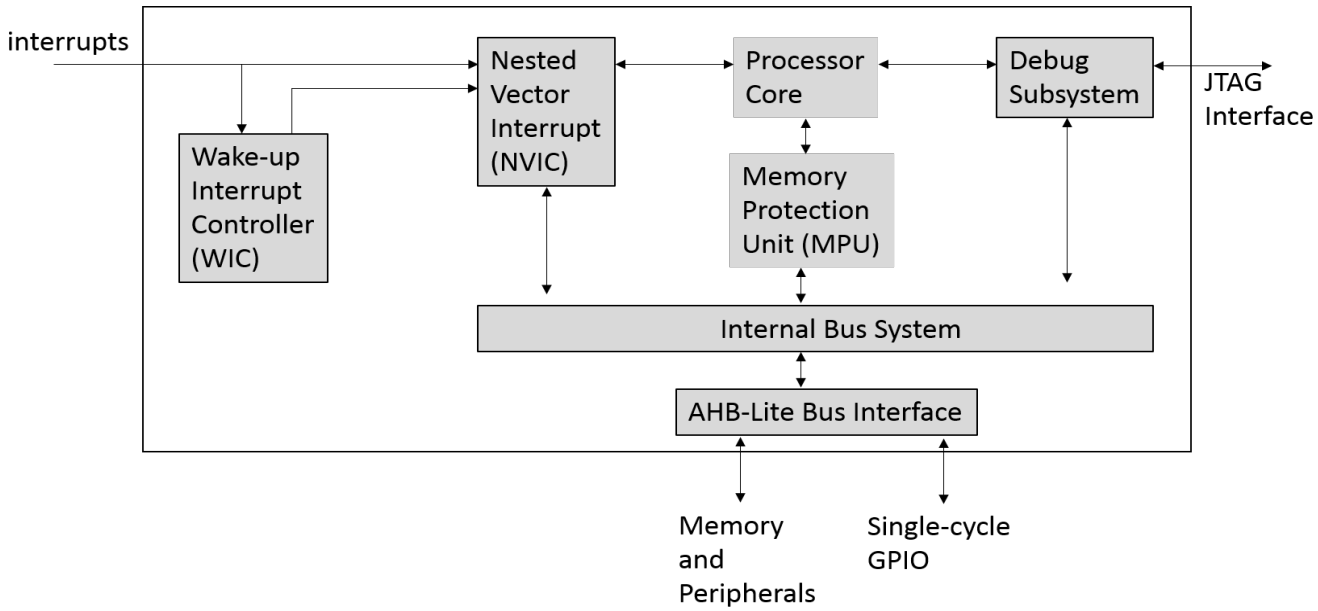


Figure 2.1: Cortex M0+ Block Diagram

2.2 Processor Core

The processor core contains internal registers: includes sixteen 32-bit registers for both general and special usage, the ALU, data path and control logic.

2.3 2-stage Pipeline

The processor is a load/store architecture. It loads data from memory to a register, processes it inside the core, and if needed, writes back to memory. All load/store always complete in program order and up to two 16-bit instructions can be fetched in one transfer

2.4 NVIC

The Nested Vector Interrupt Controller (NVIC) automatically handles nested Interrupts: compares priorities between Interrupts and the current priority level.

For more information on the NVIC, see Chapter 7.

2.5 WIC

The Wake-up Interrupt controller (WIC) masks Interrupt signals while the processor is in deep sleep.

For more information on the WIC, see section 6.12.

2.6 Bus Interface

The Cortex M0+ processor uses AMBA 3 AHB-Lite bus to communicate with other parts of the system, such as memory blocks, peripherals and application-specific logic. AHB-Lite is a subset of the AHB specifications, designed for single master systems with simple peripherals.

2.7 Debug System

The debug system provides a standard debug interface. It handles debug control, program breakpoints and data watch points. When a debug event occurs, the processor is halted and the debug subsystem allows the execution of one instruction at a time in order to analyze register values and flags. Supports unlimited software breakpoints using BKPT instruction. Non-intrusive access to core peripherals including memory.

2.8 Cortex M0+ Registers

The processor has sixteen 32-bit generic registers: thirteen general-purpose registers (R0-R12) and several special-purpose registers a Stack Pointer (SP) R13, a Link Register (LR) R14, a Program Counter (PC) R15 and Special-purpose Program Status Registers, (xPSR)).

For more information on the Cortex M0+ registers, see Chapter 3.

For more detailed information on Arm Cortex M0+ registers, refer to ARMv6-M Architecture Reference Manual.

2.9 Memory Protection Unit (MPU)

The MPU can be used to make an embedded system more robust and more secure by:

- Prohibiting the user applications from corrupting data used by critical tasks (such as the operating system kernel).
- Defining the SRAM memory region as a non-executable to prevent code injection attacks.
- Changing the memory access attributes.
- Independent attribute settings for each region
- Overlapping regions
- The export of memory attributes to the system.

The MPU can be used to protect up to eight memory regions. These, in turn can have eight sub-regions, if the region is at least 256 bytes. The sub-regions are always of equal size and can be enabled or disabled by a sub-region number. Because the minimum region size is driven by the cache line length (32 bytes), 8 sub-regions of 32 bytes correspond to a 256 bytes size.

The regions are numbered 0-7. In addition, there is another region called the default region with an id of -1. All the 0-7 memory regions take priority over the default region.

The regions can overlap and can be nested. The region 7 has the highest priority and the region 0 has the lowest one and this governs how overlapping the regions behave. The priorities are fixed and cannot be changed. Every region can have an access permission attribute (no accesses allowed, read-only, or read/write). Those can be further refined by fetch or no-fetch guards.

Further information on the details for accessing and using the MPU can be found at Arm Developer Arm Developer Infocenter - Cortex-M0+ Devices Generic User Guide and/or Arm Developer Infocenter - ARMv6-M Architecture Reference Manual.

2.10 Clocking System

The UT32M0R500 implements a flexible clocking system that allows for using an internal or external clock source. This selection is based on the CLKSEL pin. The internal clock source is an oscillator running at a fixed frequency of 50 MHz. For the external clock source, the UT32M0R500 supports either a single-ended square wave or a crystal oscillator-based input. From the clock source selection, the system clock, called PCLK, is generated. The system clock has user programmable frequency through the CLK_DIVIDE register in the system control module (SYSCON) and drives the Arm core and all the peripherals. A notable exception is that the ADC module operates directly from the internal oscillator. Within each peripheral, there are various programmable options for changing the clock frequency for that particular peripheral.

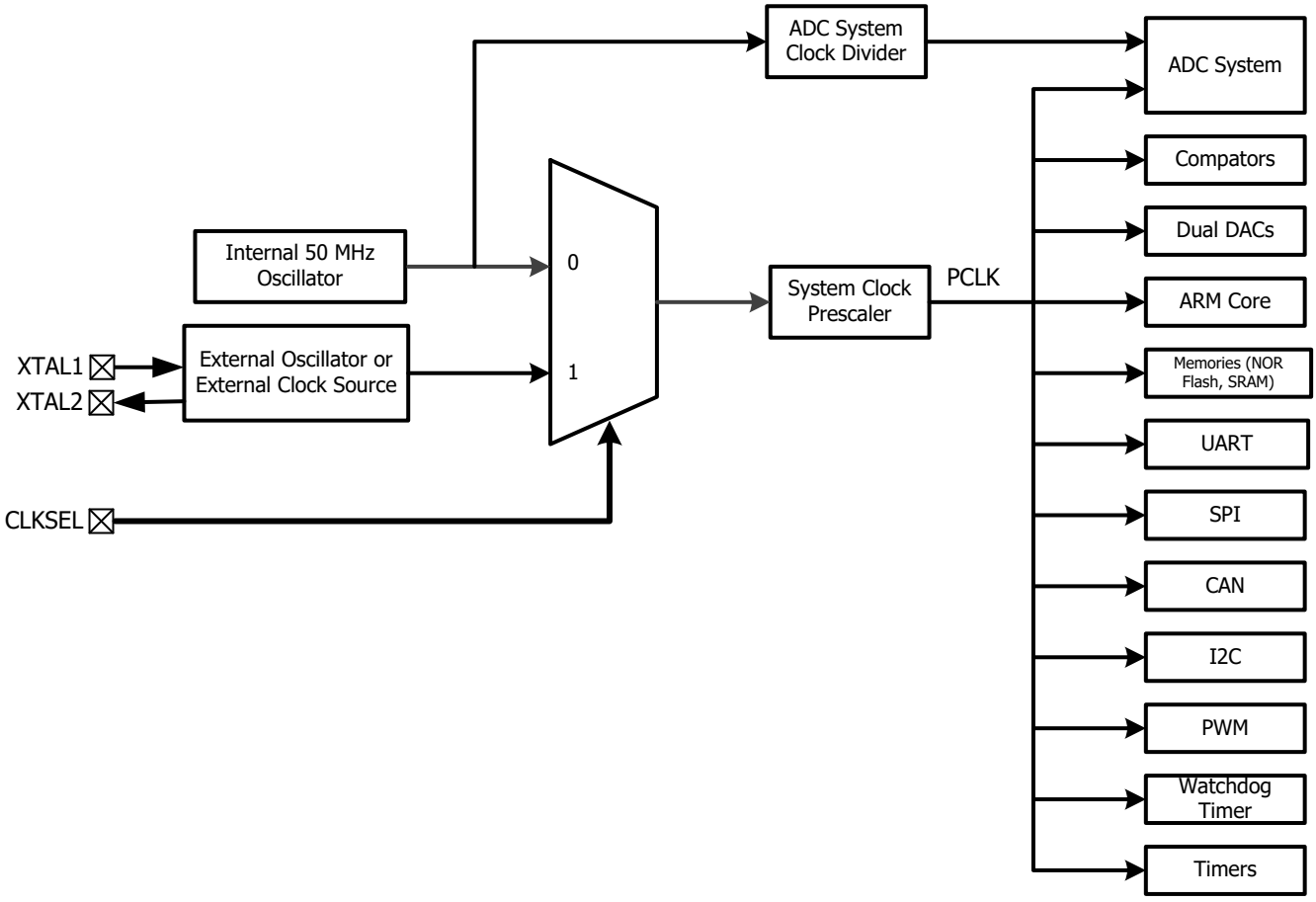


Figure 2.1: Block diagram of UT32M0R500 clocking architecture

3 ARM Cortex M0+ Registers

3.1 Overview

The processor has sixteen 32-bit registers that includes thirteen general-purpose registers (R0-R12) and several special-purpose registers: R13 is the Stack Pointer (SP), R14 is the Link Register (LR), R15 is the Program Counter (PC). The Special-purpose Program Status Registers is xPSR. The exception mask register (PRIMASK) disables the handling of exceptions by the processor; it can be used for timing critical tasks. The CONTROL register controls the stack used and the privilege level for software execution when the processor is in Thread mode.

Figure 3.1 shows the Cortex M0+ registers.

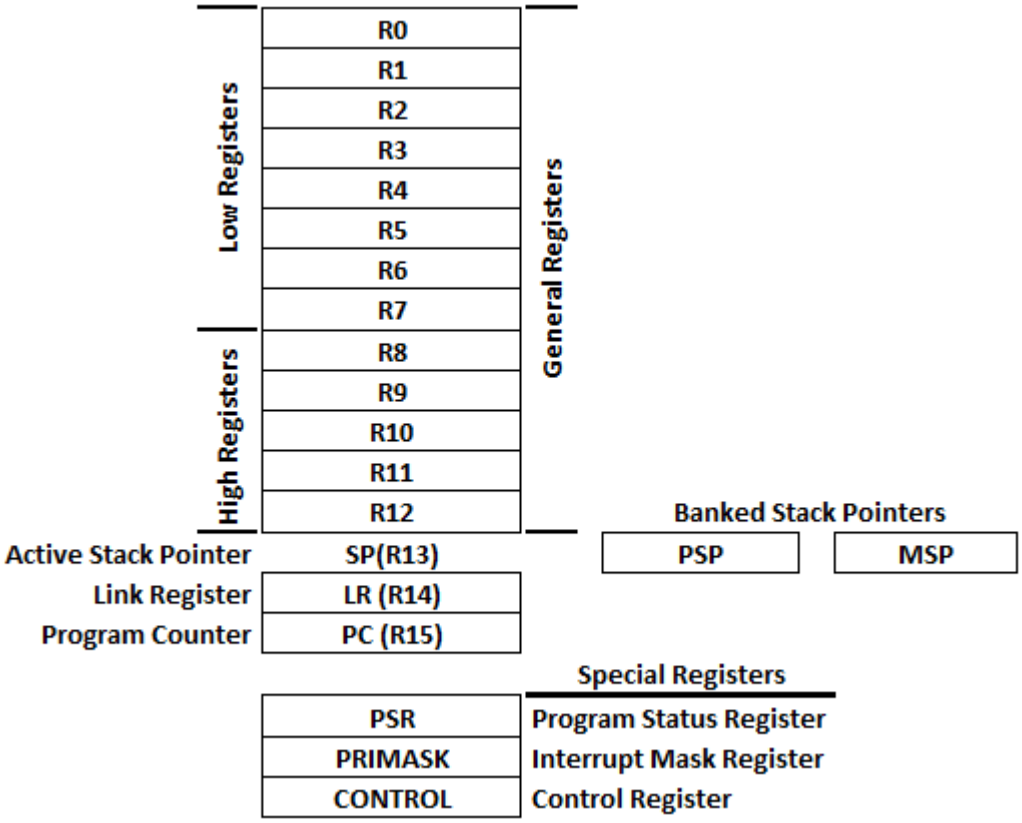


Figure 3.1: Processor Core Register

3.2 Cortex M0+ General-Purpose Registers

The general-purpose registers R0-R12 have no special architecturally-defined uses.

Most instructions that can specify a general-purpose register can use R0-R12.

Low registers: R0-R7 are accessible by all instructions that specify a general-purpose register.

High registers: R8-R12 are accessible by all 32-bit instructions that specify a general-purpose register and are not accessible by most 16-bit instructions.

3.3 Stack Pointer (SP)

R13 is used as the Stack Pointer (SP) and it's auto aligned to a word, four-byte boundary.

Handler mode always uses the main stack pointer (MSP), but for Thread mode, it can use either process stack pointer or main stack pointer. The SP keeps track of the memory's address that holds the last value stored on the stack. It is used for saving the context of a program while switching between tasks. There are two SPs in the Cortex-M0 processor: main (MSP) and process (PSP). For simple applications, the MSP is the only stack pointer used. For applications that require privileged access, the MSP is used for OS kernel and exception handlers, while the PSP is used for other tasks.

3.4 Link Register (LR)

R14 is the Link Register (LR). LR is used to store the return address of a function call; the PC will load the value from the LR after a function is finished.

3.5 Program Counter (PC)

R15 is the Program Counter (PC). PC contains the address of the next assembly instruction to be executed. For 32-bit instruction code, the PC is automatically incremented by 4, except for branching instructions.

3.6 Program Status Registers (PSR)

The Program Status Register (PSR) combines mutually exclusive the Application Program Status Register (APSR), Interrupt Program Status Register (IPSR) and the Execution Program Status Register (EPSR).

- APSR contains the current state of the condition flags (N, Z, C, V) from previous instruction executions.
- IPSR contains the exception type number of the current Interrupt Service Routine (ISR).
- The EPSR contains the Thumb state bit.

3.7 Interrupt Mask Register (PRIMASK)

When bit [0] is set to 1, the PRIMASK register blocks the handling of exceptions by the processor, can be used to prevent data race conditions.

3.8 Control Register (CONTROL)

The CONTROL register controls the stack used and the privilege level for software execution when the processor is in Thread mode. After reset, the MSP is used but when using an OS, PSP can be used for Threads and MSP for OS and ISR's implementation.

4 ARM Cortex M0+ Instruction Set

4.1 Overview

The Cortex-M0+ Instruction Set Architecture is developed based on the Arm v6-M architecture, which is a combination of Armv6 architecture's Thumb instruction set and Armv7-M architecture memory map, exception model, and Thumb-2 system.

Armv6-M supports the Thumb instruction set including a small number of 32-bit instructions introduced with Thumb-2 technology. The 16-bit instruction support is equivalent to the Thumb instruction set support in Armv6 prior to the introduction of Thumb-2 technology. Thumb instructions are either 16-bit or 32-bit and are aligned on a two-byte boundary. 16-bit and 32-bit instructions can be intermixed freely.

For more detailed information on Arm Cortex M0+ Instruction Set, refer to *Cortex-M0+ Devices Generic User Guide*

Below is the Cortex-M0+ Instruction Set assembly syntax.

Label

Mnemonic Operand1, Operand2,… ; Comments

- **Labels** are symbolic representations of addresses; labels are used to mark specific addresses to refer to from other parts of the code
- **Mnemonic** is the name of the instruction. Most data processing instructions can optionally update the condition flags according to the result of the operation
- **operand1** is the destination register
- **Operand2** is the source register
- **";"** are comments and don't affect the program

Note: mnemonics, pseudo-instructions, directives and symbolic register names can be written in all uppercase or all lowercase, but not mixed. Labels and comments can be in uppercase, lowercase, or mixed.

Table 4.1: Cortex M0+ Instruction Set

16-Bit Thumb Instructions Supported on Cortex-M0+									
ADCS	ADDS	ADR	ANDS	ASRS	B	BIC	BLX	BKPT	BX
CMN	CMP	CPS	EORS	LDM	LDR	LDRH	LDRSH	LDRB	LDRSB
LSLS	LSRS	MOV	MVN	MULS	NOP	ORRS	POP	PUSH	REV
REV16	REVSH	ROR	RSB	SBCS	SEV	STM	STR	STRH	STRB
SUBS	SVC	SXTB	SXTH	TST	UXTB	UXTH	WFE	WFI	YIELD

32-Bit Thumb-2 Instructions Supported on Cortex-M0+									
BL	DSB	DMB	ISB	MRS	MSR				

- The special registers **Control, MASK** and **xPSR** are accessed using the **MRS** and **MSR** instructions.
- **MRS** moves the data from a special register into a general purpose register, and
- **MSR** moves the data from a general purpose register into a special register.
- **ISB, DSB,** and **DMB** are memory synchronization instructions.
- **BL** was supported in the traditional Thumb instruction set, but the bit field definition was extended in Thumb-2.

For **WFE, WFI** and **SEV** sleep mode instructions, see Power Management for information.

Table 4.2: Cortex M0+ Instruction Set Categories

Category	Type	Mnemonic
Memory Access	Load	LDR, LDRB, LDRH, LDRSH, LDRSB, LDM
	Store	STR, STRB, STRH, STM
	Stack	PUSH, POP
Data movement	Move	MOV, MOVS, MRS, MSR
Control flow	Branch	B, BL, BX, BLX, BCC
Data processing	Math	ADD, ADDS, ADCS, ADR,MULS, RSBS,SBCS, SUB(S)
	Logic	ANDS, EORS, ORRS, BICS, MVNS, TST
	Compare	CMP, CMN, TST
	Shift and Rotate	ASRS, LSLS, LSRS, RORS
	Extend	SXTB, SXTH, UXTB, UXTH
	Reverse	REV, REV16, REVSH
Miscellaneous		BKPT, CPSID, CPSIE, WFE, WFI, SVC, DMB, DSB, ISB, SEV, NOP

4.2 Memory Access: LOAD

Syntax	Example	Description
LDR <Rt>,=Direct Number	LDR R0,=0x01234567	R0=0x01234567
LDR <Rt>,[<Rn>,Constant]	LDR R0,[R1,4]	R0=memory[R1+4], load word from mem
LDR <Rt>,[<Rn>,<Rm>]	LDR R0,[R1,R2]	R0=memory[R1+R2], load word from mem
LDRH <Rt>,[<Rn>,<Rm>]	LDRH R0,[R1,R2]	R0=memory[R1+R2], load signed half word from mem
LDRB <Rt>,[<Rn>,<Rm>]	LDRB R0,[R1,R2]	R0=memory[R1+R2], load signed byte word from mem
LDRSH <Rt>,[<Rn>,<Rm>]	LDRSH R0,[R1,R2]	R0=memory[R1+R2], load word from mem
LDRSB <Rt>,[<Rn>,<Rm>]	LDRSB R0,[R1,R2]	R0=memory[R1+R2], load word from mem

4.3 Memory Access: STORE

Syntax	Example	Description
STR <Rt>,[<Rn>,<Rm>]	STR R0,[R1,R2]	memory[R1+R2]=R0, write word to mem
STRH <Rt>,[<Rn>,<Rm>]	STRH R0,[R1,R2]	memory[R1+R2]=R0, write half word to mem
STRB <Rt>,[<Rn>,<Rm>]	STRB R0,[R1,R2]	memory[R1+R2]=R0, write a byte to mem

4.4 Multiple Data Access

Syntax	Example	Description
LDM <Rn>,{<Ra>,<Rb>…}	LDM R0,{R1,R2-R7}	R1=memory[R0] R2=memory[R0+4] …
LDMIA <Rn>,{<Ra>,<Rb>…}	LDMIA R0,{R1,R2-R7}	R1=memory[R0], load multiple and inc last address + 4 R2=memory[R0+4] …
STMIA <Rn>,{<Ra>,<Rb>…}	STMIA R0,{R1,R2-R7}	R1=memory[R0], store multiple and inc last address + 4 R2=memory[R0+4] …

4.5 Stack Access: PUSH and POP

Syntax	Example	Description
PUSH {<Ra>,<Rb>…}	PUSH {R0,R1, R2}	Memory[SP-4]=R0, Memory[SP-8]=R1, Memory[SP-12]=R2, SP=SP-12
POP {<Ra>,<Rb>…}	POP {R0,R1, R2}	R2= Memory[SP], R1= Memory[SP+4], R0= Memory[SP+8], SP=SP+12

4.6 Arithmetic ADD

Syntax	Example	Description
ADDS <Rd>,<Rn>,<Rm>	ADDS R0,R1,R2	R0= R1+R2, update APSR
ADDS <Rd>,<Rn>,#imm	ADDS R0,R1,#0x2	R0= R1+2, update APSR
ADDS <Rd>,#imm	ADDS R0,#0x2	R0= R0+2, update APSR
ADD <Rd>,<Rn>	ADDS R0,R1	R0= R1+R2
ADDC <Rd>,<Rm>	ADDS R0,R1	R0= R1+R2 + carry, update APSR
ADDS <Rd>, PC, #imm	ADDS R0,PC,#0x04	R0= PC+4

4.7 Arithmetic SUB, MUL

Syntax	Example	Description
SUBS <Rd>,<Rn>,<Rm>	SUBS R0,R1,R2	R0= R1-R2, update APSR
SUBS <Rd>,<Rn>,#imm	SUBS R0,R1,#0x2	R0= R1-2, update APSR
SUBS <Rd>,#imm	SUBS R0,#0x2	R0= R0-2, update APSR
SBCS <Rd>,<Rd>,<Rm>	SBCS R0,R0,R1	R0= R0-R1 - Borrow, update APSR
RSBS <Rd>,<Rd>,#0	RSBS R0,R0,#0	R0= -R0
MULS <Rd>,<Rm>,<Rd>	MULS R0,R1,R0	R0=R0*R1

4.8 Arithmetic CMP

Syntax	Example	Description
CMP <Rn>,<Rm >	CMP R0,R1	R0-R1, update APSR
CMP <Rn>,#imm	CMP R0,#0x1	R0-1, update APSR
CMN <Rn>,<Rm>	CMN R0,R1	R0+R1, update APSR

4.9 Logic Operations

Syntax	Example	Description
ANDS <Rd>,<Rm>	ANDS R0,R1	update APSR
ORRS <Rd>,<Rm>	ORRS R0,R1	update APSR
EORS <Rd>,<Rm>	EORS R0,R1	XOR(R0,R1), update APSR
MVNS <Rd>,<Rm>	MVNS R0,R1	R0=NOT(R1), update APSR
BICS <Rd>,<Rm>	BICS R0,R1	AND(R0, NOT(R1)), update APSR
TST <Rd>,<Rm>	TST R0,R1	AND(R0, R1), update APSR

4.10 Arithmetic Shift Operation

Syntax	Example	Description
ASRS <Rd>,<Rm>	ASRS R0,R1	R0=R0>>R1, update APSR
ASRS <Rd>,<Rm>#imm	ASRS R0,R1,#0x1	R0=R1>>1, update APSR

4.11 Logic Shift Operation

Syntax	Example	Description
LSRS <Rd>,<Rm>	LSRS R0,R1	R0=R0>>R1, update APSR
LSRS <Rd>,<Rm>#imm	LSRS R0,R1,#0x1	R0=R1>>1, update APSR
LSLS <Rd>,<Rm>	LSLS R0,R1	R0=R0<<R1, update APSR
LSLS <Rd>,<Rm>#imm	ASRS R0,R1,#0x1	R0=R1<<1, update APSR

4.12 Rotate Operation

Syntax	Example	Description
RORS <Rd>,<Rm>	RORS R0,R1	R0=R0>>R1(rotate), update APSR

4.13 Reverse Ordering Operation

Syntax	Example	Description
REV <Rd>,<Rm>	REV R0,R1	R0={R1 [7:0], R1 [15:8] , R1 [23:16] , R1 [31:24]}
REV16<Rd>,<Rm>	REV16 R0,R1	R0={R1 [23:16], R1 [31:24] , R1 [7:0] , R1 [15:8]}
REVSH <Rd>,<Rm>	REVSH R0,R1	R0=signed extend{R1 [7:0], R1 [15:8]}

4.14 Extended Operation

Syntax	Example	Description
SXTB <Rd>,<Rm>	SXTB R0,R1	R0=signed extend(R1 [7:0])
SXTH <Rd>,<Rm>	SXTH R0,R1	R0=signed extend(R1 [15:0])
UXTB <Rd>,<Rm>	UXTB R0,R1	R0=zero extend(R1 [7:0])
UXTH <Rd>,<Rm>	UXTH R0,R1	R0=zero extend(R1 [15:0])

4.15 Program Flow Control

Syntax	Example	Description
B <label>	B loop	Change PC to loop
B <cond><label>	BEQ loop	If Z=1 in APSR reg, change PC to loop
BL <label>	BL function	Change PC to function, LR = PC+4
BX <Rm>	BX R0	PC = R0
BLX	BLX R0	PC = R0, LR = PC+4

5 Arm Cortex M0+ Memory Map

5.1 Overview

The Arm Cortex-M0+ Memory Map describes the organization of the processor's address space. The address space contains internal and external RAM and ROM as well as buses and peripherals. The total memory space of a 32-bit system is 4GB. It is split and separated into regions with different functionality which can be individually changed, except for internal private peripheral bus of the processor.

The processor stores 8-bit bytes, 16-bit half words or 32-bit words, so a particular memory address specifies a particular byte in memory, and in order to get a byte from memory, the processor accesses the next three-byte addresses to get a full 32-bit word.

Figure 5.1 shows the predefined regions within the memory map of the UT32M0R500 microcontroller.

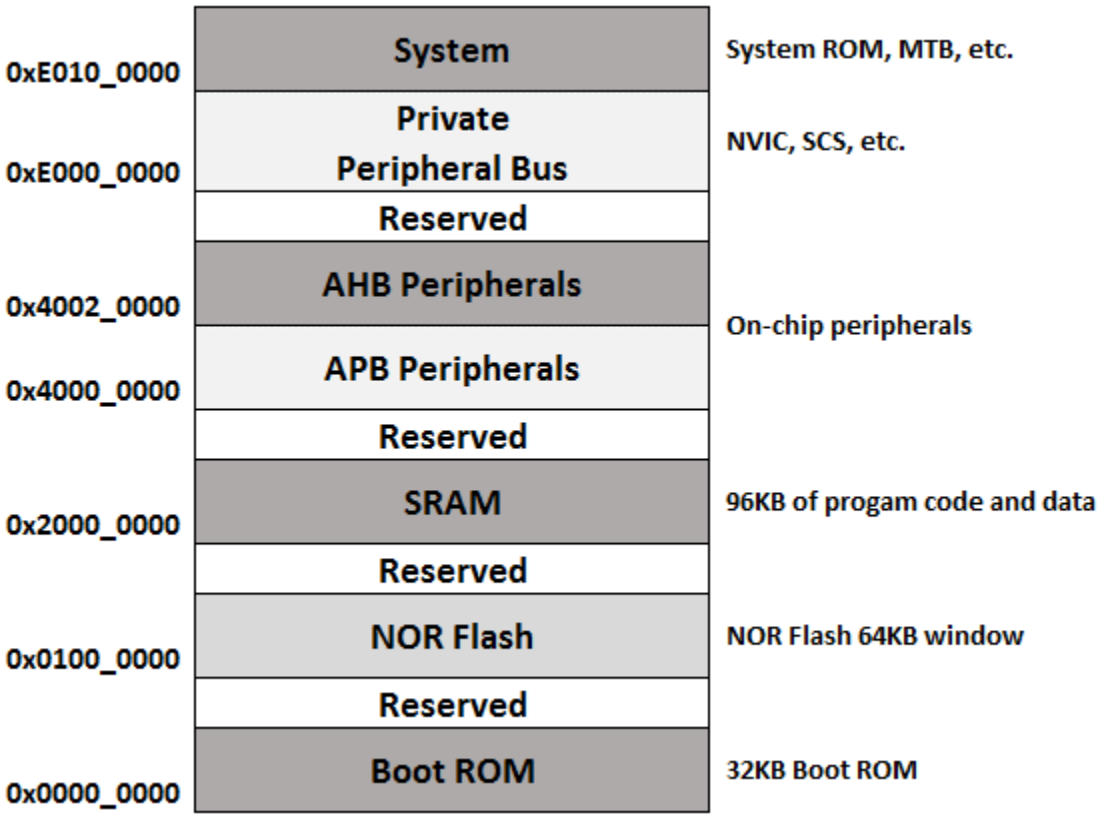


Figure 5.1: UT32M0R500 Memory Map

Table 5.1 and Table 5.2 specify the predefined regions for the different peripherals with boundary addresses.

Table 5.1. AHB Memory Map Table

Address Range	Size (bytes)	UT32M0R500 Peripheral	Description
0x0000 0000 - 0x0000 7FFF	32K	Boot ROM	See 5.2 Boot ROM for info
0x0100 0000 - 0x0100 FFFF	64K	NOR Flash	See section 5.2 and chapter 18 for info
0x2000 0000 - 0x2001 7FFF	96K	SRAM	See section 5.3 Embedded SRAM for more info
0x4000 0000 - 0x4001 FFFF	128K	APB Peripherals	See Table 5.2
0x4002 0000 - 0x4002 0FFF	4K	AHB GPIO Bank 0	See Table 12.2 and Table 12.3 for register map
0x4002 1000 - 0x4002 1FFF	4K	AHB GPIO Bank 1	See Table 12.2 and Table 12.3 for register map
0x4002 2000 - 0x4002 2FFF	4K	AHB GPIO Bank 2	See Table 12.2 and Table 12.3 for register map
0x4002 3000 - 0x4002 3FFF	4K	AHB CAN 0	See Table 24.1 for register map
0x4002 4000 - 0x4002 4FFF	4K	AHB CAN 1	See Table 24.1 for register map
0x4002 F000 - 0x4002 FFFF	4K	System Controller	See Table 8.1 for register map
0xE000 0000 - 0xE00F FFFF	4K	Private peripheral bus addresses in the Cortex-M0+	See CMSIS section 6.11 for info
0xF000 0000 - 0xF000 0FFF	4K	System ROM	See CoresSight-MTB-M+ for more info
0xF020 0000 - 0xF020 0FFF	4K	MTB SFR	See CoresSight-MTB-M+ for more info
0xF021 0000 - 0xF021 FFFF	64K	4kB MTB SRAM	See CoresSight-MTB-M+ for more info
0xF022 0000 - 0xFFFF FFFF		Cortex M0+	

Table 5.2. APB Peripheral Memory Map Table

Address Range	Size (bytes)	UT32M0R500 Peripheral	Description
0x4000 0000 - 0x4000 0FFF	4K	RTC	See Table 20.1 for register map
0x4000 1000 - 0x4000 1FFF	4K	Dual Timer0	See Table 17.1 for register map
0x4000 2000 - 0x4000 2FFF	4K	Dual Timer1	See Table 17.1 for register map
0x4000 3000 - 0x4000 3FFF	4K	PWM	See Table 18.5 for register map
0x4000 4000 - 0x4000 4FFF	4K	UART0	See Table 21.2 for register map
0x4000 5000 - 0x4000 5FFF	4K	UART1	See Table 21.2 for register map
0x4000 6000 - 0x4000 6FFF	4K	SPI	See Table 22.1 for register map
0x4000 7000 - 0x4000 7FFF	4K	DACs	See Table 15.1 for register map
0x4000 8000 - 0x4000 8FFF	4K	Watchdog	See Table 19.1 for register map
0x4000 9000 - 0x4000 9FFF	4K	I2C0	See Table 23.3 for register map
0x4000 A000 - 0x4000 AFFF	4K	I2C1	See Table 23.3 for register map
0x4000 B000 - 0x4000 BFFF	4K	Trim Control	
0x4000 C000 - 0x4000 CFFF	4K	NFC	See Table 10.1 for register map
0x4000 D000 - 0x4000 DFFF	4K	SRAM EDAC/Scrub	See Table 9.1 for register map
0x4000 E000 - 0x4000 EFFF	4K	Comparators	See Table 16.1 for register map
0x4000 F000 - 0x4000 FFFF	4K	ADC	See Table 14.5 for register map

5.2 Boot ROM

The UT32M0R500 bootloader is stored on-chip in ROM code; it is programmed into the chip at manufacture time, so it cannot be replaced. The bootloader copies an executable image from NOR Flash into SRAM and jumps to the image. The executable image located in NOR Flash can be updated via UART0 or CAN0 interfaces. The NOR Flash is an 8MB of on-chip flash memory. It specifies different memory regions for four program images. Each program image has two sectors of 64KB each. The maximum program image size is 90KB or 0x167FE (0x16800 - 2 bytes for CRC). At offset 0x167FE from the image base address, the bootloader appends a 16-bit CRC to the particular image. If the image override word and crc are valid, the bootloader will attempt to load the specified image number first. If no image override is specified or the specified image is not valid (stored CRC does not match calculated CRC), then the bootloader will begin searching for a valid image starting with image 0. At the start of sector 16, address 0x0009_0000, 4 bytes are reserved for image override. Out of the 4 bytes, two bytes or a 16-bit half word is for the image number and the other two bytes or 16-bit half word is for computing a CRC on the 16-bit image number; this ensures error correction on the image number.

The NOR Flash controller (NFC) provides the interface between the on-chip NOR Flash and the AHB bus. For more information on the NFC, see NOR Flash Controller (NFC).

Figure 5.2 specifies the four program images memory boundaries.

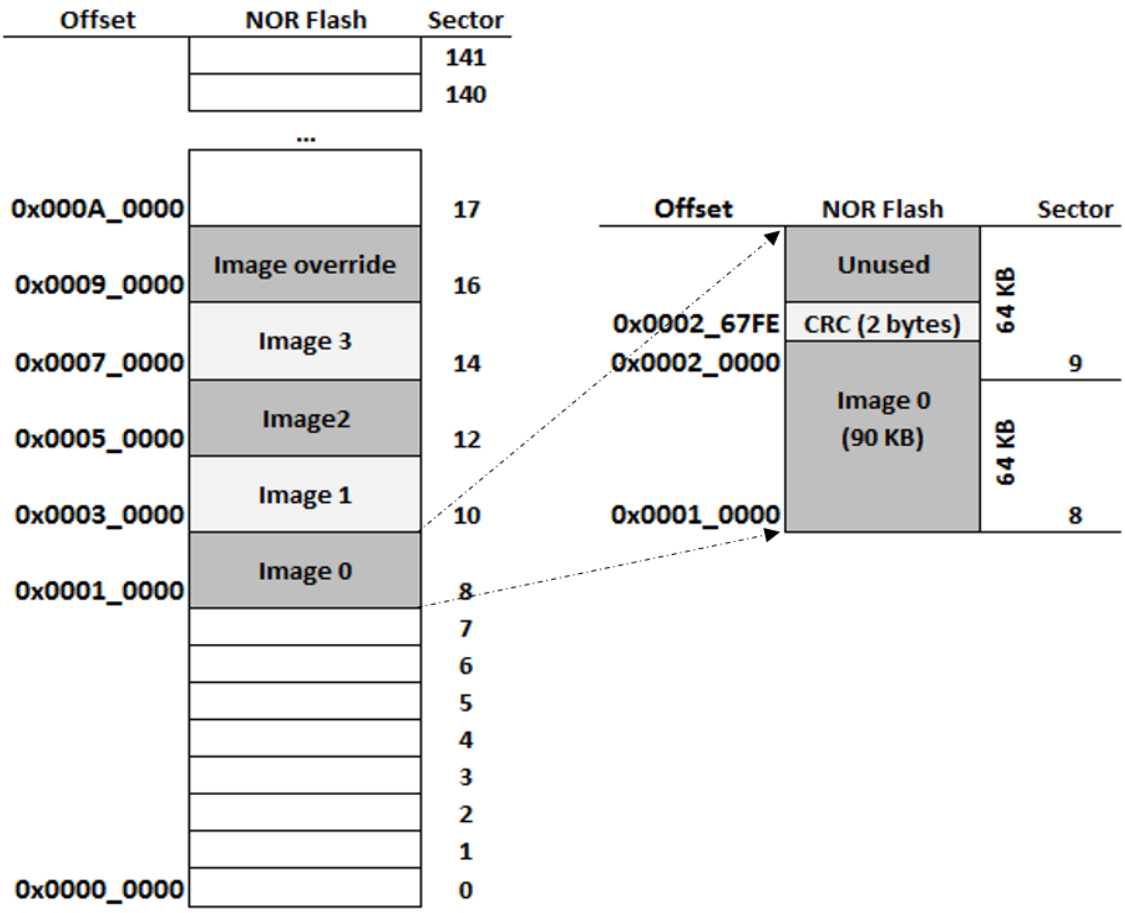


Figure 5.2: NOR Flash Memory Map for firmware image locations

5.3 Embedded SRAM

The UT32M0R500 device includes 96KB of SRAM, located at start address 0x2000_0000, for both program code as well as data. Data stored in RAM can be divided into static data, stack and heap. Static data contains global or static variables, and is a fixed amount of data which can be calculated upfront by the compiler and reserved in the RAM (SRAM) memory. This data is stored at the beginning of the RAM (data) memory space, and reduces the available memory permanently. The stack contains all local or temporary data required for subprograms; it grows from the upper limit of the data region downward. The heap uses the function malloc() for dynamic memory allocation; It can last for the duration of the program; and it grows upward above the static data region toward the stack.

For more detailed information on Arm Cortex M0+ data storage and data types, refer to *Cortex-M0+ Devices Generic User Guide*

Figure 5.3 shows the different data memory storage uses.

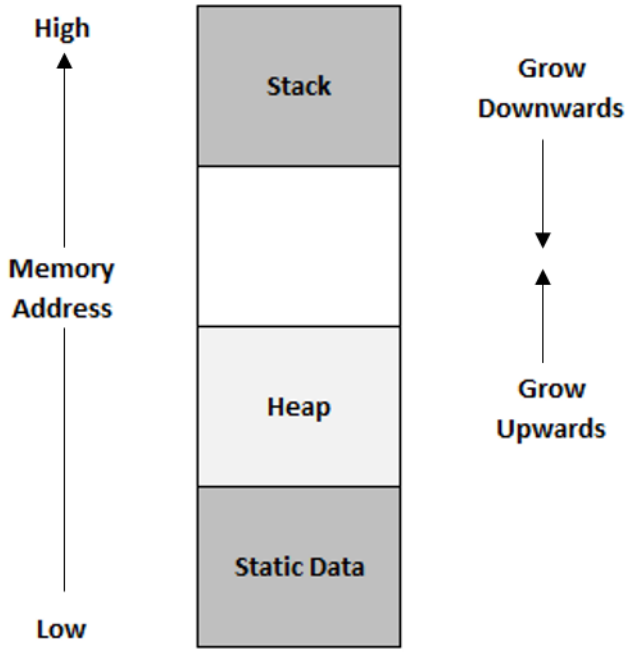


Figure 5.3: Data Memory Storage

6 Nested Vector Interrupt Controller (NVIC)

6.1 Overview

The Nested Vector Interrupt Controller (NVIC) controls external Interrupts and system exceptions. External Interrupts can have up to four levels of priority—possible priorities are 0x00, 0x40, 0x80 and 0xC0. Arm Cortex M0 architecture supports a maximum of 32 external Interrupts and one non-maskable Interrupt (NMI); Table 6.1 shows the number of external Interrupts supported by the UT32M0R500 microcontroller. NVIC provides very fast Interrupt handling and features the Wake-up Interrupt Controller for ultra-low power sleep mode support. Interrupts can be either level or edge-triggered. NVIC registers enable or disable each external Interrupt and its pending status. Cortex Microcontroller Software Interface Standard (CMSIS) provides the functions to configure the Nested Vector Interrupt Controller. The processor architecture has a hardware block called Wake-up Interrupt Controller (WIC) that masks Interrupt signals while the processor is in deep sleep mode. Figure 6.1 shows a block diagram for NVIC and WIC.

For more detailed information on NVIC operation, registers and programing, refer to ARMv6-M Architecture Reference Manual.

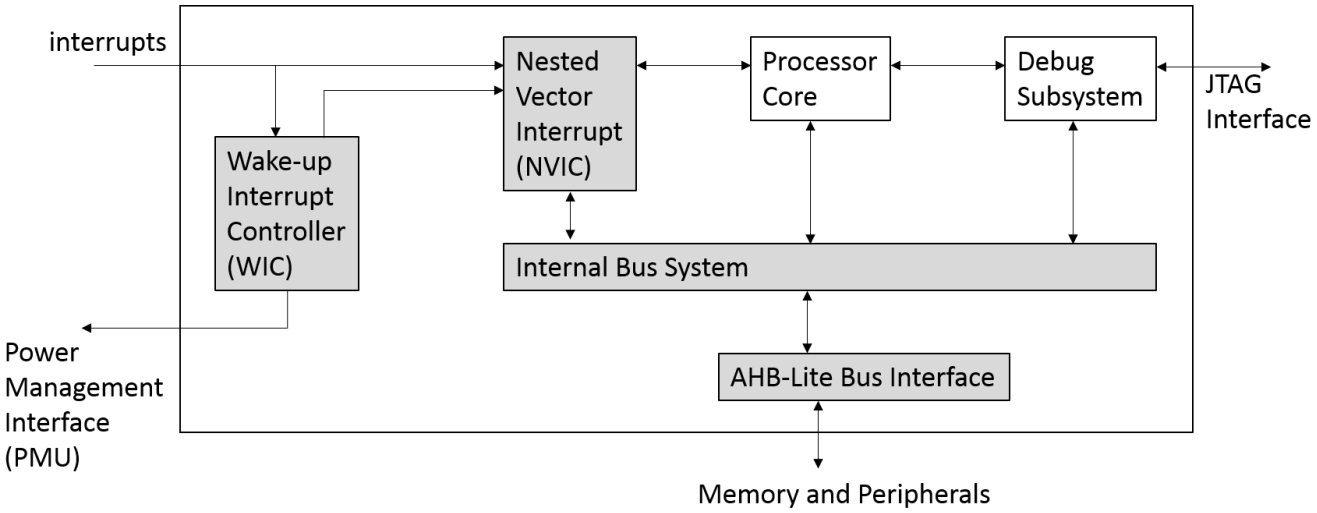


Figure 6.1: NVIC and WIC Block Diagram

6.2 Interrupts

Table 6.1 lists all the vectors for the UT32M0R500 vector table.

Table 6.1: Vector Table

Exception Number	IRQ Number	Priority	Vector	Offset
			Initial SP	0x00
1	-15	-3	Reset	0x04
2	-14	-2	NMI	0x08
3	-13	-1	Hard Fault	0x0C
	[-12:-6]		reserved	
11	-5	Programmable	SVCall	0x2C
	[-4:-3]		reserved	
14	-2	Programmable	PendSV	0x38
15	-1	Programmable	SysTick	0x3C
16	0	Programmable	MBEA	0x40
17	1	Programmable	DualTimer0	0x44
18	2	Programmable	DualTimer1	0x48
19	3	Programmable	PWM	0x4C
20	4	Programmable	RTC	0x50
21	5	Programmable	GPIO16 [GPIO1, Pin0]	0x54
22	6	Programmable	GPIO17 [GPIO1, Pin1]	0x58
23	7	Programmable	GPIO18 [GPIO1, Pin2]	0x5C
24	8	Programmable	GPIO19 [GPIO1, Pin3]	0x60
25	9	Programmable	GPIO20 [GPIO1, Pin4]	0x64
26	10	Programmable	GPIO21 [GPIO1, Pin5]	0x68
27	11	Programmable	GPIO22 [GPIO1, Pin6]	0x6C
28	12	Programmable	GPIO23 [GPIO1, Pin7]	0x70
29	13		reserved	
30	14	Programmable	UART0	0x78
31	15	Programmable	UART1	0x7C
32	16	Programmable	CAN0	0x80
33	17	Programmable	CAN1	0x84
34	18	Programmable	I2C0	0x88
35	19	Programmable	I2C1	0x8C
36	20	Programmable	SPI	0x90
37	21	Programmable	ADC	0x94
38	22	Programmable	GPIO0 combined	0x98
39	23	Programmable	GPIO1 combined	0x9C
40	24	Programmable	GPIO2 combined	0xA0
41	25	Programmable	NFC	0xA4
	[31:26]		Reserved	

6.3 Reset Vector

The Cortex M0+ supports power-on reset and local reset. Power-on reset resets the processor, System Control Space (SCS), and debug logic; local reset resets the processor and SCS, but not the debug logic. The reset exception vector has a fixed priority of -3 (highest) and is permanently enabled.

6.4 Non Maskable Interrupt (NMI)

NMI cannot be disabled, and can be used for safety critical systems, power failure handling or as a watchdog.

6.5 HardFault

HardFault handles all the generic faults that don't have a specific exception, i.e., unknown opcode or memory fault.

6.6 SVCall (SuperVisor Call)

SVCall for OS control when a supervisor instruction is executed.

6.7 PendSV (Pendable Service Call)

PendSV for OS control when high-priority tasks are completed.

6.8 SysTick

The system tick timer (SysTick) is the OS timer used for context switching. SysTick is a 24-bit timer that counts down from the reload value to zero.

6.9 External Interrupts

The Cortex-M0+ supports up to 32 external Interrupts. NVIC accepts active high or active low, or programmable configurations from external Interrupt signals. NVIC control registers can be used to independently enable or disable each external Interrupt and its pending status.

6.9.1 Level and Edge-Triggered Interrupts

External Interrupts are both level and edge-triggered Interrupts. Level Interrupt is de-asserted by the peripheral. Edge-triggered Interrupt is sampled synchronously on the rising edge of PCLK (system clock). In order for the NVIC to detect and latch the Interrupt, the peripheral must assert the Interrupt for at least one clock cycle.

Servicing the ISR automatically removes the pending state from the Interrupt. A level Interrupt holds the asserted signal until it is done servicing it.

6.10 Interrupt Priorities

External Interrupts have four levels of priority. Priority level registers are eight-bit wide, but only two bits [7:6] are implemented, with possible priorities: 0x00, 0x40, 0x80, 0xC0; 0xC0 has the lowest priority and 0x00 has the highest.

6.11 CMSIS functions for NVIC registers

CMSIS is a software interface standard1 to enhance software portability. CMSIS provides a standardized software interface to configure the nested vectored Interrupt controller (NVIC).

Table 6.3: CMSIS Function Table

CMSIS Functions	Description
Void NVIC_EnableIRQ(IRQn_Type IRQn)[1]	Enable an Interrupt or exception
Void NVIC_DisableIRQ(IRQn_Type IRQn) [1]	Disables an Interrupt or exception
Void NVIC_SetPendingIRQ(IRQn_Type IRQn) [1]	Sets the pending status of Interrupt or exception to 1
Void NVIC_ClearPendingIRQ(IRQn_Type IRQn) [1]	Clears the pending status of Interrupt or exception to 1
uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn) [1]	Reads the pending status of Interrupt or exception; returns a non-zero value if the pending status is set to 1
Void NVIC_SetPriority(IRQn_Type IRQn) [1]	Sets priority of an Interrupt or exception with configurable priority to 1
uint32_t NVIC_GetPriority(IRQn_Type IRQn) [1]	Reads the priority of an Interrupt or exception with configurable priority level; returns the current priority level

1. IRQn is the IRQ number specified in Table 6.1.

6.12 Wake-up Interrupt Controller

The UT32M0R500 supports the Wake-up Interrupt Controller (WIC) and a system-level power management unit (PMU). The WIC facilitates clock gating implementation during deep sleep mode, which further reduces power consumption. The WIC masks NVIC Interrupt signals, so no extra programmable registers are required. When an Interrupt is detected, the WIC sends a request to the PMU to restore power and clock signals to the processor, and then the processor can wake up and process the Interrupt request.

7 Power Management

7.1 Overview

The UT32M0R500 datasheet specifies the hardware characteristic for power management, and for low-power modes, the Cortex M0+ architecture supports normal and deep sleep modes. In normal sleep mode, the processor turns off some of the clock signals; in deep sleep mode, it turns off components and reduces power supply to memory blocks. WFE and WFI instruction are used to get the processor into sleep mode; sleep on exit feature speeds up the transition from sleep mode by allowing the processor to stay asleep as long as possible; finally, the wakeup Interrupt controller (WIC) facilitates clock gating implementation during deep sleep mode.

For detailed information on Arm Cortex M0+ Instruction Set, refer to *Cortex-M0+ Devices Generic User Guide.*

7.2 Wakeup Interrupt Controller

The UT32M0R500 includes the wakeup Interrupt controller (WIC) along with the system-level power management unit (PMU). Activated in deep sleep mode, the WIC block masks Interrupt signals while the processor is in sleep mode, so no extra programmable registers are needed. When an Interrupt is detected, the WIC sends a request to the PMU, and then the processor can wake up and process the Interrupt request.

7.3 Sleep-On-Exit

The sleep-on-exit feature allows the processor to enter sleep mode as soon as all the exception handlers are completed. It reduces the power consumption of the processor by avoiding unnecessary programs in the main thread and stacking, unstacking operation. For Interrupt-driven applications, the sleep-on-exit maximizes the amount of time the processor can stay in sleep mode.

7.4 System Control Register

The Cortex M0+ system control register (SCR) controls features of entry to and exit from low power mode, and for more information, refer to *Cortex-M0+ Devices Generic User*.

Table 7.1 gives an overview of the register features.

Table 7.1: System Control Register

Bit Number	Bit Name	Description
1	SLEEPONEXIT	Sleep-on-exit enable bit
2	SLEEPDEEP	0: normal mode 1: deep sleep
4	SEVONPEND	Send event on pend bit; enable generation of event by a new Interrupt pending status

7.5 CMSIS functions for SCR register

CMSIS is a software interface standard to enhance software portability. CMSIS provides a standardized software interface to configure the system control register.

Table 7.2: CMSIS Function Table

CMSIS Functions	Description
void __WFE(void)	Wait for event
void __WFI(void)	Wait for Interrupt
void __SEV(void)	Send event

8 System Controller (SYSCON)

8.1 Overview

The system controller in the UT32M0R500 is an AHB mapped module that contains system level control logic and features. The functionality of the SYSCON module includes:

- Reset status – Last reset source, reset counter
- Power Management Unit (PMU) Enable
- Reset control logic
- Boot Configuration Access
- Clock Divide Control
- Oscillator Shutdown
- Analog Shutdowns for
  - Precision current source
  - Temperature Sensor shutdown
  - Low Noise Voltage Reference to ADC/DAC/Comparators shutdown
  - Current reference to ADC/DAC/Comparators shutdown
- General Purpose Registers with “Keep Alive” option

8.2 “Keep Alive” Functionality

Based on setting in the SYSCON the “Keep Alive” feature allows for data and/or settings to persist during a soft reset. The main areas that are supported by this functionality are:

- The DAC settings in the DAC.CONTROL register are preserved. This functionality is enabled by writing a 1 to the DAC.CONTROL.SOFT_RST_DIS bit.
- The setting for the GPIO can be preserved. This functionality is enabled by writing a 1 to the GPIO.SOFT_RESET.SOFT_RESET bit

8.3 System Controller Register Details

Table 8.4: System Controller Registers

Register	Offset (Hex)
Boot Done Register (BOOT_DONE)	0x0000
Power Management Unit Control Register (PMU_CTRL)	0x0004
Reset Control Register (RST_CTRL)	0x0008
Boot Configuration Value Register (BOOTCFG_VALUE)	0x000C
Reset Information Register (RESET_INFO)	0x0010
Reset Counter Register (RESET_COUNT)	0x0014
Clock Divider Register (CLK_DIVIDE)	0x0018
Clock Control Register (OSC_SHUTDOWN)	0x001C
Analog Circuitry Shutdown Register (ANALOG_SHOTDOWNS)	0x0020
General Purpose Register 0 (GPREG0)	0x0024
General Purpose Register 1 (GPREG1)	0x0028
General Purpose Register 2 (GPREG2)	0x002C

8.3.1 Boot Done Register (BOOT_DONE)

BOOT_DONE																	Offset = 0x0000
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R																	
W																	
Reset	[UU…U]																

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																BD
W																
Reset	[UU…U]															0

Table 8.2: Description of Boot Done Register

Bit Number(S)	Bit Name	Reset State	Description
31-1	RESERVED	[UU…U]	
0	BOOT_DONE (BD)	0	The BOOT_ROM writes this bit. The user should never need to write this bit. Writing 1 exits boot mode (always internal oscillator) and allows the CLKSEL pin to select the clock source. 1: UT32M0R500 completed boot successfully 0: no action

8.3.2 Power Management Unit Control Register (PMU_CTRL)

PMU_CTRL															Offset = 0x0004
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																PE
W																
Reset	[UU…U]															0

Table 8.5: Description of Power Management Unit Control Register

Bit Number(S)	Bit Name	Reset State	Description
31-1	RESERVED	[UU…U]	
0	PMUENABLE (PE)	0	PMUENABLE Value 1: Enable the PMU 0: Disable the PMU

8.3.3 Reset Control Register (RST_CTRL)

RST_CTRL																Offset = 0x0008
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R															RE	LP
W																
Reset	[UU…U]														0	0

Table 8.6: Description of Reset Control Register

Bit Number(S)	Bit Name	Reset State	Description
31-2	RESERVED	[UU...U]	
1	SA_RST_EN (RE)	0	1: GPREGx registers only reset on a hard reset (RSTN low) 0: no action
0	LOCKUPPRESETN (LP)	0	1: System reset on LOCKUP 0: no action

8.3.4 Boot Configuration Value Register (BOOTCFG_VALUE)

BOOTCFG_VALUE																**Offset = 0x000C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R															BC[1:0]	
W																
Reset	[UU…U]														00	

Table 8.7: Description of Boot Configuration Value Register

Bit Number(S)	Bit Name	Reset State	Description
31-2	RESERVED	[UU...U]	
1-0	BOOTCFG (BC)	00	Written by system based on BOOTCFG pins 11: Update from CAN 10: Update from UART 01: reserved 00: Boot from NOR Flash

8.3.5 Reset Information Register (RESET_INFO)

This register contains information describing the type of soft reset that occurred. This register is only reset on hard reset. It can be write-cleared or will be updated on a new soft reset condition.

RESET_INFO																Offset = 0x0010
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														LP	WR	SR
W																
Reset	[UU…U]													0	0	0

Table 8.8: Description of Reset Information Register

Bit Number(S)	Bit Name	Reset State	Description
31-3	RESERVED	[UU...U]	
2	LOCKUPRESET (LP)	0	1: If the last soft reset was caused by lockup. Used if RST_CTRL[0] bit is high 0: no action
1	WDOGRESET (WR)	0	1: If the last soft reset was caused by the watchdog timer, only set if reset enabled in watchdog circuit 0: no action
0	SYSRESET (SR)	0	1: If last soft reset was caused by SYSRESETREQ 0: no action

8.3.6 Reset Count Register (RESET_COUNT)

This register is only reset on hard reset. It can be write-cleared or will be updated on a new soft reset condition.

RESET_COUNT																Offset = 0x0014
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									RESET_COUNT [7:0]							
W																
Reset	[UU…U]								0x00							

Table 8.9: Description of Reset Count Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[UU...U]	
7-0	RESET_COUNT	0x00	Count value of the number of times there have been a soft-reset since being cleared or hard reset

8.3.7 Clock Divider Register (CLK_DIVIDE)

This register defines the clock divider value. This value divides the master system clock that is used by the microcontroller. Also this is a glitch free transition when changing the register value. This register is only reset on hard reset.

CLK_DIVIDE																**Offset = 0x0018**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														CLKDIV[2:0]		
W																
Reset	[UU…U]													000		

Table 8.10: Description of Clock Divider Register

Bit Number(S)	Bit Name	Reset State	Description
31-3	RESERVED	[UU...U]	
2-0	CLKDIV	000	Divide factor for clock in binary 000: no divide 001: divide by 2 010: divide by 4 011: divide by 8 100: divide by 16 101: divide by 32 110: divide by 64 111: unsupported

8.3.8 Clock Control Register (OSC_SHUTDOWN)

The XTAL1/2 pins can be configured to work with either a crystal oscillator or with a single ended clock signal. In crystal oscillator mode (0x4), XTAL1 in the crystal input and XTAL2 is the output feedback for the crystal. In single ended clock mode, either XTAL1 or XTAL2 can operate as the input pin, although XTAL2 is recommended. See the below table:

Table 8.8: External Clock Configuration

Clock Source	CLKSEL	XTAL1	XTAL2	OSC_SHUTDOWN [2:0] [1]
Internal 50MHz Oscillator	LOW	Unused, pulldown to VSS with resistor	Unused, reset state is output, so leave floating	b'000
Crystal Oscillator	HIGH	Clock Input	Clock Feedback Output	b'00X
Single-Ended Oscillator (Recommended)	HIGH	Unused, tie low with 10kΩ	Clock Input	b'11X
Single-Ended Oscillator (Alternate)	HIGH	Clock Input	Enabled output, leave floating	b'10X

Note:

1) 'X' is Don't Care, depending on if the user wants to disable the internal oscillator while using an external clock. The Internal oscillator is required for ADC operation, even when using an external oscillator.

OSC_SHUTDOWN																**Offset = 0x001C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														XI	XO	OS
W																
Reset	[UU…U]													1	0	0

Table 8.11: Description of Clock Control Register

Bit Number(S)	Bit Name	Reset State	Description
31-3	RESERVED	[UU...U]	
2	XTAL_IE (XI)	1	1: Clock input enabled as system clock 0: Disabled
1	XTAL_OEB (XO)	0	1: Disable output for xtal buffer; XTAL2 = Hi-Z 0: Enable output of XTAL1 to XTAL2; XTAL2 = -XTAL1

Bit Number(S)	Bit Name	Reset State	Description
0	OSC_SHUTDOWN (OS)	0	1: Shutdown internal oscillator 0: Enable internal oscillator (only shut down if not using the ADC, and are using an external clock input selected by the CLKSEL pin)

8.3.9 Analog Circuitry Shutdown Register (ANALOG_SHUTDOWNS)

Note: LNREF and IREF are needed for the ADC/DAC/Comparators to work.

ANALOG_SHUTDOWNS																**Offset = 0x0020**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													TS	LS	IS	PS
W																
Reset	[UU…U]												1	1	1	1

Table 8.12: Description of Analog Circuitry Shutdown Register

Bit Number(S)	Bit Name	Reset State	Description
31-4	RESERVED	[UU...U]	
3	TMON_ SHUTDOWN (TS)	1	Temperature sensor shutdown 1: Disable temperature monitor 0: Enable temperature monitor
2	LNREF_ SHUTDOWN (LS)	1	Low noise reference to ADC/DAC/Comparators shutdown 1: Disable low noise reference 0: Enable low noise reference
1	IREF_ SHUTDOWN (IS)	1	Current reference to ADC/DAC/Comparators shutdown 1: Disable current reference 0: Enable current reference
0	PCS_ SHUTDOWN (PS)	1	Precision current source shutdown 1: Disable precision current source 0: Enable precision current source

8.3.10 General Purpose Register 0 (GPREG0)

This register can be used for storing any set of data with a write.

GPREG0																**Offset = 0x0024**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPREG0[31:16]															
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPREG0[15:0]															
Reset	[00...0]															

Table 8.13: Description of General Purpose Register 0

Bit Number(S)	Bit Name	Reset State	Description
31-0	GPREG0	[00...0]	General purpose register 0

8.3.11 General Purpose Register 1 (GPREG1)

This register can be used for storing any set of data with a write.

GPREG1																**Offset = 0x0028**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPREG1[31:16]															
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPREG1[15:0]															
Reset	[00…0]															

Table 8.14: Description of General Purpose Register 1

Bit Number(S)	Bit Name	Reset State	Description
31-0	GPREG1	[00…0]	General purpose register 1

8.3.12 General Purpose Register 2 (GPREG2)

This register can be used for storing any set of data with a write.

GPREG2																**Offset = 0x002C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPREG2[31:16]															
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPREG2[15:0]															
Reset	[00…0]															

Table 8.15: Description of General Purpose Register 2

Bit Number(S)	Bit Name	Reset State	Description
31-0	GPREG2	[00…0]	General purpose register 2

9 SRAM Controller with EDAC and Scrubbing

9.1 Overview

The AHB to SRAM interface module connects an on-chip synchronous SRAM block to the AHB interface for zero wait state accesses. This block uses dual port compiled SRAM. It contains a passive EDAC interface and active scrub engine. The interface supports a no-wait state AHB response for read and write as well as word, half-word, and byte accesses.

The scrubber engine performs a scrubbing operation on the SRAM. Scrubbing is an error correction technique in which a background process periodically inspects the contents of memory and fixes any errors or inconsistencies by replacing them with a functional copy of the data. The scrubbing function operates independently from the EDAC.

The scrub rate is programmable as is the depth of memory to scrub. Single bit errors are corrected and counted. Double bit errors are detected and counted.

The Scrub timer reload value is the number of clock cycles before moving to the next scrub address.

For a 50Mhz system clock, the scrub address rate is given by:

*scrub rate=pclk*ReloadValue*
*scrub rate=20ns*50000*
scrub rate=1 msecs

For the full SRAM, 0x6000 addresses (24576 address * 32 bits = 96kB), the scrubber would take 1ms * 24576 addresses = 24.576 seconds to go through the whole SRAM memory.

Note: Timer_Reload minimum is 4 clock cycles.

9.2 AHB-to-SRAM Interface Register Details

Table 9.16: AHB-to-SRAM Interface Registers

Register	Offset (Hex)
Control Register (CONTROL)	0x0000
Current Scrub Timer Count Register (TIMER_COUNT)	0x0004
Scrub Timer Reload Value Register (TIMER_RELOAD)	0x0008
MBEA Interrupt Status Register (INT_STATUS)	0x000C
Scrub Max Address Register (MAX_ADDRESS)	0x0010
Single Bit Error Count Port A Register (SBE_COUNT_A)	0x0014
Single Bit Error Count Port B Register (SBE_COUNT_B)	0x0018
Multiple Bit Error Count Port A Register (MBE_COUNT_A)	0x001C
Multiple Bit Error Count Port B Register (MBE_COUNT_B)	0x0020
Address of Last MBEA Event (LAST_ADDRESS)	0x0024

9.2.1 Control Register (CONTROL)

CONTROL																Offset = 0x0000
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														MIE		SE
W																
Reset	[00...0]													0	0	1

Table 9.17: Description of the Control Register

Bit Number(S)	Bit Name	Reset State	Description
31-3	RESERVED	[00...0]	
2	MBEA_INT_EN (MIE)	0	1: Causes an Interrupt to be generated when a multiple bit error is encountered on port A data 0: No Interrupts will be generated when a multiple bit error is encountered
1	RESERVED	0	
0	SCRUB_EN (SE)	1	1: Scrubbing is enabled 0: Scrubbing is disabled

9.2.2 Timer Count Register (TIMER_COUNT)

TIMER_COUNT																Offset = 0x0004
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TIMER_COUNT															
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TIMER_COUNT															
W																
Reset	[00...0]															

Table 9.18: Description of the Timer Count Register

Bit Number(S)	Bit Name	Reset State	Description
31-0	TIMER_COUNT	[00...0]	Current Value of the scrub timer

9.2.3 Timer Reload Value Register (TIMER_RELOAD)

TIMER_RELOAD																Offset = 0x0008
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TIMER_RELOAD															
Reset	0x0001															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TIMER_RELOAD															
Reset	[00…0]															

Table 9.19: Description of the Timer Reload Value Register

Bit Number(S)	Bit Name	Reset State	Description
31-0	TIMER_RELOAD	0x0001_0000	Countdown value of the scrub timer. Note: Timer_Reload minimum is 4 clock cycles.

9.2.4 MBEA Interrupt Status Register (INT_STATUS)

INT_STATUS																Offset = 0x000C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W																MI
Reset	[00…0]															0

Table 9.20: Description of the MBEA Interrupt Status Register

Bit Number(S)	Bit Name	Reset State	Description
31-1	RESERVED	[00…0]	
0	MBEA_INT (MI)	0	Set if MBEA_INT_EN and multiple bit error encountered on port A. Will stay set until written with any value

9.2.5 Scrub Max Address Register (MAX_ADDRESS)

MAX_ADDRESS																Offset = 0x0010
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		Max_Scrub_Address														
W																
Reset	0	0x6000														

Table 9.21: Description of the Scrub Max Address Register

Bit Number(S)	Bit Name	Reset State	Description
31-15	RESERVED	[00…0]	
14-0	Max_Scrub_Address	0x6000	Maximum address to scrub through before starting back at the beginning of address space

9.2.6 Single Bit Error Port A Register (SBE_COUNT_A)

REG NAME						Offset = 0x0004										
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SBEA_COUNT															
W																
Reset	[00…0]															

Table 9.22: Description of the Single Bit Error Port A Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-0	SBEA_COUNT	[00…0]	Count of Single Bit Errors detected and corrected during port A reads. Write any value to this register to clear

9.2.7 Single Bit Error Port B Register (SBE_COUNT_B)

SBE_COUNT_B						Offset = 0x0018										
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SBEB_COUNT															
W																
Reset	[00…0]															

Table 9.23: Description of the Single Bit Error Port B Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-0	SBEB_COUNT	[00…0]	Count of Single Bit Errors detected and corrected during scrubbing. Write any value to this register to clear

9.2.8 Multiple Bit Error Port A Register (SCRUB.MBE_COUNT_A)

REG NAME																**Offset = 0x0004**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MBEA_COUNT															
W																
Reset	[00…0]															

Table 9.24: Description of the Multiple Bit Error Port A Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-0	MBEA_COUNT	[00…0]	Count of Multiple Bit Errors detected during port A reads. Write any value to this register to clear

9.2.9 Multiple Bit Error Port B Register (MBE_COUNT_B)

MBE_COUNT_B																Offset = 0x0020
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset																

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MBEB_COUNT															
W																
Reset	[00…0]															

Table 9.25: Description of the Multiple Bit Error Port B Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-0	MBEB_COUNT	[00…0]	Count of Multiple Bit Errors detected during scrubbing. Write any value to this register to clear

9.2.10 Address of the Last MBE Event Register (LAST_ADDRESS)

LAST_ADDRESS																Offset = 0x0024
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																LA
W																
Reset	[00…0]															0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LAST_ADDRESS															
W																
Reset	[00…0]															

Table 9.26: Description of the Address of the Last MBE Event Register

Bit Number(S)	Bit Name	Reset State	Description
31-17	RESERVED	[00…0]	
16-0	LAST_ADDRESS (LA)	[00…0]	Last Port A address that received a multiple bit upset. Overwritten by new MBE addresses.

10 NOR Flash Controller (NFC)

10.1 Overview

The NOR Flash Controller (NFC) is the UT32M0R500 interface to the 64 Mbit NOR Flash Memory (NFM). It provides the functionality to control and access the Flash using the JEDEC 42.4 Flash command set standard. The NFC provides the data interface and control protocols to operate the NOR Flash via the Nor Flash Memory I/O.

To meet radiation requirements, the NOR Flash requires to remain unpowered for 90% of the mission life or 10% duty cycle. Also the boot rom automatically powers the flash down after boot.

The NFC supports 4 of the 11 Nor Flash Commands

1. Read
2. NVMEM Reset
3. Program
4. Sector Erase

The commands not supported:

1. Unlock Bypass
2. Unlock Bypass Program
3. Unlock Bypass Reset
4. Chip Erase
5. CFI Query
6. Erase Suspend
7. Erase Resume

10.2 NFC Register Details

Table 10.27: NFC Registers

Register	Offset (Hex)
Control Register (CONTROL)	0x0000
Status Register (STATUS)	0x0004
Sector Address Register (SECTOR_ADDR)	0x0008
Peripheral ID0 Register (PERIPH_ID0)	0x0FC0
Peripheral ID1 Register (PERIPH_ID1)	0x0FC4
Peripheral ID2 Register (PERIPH_ID2)	0x0FC8
Peripheral ID3 Register (PERIPH_ID3)	0x0FCC
Peripheral ID4 Register (PERIPH_ID4)	0x0FD0
Peripheral ID5 Register (PERIPH_ID5)	0x0FD4
Peripheral ID6 Register (PERIPH_ID6)	0x0FD8
Peripheral ID7 Register (PERIPH_ID7)	0x0FDC
Peripheral ID8 Register (PERIPH_ID8)	0x0FE0
Peripheral ID9 Register (PERIPH_ID9)	0x0FE4
Peripheral ID10 Register (PERIPH_ID10)	0x0FE8
Peripheral ID11 Register (PERIPH_ID11)	0x0FEC
Peripheral ID12 Register (PERIPH_ID12)	0x0FF0
Peripheral ID13 Register (PERIPH_ID13)	0x0FF4
Peripheral ID14 Register (PERIPH_ID14)	0x0FF8
Peripheral ID15 Register (PERIPH_ID15)	0x0FFC

10.2.1 Control Register (CONTROL)

CONTROL																**Offset = 0x0000**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																NPD
W																
Reset	[00…0]															0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EBM	CIQ	IEQ	EEP	SSE	ESE	DN	SM	WP						RNV	RNF
W																
Reset	0	0	0	0	0	0	0	0	0	[00…0]					0	0

Table 10.28: Description of the Control Register

Bit Number(S)	Bit Name	Reset State	Description
31-17	RESERVED	[00...0]	
16	NVM POWER DOWN (NPD)	0	Remove power to NOR flash when set 1: Enable power down (no power to NOR flash) 0: Disable power down
15	ENABLE BYTE MODE (EBM)	0	When enabled, all transfers to and from NOR flash will be byte length only (disables 16 bit transfers) 1: Enable 0: Disable
14	CLEAR IRQ (CIQ)	0	Clears Interrupt Request 1: Clears 0: No action
13	ENABLE IRQ (EIQ)	0	Enable Interrupt request. When enabled, an Interrupt will be generated on rising edge of OP_COMPLETE bit (STATUS Register) 1: Enable 0: No action
12	ENABLE EMBEDDED PROGRAM (EEP)	0	Enable write operations to NOR flash 1: Enable 0: Disable
11	START SECTOR ERASE (SSE)	0	1: Start sector erase operation 0: No action NOTE: Enable Sector Erase bit must be '1' prior to setting Start Sector Erase to '1'
10	ENABLE SECTOR ERASE (ESE)	0	1: Enable sector erase operation 0: Disable sector erase operation
9	DISABLE NOR FLASH (DN)	0	1: Disable NOR flash, NOR flash output pins place in high-Z state 0: Enable NOR flash
8	STANDBY MODE (SM)	0	1: NOR flash placed in low power (standby) mode 0: Disable low power (standby) mode
7	WRITE PROTECT MODE (WP)	0	1: Disable programming and erasing in Sectors SA0.1 (Bank 1) and SA140, 141 (Bank 2) 0: Enable programming and erasing
6-2	RESERVED	[00…0]	
1	RESET NOR FLASH (RNV)	0	1: Perform reset operation on NOR flash. This consists of a 500ns minimum reset operation. The bit resets itself at the end of the operation 0: No action
0	RESET NFC (RNF)	0	1: Perform reset operation on NFC only 0: No action

10.2.2 Status Register (STATUS)

STATUS																**Offset = 0x0004**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					WH	WL	ES	IS	DQ7	DQ6	DQ5	DQ3	DQ2	OC	OB	RB
W																
Reset	[00…0]				0	0	0	0	0	0	0	0	0	0	0	0

Table 10.29: Description of the Status Register

Bit Number(S)	Bit Name	Reset State	Description
31-12	RESERVED	[00...0]	
11	WRITE STARTED LOW (WH)	0	This bit is set when the Op_Busy_n line transitions hi-low during a Wr Hi word operation. This is NVMEM Handshake that it accepted Write operation. The signal can be checked directly by reading bit 0, but transition may likely be missed. Bit is sticky, will stay set until a new Erase Operation is started 1: Op Busy N has transitioned 0: No action
10	WRITE STARTED LOW (WL)	0	This bit is set when the Op_Busy_n line transitions hi-low during a Wr Low word operation. This is NVMEM Handshake that it accepted Write operation. The signal can be checked directly by reading bit 0, but transition may likely be missed. Bit is sticky, will stay set until a new Write Operation is started 1: Op Busy N has transitioned due to a write low word operation 0: No action
9	ERASE STARTED (ES)	0	This bit is set when the Op_Busy_n line transitions hi-low during an Erase operation. This is NVMEM Handshake that it accepted Erase operation. The signal can be checked directly by reading bit 0, but transition may likely be missed. Bit is sticky, will stay set until a new Erase Operation is started 1: Op Busy N bit has transitioned during an erase operation 0: No action
8	INT STATUS (IS)	0	1: An Interrupt has occurred 0: No action
7	DQ[7] (DQ7)	0	Value of NOR flash DQ[7] line
6	DQ[6] (DQ6)	0	Value of NOR flash DQ[6] line
5	DQ[5] (DQ5)	0	Value of NOR flash DQ[5] line
4	DQ[3] (DQ3)	0	Value of NOR flash DQ[3] line
3	DQ[2] (DQ2)	0	Value of NOR flash DQ[2] line
2	OP COMPLETE (OC)	0	Internal NFC operation complete 1: Operation complete 0: No action
1	OP BUSY N (OB)	0	Internal NFC is busy 1: NFC is idle 0: NFC is busy with operation
0	READY BUSY N (RB)	0	NOR flash busy indicator 1: NOR flash is idle 0: NOR flash busy with operation

10.2.3 Sector Address Register (SECTOR_ADDR)

SECTOR_ADDR															Offset = 0x0008
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W											SECTADDR [21:16]					
Reset	[00…0]										[00…0]					

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SECTADDR 15:0]															
Reset	[00…0]															

Table 10.30: Description of the Sector Address Register

Bit Number(S)	Bit Name	Reset State	Description
31-22	RESERVED	[00…0]	
21-0	SECTADDR	[00…0]	Sector address of flash sector to perform operation (read, write, or erase)

10.2.4 Peripheral ID0 Register (PERIPH_ID0)

PERIPH_ID0															Offset = 0x0FC0
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W									PERIPH_ID0							
Reset	[00…0]								[00…0]							

Table 10.31: Description of the Peripheral ID0 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID0	[00…0]	ID0 Value

10.2.5 Peripheral ID1 Register (PERIPH_ID1)

PERIPH_ID1																Offset = 0x0FC4
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID1							
W																
Reset	[00…0]								[00…0]							

Table 10.32: Description of the Peripheral ID1 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID1	[00…0]	ID1 Value

10.2.6 Peripheral ID2 Register (PERIPH_ID2)

PERIPH_ID2																Offset = 0x0FC8
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID2							
W																
Reset	[00…0]								[00…0]							

Table 10.33: Description of the Peripheral ID2 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID2	[00…0]	ID2 Value

10.2.7 Peripheral ID3 Register (PERIPH_ID3)

PERIPH_ID3																Offset = 0x0FCC
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID3							
W																
Reset	[00…0]								[00…0]							

Table 10.34: Description of the Peripheral ID3 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID3	[00…0]	ID3 Value

10.2.8 Peripheral ID4 Register (PERIPH_ID4)

PERIPH_ID4																Offset = 0x0FD0
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID4							
W																
Reset	[00…0]								0x04							

Table 10.35: Description of the Peripheral ID4 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID4	0x04	ID4 Value

10.2.9 Peripheral ID5 Register (PERIPH_ID5)

PERIPH_ID5																Offset = 0x0FD4
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID5							
W																
Reset	[00…0]								[00…0]							

Table 10.36: Description of the Peripheral ID5 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID5	[00…0]	ID5 Value

10.2.10 Peripheral ID6 Register (PERIPH_ID6)

PERIPH_ID6																Offset = 0x0FD8
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID6							
W																
Reset	[00…0]								[00…0]							

Table 10.11: Description of the Peripheral ID6 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID6	[00…0]	ID6 Value

10.2.11 Peripheral ID7 Register (PERIPH_ID7)

PERIPH_ID7																Offset = 0x0FDC
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID7							
W																
Reset	[00…0]								[00…0]							

Table 10.12: Description of the Peripheral ID7 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID7	[00…0]	ID7 Value

10.2.12 Peripheral ID8 Register (PERIPH_ID8)

PERIPH_ID8																Offset = 0x0FE0
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID8							
W																
Reset	[00…0]								0x18							

Table 10.13: Description of the Peripheral ID8 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID8	0x18	ID8 Value

10.2.13 Peripheral ID9 Register (PERIPH_ID9)

PERIPH_ID9																Offset = 0x0FE4
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID9							
W																
Reset	[00…0]								0xBB							

Table 10.14: Description of the Peripheral ID9 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID9	0xBB	ID9 Value

10.2.14 Peripheral ID10 Register (PERIPH_ID10)

PERIPH_ID10																Offset = 0x0FE8
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID10							
W																
Reset	[00…0]								0x1B							

Table 10.15: Description of the Peripheral ID10 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID10	0x1B	ID10 Value

10.2.15 Peripheral ID11 Register (PERIPH_ID11)

PERIPH_ID11																Offset = 0x0FEC
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID11							
W																
Reset	[00…0]								[00…0]							

Table 10.16: Description of the Peripheral ID11 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID11	[00…0]	ID11 Value

10.2.16 Peripheral ID12 Register (PERIPH_ID12)

PERIPH_ID12																Offset = 0x0FF0
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID12							
W																
Reset	[00…0]								0x0D							

Table 10.17: Description of the Peripheral ID12 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	PERIPH_ID12	0x0D	ID12 Value

10.2.17 Peripheral ID13 Register (PERIPH_ID13)

PERIPH_ID13																Offset = 0x0FF4
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID13							
W																
Reset	[00...0]								0xF0							

Table 10.18: Description of the Peripheral ID13 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00...0]	
7-0	PERIPH_ID13	0xF0	ID13 Value

10.2.18 Peripheral ID14 Register (PERIPH_ID14)

PERIPH_ID14																Offset = 0x0FF8
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID14							
W																
Reset	[00...0]								0x05							

Table 10.19: Description of the Peripheral ID14 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00...0]	
7-0	PERIPH_ID14	0x05	ID14 Value

10.2.19 Peripheral ID15 Register (PERIPH_ID15)

PERIPH_ID15																Offset = 0x0FFC
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									PERIPH_ID15							
W																
Reset	[00…0]								0xB1							

Table 10.20: Description of the Peripheral ID15 Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00...0]	
7-0	PERIPH_ID15	0xB1	ID15 Value

11 ARM Keil uVsion Tools

11.1 Overview

The Keil Microcontroller Development Kit (MDK) includes uVision's integrated development environment (IDE), which includes project management, make facilities, source code editing, program debugging and complete simulation. Keil µVision tools include the compiler, assembler, linker and Libraries. The MDK produces the code and uses the debugger to run it on the UT32MR500 eval board.

Software Packs includes devices, CMSIS and MDK-Middleware. Device support includes device startup, HAL and CMSIS drivers; CMSIS support includes CMSIS-Core, CMSIS-DSP and CMSIS-RTOS; and MDK-Middleware support includes Network, USB, file system, graphics, mbed and IoT.

The µVision debugger connects to the ULINK2 adapter which connects to the UT32M0R500 JTAG interface connector. The debugger supports breakpoints, watch windows, and execution control. It also supports event/exception viewers, logic analyzer, execution profiler, and code coverage.

NOTE: ULINK2 JTAG only loads the application into SRAM for debugging, and for Flash programming, a Terminal Window is needed to download the image via UART0 or CAN0.

For programming the Flash memory, see Boot ROM.

For more detailed information on Arm Keil MDK, refer to http://www2.keil.com/mdk5.

For more detailed information on the ULINK2 JTAG adapter, refer to www.keil.com/ulink.

Figure 11.1 shows a simplified block diagram of debugging applications.

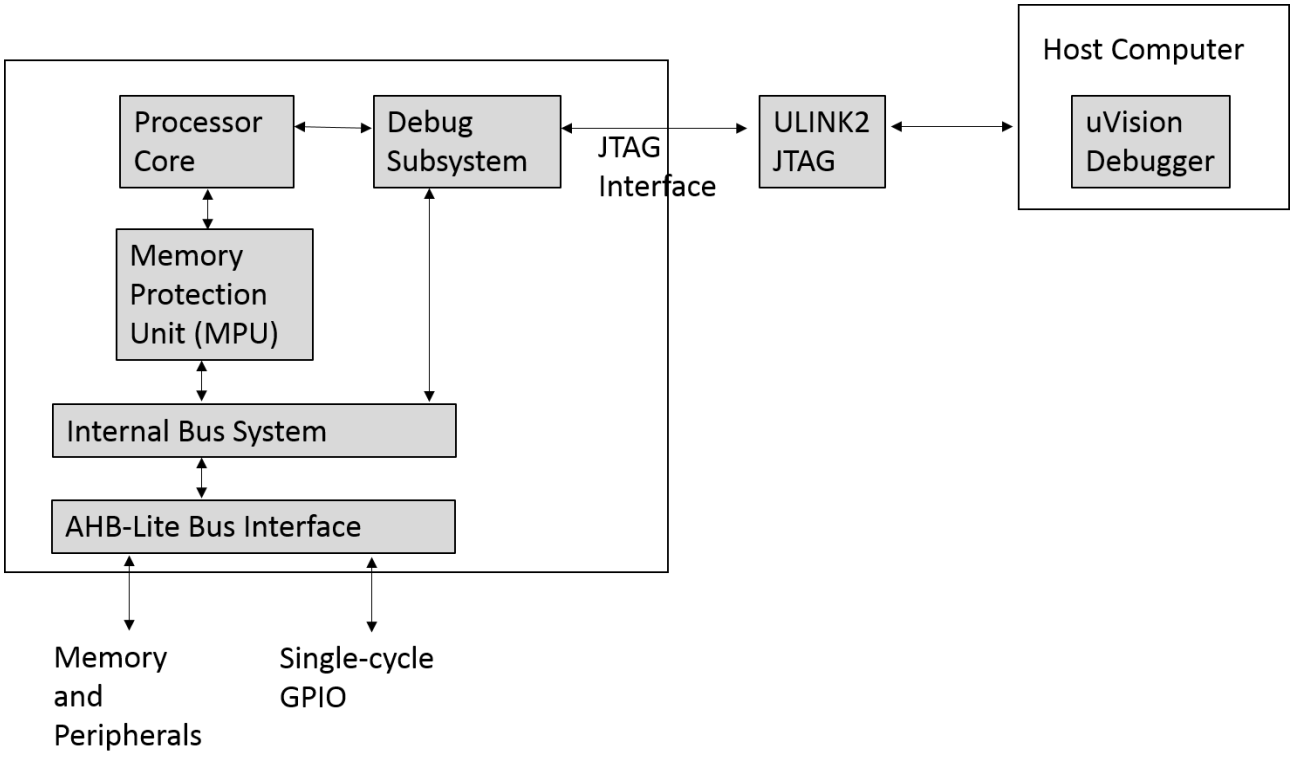


Figure 11.1: Debugging Applications

11.2 MDK Tools

The MDK tools includes all the components for creating, building and debugging applications the UT32M0R500 microcontroller. It also includes the ARM C/C++ compiler with assembler, linker and libraries.

11.3 Software Packs

Software Packs include CMSIS libraries, middleware, board support, code templates and example projects. Software packs contain specific information for the microcontroller device and software components that are available for applications.

The device information pre-configures development tools for the specific microcontroller and shows only the options that are relevant for the selected device.

11.4 MDK Editions

The Keil uVision provides the MDK-Lite license free version restricted to 32KB of code, but it allows for complete software development.

For more detailed information on all MDK editions, refer to www.keil.com/editions.

12 Boot Configuration

The UT32M0R500 supports four (4) modes of booting the device where the mode selection is based on the configuration of the BOOTCFG0 and BOOTCFG1 pins as specified in Table 12.1.

Table 12.1: Boot Mode Selection Description

Boot mode selection pins		Boot Mode	Description
BOOTCFG1	BOOTCFG0		
0	0	0	Load image from internal Flash memory into SRAM and execute
0	1	1	Reserved
1	0	2	Load/Update image over UART0 into flash (reset required)
1	1	3	Load/Update image over CAN0 into flash (reset required)

Note: Boot modes will only be switched or interpreted on a RESET event or external reset.
The following sections give further details of the three boot modes supported by the UT32M0R500.

12.1 Boot Mode 0 (BOOTCFG = 2'b00)

This mode describes the loading of the firmware image from NOR Flash into internal SRAM memory mode of operation. This mode is considered the normal boot operation mode. In this mode, the bootloader performs a system initialization where the device is placed in the default state and initializes communication with the NOR Flash. The bootloader copies the firmware image from NOR Flash memory to internal SRAM. After copying the user code, a CRC verification of the code is performed to determine if the transfer was successful. Upon a successful code transfer, the bootloader checks the CLKSEL pin to determine if an external clock is to be used. If the CLKSEL pin is in a High state, then the system clock is switched over to using the external clock on the XTAL[1:0] pins. All bootloader operations are performed using the internal oscillator operation at 50 MHz until the CLKSEL pin is checked. After the clock selection is performed, the stack pointer is initialized, then the program counter is set to point to the beginning of the transferred code, and code execution starts.

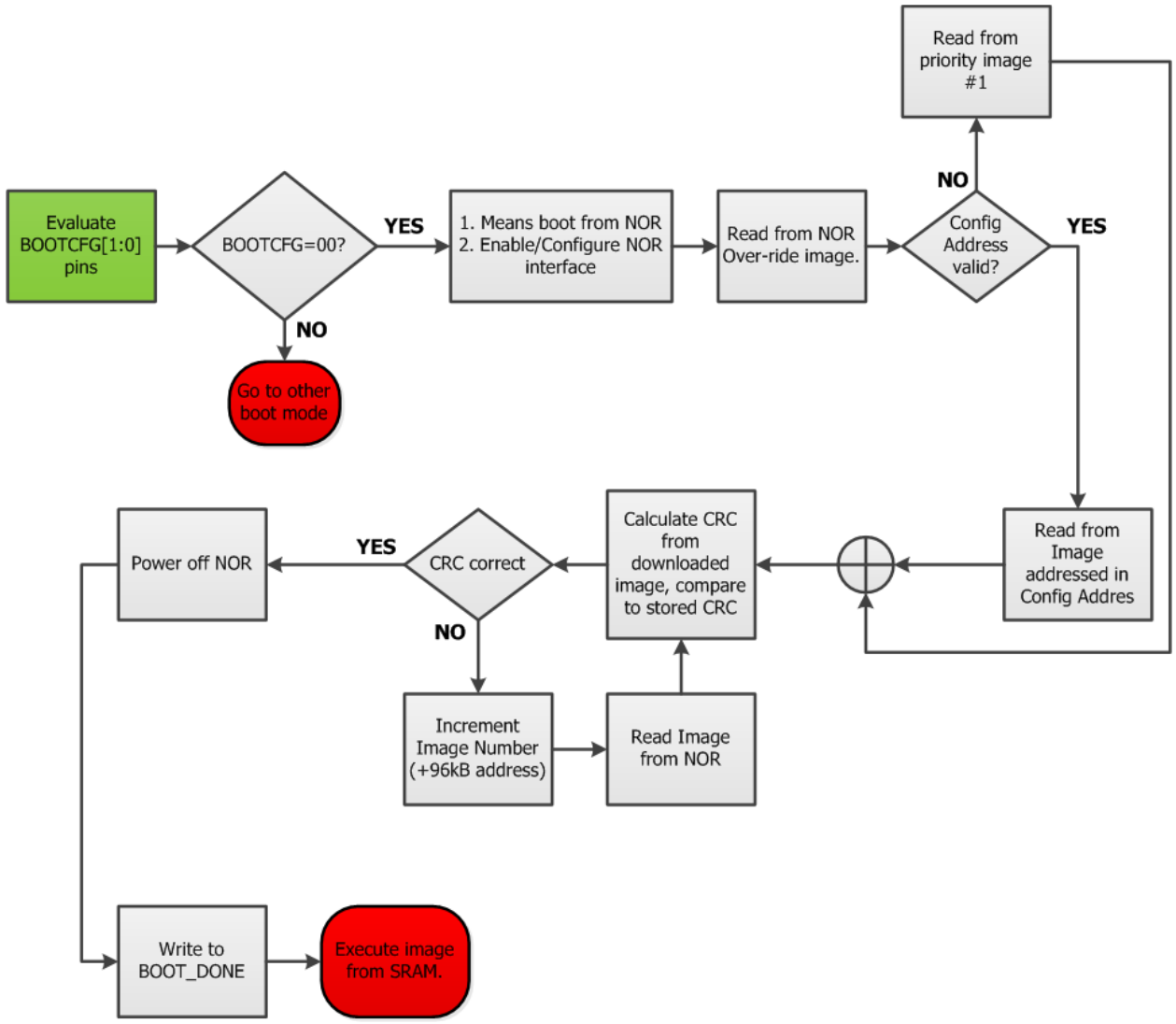


Figure 12.1: Flow diagram for BOOTCFG = 2'b00

12.2 Boot Mode 1 (BOOTCFG = 2'b01)

Reserved.

12.3 Boot Mode 2 (BOOTCFG = 2'b10)

This mode describes the loading of the firmware image over UART (UART0) mode of operation. In this mode, the bootloader first performs a system initialization where the device is placed in the default state. The bootloader then configures the UART0 for operating at 19200 Baud/8Baud/8bits/noparity/1 stop bit. After configuring the UART0, the bootloader loads the firmware image transmitted over UART0 to the addressed memory of the internal Flash memory location based on the image number selected. All bootloader operations are performed using the internal oscillator operation at 50 MHz. To execute the firmware, the BOOTCFG pins must be set to 2'b00 and a reset applied.

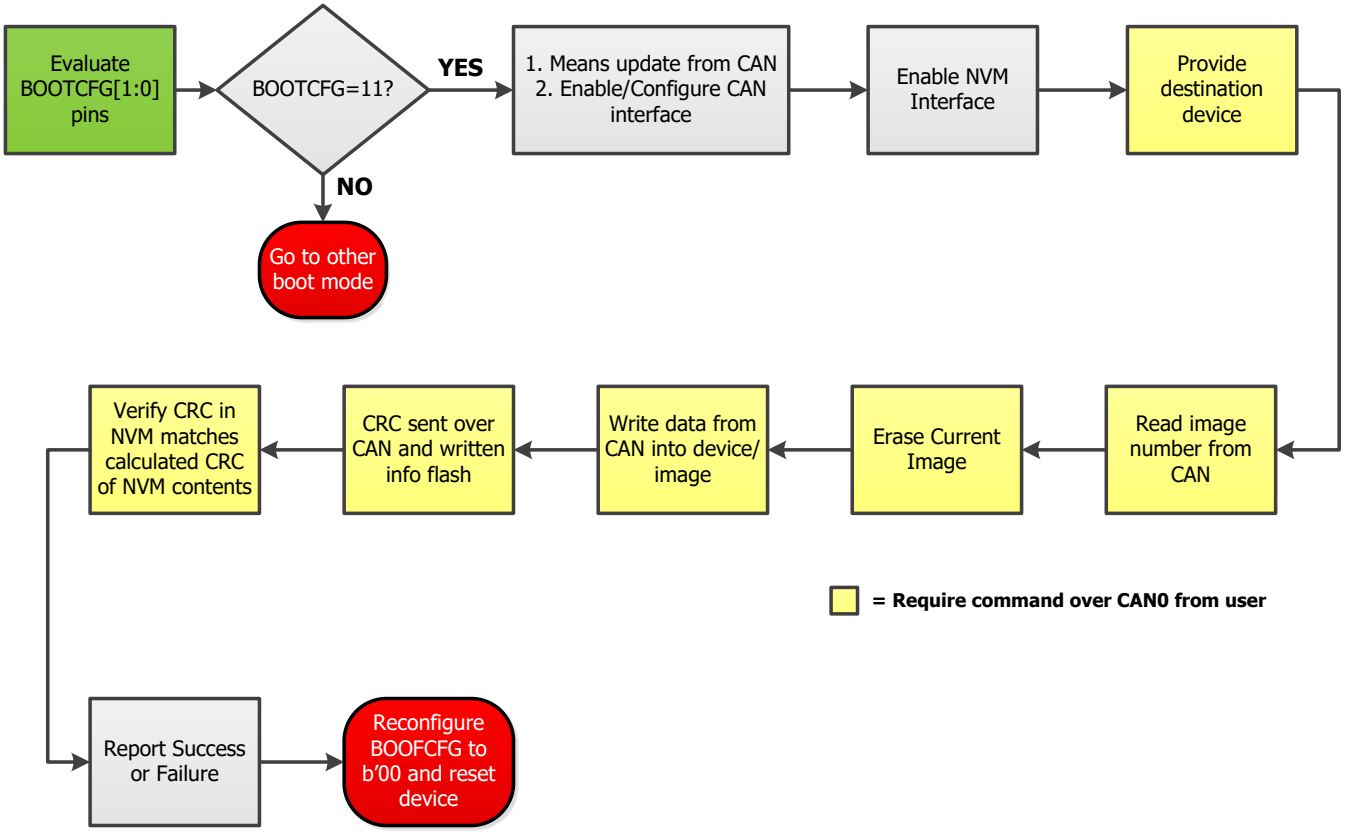


Figure 12.2: Flow diagram for BOOTCFG = 2'b10

12.4 Boot Mode 3 (BOOTCFG = 2'b11)

This mode describes the loading of the firmware image over the CAN bus (CAN0) mode of operation. In this mode, the bootloader first performs a system initialization where the device is placed in the default state. The bootloader then configures the CAN0 for operating at 125 kHz. After configuring the CAN0, the bootloader copies the firmware image transmitted over CAN0 to the addressed NOR Flash location as prescribed by the image number. After a successful load of the firmware, the BOOTCFG pins must be set to 2'b00 and a reset applied for the firmware to start executing. All bootloader operations are performed using the internal oscillator operation at 50 MHz.

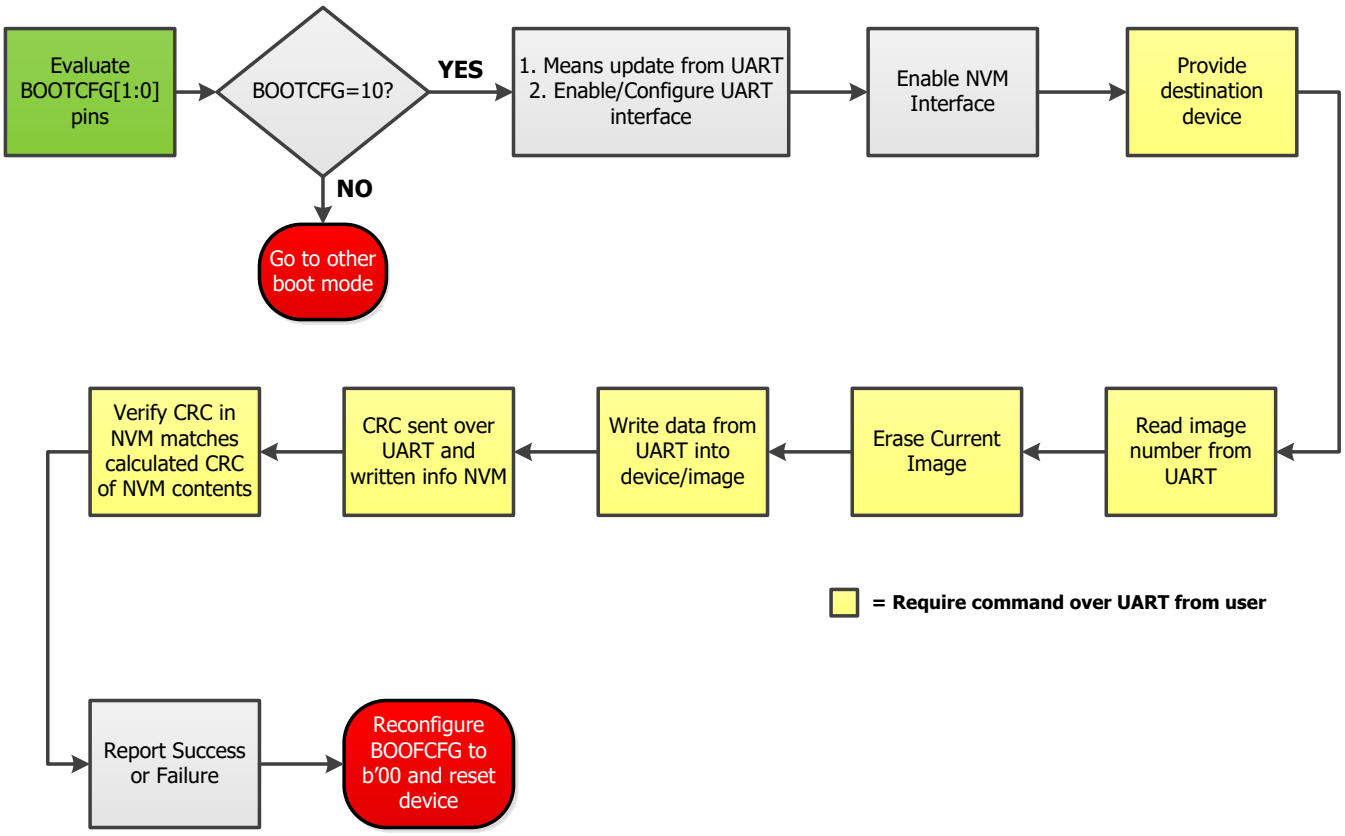


Figure 12.3: Flow diagram for BOOTCFG = 2'b11

13 General Purpose Input/Output (GPIO) Port

13.1 Overview

The general-purpose input/output (GPIO) peripheral implements three 16-bit ports with Interrupt support. GPIO pins represent a 1 or 0 state; Each GPIO can be individually set as an input or output and can optionally generate an Interrupt. In addition to GPIO implementation, many GPIO pins have alternate functions. After hard reset specified by the datasheet, GPIOs with alternate functions default to alternate functions, so the user has to configure the GPIO pin as a GPIO to access its functionality.

Figure 13.1 shows a simplified block diagram for a single GPIO pin.

For GPIO example code and application specific interface (API), refer to the **UT32MOR500 Software Development Kit (SDK).**

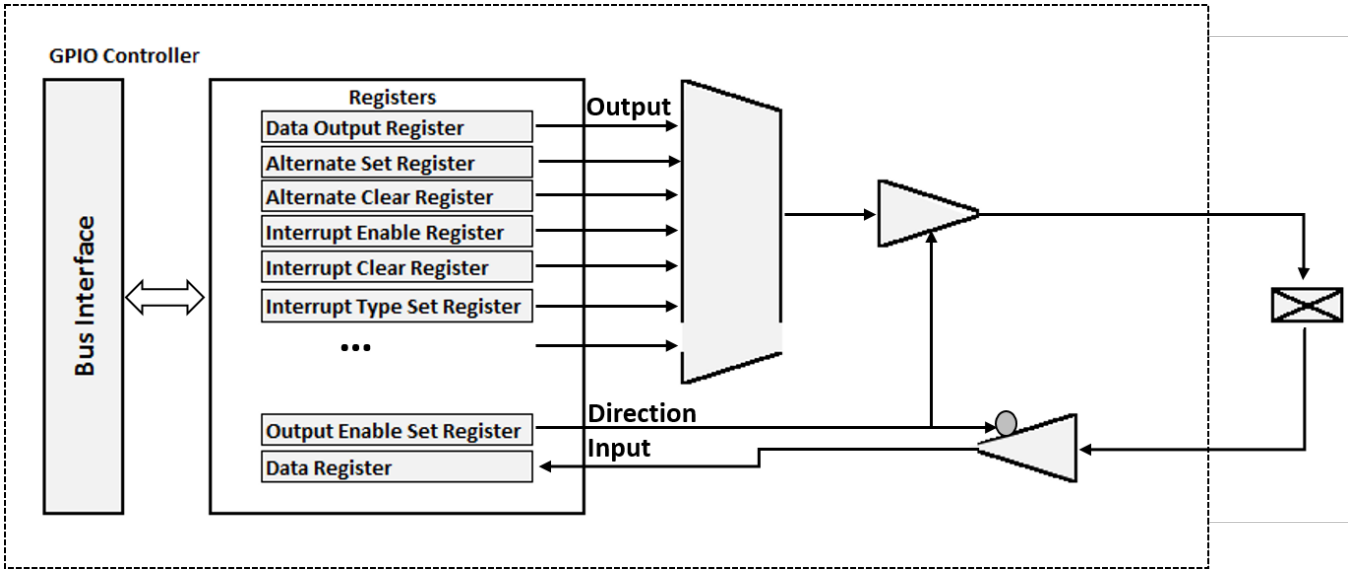


Figure 13.1: GPIO Block Diagram

13.2 Functional Description

The datasheet specifies hardware characteristics for each GPIO, and by software, the user can dynamically configure each GPIO in several modes.

Input mode:

- Pull-up
- Pull-down
- Floating, High-impedance or Tri-state

Output mode:

- Push-pull
- Open-drain

Alternate Function mode:

- Push-pull
- Open-drain

Note: Open-drain is not a true “open-drain” but implemented in software by floating (input) the GPIO pin, connecting an external pull up/down resistor, and finally, driving (output) the GPIO pin high or low.

13.3 Operation

TheUT32M0R500 GPIO pins have multiple functions. Configuration registers control the functionality of the pin and its connectivity to the on-chip peripherals. Peripherals should be connected to the appropriate pins prior to being activated and prior to any related Interrupts being enabled. Dedicated GPIO are initialized as inputs, whereas GPIO with alternate functions are initialized to use the alternate function. The GPIO’s are configured in three banks of 16 pins each: Bank 0 GPIO0[15:0], Bank 1 GPIO1[15:0], Bank 2 GPIO215:0] for a total of 48 GPIO’s, GPIO0-47. Each bank has a shared Interrupt (one Interrupt for each of the 16 pins within the bank). In addition, each of the inputs to the GPIO (when configured as input) can be used as an IRQ (Interrupt request). Additionally, GPIO1[7:0], GPIO16-23 can each have a dedicated external Interrupt. All pins can be configured to have a pull-up, pull-down, or tri-state (for open-drain operation). The GPIO are half-word (16 bit), byte (8 bit) or half-byte (4 bit) addressable where read or writes occur in a single cycle. The GPIO support upper/lower byte mask registers for access control.

For more information on external Interrupts and the nested vector Interrupt (NVIC), refer to Nested Vector Interrupt Controller (NVIC).

13.4 External Interrupt generation

The GPIO provides external Interrupt capabilities. There is one external Interrupt per GPIO Bank, Interrupt vectors 22-24 for GPIO0-GPIO2 respectively. Additionally, GPIO1[7:0], GPIO16-23 can each have a dedicated external Interrupt, vectors 5-12 respectively. The UTM0R500 microcontroller based on the Cortex M0+ processor contains an internal nested vector Interrupt controller (NVIC) that supports up to 32 external IRQ’s and a non-maskable Interrupt (NMI). NVIC handles the nested Interrupts automatically based on priorities and Interrupt number, and If an Interrupt is accepted, it communicates with the processor to allow it to execute the correct Interrupt handler.

For more information on external Interrupts and the nested vector Interrupt (NVIC), refer to Nested Vector Interrupt Controller (NVIC).

Within each bank, each individual pin can be configured as an Interrupt with separate polarity and Interrupt type, as shown in Table 13.1. After an Interrupt is triggered, the corresponding bit in the INTSTATUS Register is set. This also causes the corresponding bit of the GPIOINT[15:0] signal to be asserted. As a result, the combined Interrupt signal, COMBINT, is also asserted. The Interrupt status flag is cleared by writing a 1 to the corresponding bit of the INTCLEAR Register, the same address as the INTSTATUS Register.

The system level clock signal, PCLK, must be active during Interrupt detection, because of the double flip-flop synchronization logic. There is also a three-cycle latency for Interrupt generation that consists of two cycles for input signal synchronization, and one cycle for registering of the Interrupt status.

Table 13.1: Interrupt Settings

Interrupt Enable (INTENSET/INTENCLR Registers)	Interrupt Polarity (INTPOLSET/ INTPOLCLR Registers)	Interrupt Type (INTTYPESET/ INTTYPECLR Registers)	Interrupt features
0	-NA-	-NA-	Disable
1	0	0	Low-level
1	0	1	Falling edge
1	1	0	High-level
1	1	1	Rising edge

13.5 Masked access

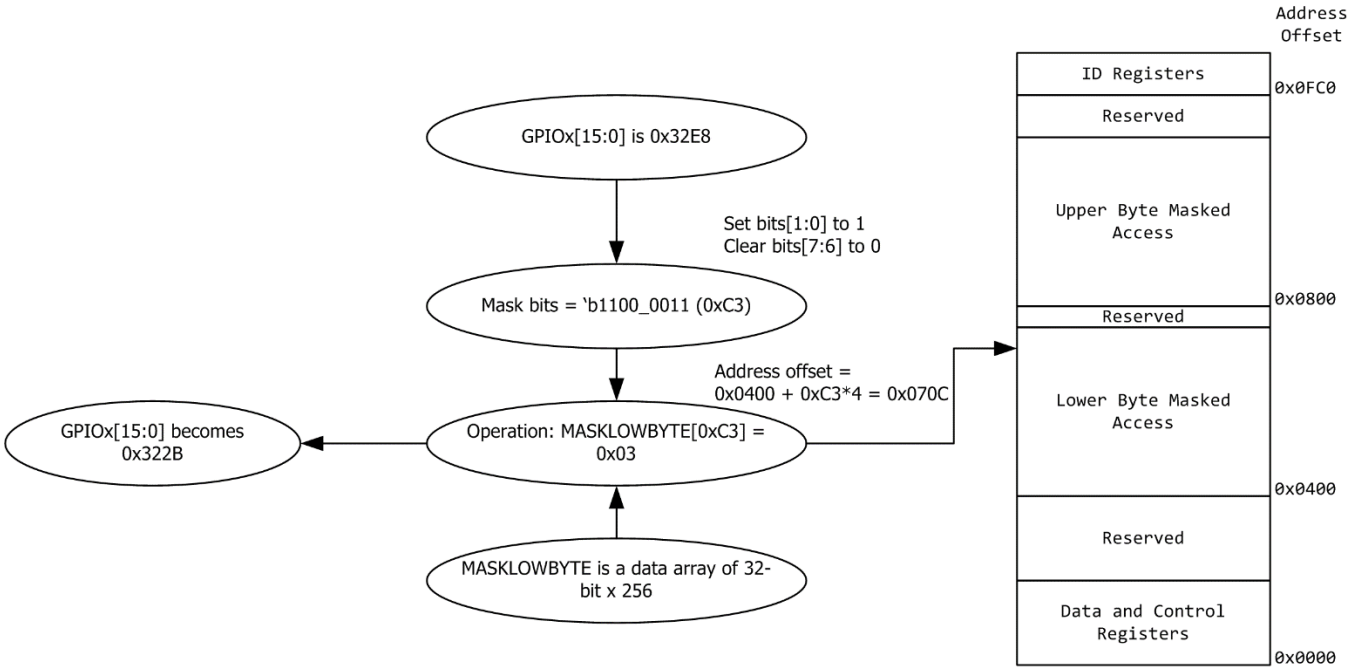
The masked access feature permits individual bits or multiple bits to be read from or written to in a single transfer. This avoids software-based read-modify-write operations that are not thread safe. With the masked access operations, the 16-bit I/O is divided into two halves, lower byte and upper byte. The bit mask address spaces are defined as two arrays, each containing 256 words.

For example, to set bits[1:0] to 1 and clear bits[7:6] in a single operation, you can carry out the write to the lower byte mask access address space. The required bit mask is 0xC3, and you can write the operation as MASKLOWBYTE[0xC3] = 0x03 as shown below.

MASKLOWBYTE Operation (Mask access 1) :

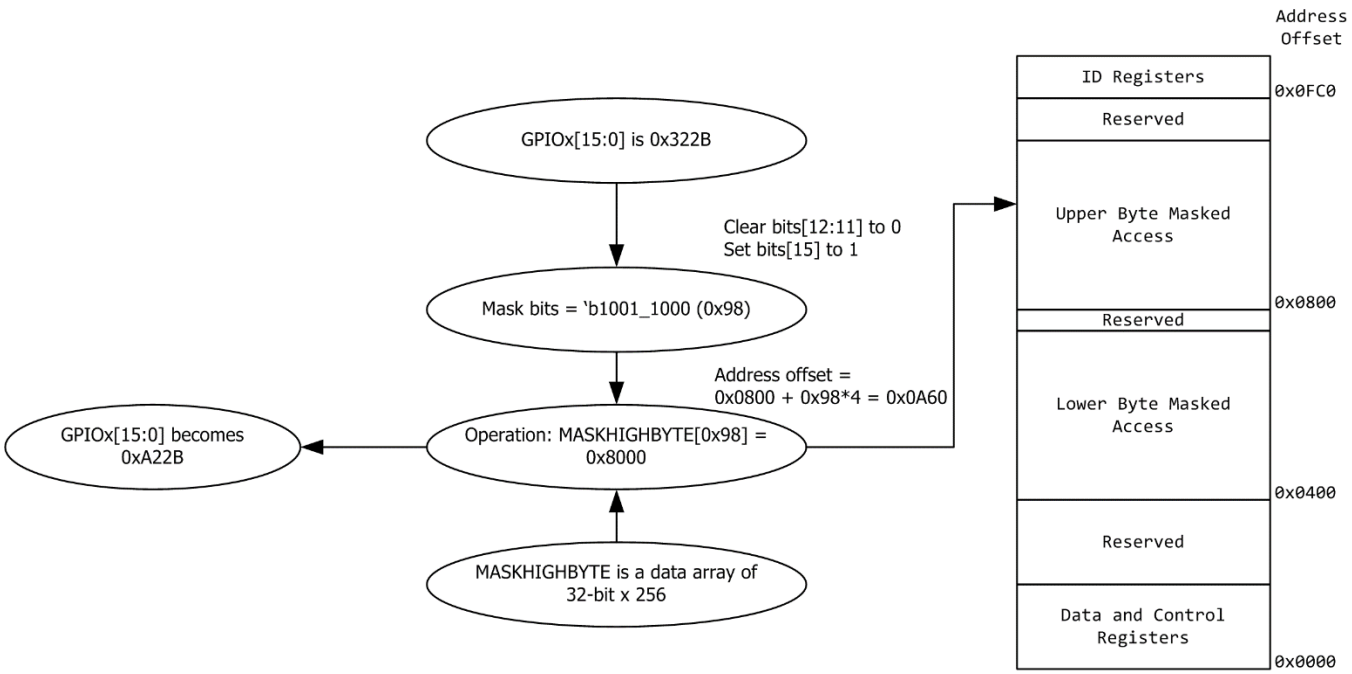
Description of the detailed of Mask access 1 as per the flow diagram

1	Data bits =	00xx xx11	//Clear bits[7:6] to 0 and set //bits[1:0] to 1
2	Mask bits =	1100 0011	//0xC3
3	MASKLOWBYTE =	0000 0011	//0x03 with AND operation
4	GPIOx[15:0] =	0011 0010 1110 1000	//Original input data is 0x32E8
5		0000 0011	//using MASKLOWBYTE from line 3
6		1110 1011	//OR operation as intermediate //results
7	Complement of Mask bits =	0011 1100	
8	Data bits =	00xx xx11	
9	Data bits after OR operation	0011 1111	//00xx xx11 -> 0011 1111
10		1110 1011	//Intermediate result from line 6
11	Data bits =	0011 1111	//Data bits after OR operation from //line 9
12	Data bits after AND operation =	0010 1011	//2B
13	New GPIOx =	0011 0010 0010 1011	//0x322B



MASKHIGHBYTE Operation (Mask access 2):

Same operation as MASKLOWBYTE (see preceding section)



13.6 Soft Reset Mode

Soft Reset Register **(SOFT_RESET)** allows control whether to perform or ignore a reset to the GPIOS. Setting bit [0] to 1 ignores reset for GPIO's set as input/output and setting bit [0] to 0 allows the GPIO's to reset to their default hard reset state.

13.7 Alternate Function Configuration

Some of the GPIO interfaces are multi-function (i.e., GPIO or other function such as PWM, UART, etc.). This allows sharing the GPIO's among different interfaces. The use of alternate function (i.e., not GPIO) is controlled via the ALTFUNCSET and ALTFUNCCLR registers. ALTFUNCSET sets the GPIO alternate function bits to 1 for alternate function or 0 for input/output. ALTFUNCCLR clears the alternate function bit.

13.8 Input Configuration

In order to configure a GPIO as an input, the Alternate Function register must be disabled by clearing the particular bit. OUTENABLESET register controls the input/output direction. A 0 in bits [15:0] makes the corresponding GPIO an input. As an input, the GPIO output buffer is disabled, Schmitt trigger input is activated, pull up/down resistors are activated by setting the particular bit in the PULL_ENABLE register and setting a 1 for pull-up, 0 for pull-down in the PULL_UP_DOWN register, and reading the DATA register provides the GPIO state.

13.9 Output Configuration

In order to configure a GPIO as an output, the Alternate Function register must be disabled by clearing the particular bit. OUTENABLESET register controls the input/output direction. A 0 in bits [15:0] makes the corresponding GPIO an output. As an output, the GPIO output buffer is enabled in push-pull mode--with a 0 sinking current and 1 sourcing current, "open-drain" is implemented in software by clearing the PULL_ENABLE register bit and floating the GPIO pin, connecting an external pull up/down resistor, and finally, configuring the GPIO as an output and driving the GPIO pin high or low.

13.10 Registers

Table 13.2 shows the GPIO banks with the following AHB memory map address.

Table 13.2: GPIO Bank AHB Addresses

GPIO BANKS	AHB ADDRESS
Bank 0	0x4002_0000 – 0x4002_0FFF
Bank 1	0x4002_1000 – 0x4002_1FFF
Bank 2	0x4002_2000 – 0x4002_2FFF

Table 13.3 shows the GPIO Port register offset addresses from GPIO bank AHB addresses.

Table 13.3: GPIO Registers

Register	Offset (Hex)
Port Data Register (DATA)	0x0000
Port Data Out Register (DATAOUT)	0x0004
Output Enable Set Register (OUTENABLESET)	0x0010
Output Enable Clear Register (OUTENABLECLR)	0x0014
Alternate Function Set Register (ALTFUNCSET)	0x0018
Alternate Function Clear Register (ALTFUNCCLR)	0x001C
Interrupt Enable Set Register (INTENSET)	0x0020
Interrupt Enable Clear Register (INTENCLR)	0x0024
Interrupt Type Set Register (INTTYPESET)	0x0028
Interrupt Type Clear Register (INTTYPECLR)	0x002C
Interrupt Polarity Set Register (INTPOLSET)	0x0030
Interrupt Polarity Clear Register (INTPOLCLR)	0x0034
Interrupt Status/Clear Register (INTSTATUS/INTCLEAR)	0x0038
Soft Reset Option Register (SOFT_RESET)	0x003C
Pull Enable Register (PULL_ENABLE)	0x0040
Pull Up/Down Register (PULL_UP_DOWN)	0x0044
Lower Bits Masked Register (LB_MASKED)	0x0400:0x07FC
Higher Bits Masked Register (HB_MASKED)	0x0400:0x07FC
Peripheral ID 4 Register (PID4)	0x0FD0
Peripheral ID 5 Register (PID5)	0x0FD4
Peripheral ID 6 Register (PID6)	0x0FD8
Peripheral ID 7 Register (PID7)	0x0FDC
Peripheral ID 0 Register (PID0)	0x0FE0
Peripheral ID 1 Register (PID1)	0x0FE4
Peripheral ID 2 Register (PID2)	0x0FE8
Peripheral ID 3 Register (PID3)	0x0FEC
Component ID 0 Register (CID0)	0x0FF0
Component ID 1 Register (CID1)	0x0FF4
Component ID 2 Register (CID2)	0x0FF8
Component ID 3 Register (CID3)	0x0FFC

13.10.1 DATA Register (DATA)

DATA																Offset = 0x0000
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA[15:0]															
W																
Reset	[UU…U]															

Table 13.4: Description of Data Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	DATA	[UU…U]	Read indicates the state of the DATA port pin n. Write sets the state of the DATAOUT port pin n. 0: Data='0' 1: Data='1'

13.10.2 Data Out Register (DATAOUT)

DATAOUT																Offset = 0x0004
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATAOUT[15:0]															
W																
Reset	[00…0]															

Table 13.5: Description of Data Out Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	DATAOUT	[00...0]	Read indicates the state of the DATAOUT port pin n. Write sets the state of the DATAOUT port pin n. 1: Data='1' 0: Data='0'

13.10.3 Output Enable Set Register (OUTENABLESET)

OUTENABLESET																**Offset = 0x0010**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU...U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OUTENABLESET[15:0]															
W																
Reset	[00...0]															

Table 13.37: Description of Output Enable Set Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	OUTENABLESET	[00...0]	Write 1: Sets the output enable bit 0: No action Read 1: Indicates the signal direction as output 0: Indicates the signal direction as input

13.10.4 Output Enable Clear Register (OUTENABLECLR)

OUTENABLECLR																Offset = 0x0014
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OUTENABLECLR[15:0]															
W																
Reset	[00…0]															

Table 13.38: Description of Output Enable Clear Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU…U]	
15-0	OUTENABLECLR	[00…0]	Write 1: Clears the output enable bit 0: No action Read 1: Indicates the signal direction as output 0: Indicates the signal direction as input

13.10.5 Alternate Function Set Register (ALTFUNCSET)

ALTFUNCSET																Offset = 0x0018
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ALTFUNCSET[15:0]															
W																
Reset	[00…0]															

Table 13.39: Description of Alternate Function Set Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	ALTFUNCSET	Bank 0 and 1: 0x0000 Bank 2: 0xFFFF	Write 1: Sets the ALTFUNC 0: No action Read 1: Indicates alternate function 0: Indicates GPIO

13.10.6 Alternate Function Clear Register (ALTFUNCCLR)

ALTFUNCCLR																**Offset = 0x001C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ALTFUNCCLR[15:0]															
W																
Reset	[00…0]															

Table 13.40: Description of Alternate Function Clear Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	ALTFUNCCLR	Bank 0 and 1: 0x0000 Bank 2: 0xFFFF	Write 1: Clears the ALTFUNC bit 0: No action Read 1: Indicates alternate function 0: Indicates GPIO

13.10.7 Interrupt Enable Set Register (INTENSET)

INTENSET																Offset = 0x0020
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTENSET[15:0]															
W																
Reset	[00…0]															

Table 13.41: Description of Interrupt Enable Set Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[UU…U]	
15-0	INTENSET	[00…0]	Write 1: Enable Interrupt 0: No action Read 1: Indicates Interrupt enabled 0: Indicates Interrupt disabled

13.10.8 Interrupt Enable Clear Register (INTENCLR)

INTENCLR																Offset = 0x0024
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTENCLR[15:0]															
W																
Reset	[00…0]															

Table 13.42: Description of Interrupt Enable Clear Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	INTENCLR	[00...0]	Write 1: Disables Interrupt 0: No action Read 1: Indicates Interrupt enabled 0: Indicates Interrupt disabled

13.10.9 Interrupt Type Set Register (INTTYPESET)

INTTYPESET																**Offset = 0x0028**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU...U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTTYPESET[15:0]															
W																
Reset	[00...0]															

Table 13.12: Description of Interrupt Type Set Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	INTTYPESET	[00...0]	Write 1: Enable Interrupt type 0: No action Read 1: For falling or rising edge 0: For low or high level

13.10.10 Interrupt Type Clear Register (INTTYPECLR)

INTTYPECLR																Offset = 0x002C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTTYPECLR[15:0]															
W																
Reset	[00…0]															

Table 13.13: Description of Interrupt Type Clear Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU…U]	
15-0	INTTYPECLR	[00…0]	Write 1: Disables Interrupt type 0: No action Read 1: For falling or rising edge 0: For low or high level

13.10.11 Interrupt Polarity Level and Edge Configuration Set Register (INTPOLSET)

INTPOLSET																Offset = 0x0030
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTPOLSET[15:0]															
W																
Reset	[00…0]															

Table 13.14: Description of Interrupt Polarity Level and Edge Configuration Set Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	INTPOLSET	[00...0]	Write 1: Set Interrupt polarity 0: No action Read 1: For high level or rising edge 0: For low lever or falling edge

13.10.12 Interrupt Polarity Level and Edge Configuration Clear Register (INTPOLCLR)

INTPOLCLR																**Offset = 0x0034**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU...U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTPOLCLR[15:0]															
W																
Reset	[00...0]															

Table 13.15: Description of Interrupt Polarity Level and Edge Configuration Clear Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	INTPOLCLR	[00...0]	Write 1: Clears Interrupt polarity 0: No action Read 1: For high level or rising edge 0: For low level or falling edge

13.10.13 Interrupt Request Clear Register (INTSTATUS, INTCLEAR)

INTSTATUS, INTCLEAR																Offset = 0x0038
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTSTATUS/INTCLEAR[15:0]															
W																
Reset	[00…0]															

Table 13.16: Description of Interrupt Request Clear Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	INTSTATUS/ INTCLEAR	[00…0]	Write 1: Clears Interrupt request 0: No action Read indicates the status of the IRQ port pin n.

13.10.14 Soft Reset Option Enable Register (SOFT_RESET)

SOFT_RESET																Offset = 0x003C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																SR
W																
Reset	[UU…U]															0

Table 13.17: Description of Soft Reset Option Enable Register

Bit Number(S)	Bit Name	Reset State	Description
31-1	RESERVED	[UU...U]	
0	SOFT_RESET (SR)	0	1: Ignores the soft reset on GPIO's 0: Sets GPIO's to default hard rest state

13.10.15 Pullup/Pulldown Enable Register (PULL_ENABLE)

PULL_ENABLE																Offset = 0x0040
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PULL_ENABLE[15:0]															
W																
Reset	[UU…U]															

Table 13.18: Description of Pullup/Pulldown Enable Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	PULL_ENABLE	[UU…U]	1: Pull up/down enabled 0: Pull up/down disabled

13.10.16 Pullup/Pulldown Configuration Register (PULL_UP_DOWN)

PULL_UP_DOWN																Offset = 0x0044
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU…U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PULL_UP_DOWN[15:0]															
W																
Reset	[00…0]															

Table 13.43: Description of Pullup/Pulldown Configuration Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[UU...U]	
15-0	PULL_UP_DOWN	[00...0]	1: Corresponding GPIO pin n is pull up 0: Corresponding GPIO pin n is pull down

Low Byte Access Mask Register (LB_MASKED)

LB_MASKED																**Offset = 0x0400: 0x07FC**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU...U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									LB_MASKED[7:0]							
W																
Reset	[UU...U]								[UU...U]							

Table 13.20: Description of Low Byte Access Mask Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[UU...U]	
7-0	LB_MASKED	[UU...U]	Lower eight bits masked access

13.10.17 Upper Byte Access Mask Register (UB_MASKED)

UB_MASKED																**Offset = 0x0800: 0x0BFC**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[UU...U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UB_MASKED[15:8]															
W																
Reset	[UU...U]								[UU...U]							

Table 13.12: Description of Upper Byte Access Masked Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[UU…U]	
15-8	UB_MASKED	[UU…U]	Upper eight bits masked access
7-0	RESERVED	[UU…U]	

14 Analog-to-Digital Converter (ADC)

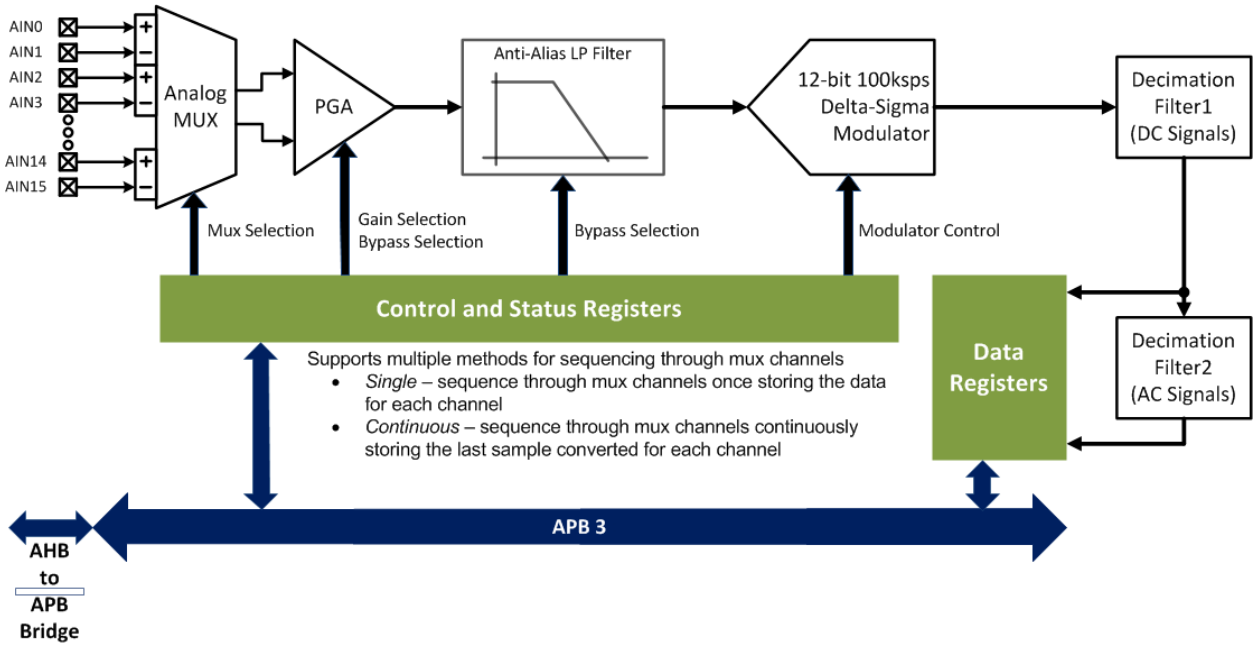


Figure 14.1: Simplified block diagram for the 12-Bit ADC Module

14.1 Overview

The Analog-to-Digital Converter (ADC) has 12-bits of resolution and can convert at up to 100KSPS rate. As shown in the above Figure 14.1, the Analog Multiplexer (AMUX) provides 16 analog inputs which can be configured single-ended, or up to 8 differential input pairs, or a combination. The Temperature Sensor is provided as an additional 17th selectable input to the ADC (not shown). The Programmable-Gain Amplifier (PGA) conditions the analog input signal with selectable gain of 0.5, 1, 2, 4, 8, or 16 V/V. The gain setting is independently programmed per analog input channel. A 3rd-order anti-alias low-pass analog filter prepares the signal for the sampling modulator. The Delta-Sigma Modulator (DSM) converts the analog output signal to a digital bitstream, which is decimated and filtered by one of two Digital Decimation Filters (DDF) to produce the 12-bit wide parallel data output.

The ADC supports a powerdown mode by ADC.SPB_CFG_0.ADC_EN=0.

The 12-bit ADC conversion results are included within a 32-bit data word that is available at the ADC Output Word Register (DATA). The format of this 32-bit word is shown in the ADC Output Word Register (DATA) description. A result must be read by the microcontroller core before the next conversion is completed, or the earlier data is overwritten. An Interrupt signal will be available at the completion of each conversion.

A trigger command as set by the ADC_TRIGGER bit field in the ADC.SPB_CFG_1 Configuration Register will start the conversion process. This software-controlled write can be applied at a deterministic period by using an Interrupt service routine for one of the M0+ timers. Note that since the ADC clock is divided from the system clock, it is possible to coordinate the timer count-down to generate an Interrupt at a constant relationship to the ADC clock.

The ADC Module supports either a SingleSweep or Continuous conversion mode through the ADC.SPB_CFG_0.ADC SingleSweep bit. In SingleSweep mode the ADC performs a conversion on all of the enable channels (AIN0 - AIN15) based on a user supplied trigger command, then waits for a subsequent trigger to start the next conversion sweep. In Continuous mode the ADC performs continuous conversions sequentially on all enabled channels. In both modes, all enabled single-ended channels will be sequenced through, followed by any enabled differential channels, followed by the temperature channel, if enabled. In Continuous mode, the first enabled channel is selected for the conversion immediately following conversion of the last enabled channel.

In both modes, if more than one channel is enabled, the ADC will sequence through all enabled channels, with a user-controlled delay between when the next channel is enabled and the start of filter accumulation, using the ADC.SEQ_CTRL.ADC_SEQDLY setting. *Note: If a differential and single-ended conversion is enabled for a given channel, the differential channel data will overwrite the ADC output register.*

The following gives details of possible conversion modes:

- **With ADC_SINGLESWEEP = 1:** single-sequence sweep of enabled channels, manual software trigger required for next sweep
  - — Enable one or more channels
  - — ADC Interrupt available after sweep of conversions
  - — software read of output registers (prior to completion of first channel in next sweep)
  - — software write to re-trigger
- **With ADC_SINGLESWEEP = 0:** auto-sequence of enabled channels, continuously
  - — Enable one or more channels
  - — ADC Interrupt available after each sweep of conversions
  - — software read of output registers sometime after each conversion sweep
  - — (hardware will re-trigger at each conversion-complete)
- **With ADC_SINGLESWEEP = 1 (Software-controlled sample rate):** Software trigger required to increment to next channel (this mode allows a precise sample rate to be determined by user, while the OSR-defined conversion may be slightly faster.)
  - — select SingleSweep = 1, enable one channel
  - — software read of output after each conversion
  - — software write to disable channel, enable next channel after each conversion
  - — software write to trigger after each conversion

- **With ADC_SINGLESWEEP = 1 (Software controlled multi-sampled channel):** multi-conversion per channel
  - — select SingleSweep = 1, enable one channel
  - — software read of output after each conversion
  - — software write to trigger after each conversion
    - software filtering of multiple conversions
  - — software write to disable first channel / enable next channel after same-channel filtering

14.2 ADC Filter Selection

Two ADC Digital Decimation Filter (DDF) types are available and are selectable independent of the SingleSweep setting. DDF1 is a 3rd-order Cascade of Integrators (COI) filter. DDF2 is a 4rd-order Sinc filter, selected using the DDF2 bit of each channels CHAN_CFG register. DDF2 is not expected to be used in the Continuous mode, since it is a multi-sample filter with storage of 3 previous conversions.

14.3 COI Filter

The DDF1 filter is a 3rd-order Cascade of Integrators (COI) filter, which is enabled for an ADC conversion when the currently selected input channel has the corresponding ADC.CHAN_CFG[n], bit DDF2 cleared. DDF1 is an incremental filter, which is reset at the start of each conversion period. The use of this incremental filter allows more than one of the analog inputs to be enabled, where each conversion is independent from previous conversions. When using the incremental filter, DDF1, each channel input signal is expected to be a near-DC signal.

Note: the settling time of the Anti-Alias Filter may require a delay set in the ADC.SEQ_CTRL register before switching to the next enabled channel.

14.4 Sinc4 Filter

The DDF2 user-selectable filter is a 4rd-order sinc filter, enabled for all of the ADC conversions when the DDF2_EN_CLK bit of ADC.SPB_CFG_0 register is set to ‘1’, and the currently selected input channel has the corresponding ADC.CHAN_CFG[n] bit DDF2 set to ‘1’. DDF2 is a 4-stage IIR filter, which is not reset at the start of each conversion period. The use of more than one channel enable is not expected, since each filter output word is dependent on the previous four conversions. The DDF2 mode supports an analog input signal which contains frequency content.

14.5 Output Word Range

The ADC Module supports a signed output, therefore when operating on a differential input signal the output range is from 0x0800 to 0x07FF as indicated in Table 14.1.

Within the sinc4 and coi3 filters, signed arithmetic is used. Thus a 50% ones content from the modulator will produce a 0 count, since as many +1's and -1's are added together. This is a midrange voltage, as shown here:

Table 14.1: ADC Module Output Range for a Differential Input

Vin	AIN[0], Diff @Vcm=0.75V	AIN[1], Diff @Vcm=0.75V	Signed Output(ideal)
max	1.5V	0V	7FF
max-1LSB	1.4996V	366uV	7FE
...			...
max/2	0.75	0.75	000
max/2-1LSB	0.7496V	0.7504V	FFF
...			...
min +1LSB	366uV	1.4996V	801
min	0V	1.5V	800

Furthermore, the input voltage range (Vin) for max and min on in input pairs configured for differential input are different than a Single-Ended mode input. When configured for singled-ended operation, the input voltage range is from 0 – 1.5 V with the digital output ranging from 0x0800 to 0x7FF as shown in Table 14.2.

Table 14.2: ADC Module Output Range for a Single-Ended Input.

Vin	AIN[0], SE	Signed Output(ideal)
max	1.5V	7FF
max-1LSB	1.4996V	7FE
...		...
max/2	0.75V	000
max/2-1LSB	0.7496V	FFF
...		...
min +1LSB	366 uV	801
min	0V	800

14.6 ADC Module Functionality

14.6.1 Interrupt Functionality

The ADC Interrupt signal is a combined Interrupt signal which indicates the end of a conversion, as well as the occurrence of error conditions. A global enable bit, ADC.INTR_EN is located in ADC.SPB_CFG_0. Each source of Interrupt within the ADC peripheral can be uniquely enabled or disabled using the five "IEN"-labeled control bits in the ADC.SPB_CFG_0.

End-of-Conversion Interrupt: When ADC.SPB_CFG_0.CONV_COMPL_IEN is '1', the end of a sweep of conversions (in either single- or auto-sweep mode) will send the ADC Interrupt high. The Interrupt stays active until a read occurs of any of the ADC.DATA registers or the ADC.INT_STATUS register.

Error Status Interrupts: When bits 27 to 30 of ADC.SPB_CFG_0 contain a '1' to enable a particular ADC error Interrupt, the ADC Interrupt signal is driven high at the end of the sweep of conversions (in either singlesweep- or continuous mode) which created the error condition. The Interrupt stays active until a read occurs of any of the ADC.DATA registers or the ADC.INT_STATUS register.

14.6.2 System Clock Speed Requirements

For proper ADC operation, the minimum allowed System clock (PCLK) is affected by the OSR, ADC_CLK frequency, and number of enabled channels. If data from all enabled channels is expected to be read out within one conversion time (the first conversion of the next sequence), the following relationship must be observed so as to not over-write the first channel's data of a sequence:

$$\#Enabled\ Channels * nPCLKs\ per\ readchannel * PCLK_{period}$$

$$\leq (OSR * ADC\ CLK_{period}) + (25 * ADC\_CLK_{period} * ADC\_SEQDLY)$$

The following is an example calculation for determining the minimum PCLK frequency required:

With 18 channels enabled, an estimated 4 PCLKs/read, OSR=127, ADC_CLK = 12.5MHz, and ADC_SEQDLY = 0:

18*4*PCLK period <= 127 * 80ns + (25 * 80ns * 0)
PCLK period <= 141.1 ns
PCLK frequency >= 7.1 MHz

Note: this calculations show 4 PCLKs per channel read, which is as quick as possible on the APB bus. Any other APB bus activity, or C-code overhead, will slow down this reading and demand a faster PCLK.

14.6.3 ADC_SEQDLY Requirements

The ADC_SEQDLY setting controls the amount of time the ADC delays between conversions. To determine the minimum value for ADC_SEQDLY, the following must be true:

$$ADC\_SEQDLY_{min} \geq ADC\_SEQDLY_{min\ Input\ Settling\ Time}$$
$$ADC\_SEQDLY_{min} \geq ADC\_SEQDLY_{min\ Conversion\ Storage\ Time}$$

The first of these two values, $ADC\_SEQDLY_{min\ Input\ Settling\ Time}$, accounts for the Input Settling Time, which is the amount of time between the analog mux changing to the next enabled channel and that channel being converted. The following equation calculates the Input Settling Time:

$$Input\ Settling\ Time = ADC\_SEQDLY_{min\ Input\ Settling\ Time} * ADC\_CLK\ Period * 25$$

Re-written, this equation is:

$$ADC\_SEQDLY_{min\ Input\ Settling\ Time} = \frac{Input\ Settling\ Time}{ADC\_CLK\ period * 25}$$

The maximum input settling time listed in the datasheet is 28µs. To ensure that the ADC_SEQDLY accounts for this settling time, using an ADC_CLK setting of 12.5MHz (or 0.08µs):

$$ADC\_SEQDLY_{min\ Input\ Settling\ Time} = \frac{28\mu s}{0.08\mu s * 25}$$

$$ADC\_SEQDLY_{min\ Input\ Settling\ Time} = 14\ (decimal)$$

The second of the two values, $ADC\_SEQDLY_{min\ Conversion\ Storage\ Time}$, accounts for the amount of time required for the ADC to store the conversion data. This data is transferred from the ADC_CLK domain into the PCLK domain and requires 4 PCLK periods minimum to complete in the subsequent conversion period. The equation is more complex, as both CLKs and the OSR value affect the calculation:

$$ADC\_SEQDLY_{min\ Conversion\ Storage\ Time} = \frac{\left(4* \left(\frac{ADC\_CLK\ frequency}{PCLK\ frequency}\right) - OSR\right)}{25}$$

To continue the above example, assuming a PCLK frequency of 1MHz, and an OSR of 101:

$$ADC\_SEQDLY_{min\ Conversion\ Storage\ Time} = \frac{\left(4* \left(\frac{12.5MHz}{1MHz}\right) - 101\right)}{25}$$

$$ADC\_SEQDLY_{min\ Conversion\ Storage\ Time} = 0\ (decimal)$$ **(Negative numbers round up to 0)**

With these two calculations complete, the true minimum value for ADC_SEQDLY is:

$$ADC\_SEQDLY_{min} = 14\ (decimal)$$

The OSR values listed in Table 14.3 and Table 14.4 are the recommended OSR settings found in ADC.TIM_CTRL register – Optimal OSR Settings, for the COI3 filter and SINC4 filter.

Table 14.3: Minimum ADC_SEQDLY Setting, COI3 Filter

(highlighted where value is above 10)

PCLK (MHz)	OSR for COI3 Filter						
	63	80	101	127	160	202	255
0.032	60	60	59	58	57	55	53
0.064	29	29	28	27	25	24	22
0.128	14	13	12	11	10	8	6
0.256	6	5	4	3	2	0	0
0.512	2	1	0	0	0	0	0
0.750	1	0	0	0	0	0	0
1.024	0	0	0	0	0	0	0
2.048	0	0	0	0	0	0	0
4.096	0	0	0	0	0	0	0
32.768	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0

Table 14.4: Minimum ADC_SEQDLY Setting, SINC4 Filter

(highlighted where value is above 10)

PCLK (MHz)	OSR for SINC4 Filter								
	58	69	82	98	116	138	164	195	232
0.032	61	60	60	59	58	57	56	55	54
0.064	29	29	28	28	27	26	25	24	22
0.128	14	13	13	12	11	11	10	8	7
0.256	6	6	5	4	4	3	2	1	0
0.512	2	2	1	0	0	0	0	0	0
0.750	1	0	0	0	0	0	0	0	0
1.024	0	0	0	0	0	0	0	0	0

14.6.4 Calculating Channel Conversion Time

To calculate the time for a single channel to convert, use following formula:

Total Conversion Time= Enabled Channels(Setup Time+Conversion Time)*

*Setup Time=Input Settling Time= ADC_SEQDLY*ADC_CLK Period*25*

*Conversion Time=ADC_CLK Period *OSR*

Total Conversion Time=Enabled Channels((ADC_SEQDLY *ADC_CLK Period *25) +(ADC_CLK Period *OSR))*

For an ADC_SEQDLY of 14, ADC_CLK of 12.5MHz (0.08µs period), and OSR of 101, the Total Conversion Time for one channel is equal to:

$$Total\ Conversion\ Time_{One\ Channel} = 1*((14*0.08\mu s*25)+(0.08\mu s*101))$$

$$Total\ Conversion\ Time_{One\ Channel} = 28.08\mu s$$

14.6.5 Conversion Error Flags and Their Meanings

Each channel has its own ADC->DATA register, which means each channel has its own set of error flags, and the configuration of each channel affects what flags are applicable. Knowing what each error flag means is important for ensuring the ADC is operating as intended. The below paragraphs describe what is actually happening when one or more of these bits are set.

- DATA_ERROR
- COI_OVER
- SINC4_OVER
- DSM_OVL_FLAG
- TRIG_UNDER

The **DATA_ERROR** flag (bit 14) is a combinational OR of all the other error flags, including the reserved error flag located in bit 15. Use the DATA_ERROR flag to check if any of the other error flags are set.

The **DATA_VALUE_STALE** flag (bit 15, now reserved, see Appendix A: Errata) sets when either a conversion has not completed for this channel since a reset state, or a conversion has not completed since the last read of this specific DATA register. Due to errata concerning the ADC_CONV_COMPL bit, a bit the DATA_VALUE_STALE bit depended upon, this bit is reserved.

The **COI_OVER** flag (bit 23) and the **SINC4_OVER** flag (bit 24) perform similarly. One of these two flags, depending on which of the two digital filters the channel in question uses, will set if the data coming out of the Delta-Sigma Modulator (DSM) has:

- Exceeded the maximum input range. The number of 1's sent from the DSM to the filter indicate the input voltage has exceeded the maximum input range.
- Fallen below the minimum input range. The number of 0's sent from the DSM to the filter indicate the input voltage has fallen below the minimum input range.

Note: Users can pick an OSR value outside of the recommended table, but this will change the full-scale data range. The full-scale data range is a value used by the COI_OVER or SINC4_OVER flags to determine if an input value is out of range based upon a channel's gain setting. Using a non-recommended OSR value will set the full-scale data range to the maximum value of 0x7FF, regardless of the PGA gain.

[See errata note for the COI3 filter: ADC COI3_OVER Low Voltage False Flag]

The **DSM_OVL_FLAG** flag (bit 25) is a copy of the ADC->STAB_CTRL.DSM_OVL_FLAG. This bit sets when the Delta-Sigma Modulator (DSM) output bitstream repeats the same value too many times (determined by STAB_CTRL.DSM_OVL_CNT), indicating the DSM is overloaded. The overloaded DSM will automatically reset without affecting either of the digital filters (COI3 or SINC4), or the data read from those filters. STAB_CTRL.DSM_OVL_RST determines the number of ADC_CLK cycles taken to reset. This feature has been included as a fail-safe backup. As the DATA registers' bit copies the STAB_CTRL bit, clearing the ADC->STAB_CTRL version is the only way to clear the DATA bit.

The **TRIG_UNDER** flag (bit 26) is set to indicate a trigger happened mid-conversion. Note that triggers that happen mid-conversion cause a new conversion to start in addition to setting this bit. The start of a new conversion (conversion caused by a new trigger, not a mid-conversion trigger) clears this bit.

14.7 ADC Register Details

All registers have read & write access to the defined fields, unless otherwise indicated.

Table 14.5: ADC Registers

Register	Offset (Hex)
SPB Configuration Register 0 (SPB_CFG_0)	0x00
SPB Configuration Register 1 (SPB_CFG_1)	0x04
Single-Ended Channel Configuration Register (SECHAN_CFG)	0x10 – 0x4C
Differential Channel Configuration Register (DIFFCHAN_CFG)	0x50 – 0x6C
Temperature Channel Configuration Register (TEMPCHAN_CFG)	0x70
Timing Control Register (TIM_CTRL)	0x74
Sequence Control Register (SEQ_CTRL)	0x78
DSM Digital Stability Control (STAB_CTRL)	0x84
Interrupt Status Register (INT_STATUS)	0x8C
Data Output Word Register (DATA)	0x90 – 0xD4

All registers have read & write access to the defined fields, unless otherwise indicated.

The ADC_TRIGGER bit of SPB_CFG_1 is used to start a conversion, whether it is in ADC_SINGLESWEEP mode, or the continuous-sweep mode (SPB_CFG_0.ADC_SINGLESWEEP = 0). Prior to setting the ADC_TRIGGER bit, all configuration settings should be entered to create a valid setup for the conversion(s). The figure below outlines the steps need for proper configuration of the ADC for a conversion.

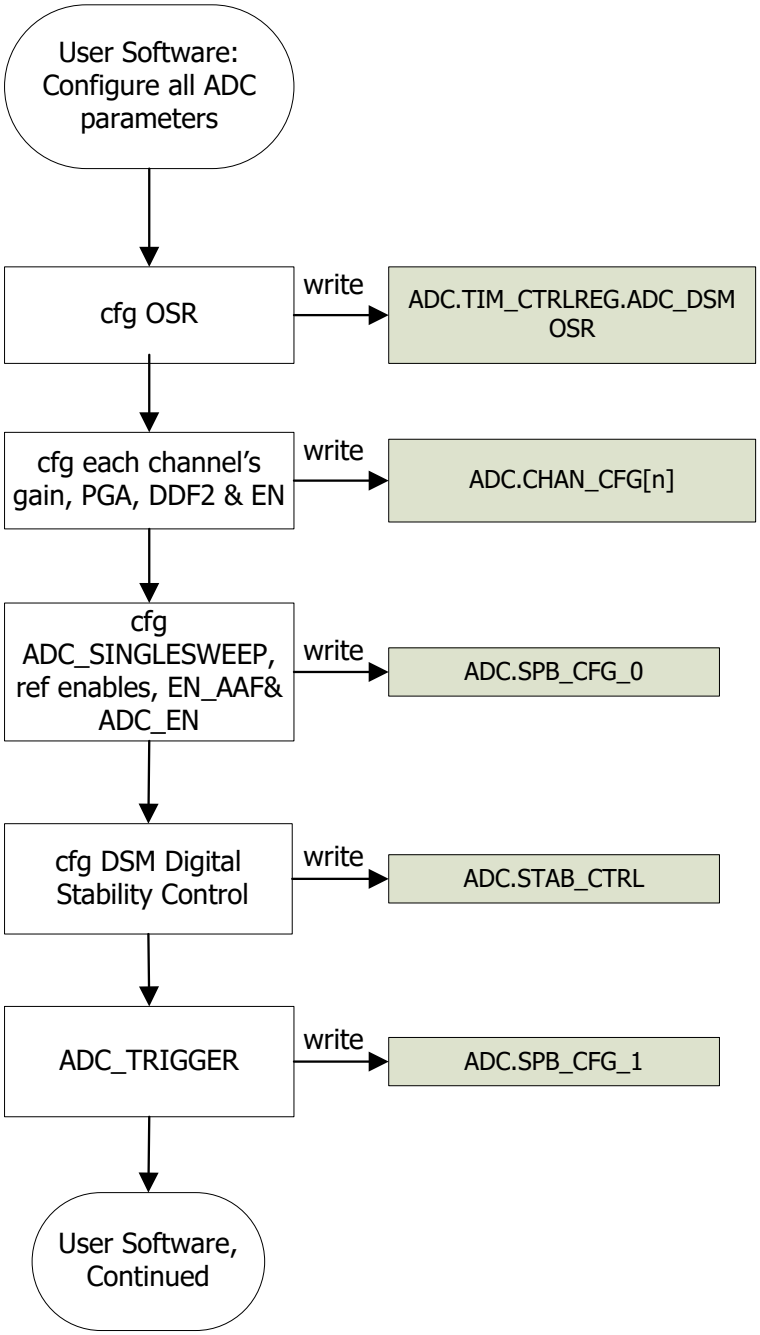


Figure 14.2: ADC configuration and trigger

Furthermore, when a change to the ADC configuration is required, a sequence of operations should be followed to ensure no error status bits are set and the next conversion sequence starts properly. Therefore, the following sequences will properly configure the ADC for various scenarios:

Scenario #1: Startup of Continuous Conversions

1. Configure the ADC options as listed in Figure 14.2, with ADC_SINGLESWEEP = 0.
2. Delay for an Input Settling Time.
3. Write an ADC_TRIGGER as listed in Figure 14.2.
4. Read the highest channel interrupt bit of ADC.INT_STATUS until it and the CONV_COMPL _COMB bit reads a '1'.
5. Read the proper ADC.DATA words (0x90 through 0xD4) per the enabled channels.
6. (Conversions automatically continue)
7. Return to step 3.

Scenario #2: Startup of Single-Sweep Conversions

1. Configure the ADC options as listed in Figure 14.2, with ADC_SINGLESWEEP = 1.
2. Delay for an Input Settling Time.
3. Write an ADC_TRIGGER as listed in Figure 14.2.
4. Read the highest channel interrupt bit of ADC.INT_STATUS until it and the CONV_COMPL _COMB bit reads a '1'.
5. Read the proper ADC.DATA words (0x90 through 0xD4) per the enabled channels.
6. Return to step 3 (Trigger).

Scenario #3: Reconfiguration of Continuous Conversions

7. During a continuous conversion mode operation, at some time following step 5 of Scenario #1, write a '1' to the ADC_RST_CCONV bit to halt the conversion sequence at the end of the current sequence.
8. Read the highest channel interrupt bit of ADC.INT_STATUS until it and the CONV_COMPL _COMB bit reads a '1'.
9. Write to the desired configuration registers to reconfigure the ADC settings.
10. If any channel enables have been altered, wait a period equal to ADC_SEQDLY for the input mux to settle.
11. Write an ADC_TRIGGER.
12. Read the highest channel interrupt bit of ADC.INT_STATUS until it and the CONV_COMPL _COMB bit reads a '1'.
13. Read the proper ADC.DATA words (0x90 through 0xD4) per the enabled channels.
14. Return to step 6 (Conversion Complete polling).

Scenario #4: Reconfiguration of Single-Sweep Conversions

1. Read the highest channel interrupt bit of ADC.INT_STATUS until it and the CONV_COMPL _COMB bit reads a '1'.
2. Write to the desired configuration registers to reconfigure the ADC settings.
3. If any channel enables have been altered, wait a period equal to ADC_SEQDLY for the input mux to settle.
4. Write an ADC_TRIGGER.
5. Read the highest channel interrupt bit of ADC.INT_STATUS until it and the CONV_COMPL _COMB bit reads a '1'.
6. Read the proper ADC.DATA words (0x90 through 0xD4) per the enabled channels.
7. Return to step 4 (Trigger)

14.7.1 SPB Configuration Register 0 (SPB_CFG_0)

Power down/enables. The independent block power-downs are needed for evaluation. A setting of '0' at any of these enable bits will force the power-down of that circuit.

SPB_CFG_0																**Offset = 0x0000**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CC	TU	DO	SO	CO			AI								
Reset	0	0	0	0	0	00		0	[00…0]							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ODB[1:0]		AS			EC	EB		ERC		ERP	DC	EA	ED	EP	AE
Reset	0	0	0	00		0	0	0	0	0	0	0	0	0	0	0

Table 14.6: Description of SPB Configuration Register 0

Bit Number(s)	Bit Name	Reset State	Description
31	CONV_COMPL _IEN (CC)	0	1: Enable the Interrupt(1,2) 0: Disable the ADC_CONV_COMPL status bit (of ADC.SPB_CFG_1) from creating an Interrupt **NOTE:** See ADC External vs Internal Oscillator CONV_COMPL Conflict
30	TRIG_UNDER _IEN (TU)	0	1: Enable the Interrupt(1,2) 0: Disable the TRIG_UNDER status bit (of ADC.DATA) from creating an Interrupt
29	DSM_OVL_IEN (DO)	0	1: Enable the Interrupt(1,2) 0: Disable the DSM_OVL_FLAG status bit (of ADC.DATA) from creating an Interrupt
28	SINC4_OVER _IEN (SO)	0	1: Enable the Interrupt(1,2) 0: Disable the SINC4_OVER status bit (of ADC.DATA) from creating an Interrupt
27	COI_OVER_IEN (CO)	0	1: Enable the Interrupt(1,2) 0: Disable the COI_OVER status bit (of ADC.DATA) from creating an Interrupt
26-25	RESERVED	00	
24	ADC_INTR_EN (AI)	0	1: Enable the Interrupts, when bits 31:27 are enabled 0: Disable the combined ADC_INTR Interrupt signal. Also disables the Interrupt enable bits 31:27
23-16	RESERVED	[00…0]	
15-14	ODB[1:0]	0	Count of modulator clocks to delay the start of a conversion after the modulator is reset Default = 00 (One clock delay)
13	ADC _SINGLESWEEP (AS)	0	1: Single Sweep Mode. One sweep of all enable channels, then the user must provide trigger to begin new sweep through enabled channels 0: Continuous Mode. Automatic re-trigger of sweep through all enabled channels

Bit Number(s)	Bit Name	Reset State	Description
12	RESERVED	0	Always write as '0'
11	RESERVED	0	
10	EN_CLKGEN (EC)	0	1: Non-overlapping clock phase generator enabled 0: Power-down the non-overlapping clock phrase generator
9	EN_BIASGEN (EB)	0	1: Current bias/reference generator enabled 0: Continuous mode. Automatic re-trigger of sweep through all enabled channels
8	RESERVED	0	
7	EN_REFC (ERC)	0	1: Enable the current bias/reference generator 0: Power-down the current bias/reference generator
6	RESERVED	0	
5	EN_REFP (ERP)	0	1: Enable the positive reference buffer 0: Power-down the positive reference buffer
4	DDF2_CLK_EN (DC)	0	1: Enable the clocking of the DDF2, constant logic 1 0: No action
3	EN_AAF (EA)	0	1: Enable the anti-alias filter 0: Power-down the anti-alias filter
2	EN_DSM (ED)	0	1: Enable the Delta-Sigma-Modulator. This bit does not control the clocking or enabling of the digital filters to allow test data through the filters. ADC_EN can be used to shut down the digital filter clocks 0: Power down the Delta-Sigma-Modulator
1	EN_PGA (EP)	0	1: Enable the programmable gain amplifier 0: Power down the programmable gain amplifier
0	ADC_EN (AE)	0	1: Enable the blocks with the ADC 0: Assert the power-down for circuit blocks within the ADC independent of any other enable bit of this register. This bit set to logic 0 will disable the COI3 and SINC4 clock signals and assert the reset to the analog macro. The ADC_CLK to the ADC Module is also disabled.

Note 1: active only when the ADC_INTR_EN bit is set to logic 1

Note 2: The Interrupt enable bits do not affect the ADC.INT_STATUS or ADC.DATA bits, where conversion complete and error status can always be read.

14.7.2 SPB Configuration Register 1 (ADC.SPB_CFG_1)

These following settings are operational settings and control commands. All write-able bits are momentary active high, to allow a single write to create a momentary command, instead of read-modify-write.

SPB_CFG_1																**Offset = 0x0004**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	AR															
Reset	0	[00…0]														

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							RF									
W									RS							TR
Reset	[00…0]						0	0	0	[00…0]						0

Table 14.7: Description of SPB Configuration Register 1

Bit Number(s)	Bit Name	Reset State	Description
31	ADC_REGDEF (AR)	[UU...U]	1: Return all ADC register values to default 0: No action (Write only, reads back as '0')
30-10	RESERVED	[00…0]	
9	ADC _READYFLAG (RF)	0	1: A minimum of necessary settings are complete, e.g., the ADC modulator is enabled, and at least one channel is enabled 0: No action (Read only, writes are ignored)
8	RESERVED	0	NOTE: See ADC External vs Internal Oscillator CONV_COMPL Conflict
7	ADC_RST _CCONV (RS)	0	1: Cancel continuous Auto-Sequence convert mode. If a current sequence is in progress, it will finish the sweep through all the enabled channels before halting to wait for a trigger 0: No action
6-1	RESERVED	[00…0]	
0	ADC_TRIGGER (TR)	0	1: Triggers ADC to begin conversion 0: No action

Note 1: when bit 8 is set and the last channel enabled wraps around, the predefined delay for setup time is reduced and causes an inaccurate ADC conversion reading in the first enabled channel, see Appendix A: Errata for work around.

14.7.3 Single-Ended Channel Configuration Register (SECHAN_CFG[0] to SECHAN_CFG[15])

Each of the _CFG[0] to _CFG[15] define the registers for each of the sixteen single-ended channels ADC.SECHAN_CFG_REG registers _0 to _15.

An input which is enabled is thereby selected for conversion by single or continuous mode. Since each channel has unique enable bits, the system can be configured with combinations of differential, single-ended, and disabled channels.

Whenever an even-numbered SE channel (0-14) is enabled and currently selected through the 16:1 mux for the ADC conversion, the digital hardware will assert the ONG to force a ground on the negative input of the PGA. Likewise, for the odd-numbered channels (1-15) the OPG will be asserted to force a ground on the positive PGA input. Note that there is no internal checking that an enabled single-ended channel is also enabled for differential.

All odd-numbered SE channels (1-15) will require an inversion of the filter output word in the digital logic. This accommodates the VIN signal connecting to the negative input of the PGA.

All single-ended channels in COI3 filter mode will accommodate the full range of the 12-bit output word by shifting right one bit the selected bits of the 32-bit counter and inverting the MSbit.

SECHAN_CFGx																**Offset = 0x0010 – 0x004C (Increments of 0x04)**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R										GAIN[2:0]			DD			EN
W																
Reset	[00…0]									001			0	00		0

Table 14.8: Description of Single-Ended Channel Configuration Registers

Bit Number(s)	Bit Name	Reset State	Description
31-7	RESERVED	[00…0]	
6-4	GAIN[2:0]	001	Selects the PGA gain for the appropriate channel 000: 0.5 V/V 001: 1 V/V 010: 2 V/V 011: 4 V/V 100: 8 V/V 101, 110, 111: 16 V/V
3	DDF2 (DD)	0	1: Select the SINC4 filter (DDF2). Can be unique for each Single-Ended channel 0 to 15, should only be enabled on one channel at a time. 0: Select the COI3 filter (DDF1)
2-1	RESERVED	00	
0	EN	0	1: Enable channel for single-ended operation 0: No action

14.7.4 Differential Channel Configuration Register (DIFFCHAN_CFG[0] to DIFFCHAN_CFG[7])

Bit definition for each of 8 ADC_DIFFCHAN_CFG registers, _0 to _7

Differential pairs are restricted by definition to neighboring pins 1&2, 3&4, etc. The 8 registers control these pairs:

ADC.DIFFCHAN_CFG_0 (offset 50) -> 1&2
ADC.DIFFCHAN_CFG_1 (offset 54) -> 3&4
ADC.DIFFCHAN_CFG_2 (offset 58) -> 5&6
ADC.DIFFCHAN_CFG_3 (offset 5c) -> 7&8
ADC.DIFFCHAN_CFG_4 (offset 60) -> 9&10
ADC.DIFFCHAN_CFG_5 (offset 64) -> 11&12
ADC.DIFFCHAN_CFG_6 (offset 68) -> 13&14
ADC.DIFFCHAN_CFG_7 (offset 6c) -> 15&16

DIFFCHAN_CFGx																**Offset = 0x0050-0x006C (Increments of 0x04)**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R										GAIN[2:0]			DD			EN
W																
Reset	[00…0]									001			0	00		0

Table 14.9: Description of the Differential Channel Configuration Registers

Bit Number(s)	Bit Name	Reset State	Description
31-7	RESERVED	[00…0]	
6-4	GAIN[2:0]	001	Selects the PGA gain for the appropriate channel pair 000: 0.5V/V 001: 1 V/V 010: 2 V/V 011: 4 V/V 100: 8 V/V 101, 110, 111: 16 V/V
3	DDF2	0	1: Select the SINC4 filter (DDF2). Can be unique for each differential pair channel 0 to 7, but only one differential channel should enable this filter 0: Select the COI3 filter (DDF1)
2-1	RESERVED	00	
0	EN	0	1: Enable channel for differential input operation 0: No action

14.7.5 Temperature Channel Configuration Register (TEMPCHAN_CFG)

The conversion order of the temperature channel signals is: single-ended positive terminal, single-ended negative terminal, differential terminals (if all three are enabled).

TEMPCHAN_CFG																**Offset = 0x0070**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
R										GAIN[2:0]			DD			EN
W																
Reset	[00…0]									001			0	00		0

Table 14.10: Description of the Temperature Channel Configuration Register

Bit Number(s)	Bit Name	Reset State	Description
31-7	RESERVED	[00…0]	
6-4	GAIN[2:0]	001	Selects the PGA gain for the appropriate channel 000: 0.5 V/V 001: 1 V/V 010: 2 V/V 011: 4 V/V 100: 8 V/V 101, 110, 111: 16 V/V
3	DDF2 (DD)	0	Digital Filter Selection applies to all enabled terminals. (COI3 recommended) 1: Selects the SINC4 filter (DDF2). 0: Selects the COI3 filter (DDF1)
2-1	RESERVED	00	
0	DIFF_EN (EN)	0	1: Enable channel for differential input operation 0: No action

14.7.6 Timing Control Register (TIM_CTRL)

There is flexibility in the programmable timing control values. This will allow the converter to be applied optimally and the circuit design evaluated thoroughly. The discrete-time ADC circuit design will operate correctly at reduced clock rates. No access to the Timing Control Register is necessary to operate the ADC as nominally designed.

TIM_CTRL																**Offset = 0x0074**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							OD		ADC_DSMOSR[7:0]							
W																
Reset	[00…0]						00		0110_0100							

Table 14.11: Description of the Timing Control Register

<table>
<tr><th>Bit Number(S)</th><th>Bit Name</th><th>Reset State</th><th>Description</th></tr>
<tr><td>31-10</td><td>RESERVED</td><td>[00…0]</td><td></td></tr>
<tr><td>9-8</td><td>ADC _OSCDIV[1:0] (OD)</td><td>00</td><td>ADC oscillator divider to set the modulator clock frequency (ADC_CLK) from the 50MHz internal oscillator
00: Divider of 4 (12.5 MHz)
01: Divider of 8 (6.25 MHz)
10: Divider of 16 (3.125 MHz)
11: Divider of 2 (25 MHz) – not guaranteed over the full temperature or voltage range</td></tr>
<tr><td>7-0</td><td>ADC _DSMOSR[7:0]</td><td>0110 _0100</td><td>Allows for setting the number of modulator samples to produce a conversion, smaller OSR is used for less resolution and faster convert rate while a larger OSR is used for more accuracy and longer convert time. The analog input Gaussian noise (thermal and quantization noise) is reduced by a higher OSR value. The ‘gain’ of the filter changes with the OSR/N value, but is accounted for internally so as to provide a 12-bit output word.
Default = “0110 0100” = 100 decimal = 8 us at a 12.5 MHz ADC_CLK clock.
Range = 0 to 255 modulator clocks. Equal to 0us to 20.32us at a 12.5MHz ADC_CLK clock.
The settings for the DSMOSR listed in the table below identify the optimal settings to use which will provide 0x07FF data output word with a full-scale analog input value. Settings below these will not provide 0x07FF for a full-scale analog input, but a value lower than 0x07FF.
Optimal OSR Settings
<table>
<tr><th>Optimal COI3 OSR Settings (decimal values)</th><th>Optimal SINC4 OSR Settings (decimal values)</th></tr>
<tr><td>63</td><td>58</td></tr>
<tr><td>80</td><td>69</td></tr>
<tr><td>101</td><td>82</td></tr>
<tr><td>127</td><td>98</td></tr>
<tr><td>160</td><td>116</td></tr>
<tr><td>202</td><td>138</td></tr>
<tr><td>255</td><td>164</td></tr>
<tr><td></td><td>195</td></tr>
<tr><td></td><td>232</td></tr>
</table>
</td></tr>
</table>

14.7.7 Sequence Control Register (SEQ_CTRL)

In both SingleSweep and Continuous, if more than one channel is enabled, the ADC will sequence through all enabled channels, with a user-controlled delay between the next channel enable and the start of filter accumulation. The ADC.SEQ_CTRL register sets the delay time (in us).

SEQ_CTRL																**Offset = 0x0078**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R											ADC_SEQDLY[5:0]					
W																
Reset	[00…0]										[00…0]					

Table 14.12: Description of the Sequence Control Register

Bit Number(s)	Bit Name	Reset State	Description
31-6	RESERVED	[00…0]	
5-0	ADC _SEQDLY[5:0]	[00…0]	The delay between when the next channel is enabled and the start of the modulator data accumulation in the filters. Delay = 25 ADC_CLK periods * ADC_SEQDLY (2us per ADC_SEQDLY count at 12.5MHz ADC_CLK). The max value of ADC_SEQDLY is 63 which is 126 usec nominal. ADC_SEQDLY = 0 will disable the delay of internal signal within the signal path, but this is not recommended since a positive settling time is required for accurate data conversion. In order to properly store the conversion data from the ADC_CLK domain into the system clock domain, four(4) PCLK periods minimum are required within the next conversion period, if a sequence is being processed. Therefore, the conversion time + ADC_SEQDLY must be longer than 4 PCLK periods. Refer to ADC_SEQDLY Requirements for minimum ADC_SEQDLY calculations to support various system clock (PCLK) speeds.

14.7.8 DSM Digital Stability Control (STAB_CTRL)

(This is a necessary addition for using the ADC outside of the incremental-delta-sigma method. This Stability Control is also valid for the Incremental-ADC, but resets occur normally with each convert operation.)

The modulator is conditionally stable and depends on the input signal amplitude, the loop-gain, and the initial conditions. Also, during power-up the modulator may become unstable. The Stability Control should detect a modulator overload by monitoring the single bit output stream of the delta-sigma modulator and, if necessary, assert the modulator reset signal. The digital filters DDF1 and DDF2 will not be reset by this signal, to allow the OSR counter to complete as normal, without upsetting the sample period. Also, the filter data will still be readable, and may only have a small perturbation due to the modulator overload. The DSM_OVL_FLAG will be echoed into the DSM_OVL_FLAG and DATA_ERROR bits of the ADC.DATA register for the appropriate conversion output only.

An overload is detected if a pattern of DSM_OVL_CNT repeated ones or zeroes occurs. After the overload event is detected, the modulator integrator reset signal is asserted for DSM_OVL_RST modulator clock periods. DSM_OVL_FLAG is a flag operation for the overload condition, which will remain latched until the software reads the flag.

STAB_CTRL																**Offset = 0x0084**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																OF
W																
Reset	[00…0]															0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R				DSM_OVL_RST[4:0]						DSM_OVL_CNT[6:0]						
W																
Reset	000			0_1000					0	001_1110						

Table 14.13: Description of the DSM Digital Stability Control Register

Bit Number(s)	Bit Name	Reset State	Description
31-17	RESERVED	[00…0]	
16	DSM_OVL_FLAG (OF)	0	1: Overload condition. Will remain logic 1 until the software reads the flag 0: No action
15-13	RESERVED	0	
12-8	DSM_OVL_RST [4:0]	0_1000	Number of ADC_CLK cycles that the modulator reset will be applied when an overload condition is detected 00000: Disable the Stability Control function
7	RESERVED	0	
6-0	DSM_OVL_CNT [6:0]	001_ 1110	Threshold to determine an overload condition has occurred. This register increments if previous modulator bitstream output is equal to current output, otherwise the register is reset.

14.7.9 Interrupt Status Register (INT_STATUS)

This register provides status of the source of the combined ADC Interrupt output signal. These bits are completely independent of the Interrupt enable bits in ADC.SPB_CFG_0. No masking is implemented for this status register. Refer to section 9.4 for details of the ADC_INTR signal.

A read of this status register will clear the ADC Interrupt output signal, but the status bits located here require a read of the associated ADC.DATAOUT_REG before this status bit is cleared. This occurs because these status bits are actually a combinatorial representation of the ADC.DATA register bits (e.g. CIO_Over_Comb is a OR of all ADC.DATA.COI_Over bits; the SE_CHNL_INTR_PEND[n] is a combinatorial path from ADC.DATA.Conv_Compl.) In other words, these status bits are “sticky”, while the ADC_INTR signal is cleared quickly, the opposite of sticky.

INT_STATUS																Offset = 0x008C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CCC	TUC	DOF	SOC	COC			TCP	DIFF_CHNL_INTR_PEND[7:0]							
Reset	0	0	0	0	0	00		0	[00…0]							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SE_CHNL_INTR_PEND[15:0]															
Reset	[00…0]															

Table 14.14: Description of the Interrupt Status Register

Bit Number(s)	Bit Name	Reset State	Description
31	CONV_COMPL _COMB (CCC)	0	1: Indicates that any ADC_DATAOUT_REG has a CONV_COMPL status bit set 0: No action
30	TRIG_UNDER _COMB (TUC)	0	1: Indicates that any ADC_DATAOUT_REG has a TRIG_UNDER status bit set 0: No action
29	DSM_OVL_FLAG _COMB (DOF)	0	1: Indicates that any ADC_DATAOUT_REG has a DSM_OVL_FLAG status bit is not set 0: No action
28	SINC4_OVER _COMB (SOC)	0	1: Indicates that any ADC_DATAOUT_REG has a SINC4_OVER status bit is not set 0: No action
27	COI_OVER _COMB (COC)	0	1: Indicates that any ADC_DATAOUT_REG has a COI_OVER status bit is not set 0: No action
26-25	RESERVED	00	
24	TEMP_CHNL _INTR_PEN [7:0]	[00…0]	1: Indicates that the last conversion of the temperature channel (whether SE or differential) was the source of the Interrupt. This bit is cleared when the associated ADC_DATAOUT_REG is read and its status bits are cleared 0: No Interrupt occurred
23-16	DIFF_CHNL _INTR_PEND [7:0]	[00…0]	One bit per Differential Channel. When a bit is set, the associated channel was the source of an Interrupt. This bit is cleared when the associated ADC_DATAOUT_REG is read, and its status bits are cleared. Note that if the differential channel is enabled, the status in ADC_DATAOUT_REG is assumed to be from a differential conversion, and these associated INTR_PEND bit(s) are asserted.
15-0	SE_CHNL_INTR _PEND [15:0]	[00…0]	One bit per Single-Ended Channel. When a bit is set, the associated channel was the source of an Interrupt. This bit is cleared when the associated ADC_DATAOUT_REG is read, and its status bits are cleared. Note that if the differential channel is enabled, the status in ADC_DATAOUT_REG is assumed to be from a differential conversion, and

Bit Number(s)	Bit Name	Reset State	Description
			these associated INTR_PEND bits are not asserted.

14.7.10 Data Output Word Register (DATA)

The ADC.DATA[15:0] array of registers contains the converted data for each channel, and status bits applicable to the conversion which produced the data. The register for a given channel is over-written when new conversion data is available, whether or not the previous data has been read.

DATA																**Offset = 0x0090-0x00D4 (Increments of 0x04)**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATA_OUT_LSB[3:0]					TU	DOF	SO	CO	GAIN[2:0]			DD			CE
W																
Reset	[00…0]				0	0	0	0	0	000			0	00		0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		DE			DATA_OUT[11:0]											
W																
Reset	0	0	00		[00…0]											

Table 14.15: Description of Data Output Word Register

Bit Number(s)	Bit Name	Reset State	Description
31-28	DATA_OUT_LSB[3:0]	[00...0]	Least significant 4 bits of the final filter counter output, typically discarded. These four counter bits which are less significant than the 12 bits of the counter used as DATA_OUT
27	RESERVED	0	
26	TRIG_UNDER(1) (TU)	0	1: Indicates a trigger was commanded while a conversion was in progress 0: No trigger commanded while a conversion was in progress
25	DSM_OVL_FLAG(1) (DOF)	0	Copy of the ADC.STAB_CTRL.DSM_OVL_FLAG and it is latched until the ADC.STAB_CTRL register is read
24	SINC4_OVER(1) (SO)	0	1: Indicates a DDF2 filter stage exceeded min or max during a conversion 0: DDF2 filter stage remained within bounds during a conversion
23	COI_OVER(1) (CO)	0	1: Indicates a DDF1 filter stage exceeded min or max during a conversion 0: DDF1 filter stage remained within bounds during a conversion
22-20	GAIN[2:0]	000	Gain status for the respective channel (consult SECHAN_CFG, DIFFCHAN_CFG, or TEMPCHAN_CFG for details)
19	DD	0	1: DDF2 (SINC4) filter is selected for the respective channel 0: DDF1 (COI3) filter is selected for the respective channel
18-17	RESERVED	00	
16	CHNL_EN	0	Status of EN setting for the respective channel
15	RESERVED	0	
14	DATA_ERROR(1)	0	1: Logical OR of all error conditions for this conversion. Cleared when read, and updated with new status at the end of each conversion -DSM_OVL_FLAG -SINC4 counter over/under range -COI over-range -Trigger under-range (bit 24, caused by a trigger while a conversion in progress -Data value stale (see bit 15)
13-12	RESERVED	00	
11-0	DATA_OUT [11:0]	[00…0]	12-bit conversion data output. Must be read by the microcontroller core before the next conversion is completed, or earlier data is overwritten

Note 1: bits [26:23] and [14] are not valid if they are read at the very same clock cycle as an ADC conversion is completed. A proper polling sequence then is to read ADC_INT_STATUS for the desired conversion complete status, and then only if the conversion is complete, immediately read this register's status bits. The read of ADC_DATAOUT_REG clears both the status bits of this register and corresponding ADC_INT_STATUS bit(s). However, the DATA_OUT bits of this register are not reset by a read, so the value contains valid conversion data whenever it is read.

15 Digital-to-Analog (DAC)

15.1 Overview

The DAC peripheral includes two independent 12-bit voltage mode R2R DACs with parallel inputs.

The digital input for each DAC is implemented independent of each other through the DAC.DATA[1] and DAC.DATA[2] registers.

The two load registers can be loaded synchronously or non-synchronously, and at the same frequency or not at the same frequency (see description of the different options in the Normal Operational Mode section).

- The two DACs are written with unique data, allowing fully independent output voltages from 0V to the VREF input voltage. The data coming from an APB transaction is written into the associated Write Register. The DAC does not convert the data until it is loaded from the Write Register into the Load Register. Double-buffering of the data via a Write Register and a Load Register supports a synchronous application into both DAC0 and DAC1, with unique data. This is provided via the lD0 and lD1 bits in the DAC.CONTROL Register. Alternatively, an “autoload” mode is available, which updates the Load Register at the end of each APB transaction to the Write Register, alleviating the APB master from having to issue two transactions to update the DAC value.

The DATA1 and DATA2 registers will be written by the digital logic during a Data Write (Address Offset 0x04 and 0x08, respectively). These will always occur asynchronously any time prior to the associated internal control signal. The synchronous update of both DACs, as described in the cases above, is only dependent on when the LD1_n and LD0_N signals are activated.

15.2 DAC Register Details

Table 15.1: DAC Registers

Register Name	Offset (hex)
Control Register (CONTROL)	0x00
Data Register 1 (DATA[1])	0x04
Data Register 2 (DATA[2])	0x08

15.2.1 Control Register (CONTROL)

CONTROL																**Offset = 0x0000**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									SRD	ID1	AL1	DE1		ID0	AL0	DE0
W																
Reset	[00…0]								0	0	0	0	0	0	0	0

Table 15.2: Description of the Control Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00...0]	
7	SOFT_RST_DIS (SRD)	0	1: Disable the HWSW_RESETn input from generating any resets 0: Logic OR the HWSW_RESEETn input with the PRESET_n function
6	ID1	0	1: A write of '1' provides a manual pulse signal to the DAC1 to load the DAC.DATA2. Always reads back as '0'. Not necessary if AUTOLOAD_1 bit is set 0: No action
5	AUTOLOAD_1 (AL1)	0	1: Enable an automatic reload of the DAC1 data following each write to the DAC.DATA2 Register 0: No action (Write a '1' to ID1 to update DAC1)
4	DAC_EN_1 (DE1)	0	1: Enable DAC_1 0: Power down DAC_1 and clear data register
3	RESERVED	0	
2	ID0	0	1: A write of '1' provides a manual pulse signal to the DAC0 to load the DAC.DATA1. Always reads back as '0'. Not necessary if AUTOLOAD_1 bit is set 0: No action
1	AUTOLOAD_0 (AL0)	0	1: Enable an automatic reload of the DAC1 data following each write to the DAC.DATA1 Register 0: No action (Write a '1' to ID0 to update DAC0)
0	DAC_EN_0 (DE0)	0	1: Enable DAC_0 0: Power down DAC_0 and clear data register

15.2.2 DAC0 Data Register (DATA1)

REG NAME																**Offset = 0x0004**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					DATA1 [11:0]											
W																
Reset	[00...0]				[00...0]											

Table 15.3: Description of the DATA1 Register

Bit Number(s)	Bit Name	Reset State	Description
31-12	RESERVED	[00...0]	
11-0	DATA1	[00...0]	Digital input to be converted for DAC0. If the AUTOLOAD_0 control bit is active, the pulse signal to load the digital input is generated immediately following the valid DATA1 digital output

15.2.3 DAC1 Data Register (DATA2)

This register contains the countdown value used by the Watchdog timer before timing out.

DATA2																**Offset = 0x0008**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					DATA2 [11:0]											
W																
Reset	[00...0]				[00...0]											

Table 15.4: Description of the DATA2 Register

Bit Number(s)	Bit Name	Reset State	Description
31-12	RESERVED	[00...0]	
11-0	DATA2	[00...0]	Digital input to be converted for DAC0. If the AUTOLOAD_1 control bit is active, the pulse signal to load the digital input is generated immediately following the valid DATA2 digital output

16 Comparators

16.1 Overview

The UT32M0R500 contains two high speed comparators with hysteresis. Both comparators support rail-to-rail inputs, therefore can support both analog and digital inputs.

Both comparators also support using analog reference voltages on the negative terminal based on the COMP.CTRL.Cx_NEG_SEL setting.

16.2 Comparator Register Details

Table 16.1: Dual Comparator Registers

Register	Offset (Hex)
Control Register (CTRL)	0x00
Status Register (STATUS)	0x04

16.2.1 Control Register (CTRL)

CTRL																**Offset = 0x0000**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R										C1_NEG_SEL		C1E		C0_NEG_SEL		C0E
W																
Reset	[00…0]									00		0	0	00		0

Table 16.2: Description of the Control Register

Bit Number(s)	Bit Name	Reset State	Description
31-7	RESERVED	[00...0]	
6-5	C1_NEG_SEL	00	Comparator #1 negative input selector: 11: Input from DAC1 10: Input from DAC0 01: Input from pin CMP1B 00: Input from voltage reference ~VREF
4	C1_EN	0	Comparator #1 enable selector 1: Comparator #1 enabled 0: Power down Comparator #1; CMP1OUT will be driven to '0'
3	RESERVED	0	
2-1	C0_NEG_SEL	00	Comparator #0 negative input selector 11: Input from DAC1 10: Input from DAC0 01: Input from pin CMP0B 00: Input from voltage reference ~VREF
0	C0_EN	0	Comparator #0 enable selector 1: Comparator #0 enabled 0: Power down Comparator #0; CMP0OUT will be driven to '0'

16.2.2 Status Register (STATUS)

STATUS																**Offset = 0x0004**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R															C1	C0
W																
Reset	[00...0]														0	0

Table 16.3: Description of the Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-2	RESERVED	[00...0]	
1	COMPARE_OUT_1 (C1)	0	Active high when Comparator #1 input (+) is higher than the selected reference voltage. Synchronized to the PCLK
0	COMPARE_OUT_0 (C0)	0	Active high when Comparator #0 input (+) is higher than the selected reference voltage. Synchronized to the PCLK

17 Timers

17.1 Overview

UT32M0R500 includes two independent programmable 32-bit free-running timer modules; each timer can be either 16-bit or 32-bit. Each timer supports a clock prescaler value of 1, 16, or 256. Each timer module can be independently enabled or disabled. Each module supports three different modes of operation: free-running, periodic timer and one-shot timer.

For Timer example code and application specific interface (API), refer to the UT32M0R500 Software Development Kit (SDK).

Figure 17.1 shows the simplified block diagram for one Timer module.

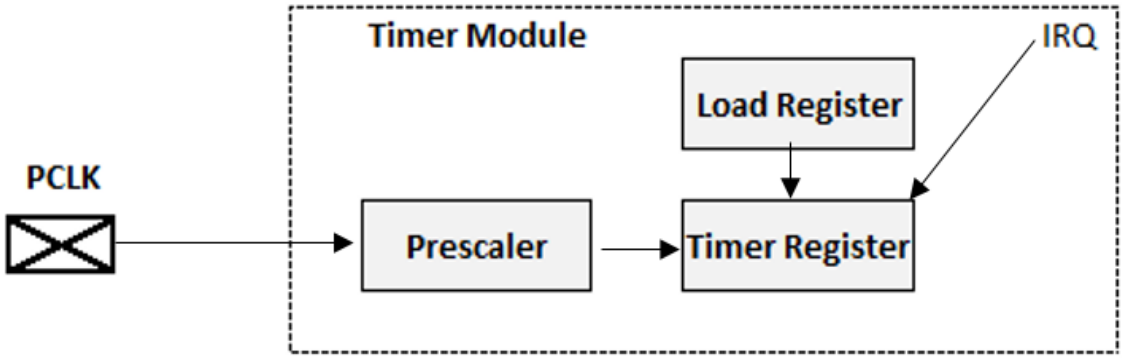


Figure 17.1: Timer Block Diagram

17.2 Functional Description

The UT32M0R500 provides two hardware timers. Hardware timers free the processor to perform other tasks during the on-time period; they provide high resolution and fixed frequency signals; they use the system clock (PCLK) as the time unit for counting up or down; and can generate an Interrupt when the counter reaches a predefined value. A timer can periodically generate an Interrupt at regular intervals for some specific action, i.e., monitoring and pre-processing tasks of closed loop control systems at a specified interval.

17.3 Operation

To generate a one second interval, set the PRESCALER to 1, MODE to periodic, COUNT to wrap, write LOAD_VAL with 781250, SIZE to 32-bit, set INTRPT_ENABLE flag to 1, and finally enable the timer module. The Interrupt service routine (ISR) generates an event when the counter counts down from 781250 to 0, then it reloads the start value to 781250.

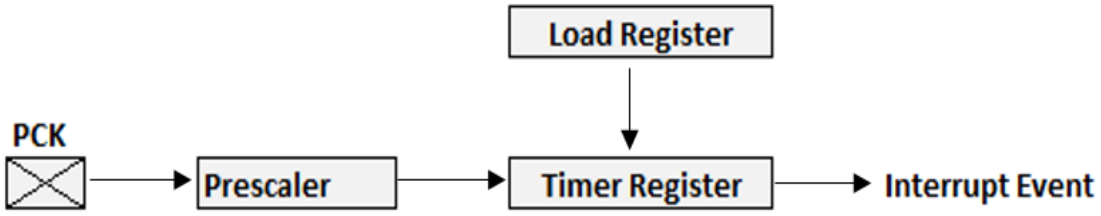


Figure 17.2: Timer Interval

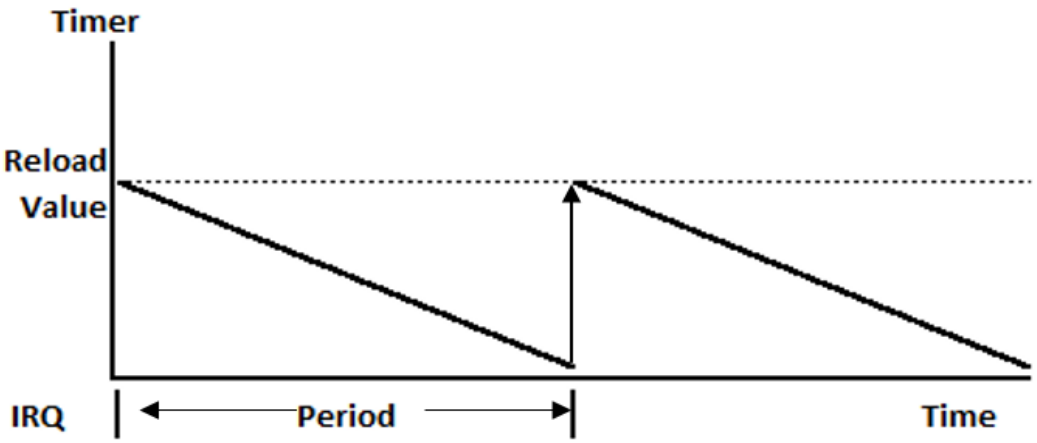


Figure 17.3: Timer Interval Diagram

17.3.1 Prescaler

The Prescaler defines the rate by which the timer register, TIMERxVALUE, increments or decrements automatically. It is derived from the system clock, PCLK. TIMER_SCALER bits in the control register, TIMERxCONTROL, have the predefined value by which the prescaler divides the PCLK frequency. The prescaler register scales the PCLK frequency down into a value from 1, 16 or 256 required by the timer, TIMERxVALUE, register.

17.3.2 Free-Running Mode

The counter wraps after reaching its zero value and continues to count down from the maximum value. This is the default mode.

17.3.3 Periodic Timer Mode

The timer generates an Interrupt at a constant interval, reloading the original value after wrapping past zero.

The TIMERxLOAD Register contains the value from which the counter is to decrement. This is the value used to reload the timer when Periodic mode is enabled, and the current count reaches 0. When this register is written to directly, the current time is immediately reset to the new value at the next rising edge of the prescaler (TIMCLK) that TIMCLKEN has enabled.

The value in TIMERxLOAD register is also overwritten if the TIMERxBGLOAD Register is written to, but the current count is not immediately affected. If values are written to both the TIMERxLOAD and TIMERxBGLOAD registers before an enabled rising edge on TIMCLK, the following occurs:

1. On the next enabled TIMCLK edge, the value written to the TIMERxLOAD value replaces the current count value.
2. Every time the counter reaches 0, the current count value is reset to the value written to TIMERxBGLOAD.

Reading from the TIMERXLOAD Register at any time after the two writes have occurred retrieves the value written to TIMERXBGLOAD. That is, the value read from TIMERxLOAD is always the value that takes effect for Periodic mode after the next time the counter reaches 0.

17.3.4 One-Shot Timer Mode

The counter generates an Interrupt once. When the counter reaches 0, it halts until reprogrammed by the following:

- Clearing the one-shot count bit in the control register, in which case the count proceeds according to the selection of Free-running or Periodic mode.
- Writing a new value to the TIMERxBGLOAD register.

17.4 Timer Register Details

Table 17.1: Timer Registers

Register	Offset (Hex)
	Timer 0 Specific Registers
Timer 0 Load Register (TIMER0LOAD)	0x0000
Timer 0 Counter Value Register (TIMER0VALUE)	0x0004
Timer 0 Control Register (TIMER0CONTROL)	0x0008
Timer 0 Interrupt Clear Register (TIMER0INTCLR)	0x000C
Timer 0 Raw Interrupt Status Register (TIMER0RIS)	0x0010
Timer 0 Masked Interrupt Status Register (TIMER0MIS)	0x0014
Timer 0 Background Load Register (TIMER0BGLOAD)	0x0018
	Timer 1 Specific Registers
Timer 1 Load Register (TIMER1LOAD)	0x0020
Timer 1 Counter Value Register (TIMER1VALUE)	0x0024
Timer 1 Control Register (TIMER1CONTROL)	0x0028
Timer 1 Interrupt Clear Register (TIMER1INTCLR)	0x002C
Timer 1 Raw Interrupt Status Register (TIMER1RIS)	0x0030
Timer 1 Masked Interrupt Status Register (TIMER1MIS)	0x0034
Timer 1 Background Load Register (TIMER1BGLOAD)	0x0038
	General Registers
Integration Test Control Register (ITCR)	0x0F00
Integration Test Output Set Register (ITOP)	0x0F04

17.4.1 Timer X Load Register (TIMERxLOAD)

TIMERxLOAD																Offset = 0x0000 (TIMER1LOAD) 0x0020 (TIMER2LOAD)
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TIMERxLOAD															
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TIMERxLOAD															
Reset	[00…0]															

Table 17.2: Description of the Timer X Load Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	TIMERxLOAD	[00…0]	The value from which the counter is to decrement

17.4.2 Timer X Counter Current Value Register (TIMERxVALUE)

TIMERxVALUE																Offset = 0x0004 (TIMER1VALUE) 0x0024 (TIMER2VALUE)
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TIMERxVALUE															
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TIMERxVALUE															
Reset	[00…0]															

Table 17.3: Description of the Timer X Counter Current Value Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	TIMERxVALUE	[00...0]	The current value of the timer

Timer X Control Register (TIMERxCONTROL)

TIMERxCONTROL																**Offset = 0x0008 (TIMER1CONTROL) 0x0028 (TIMER2CONTROL)**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									TE	TM	IE		TP		TS	OC
W																
Reset	[00...0]								0	0	1	0	00		0	0

Table 17.4: Description of the Timer X Control Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00...0]	
7	TIMER_ENABLE (TE)	0	1: Timer enabled 0: Timer disabled
6	TIMER_MIDE (TM)	0	1: Timer is in periodic mode 0: Timer is in free-running mode
5	INTERRUPT _ENABLE (IE)	1	1: Timer Interrupt enabled 0: Timer Interrupt disabled
4	RESERVED	0	
3-2	TIMER _PRESCALER (TP)	00	11: Undefined, do not use 10: 8 stages of prescale, clock is divided by 256 01: 4 stages of prescale, clock is divided by 16 00: 0 stages of prescale, clock is divided by 1
1	TIMER_SIZE (TS)	0	1: 32-bit counter 0: 16-bit counter
0	ONE_SHOT _COUNT (OC)	0	1: One-shot mode 0: Wrapping mode

17.4.4 Timer X Interrupt Clear Register (TIMERxINTCLR)

TIMERxINTCLR																Offset = 0x000C (TIMER1INTCLR) 0x002C (TIMER2INTCLR)
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																IC
Reset	[00…0]															0

Table 17.5: Description of the Timer X Interrupt Clear Register

Bit Number(S)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	INTCLR_Msk (IC)	0	1: Clear Timerx Interrupt 0: No action

17.4.5 Timer X Raw Interrupt Status Register (TIMERxRIS)

TIMERxRIS																Offset = 0x0010 (TIMER1RIS) 0x0030 (TIMER2RIS)
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																RIS
Reset	[00…0]															0

Table 17.6: Description of the Timer X Raw Interrupt Status Register

Bit Number(S)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	RAWINTSTAT _Msk (RIS)	0	1: Indicates that Interrupt conditions occurred. This value will be reflected in the Interrupt Status register only if TIMERx Interrupt Enable bit is set 0: No Interrupt conditions occurred

17.4.6 Timer X Masked Interrupt Status Register (TIMERxMIS)

TIMERxMIS																Offset = 0x0014 (TIMER1MIS) 0x0034 (TIMER2MIS)
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																MIS
W																
Reset	[00…0]															0

Table 17.7: Description of the Timer X Masked Interrupt Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	MASKINTSTAT _Msk (MIS)	0	1: The Interrupt is masked 0: The Interrupt is unmasked

17.4.7 Timer X Background Load Register (TIMERxBGLOAD)

TIMERxBGLOAD																Offset = 0x0014 (TIMER1BGLOAD) 0x0034 (TIMER2BGLOAD)
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BGLOAD_Msk															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BGLOAD_Msk															
W																
Reset	[00…0]															

Table 17.8: Description of the Timer X Background Load Register

Bit Number(S)	Bit Name	Reset State	Description
31-0	BGLOAD_Msk	[00...0]	Value for background load for TIMERx

18 Pulse Width Modulator (PWM)

18.1 Overview

The UT32M0R500 includes one pulse width modulation (PWM) module with three separate controllers. Each PWM controller can have a single output, or the three controllers can be combined to form two-paired outputs. Each PWM device is configured as a 16-bit channel. Each PWM device includes a programmable dead-band scaler with a range from 20ns to 81,920ns, programmable clock scaler for a max 332ms pulse, and all three devices have a single combined Interrupt.

For PWM example code and application specific interface (API), refer to the UT32M0R500 Software Development Kit (SDK).

Figure 18.1 shows the simplified block diagram for the PWM module.

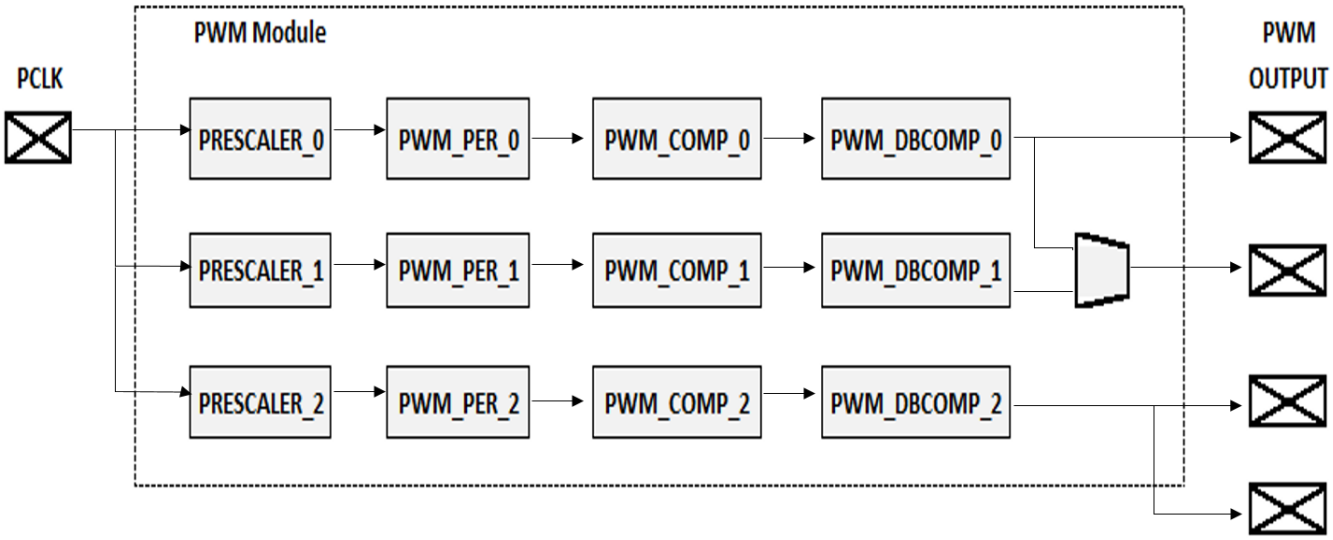


Figure 18.1: PWM Module Block Diagram

18.2 Functional Description

The PWM provides proportional power control by adjusting delivered power. It can be used to control the power delivered to a DC motor or the brightness of an LED. For a DC motor, PWM switches the power on and off such that power is applied to the motor when the signal is high and no power when the signal is low. The amount of power delivered is linearly proportional to its duty cycle. Many loads respond to the overall average value (duty cycle) of the wave by selecting a slow response. PWM signals include modulation frequency, period, on time and duty cycle.

18.3 Operation

PWM varies the width pulses of the output signal. The pulse width describes the proportion of the ON state in one pulse period variable in each cycle, which is the duty cycle. The duty cycle is defined by the following equation:

$$Duty\ Cycle = \frac{Pulse\ Width}{Total\ Period}$$

To generate a 25% duty cycle output signal, first set the particular GPIO to the corresponding PWM alternate function; then preload the scaler register, the period register with a value, i.e., 1024, the compare register with a value, i.e., 768; optionally, either period or compare match Interrupt can be enabled or dead band register can be preloaded with a desire value and enable; finally enable the particular PWM device. The hardware will reload the period value at the end of the preloaded value and compare the values of compare with the period registers. The comparator output will remain OFF until the value of the period register exceeds that of the compare register, at which point the comparator output will be set to ON. When the period register reaches 1024, the hardware will reset it, which will drive the comparator output to a logic zero, so the PWM output signal is ON 25% of the time and OFF the other 75% of the time, generating a 25% duty cycle.

Figure 18.2 shows a simplified block diagram for a PWM output signal with 25% duty cycle.

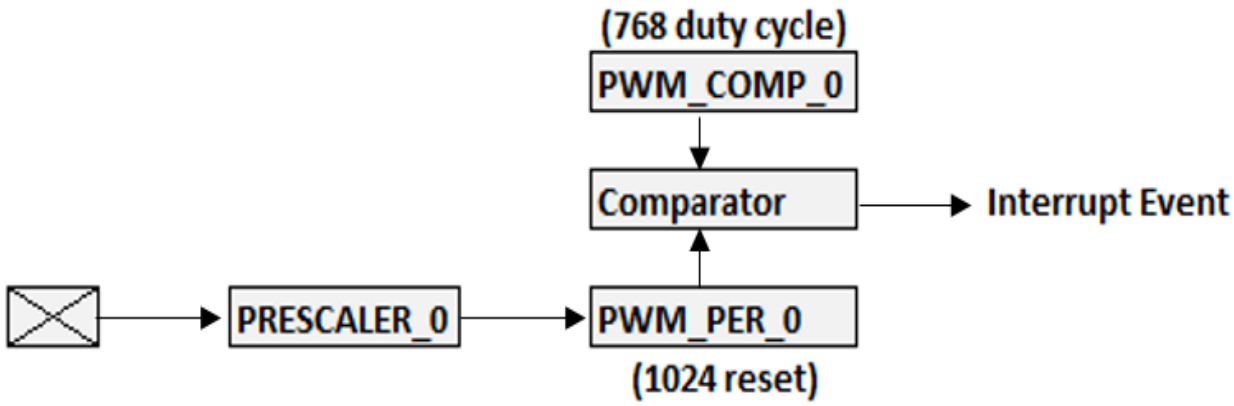


Figure 18.2: PWM Output Signal Block Diagram with 25% Duty Cycle

Figure 18.3 shows the PWM output signal timing diagram with 25% duty cycle.

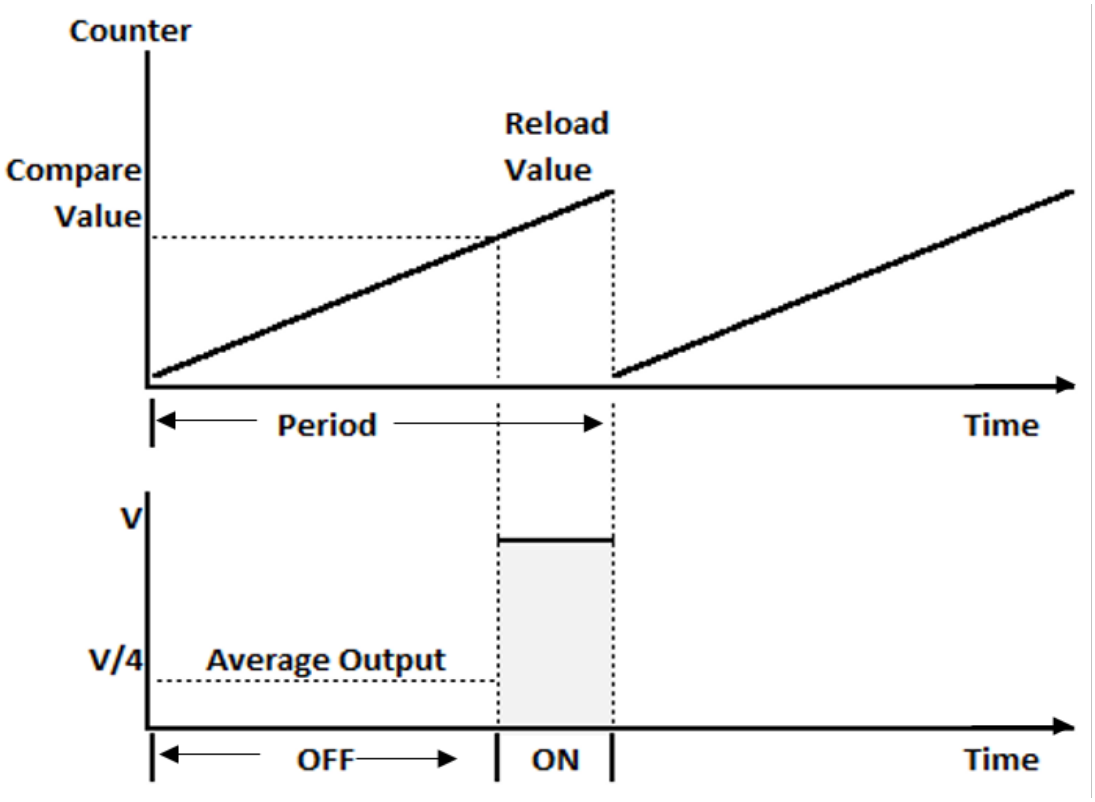


Figure 18.3: PWM Output Signal Timing Diagram with 25% Duty Cycle

18.3.1 Prescaler

The Prescaler defines the rate by which the period register, PWM_PER_x, increments or decrements automatically. It is derived from the system clock, PCLK. The control register, CTRLREG, selects which PWM devise will be read/written from the prescaler register, SCALREG. The prescaler register scales the PCLK frequency down into a value from 0-255 required by the period register.

18.3.2 Single Output Signal

Table 21.1 shows the registers used to program the UT32M0R500 PWM. CTRLREG, SCALREG and IRQREG are the core registers. CTRLREG contains control and flag bits. SCALERSEL selects which PWM channel loads the prescaler to downscale the system clock (PCLK). CORE_ENABLE flag enables the PWM module. SCALREG registers contains the prescaler value for dividing down the PCLK frequency. PWM_PER_x register contains the value, i.e., 1024, for generating the period of the particular PWM channel. PWM_COMP_x register contains the value, i.e., 768, for generating the duty cycle of the particular PWM channel. PWM_CTRL_x register contains the control and flags bits for the particular PWM channel. SCALERSEL selects which current PWM channel loads the prescaler to downscale the system clock (PCLK). POL bit set to 1 selects output signal active high; EN bit set to 1 enables the particular PWM channel.

Table 18.1 shows the Asymmetric register configurations

Table 18.1: Asymmetric Register Configurations

CTRLREG	SCALERSEL[10:8]	CORE_ENABLE[0]	
0x0001	0 :PRESCALER_0	1 :Enable PWM module	
SCALREG	RELOAD[7:0]		
0xFF	255 :Prescaler		
IRQREG			
0x00			
CAPREG1	ASYMPWM[22]		
0x1067477A	1 :Asymmetric PWM		
PWM_PER_X	PER[15:0]		
0x400	1024 :Period		
PWM_COMP_X	COMP[15:0]		
0x300	768 :Period		
PWM_DBCOMP_X	DBCOMP[7:0]		
0x00	0 :Deadband		
PWM_CTRL_X	SCALERSEL[12:10]	POL[1]	EN[0]
0x403	1 :scaler 1 selected	1 :PWM is active high	1 :Enable PWM 0

Figure 18.4 shows the single PWM output signal timing diagram.

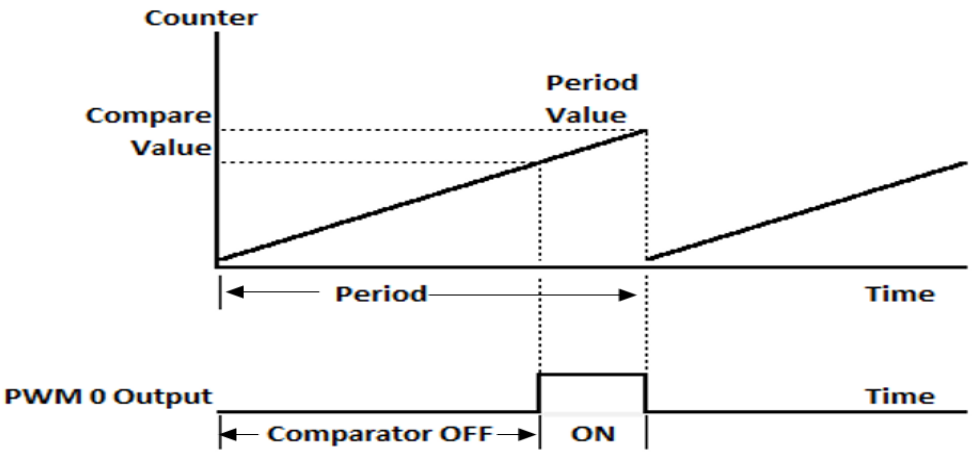


Figure 18.4: Asymmetric PWM Timing Diagram

18.3.3 Paired Output Signal

Table 21.2 shows the registers used to program the UT32M0R500 PWM. CTRLREG, SCALREG and IRQREG are the core registers. CTRLREG contains control and flag bits. SCALERSEL selects which PWM channel loads the prescaler to downscale the system clock (PCLK). CORE_ENABLE flag enables the PWM module. SCALREG registers contains the prescaler value for dividing down the PCLK frequency. PWM_PER_x register contains the value, i.e., 1024, for generating the period of the particular PWM channel. PWM_COMP_x register contains the value, i.e., 768, for generating the duty cycle of the particular PWM channel. PWM_CTRL_x register contains the control and flags bits for the particular PWM channel. SCALERSEL selects which current PWM channel loads the prescaler to downscale the system clock (PCLK). PAIR flag bit set to 1 pairs the output signals; POL bit set to 1 selects output signal active high; EN bit set to 1 enables the particular PWM channel.

Table 18.2 shows the paired register configurations.

Table 18.44: Paired Register Configurations

CTRLREG	SCALERSEL[10:8]	CORE_ENABLE[0]		
0x0001	2 :PRESCALER_0	1 :Enable PWM module		
SCALREG	RELOAD[7:0]			
0xFF	255 :Prescaler			
IRQREG				
0x00				
CAPREG1	ASYMPWM[22]			
0x1067477A	1 :Asymmetric PWM			
PWM_PER_X	PER[15:0]			
0x400	1024 :Period			
PWM_COMP_X	COMP[15:0]			
0x300	768 :Period			
PWM_DBCOMP_X	DBCOMP[7:0]			
0x00	0 :Deadband			
PWM_CTRL_X	SCALERSEL[12:10]	PAIR[2]	POL[1]	EN[0]
0x403	3 :scaler 3 selected	1 :PWM paired output	1 :PWM is active high	1 :Enable PWM 0

Figure 18.5 shows the paired PWM timing diagram.

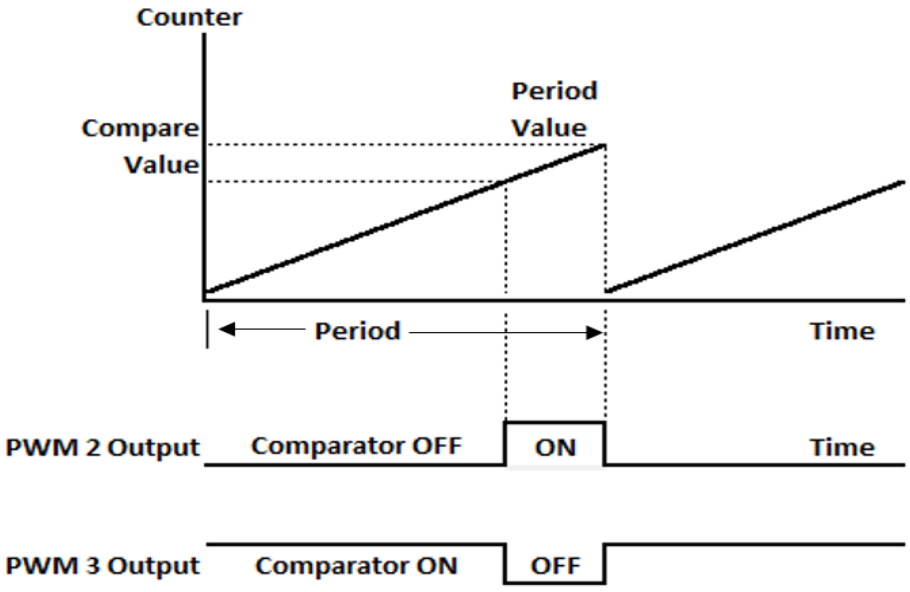


Figure 18.5: Paired PWM Timing Diagram

18.3.4 Dead Band Time

It is often desired to have a delay between when one of the PWM signals of a PWM pair goes inactive and when the other signal goes active. This delay is called dead band time. By default, the core does not generate any dead band time, but can be configured to do so by setting the DBEN bit in the PWM control register to '1'. When dead band time is enabled, the core will start a counter each time a PWM pair switches its outputs. The output going inactive is not delayed while the output going active is delayed until the counter matches the value in the PWM dead band compare register. To support a wide range of applications the amount of dead band time inserted is programmable, and each PWM pair can have a different amount of dead band time.

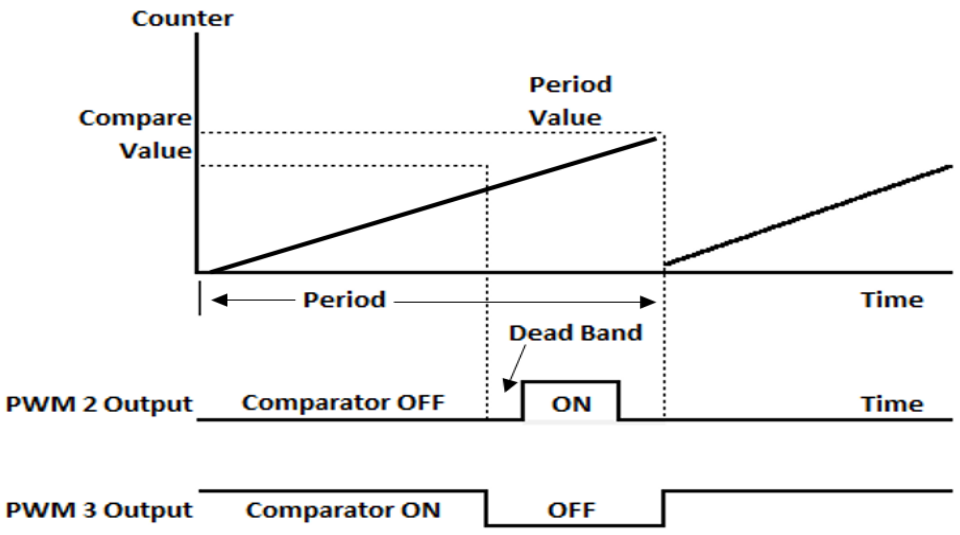
Table 18.3 shows the dead band time register configurations.

CTRLREG	SCALERSEL[10:8]	CORE_ENABLE[0]				
0x0201	2 :PRESCALER_0	1 :En PWM				
SCALREG	RELOAD[7:0]					
0xFF	255 :Prescaler					
IRQREG						
0x00						
CAPREG1	ASYMPWM[22]					
0x1067477A	1 :Asym PWM					
PWM_PER_X	PER[15:0]					
0x400	1024 :Period					
PWM_COMP_X	COMP[15:0]					
0x300	768 :Period					
PWM_DBCOMP_X	DBCOMP[7:0]					
0xFF	255 :Deadband					
PWM_CTRL_X	DBSCALER[25:22]	DBEN[21]	SCALERSEL[12:10]	PAIR[2]	POL[1]	EN[0]
0x403	15 :DB preload	1 :En DB	3 :scaler 3 selected	1 :paired output	1 :active high	1 :En PWM 0

Table 18.3: Dead Band Time Configurations

Figure 18.6 shows the paired PWM timing diagram.

Figure 18.4: Dead Band Timing Diagram



18.3.5 Interrupts

All three PWM channels share one external Interrupt (IRQ), mapped to Interrupt Vector 3. The UTM0R500 microcontroller based on the Cortex M0+ processor contains an internal nested vector Interrupt controller (NVIC) that supports up to 32 external IRQ's and a non-maskable Interrupt (NMI). NVIC handles the nested Interrupts automatically based on priorities and Interrupt number, and if an Interrupt is accepted, it communicates with the processor to allow it to execute the correct Interrupt handler.

For more information on external Interrupts and the nested vector Interrupt (NVIC), refer to Nested Vector Interrupt Controller (NVIC).

Within each channel, Interrupts are generated by setting the Interrupt pending Interrupt, IRQREG, bit [2:0] to 1 for the respective PWM device, the control register, PWM_CTRL_X, IRQ type (IRQT) bit [14] to either 1 for compare match or 0 for period match. Each PWM also has a 6-bit Interrupt counter that can be used to scale down the frequency at which the Interrupts occur.

When an Interrupt is generated the bit in the Interrupt pending register for the PWM in question is set. The bits in the Interrupt pending register stay set until software clears them by writing 1 to them. When an Interrupt is generated, or when the Interrupt scaler counter is increased, an output tick is generated on the PWM output signal.

Table 18.4 shows the Interrupt register configurations.

Table 18.4: PWM Interrupt Register Configurations

CTRLREG	SCALERSEL[10:8]	CORE_ENABLE[0]			
0x0001	0 :PRESCALER_0	1 :En PWM module			
SCALREG	RELOAD[7:0]				
0xFF	255 :Prescaler				
IRQREG	IRQ Pending[2:0]				
0x00	1 :for PWM_0				
CAPREG1	ASYMPWM[22]				
0x1067477A	1 :Asym PWM				
PWM_PER_X	PER[15:0]				
0x400	1024 :Period				
PWM_COMP_X	COMP[15:0]				
0x300	768 :Period				
PWM_DBCOMP_X	DBCOMP[7:0]				
0x00	0 :Deadband				
PWM_CTRL_X	IRQT[14]	IRQEN[13]	SCALERSEL[12:10]	POL[1]	EN[0]
0x403	0/1 :Period/Compare	1 :En IRQ	1 :scaler 1 selected	1 :PWM is active high	1 :Enable PWM 0

Figure 18.7 shows the period Interrupt timing diagram.

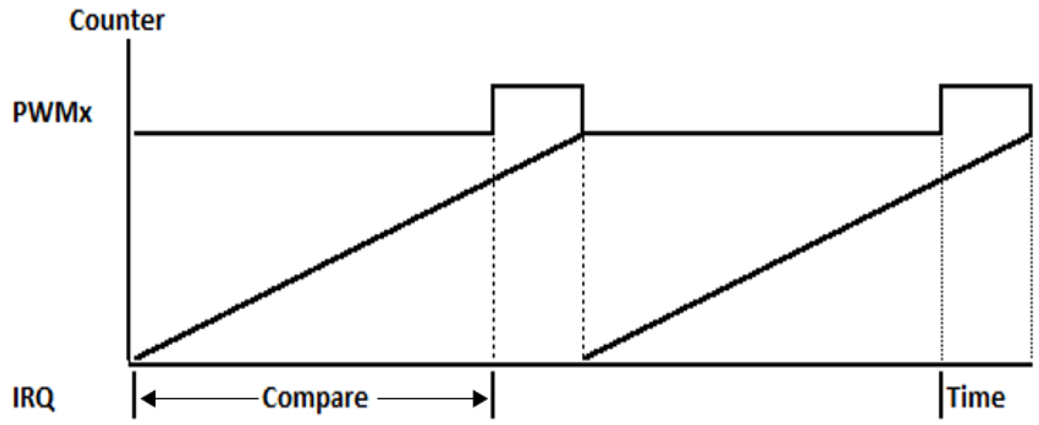


Figure 18.7: Period Interrupt Timing Diagram

Figure 18.8 shows the compare Interrupt timing diagram.

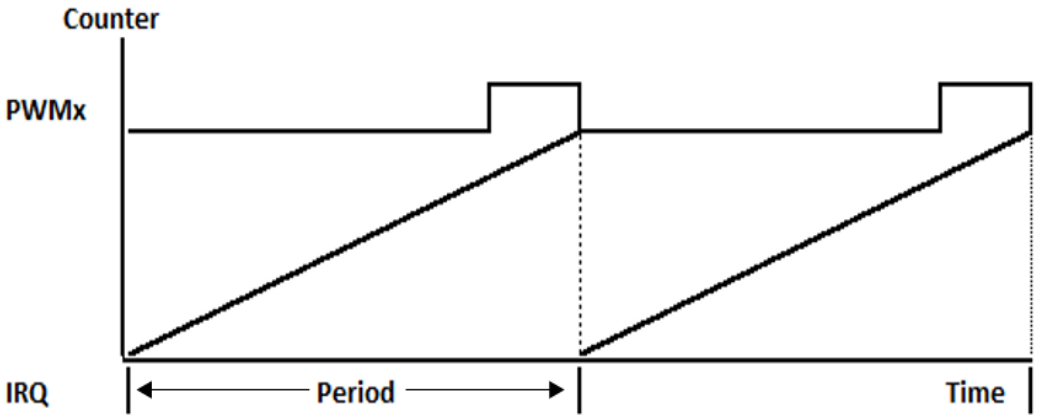


Figure 18.8: Compare Interrupt Timing Diagram

18.4 PWM Register Details

Table 18.5: PWM Registers

Register	Offset (Hex)
Core Control Register (CTRLREG)	0x0000
Scaler Reload Register (SCALREG)	0x0004
Interrupt Pending Register (IRQREG)	0x0008
Capability Register 1 (Read-only) (CAPREG1)	0x000C
Capability Register 2 (Read-only) (CAPREG2)	0x0010
RESERVED	0x0014-0x001C
PWM_0 Period Register (PWM_PER_0)	0x0020
PWM_0 Compare Register (PWM_COMP_0)	0x0024
PWM_0 Dead Band Compare Register (PWM_DBCOMP_0)	0x0028
PWM_0 Control Register for output PWM0 (PWM_CTRL_0)	0x002C
PWM_1 Period Register (PWM_PER_1)	0x0030
PWM_1 Compare Register (PWM_COMP_1)	0x0034
PWM_1 Dead Band Compare Register (PWM_DBCOMP_1)	0x0038
PWM_1 Control Register for output PWM1 (PWM_CTRL_1)	0x003C
PWM_2 Period Register (PWM_PER_2)	0x0040
PWM_2 Compare Register (PWM_COMP_2)	0x0044
PWM_2 Dead Band Compare Register (PWM_DBCOMP_2)	0x0048
PWM_2 Control Register for output PWM2 (PWM_CTRL_2)	0x004C

18.4.1 Core Control Register (CTRLREG)

CTRLREG																**Offset = 0x0000**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		NOUP				SCALERSEL										EN
W																
Reset	0	000			0	000			[00…0]							0

Table 18.6: Description of the Core Control Register

Bit Number(S)	Bit Name	Reset State	Description
31-16	RESERVED	[00...0]	
15-12	NOUP	000	No update (noup) bits for each PWM. Bit 12 for PWM_0, bit 13 for PWM_1, bit 14 for PWM_2. 1: The corresponding PWM's internal period register, compare register, and dead band compare registers are not updated from the corresponding APB registers. 0: The corresponding PWM's internal registers are updated from the corresponding APB registers. This is a write-protect to the actual PER/OCMP/DBCOMP regs, while the APB writes are allowed into shadow registers. These bits can be used by software if it wants to change more than one of the values and it is required that all values change in the same PWM period. It can also be used to synchronize the use of new values for different PWMs.
11	RESERVED	0	
10-8	SCALERSEL	000	System Clock Scaler Select bits. These bits determine which of the implemented system clock scalers' reload value that can be read/written from the scaler reload register. The first scaler register is addressed by "000".
7-1	RESERVED	[00…0]	
0	ENABLE (EN)	0	1: PWM can generate outputs 0: No ops. All outputs are disabled, and in their reset state per PWM_CTRL_0[1]

18.4.2 Scaler Reload Register (SCALREG)

SCALREG																**Offset = 0x0004**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									Word Received							
W																
Reset	[00…0]								0xFF							

Table 18.7: Description of the Scaler Reload Register

Bit Number(S)	Bit Name	Reset State	Description
31-8	RESERVED	[00...0]	
7-0	Word Received	0xFF	This field is used to reload the system clock scaler when it underflows. The SCALERSEL bits in the CTRLREG register determines which of the scalers that is read/written.

18.4.3 Interrupt Pending Register (IRQREG)

IRQREG																**Offset = 0x0008**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														IRQ Pending		
W																
Reset	[00...0]													000		

Table 18.8: Description of the Interrupt Pending Register

Bit Number(S)	Bit Name	Reset State	Description
31-3	RESERVED	[00...0]	
2-0	IRQ Pending	000	Interrupt pending bits for the PWM(s), bit 0 for PWM_0, bit 1 for PWM_1, and bit 2 for PWM_2. When an Interrupt event for a specific PWM occurs the core sets the corresponding bit in the Interrupt pending register and generates an Interrupt. Software can read this register to see which PWM that generated the Interrupt. The bits are cleared by writing 1 to them.

18.4.4 Capability Register 1 (CAPREG1)

CAPREG1																Offset = 0x000C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R				DP	DM	SEPIRQ			SP	AP	DS	DBBITS				
W																
Reset	000			1	0	00		0	0	1	1	0x07				

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NSCALERS			SBITS					PBITS					NPWM		
W																
Reset	010			0x07					0x07					010		

Table 18.9: Description of the Capability Register 1

Bit Number(S)	Bit Name	Reset State	Description
31-29	RESERVED	[00...0]	
28	DEFPOL (DP)	1	1: Default polarity is active high (outputs low after reset/power-up) 0: Default polarity is active low (outputs are high after reset/power-up)
27	DCMODE (DM)	0	1: Dual compare mode implemented 0: Dual compare mode not implemented
26-25	SEPIRQ	00	Reports Interrupt configuration
24	RESERVED	0	
23	SYMPWM (SP)	0	1: Symmetric PWM generation is implemented 0: Symmetric PWM generation is not implemented
22	ASYMPWM (AP)	1	1: Asymmetric PWM generation is implemented 0: Asymmetric PWM generation is not implemented
21	DBSCALER (DS)	1	1: Dead band time scaler(s) is implemented 0: Dead band time scaler(s) is not implemented
20-16	DBBITS	0x07	Report number of bits, -1, for the PWMs dead band time counters
15-13	NSCALERS	010	Reports number of implemented scalers, -1
12-8	SBITS	0x07	Reports number of bits for the scalers, -1
7-3	PBITS	0x07	Reports number of bits for the PWM counters, -1
2-0	NPWM	010	Reports number of implemented PWMs, -1

18.4.5 Capability Register 2 (CAPREG2)

CAPREG2																Offset = 0x0010
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R						WS	WDEPTH				WDBITS					WP
W																
Reset	[00…0]					0	[00…0]				0x07					0

Table 18.10: Description of the Capability Register 2

Bit Number(s)	Bit Name	Reset State	Description
31-11	RESERVED	[00…0]	
10	WSYNC (WS)	0	1: Waveform PWM synch signal generation is implemented 0: Waveform PWM synch signal generation is not implemented
9-6	WDEPTH	[00…0]	Reports the number of address bits -1 used for the waveform RAM. Value is log2(wdepth) -1
5-1	WDBITS	0x07	Reports number of bits -1 for each word in the waveform RAM
0	WPWM (WP)	0	1: Waveform PWM generation is implemented 0: Waveform PWM generation is not implemented

18.4.6 Period Register (PWM_PER_X)

PWM_PER_X																Offset = 0x0020 (PWM_PER_0) 0x0030 (PWM_PER_1) 0x0040 (PWM_PER_2)
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PER															
W																
Reset	[00…0]															

Table 18.11: Description of the Period Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[00...0]	
15-0	PER	[00...0]	Min operational value = '2'. Depending on the method used to generate the PWM the output could then be switched when the counter reaches the values of PWM period. When the PWM counter reaches this value a PWM period has passed. When this register is written the actual PWM period value used inside the core is not updated immediately, instead a shadow register is used to hold the new value until a new PWM period starts.

18.4.7 Compare Register (PWM_COMP_X)

PWM_COMP_X **Offset = 0x0024 (PWM_COMP_0)**
0x0034 (PWM_COMP_1)
0x0044 (PWM_COMP_2))

Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COMP															
W																
Reset	[00...0]															

Table 18.12: Description of the Compare Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[00...0]	
15-0	COMP	[00..0]	Min operational value = '1'. The PWM output is switched when the PWM counter reaches this maximum value. When the PWM counter reaches this value the PWM output is switched. When this register is written the actual PWM compare value used inside the core is not updated immediately, instead a shadow register is used to hold the new value until a new PWM period starts.

18.4.8 Dead Band Compare Register (PWM_DBCOMP_X)

PWM_DBCOMP_X																Offset = 0x0028 (PWM_DBCOMP_0) 0x0038 (PWM_DBCOMP_1) 0x0048 (PWM_DBCOMP_2)
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W									DBCOMP							
Reset	[00…0]								[00…0]							

Table 18.13: Description of the Dead Band Compare Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	DBCOMP	[00…0]	The dead band time has passed once the dead band counter reaches the value of this field. When this register is written the actual compare value used inside the core is not updated immediately, instead a shadow register is used to hold the new value until a new PWM period.

18.4.9 Control Register (PWM_CTRL_X)

PWM_CTRL_X																Offset = 0x002C (PWM_CTRL_0) 0x003C (PWM_CTRL_1) 0x004C (PWM_CTRL_2)
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W						FL	DBSCALER				DE	IRQSCALER [5:1]				
Reset	[00…0]					0	[00…0]				0	[00…0]				

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	I[0]	IT	IE	SCALERSEL			WE	DC		ME	FIX			PA	PO	EN
Reset	0	0	0	000			0	0	0	0	000			0	0	0

Table 18.45: Description of the Control Register

Bit Number(s)	Bit Name	Reset State	Description
31-27	RESERVED	[00...0]	
26	FLIP (FL)	0	1: The PWM outputs are flipped 0: The PWM outputs are not flipped
25-22	DBSCALER	[00…0]	Dead Band Scaler These bits are used to scale the system clock when generating dead band time. a) This field is only present if the DBSCALER generic is set to 1. When these bits are written the dead band scaler register inside the core is not updated immediately. b) Instead these bits are written to a reload register which updates the actual scaler when it underflows
21	DBEN (DE)	0	Dead Band Enable 1: Dead band time will be inserted when the PWM output switches from inactive to active 0: Dead band time generation is disabled, no dead band time will be inserted when the PWM output switches from inactive to active
20-15	IRQSCALER (I)	0	Interrupt Scaler Determines how many compare/period matches that need to occur before an Interrupt is generated. All zeroes means that an Interrupt will occur every compare/period match, a one means that an Interrupt will occur every second match, etc.
14	IRQT (IT)	0	Type of Interrupt 1: Generate an Interrupt on PWM compare match 0: Generate an Interrupt on PWM period match
13	IRQEN (IE)	0	Interrupt enable/disable bit 1: Interrupt is enabled 0: Interrupt is disabled
12-10	SCALERSEL	0	These 3 bits are used to select which of the system clock scalers that will be used when generating the current PWM. This field is only present when the ‘NSCALERS’ generic is greater than 1. These bits can only be set if the PWM is disabled, i.e. en bit (see below) set to ‘0’
9	WEN (WE)	0	PWM waveform enable bit. Not writeable since the ‘wavepwm’ field in Capability register 2 (CAPREG2) is set to ‘0’
8	DCEN (DC)	0	Dual compare mode enable. Not functional since the METH bit is set to ‘0’ (asymmetric). The compare register is only updated when the counter is zero. This bit is only present if the DCMODE bit in the Capability register is set.
7	RESERVED	0	
6	METH (ME)	0	PWM generation method select bit for an asymmetric output pulse. 1: Symmetric 0: Asymmetric Always a ‘0’ since the reset of CAPREG1.ASYMPWM=1 and CAPREG1.SYMPWM=0
5-3	FIX	0	Fix value select bits for PWM is used to set the PWM output to a fix value. a) If bit 3 is set to ‘1’, then bit 4 decides what the value of the PWM output will have b) If the PAIR bit is set to ‘1’ while bit 3 is set to ‘1’ as well then bit 5 determines what value the complement output will have.

Bit Number(s)	Bit Name	Reset State	Description
			NOTE: The fix bit setting takes priority over the POL bit. The FLIP bit does not still affect the paired outputs, plus the PWM_CTRL_x.EN and CTRLREG.EN bits must be set
2	PAIR (PA)	0	PWM pair bit. The PWM outputs [1:0] can be configured to be either unique PWM signals or a pair of PWM signals (where the two signals are each other's inverse), with configurable amount of dead band time in between them. 1: A complement output for PWM[0] will create a PWM pair instead of a single PWM. The complement output will be the first output's inverse, with the exception that dead band time might be added when the values switch from deactive to active. 0: The complement output of the PWM sub-block is always 0. Refer to the PWM Output Pairing Logic diagram. (This bit should always be set to '0' for PWM_CTRL_1)
1	POL (PO)	0	PWM polarity select bit 1: PWM is active high 0: PWM is active low This bit can only be set if the PWM has been disabled if the previous write to this control register, i.e. en bit (see below) set to '0'. POL bit write takes effect immediately. Reset value is as set in CAPREG1.DEFPOL bit.
0	EN	0	Enable bit for PWM. Enabling this bit takes effect on the PWM outputs at the next scaled-clock event. 1: Enable PWM 0: Disable PWM

19 Watchdog Timer (WDT)

19.1 Overview

The watchdog begins counting down from the value loaded in the Load register when WDOGCLKEN = 1 and Interrupt Enable = 1. The timer can be reloaded by writing to the Interrupt clear register. If the timer counts down to zero without being cleared an Interrupt will be asserted and the timer will reload. If the timer counts down to zero again without being cleared the WDOGRES will be asserted.

Window mode is enabled by bit 2 of the control register and writing a value to the window value register. If the current count is higher than the window value when a clear is received this will cause a reset.

19.2 Watchdog Timer Register Details

Table 19.1: Watchdog Timer Registers

Register	Offset (Hex)
Load Register (LOAD)	0x0000
Current Count Value Register (VALUE)	0x0004
Control Register (CONTROL)	0x0008
Interrupt Clear Register (INT_CLR)	0x000C
Interrupt Status Register (RAW_INT_STAT)	0x0010
Masked Interrupt Status Register (MASK_INT_STAT)	0x0014
Window Value Register (WINDOW_VALUE)	0x0018
Lock Register (LOCK)	0x0C00

19.2.1 Load Register (LOAD)

This register contains the countdown value used by the Watchdog timer.

LOAD																**Offset = 0x0000**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	---															
W																
Reset	[11…1]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	---															
W																
Reset	[11…1]															

Table 19.2: Description of the Load Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	---	[11…1]	Count down value for watchdog timeout period

19.2.2 Current Count Value Register (VALUE)

This register reads the current value of the Watchdog timer count.

VALUE																**Offset = 0x0004**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	---															
W																
Reset	[11...1]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	---															
W																
Reset	[11...1]															

Table 19.3: Description of the Current Count Value Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	---	[11...1]	Reads the current countdown value of the watchdog timer

19.2.3 Control Register (CONTROL)

Address Offset: 0x08

Reset Value: 0x0000 0000 (read response)

This register contains the status of the Watchdog timer.

CONTROL																**Offset = 0x0008**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														WR	RE	IE
W																
Reset	[00…0]													0	0	0

Table 19.4: Description of the Control Register

Bit Number(S)	Bit Name	Reset State	Description
31-3	RESERVED	[00...0]	
2	Window Reset Enable (WR)	0	1: Enables reset output when an interrupt clear occurs before the timer has decremented below the value in the WINDOW_VALUE register. Reset Enable (bit 1) MUST be enabled for this to have an effect 0: Windowed mode is not enabled
1	Reset Enable (RE)	0	1: Enable reset output when the timer reaches 0 a second time (if the interrupt is not cleared the first time the timer reaches 0) 0: Reset output is not enabled
0	Interrupt Enable (IE)	0	1: Enable interrupt when the timer counts down to 0. 0: Interrupt not enabled, the timer will not count while this bit is '0'

19.2.4 Interrupt Clear Register (INT_CLR)

This register reads the current value of the Watchdog timer count.

INT_CLR																**Offset = 0x000C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	---															
Reset	[UU...U]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	---															
Reset	[UU...U]															

Table 19.5: Description of the Interrupt Clear Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	---	[UU...U]	Writing value 0x1357C0B1 to this location clears the watchdog Interrupt and reloads the timer from the value in LOAD. Other write values are ignored

19.2.5 Interrupt Status Register (RAW_INT_STAT)

This register contains the Interrupt status of the Watchdog timer.

RAW_INT_STAT																Offset = 0x0010
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																RS
W																
Reset	[00…0]															0

Table 19.6: Description of the Interrupt Status Register

Bit Number(S)	Bit Name	Reset State	Description
31-1	RESERVED	[UU…U]	
0	Raw Status (RS)	0	Raw status for the Watchdog Timer Interrupt 1: There is an Interrupt 0: There is not an Interrupt

19.2.6 Masked Interrupt Status Register (MASK_INT_STAT)

This register contains the Interrupt status of the Watchdog timer.

MASK_INT_STAT																Offset = 0x0014
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																MS
W																
Reset	[00…0]															0

Table 19.7: Description of the Masked Interrupt Status Register

Bit Number(S)	Bit Name	Reset State	Description
31-1	RESERVED	[00…0]	
0	Masked Status (MS)	0	Masked status for the Watchdog Timer Interrupt 1: There is an Interrupt 0: There is not an Interrupt

19.2.7 Window Value Register (WINDOW_VALUE)

This register contains the countdown value for the watchdog timer window mode.

WINDOW_VALUE																**Offset = 0x0018**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	---															
W																
Reset	[11…1]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	---															
W																
Reset	[11…1]															

Table 19.8: Description of Window Value Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	---	[11…1]	If the timer is greater than this value when the Interrupt is cleared, and Window Reset is enabled, a reset occurs

19.2.8 Lock Register (LOCK)

This register controls the locking of the load, control, and window registers associated with the watchdog timer.

LOCK																**Offset = 0x0C00**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																WL
W																
Reset	[00…0]															0

Table 19.9: Description of the Lock Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00…0]	
0	WDOGLOCK (WL)	0	1: Locks write access to the load, control, and window value registers

20 Real Time Counter (RTC)

20.1 Overview

UT32M0R500 includes a programmable 32-bit free-running real-time counter (RTC) module. The current value register is used to read the contents of the counter register at certain moments; counter wraps at matched period, or at maximum count; and counter wraps to a specified load register value.

For RTC example code and application specific interface (API), refer to the UT32M0R500 Software Development Kit (SDK).

Figure 20.1 shows the simplified block diagram for the real-time counter (RTC) module.

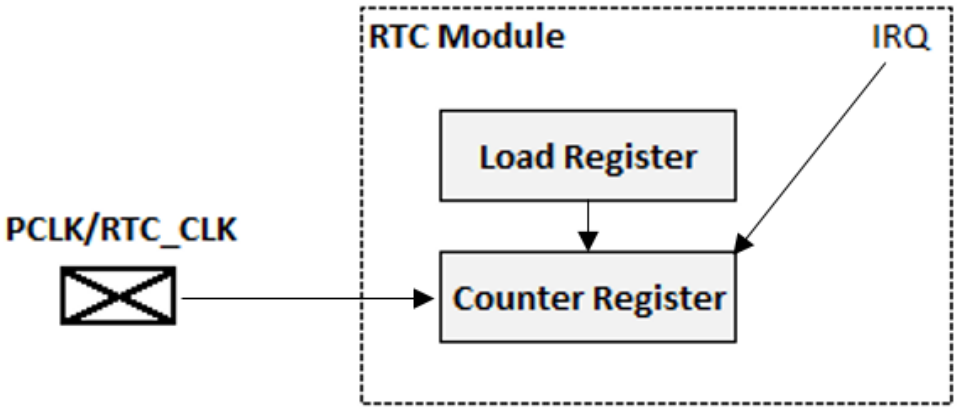


Figure 20.1: RTC Block Diagram

20.2 Functional Description

The A real-time clock is a special-purpose timer for measuring time or generating timer signals.

20.3 Operation

To generate a 20Hz interval, set WRAP_ENABLE to force the counter to wrap when a match occurs, write compare match register (CMR) with 2,500,000, set INT_MASK flag to 1, set INT_ENABLE flag to 1, and finally set the COUNTER_ENABLE flag to 1 to enable the timer module. The Interrupt service routine (ISR) generates an event when the counter matches (CMR) value of 2,500,000, then it wraps to 0.

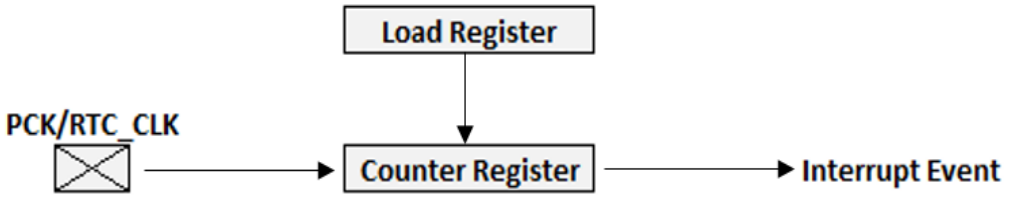


Figure 20.2: RTC Interval

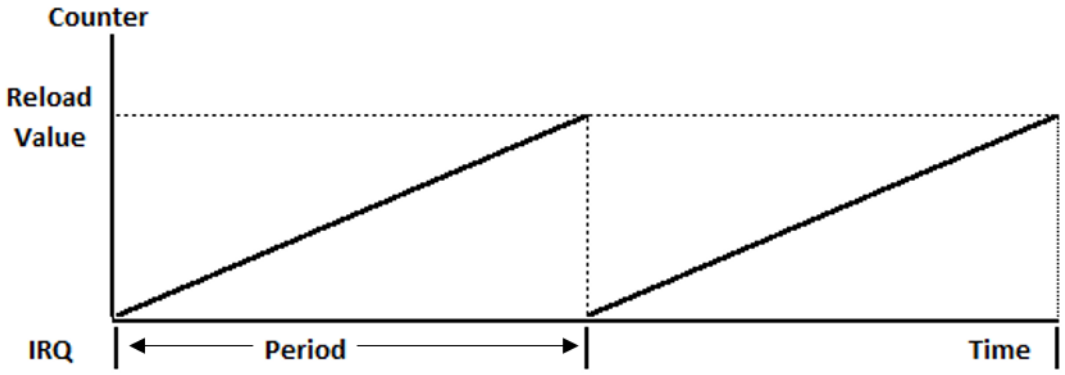


Figure 20.3: RTC Interval Diagram

20.4 RTC Register Details

Table 20.1: Up-Counter Registers

Register	Offset (Hex)
Current Counter Value Register (CCVR)	0x0000
Counter Match Register (CMR)	0x0004
Counter Load Register (CLR)	0x0008
Counter Control Register (CCR)	0x000c
Interrupt Status Register (STAT)	0x0010
Raw Interrupt Status Register (RSTAT)	0x0014
End-Of-Interrupt Register (EOI)	0x0018
Component Version Register (CMP_VER)	0x001c

20.4.1 Current Count Value Register (CCVR)

RTC_CCVR																Offset = 0x0000
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CCVR															
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CCVR															
W																
Reset	[00...0]															

Table 20.2: Description of the Current Count Value Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	CCVR	[00...0]	When read, this register is the current value of the internal counter. This value always is read coherently

20.4.2 Counter Match Register (CMR)

CMR																Offset = 0x0004
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMR															
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMR															
W																
Reset	[00...0]															

Table 20.3: Description of the Counter Match Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	CMR	[00...0]	When the internal counter register (CCVR) matches this register, an Interrupt is generated, providing Interrupt generation is enabled.

20.4.3 Counter Load Register (CLR)

CLR																**Offset = 0x0008**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLR															
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CLR															
W																
Reset	[00...0]															

Table 20.4: Description of the Counter Load Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	CLR	[00...0]	Value to be loaded into the current counter register

20.4.4 Counter Control Register (CCR)

CCR																**Offset = 0x000C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													WE	CE	IM	IE
W																
Reset	[00...0]												0	0	0	0

Table 20.5: Description of the Counter Control Register

Bit Number(s)	Bit Name	Reset State	Description
31-4	RESERVED	[00...0]	
3	WRAP_ENABLE (WE)	0	Allows the user to force the counter to wrap when a match occurs instead of waiting until the maximum count is reached 1: Wrap enable 0: Wrap disable
2	COUNTER _ENABLE (CE)	0	The user can control whether the counter is or isn't counting 1: Counter enabled 0: Counter disabled
1	INT_MASK (IM)	0	Allows the user to mask a generated Interrupt 1: Interrupt masked 0: Interrupt unmasked
0	INT_ENABLE (IE)	0	Allows the user to disable Interrupt generation 1: Interrupt enabled 0: Interrupt disabled

20.4.5 Interrupt Status Register (STAT)

STAT																**Offset = 0x0010**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																IP
W																
Reset	[00...0]															0

Table 20.6: Description of the Interrupt Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	INT_PENDING (IP)	0	Masked Interrupt status 1: Interrupt is active (regardless of polarity) 0: Interrupt is inactive

20.4.6 Interrupt Raw Status Register (RSTAT)

RSTAT																Offset = 0x0014
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																IPR
W																
Reset	[00…0]															0

Table 20.7: Description of the Interrupt Raw Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	INT_PENDING _RAW (IPR)	0	Raw Interrupt Status 1: Interrupt is active (regardless of polarity) 0: Interrupt is inactive

20.4.7 End of Interrupt Register (EOI)

EOI																Offset = 0x0018
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																IC
W																
Reset	[00…0]															0

Table 20.8: Description of the End of Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	INT_CLEAR (IC)	0	When read, clears the match Interrupt.

20.4.8 Component Version Register (CMP_VER)

CMP_VER																Offset = 0x001C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMP_VER															
W																
Reset	0x3230															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMP_VER															
W																
Reset	0x352A															

Table 20.9: Description of the Component Version Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	CMP_VER	0x3230 _352A	ASCII value for each number in the version, followed by * Ex: 0x3230_312A represents the version 2.01*

21 Universal Asynchronous Receiver/Transmitter (UART)

21.1 Overview

Universal Asynchronous Receiver/Transmitter (UART) is an asynchronous, bidirectional communication for transmitting and receiving data. At the transmitter, the UART converts the parallel data into a serial data stream, and vice versa at the receiver. It uses asynchronous communication between sender and receiver with a pre-agreed baud rate, and it uses bidirectional transmission by using separate wires for transmitting and receiving.

For UART example code and application specific interface (API), refer to the UT32M0R500 Software Development Kit (SDK).

Figure 21.1 shows the UART block diagram.

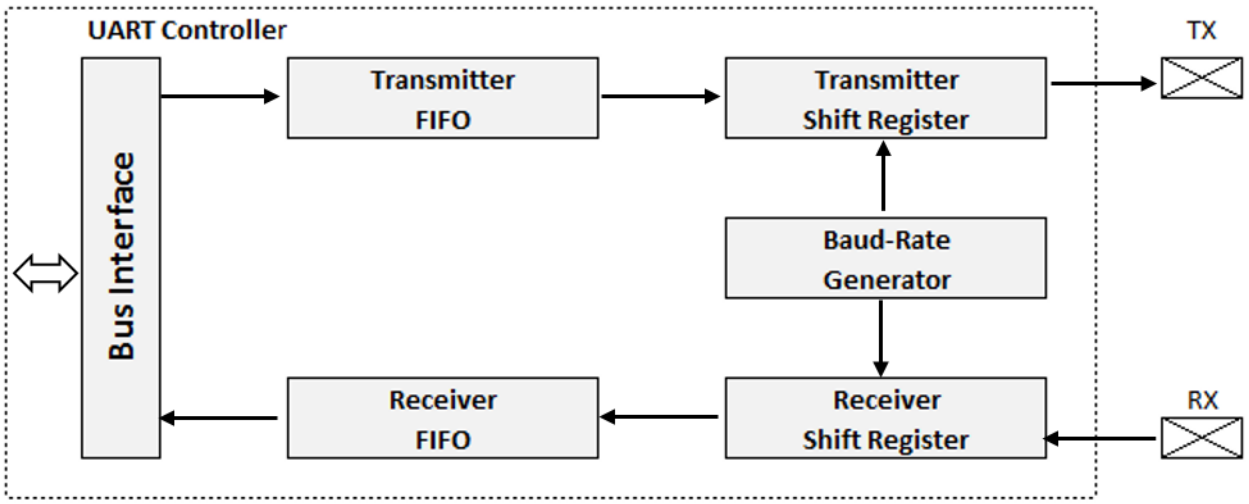


Figure 21.5: UART Block Diagram

21.2 Protocol

UART serial communication starts with the start bit, which drives the line low for one clock cycle; next comes 8-bits of data clocked sequentially from the transmitter; optionally, one parity bit can be added for error checking; and finally, the stop bit, which pulls the line high to indicate the end of transfer.

Figure 21.2 shows the UART transmit data frame.

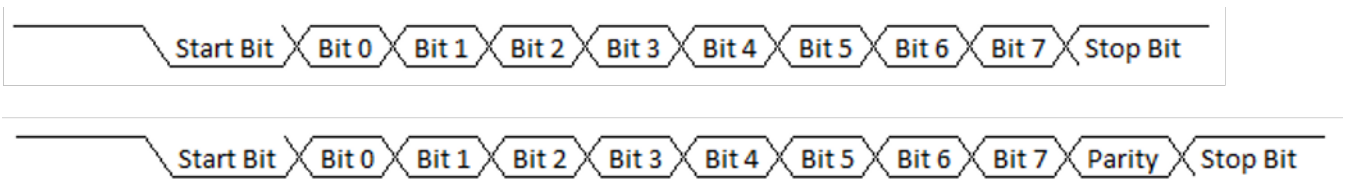


Figure 21.2: UART Transmit Data Frame

21.3 Character-Encoding Scheme

The data in the UART serial protocol encodes bytes using the ASCII code, which has 128 specified characters into 7-bit binary integers, or the UTF-8 code, which has 8 bits for international character encoding.

Table 21.1 shows both character-encoding schemes.

Table 21.1: Character-encoding Schemes

ASCII	UTF-8
Encodes 128 characters	Derived and compatible with ASCII
95 printable characters, i.e., a, b, 1, 2	Variable-width encoding
33 non-printable characters, i.e., back space, new line	Used for world wide web

21.4 Baud-Rate Generation

Each UART contains a 12-bit down-counting scaler to generate the desired baud-rate. The scaler is clocked by the system clock (PCLK) and generates a UART tick each time it underflows. It is reloaded with the value of the UART scaler reload register after each underflow. The resulting UART tick frequency should be 8 times the desired baud-rate. One appropriate formula to calculate the scaler value for a desired baud rate, using integer division where the remainder is discarded, is:

*scaler value = (PCLK) / (baud_rate * 8 + 7)*

To calculate the exact required scaler value use:

*scaler value = (PCLK) / (baud_rate * 8) - 1*

If the EC bit is set, the ticks will be generated with the same frequency as the external clock input instead of at the scaler underflow rate. In this case, the frequency of external clock must be less than half the frequency of the system clock.

21.5 Operation

21.5.1 Transmitter Operation

The transmitter is enabled through the TE bit in the UART control register. Data that is to be transferred is stored in the FIFO/holding register by writing to the data register. When ready to transmit, data is transferred from the transmitter FIFO/holding register to the transmitter shift register and converted to a serial stream on the transmitter serial output pin (TXD). It automatically sends a start bit followed by eight data bits, an optional parity bit, and one stop bit, see Figure 21.2. The least significant bit of the data is sent first.

Following the transmission of the stop bit, if a new character is not available in the transmitter FIFO, the transmitter serial data output remains high and the transmitter shift register empty bit (TS) will be set in the UART status register. Transmission resumes and the TS is cleared when a new character is loaded into the transmitter FIFO. When the FIFO is empty the TE bit is set in the status register. If the transmitter is disabled, it will immediately stop any active transmissions including the character currently being shifted out from the transmitter shift register. The transmitter holding register may not be loaded when the transmitter is disabled or when the FIFO (or holding register) is full. If this is done, data might be overwritten and one or more frames are lost.

The TF status bit (not to be confused with the TF control bit) is set if the transmitter FIFO is currently full and the TH bit is set as long as the FIFO is less than half full (less than half of entries in the FIFO contain data). The TF control bit enables FIFO Interrupts when set. The status register also contains a counter (TCNT) showing the current number of data entries in the FIFO.

When flow control is enabled, the CTSN input must be low in order for the character to be transmitted. If it is de-asserted in the middle of a transmission, the character in the shift register is transmitted and the transmitter serial output then remains inactive until CTSN is asserted again. If the CTSN is connected to a receivers RTSN, overrun can effectively be prevented.

21.5.2 Receiver Operation

The receiver is enabled for data reception through the receiver enable (RE) bit in the UART control register. The receiver looks for a high to low transition of a start bit on the receiver serial data input pin. If a transition is detected, the state of the serial input is sampled a half bit clocks later. If the serial input is sampled high the start bit is invalid and the search for a valid start bit continues. If the serial input is still low, a valid start bit is assumed and the receiver continues to sample the serial input at one bit time intervals (at the theoretical center of the bit) until the proper number of data bits and the parity bit have been assembled and one stop bit has been detected. The serial input is shifted through an 8-bit shift register where all bits have to have the same value before the new value is taken into account, effectively forming a low-pass filter with a cut-off frequency of 1/8 system clock.

The receiver also has a configurable FIFO which is identical to the one in the transmitter. As mentioned in the transmitter part, both the holding register and FIFO will be referred to as FIFO. During reception, the least significant bit is received first. The data is then transferred to the receiver FIFO and the data ready (DR) bit is set in the UART status register as soon as the FIFO contains at least one data frame. The parity, framing and overrun error bits are set at the received byte boundary, at the same time as the receiver ready bit is set. The data frame is not stored in the FIFO if an error is detected. Also, the new error status bits are OR'ed with the old values before they are stored into the status register. Thus, they are not cleared until written to with zeros from the AMBA APB bus. If both the receiver FIFO and shift registers are full when a new start bit is detected, then the character held in the receiver shift register will be lost and the overrun bit will be set in the UART status register. A break received (BR) is indicated when a BREAK has been received, which is a framing error with all data received being zero.

The RF status bit (not to be confused with the RF control bit) is set when the receiver FIFO is full. The RH status bit is set when the receiver FIFO is half-full (at least half of the entries in the FIFO contain data frames). The RF control bit enables receiver FIFO Interrupts when set. A RCNT field is also available showing the current number of data frames in the FIFO

21.6 Loop back mode

If the LB bit in the UART control register is set, the UART will be in loop back mode. In this mode, the transmitter output is internally connected to the receiver input and the RTSN is connected to the CTSN. It is then possible to perform loop back tests to verify operation of receiver, transmitter and associated software routines. In this mode, the outputs remain in the inactive state, in order to avoid sending out data.

21.7 FIFO debug mode

FIFO debug mode is entered by setting the debug mode bit in the control register. In this mode it is possible to read the transmitter FIFO and write the receiver FIFO through the FIFO debug register. The transmitter output is held inactive when in debug mode. A write to the receiver FIFO generates an Interrupt if receiver Interrupts are enabled.

21.8External Interrupt

Each UART module, UART0 and UART1, has an external Interrupt (IRQ) mapped to Interrupt Vectors 14-15 respectively. The UTM0R500 microcontroller based on the Cortex M0+ processor contains an internal nested vector Interrupt controller (NVIC) that supports up to 32 external IRQ's and a non-maskable Interrupt (NMI). NVIC handles the nested Interrupts automatically based on priorities and Interrupt number, and if an Interrupt is accepted, it communicates with the processor to allow it to execute the correct Interrupt handler.

For more information on external Interrupts and the nested vector Interrupt (NVIC), refer to **Error! Unknown switch argument.**

Within each module, two different kinds of Interrupts are available: normal Interrupts and FIFO Interrupts. For the transmitter, normal Interrupts are generated when transmitter Interrupts are enabled (TI), the transmitter is enabled and the transmitter FIFO goes from containing data to being empty. FIFO Interrupts are generated when the FIFO Interrupts are enabled (TF), transmissions are enabled (TE) and the UART is less than half-full (that is, whenever the TH status bit is set). This is a level Interrupt and the Interrupt signal is continuously driven high as long as the condition prevails. The receiver Interrupts work in the same way. Normal Interrupts are generated in the same manner as for the holding register. FIFO Interrupts are generated when receiver FIFO Interrupts are enabled, the receiver is enabled and the FIFO is half-full. The Interrupt signal is continuously driven high as long as the receiver FIFO is half-full (at least half of the entries contain data frames).

To reduce Interrupt occurrence a delayed receiver Interrupt is available. It is enabled using the delayed Interrupt enable (DI) bit. When enabled a timer is started each time a character is received and an Interrupt is only generated if another character has not been received within 4 character + 4 bit times. If receiver FIFO Interrupts are enabled a pending character Interrupt will be cleared when the FIFO Interrupt is active since the character causing the pending irq state is already in the FIFO and is noticed by the driver through the FIFO Interrupt.

There is also a separate Interrupt for break characters. When enabled, an Interrupt will always be generated immediately when a break character is received even when delayed receiver Interrupts are enabled. When break Interrupts are disabled no Interrupt will be generated for break characters when delayed Interrupts are enabled.

When delayed Interrupts are disabled the behavior is the same for the break Interrupt bit except that an Interrupt will be generated for break characters if receiver Interrupt enable is set even if break Interrupt is disabled. An Interrupt can also be enabled for the transmitter shift register. When enabled the core will generate an Interrupt each time the shift register goes from a non-empty to an empty state.

21.9 UART Register Details

Table 21.2: UART Registers

Register	Offset (Hex)
Data Register (DATA)	0x000
Status Register (STATUS)	0x004
Control Register (CONTROL)	0x008
Scaler Register (SCALER)	0x00C
FIFO Debug Register (FIFO_DEBUG)	0x010

21.9.1 Data Register (DATA)

DATA															Offset = 0x0000
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									DATA							
W																
Reset	[00…0]								[00…0]							

Table 21.3: Description of the Data Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	DATA	[00…0]	Rd = Receiver holding register or FIFO (read access) Wr = Transmitter holding register or FIFO (write access)

21.9.2 Status Register (STATUS)

STATUS																Offset = 0x0004
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RCNT						TCNT									
W																
Reset	[00…0]						[00…0]						[00…0]			

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R						RF	TF	RH	TH	FE	PE	OV	BR	TE	TS	DR
W																
Reset	[00…0]					0	0	0	1	0	0	0	0	1	1	0

Table 21.4: Description of the Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-26	RCNT	[00…0]	Receiver FIFO count (RCNT) – shows the number of data frames in the receiver FIFO NOTE: Maximum FIFO size is 8 bytes
25-20	TCNT	[00…0]	Transmitter FIFO count (TCNT) – shows the number of data frames in the transmitter FIFO NOTE: Maximum FIFO size is 8 bytes
19-11	RESERVED	[00…0]	
10	RF	0	Receiver FIFO full 1: The receiver FIFO is full 0: The receiver FIFO is not full
9	TF	0	Transmitter FIFO full 1: The transmitter FIFO is full 0: The transmitter FIFO is not full
8	RH	0	Receiver FIFO half-full 1: At least half of the FIFO is holding data 0: Less than half of the FIFO is holding data
7	TH	1	Transmitter FIFO half-full 1: The FIFO is less than half full 0: The FIFO is at least half full
6	FE	0	Framing error 1: A framing error was detected 0: No framing error detected
5	PE	0	Parity error 1: A parity error was detected 0: No parity error detected
4	OV	0	Overrun 1: One or more characters have been lost due to overrun 0: No characters lost due to overrun
3	BR	0	Break received 1: A BREAK has been received 0: No BREAK received
2	TE	1	Transmitter FIFO empty Indicates that the transmitter FIFO is empty
1	TS	1	Transmitter shift register empty 1: The transmitter shift register is empty 0: The transmitter shift register is not empty
0	DR	0	Data ready 1: New data is available in the receiver holding register 0: No new data ready

21.9.3 Control Register (CONTROL)

CONTROL																Offset = 0x0008
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FA															
W																
Reset	1	[00…0]														

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		SI	DI	BI	DB	RF	TF	EC	LB	FL	PE	PS	TI	RI	TE	RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21.5: Description of the Control Register

Bit Number(s)	Bit Name	Reset State	Description
31	FA	0	FIFOs available 1: Receiver and transmitter FIFOs are available 0: Only the holding register is available
30-15	RESERVED	[00…0]	
14	SI	0	Transmitter shift register empty Interrupt enable 1: An Interrupt will be generated when the transmitter shift register becomes empty 0: No Interrupt generated
13	DI	0	Delayed Interrupt enable 1: Delayed receiver Interrupts are enabled, and an Interrupt will only be generated for received characters after a delay of 4 character times + 4 bits if no new character has been received during that interval. This is only applicable if receiver Interrupt enable is set. 0: Delayed receiver Interrupts are disabled
12	BI	0	Break Interrupt enable 1: An Interrupt will be generated each time a break character is received 0: No Interrupt generated
11	DB	0	FIFO debug mode enable 1: The FIFO debug register can be read from and written to 0: The FIFO debug register cannot be read from or written to
10	RF	0	Receiver FIFO Interrupt enable 1: Receiver FIFO level Interrupts are enabled, and occur when RX FIFO becomes half-full 0: Receiver FIFO level Interrupts are not enabled
9	TF	0	Transmitter FIFO Interrupt enable 1: Transmitter FIFO level Interrupts are enabled, and occur when RX FIFO becomes half-full 0: Transmitter FIFO level Interrupts are not enabled
8	EC	0	External Clock 1: The UART scaler will be clocked by UARTI.EXTCLK 0: The UART scaler will be clocked by the internal clock

Bit Number(s)	Bit Name	Reset State	Description
7	LB	0	Loop back 1: Loop back mode is enabled 0: Loop back mode is disabled
6	FL	0	Flow control 1: Flow control is enabled using CTS/RTS (when implemented) 0: Flow control not enabled
5	PE	0	Parity enable 1: Parity generation and checking enabled (when implemented) 0: Parity generation and checking not enabled
4	PS	0	Parity select 1: Odd parity 0: Even parity
3	TI	0	Transmitter Interrupt enable 1: Interrupts are generated when characters are transmitted 0: Interrupts not generated
2	RI	0	Receiver Interrupt enable 1: Interrupts are generated when characters are received 0: Interrupts not generated
1	TE	0	Transmitter enable 1: Transmitter is enabled. Note that the FIFO contents and counters are not cleared when disabled. Prior to TE, the UART_SCALER must be written following each reset event 0: Transmitter not enabled
0	RE	0	Receiver enable 1: Receiver is enabled. Note that the FIFO contents and counters are not cleared when disabled. Note that prior to RE, the UART_SCALER must be written following each reset event. 0: Receiver not enabled

21.9.4 Scaler Register (SCALER)

SCALER																**Offset = 0x000C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					Scaler Reload Value											
W																
Reset	[00…0]				[00…0]											

Table 21.6: Description of the Scaler Register

Bit Number(S)	Bit Name	Reset State	Description
31-12	RESERVED	[00...0]	
11-0	Scaler Reload Value	[00...0]	Scaler reload value. This register must be written after a reset event, or serial I/O may not commence. A read after reset may have a legitimate data value, but this does not propagate into the TX and RX logic until a write takes place.

21.9.5 FIFO Debug Register (FIFO_DEBUG)

FIFO_DEBUG																**Offset = 0x0010**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									DATA							
W																
Reset	[00...0]								[00...0]							

Table 21.7: Description of the FIFO Debug Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00...0]	
7-0	SOFT_RESET	[00...0]	Read: transmitter holding register or FIFO Write: Receiver holding register or FIFO

22 Serial Peripheral Interface (SPI)

22.1 Overview

The UT32M0R500 contains one serial peripheral interface (SPI) controller configured as a master. The SPI operates using Motorola SPI protocol. It has TX and RX FIFO buffers; each buffer has a size of 16 bytes; each byte has a width of 4-16 bits. It has programmable RX sample point delay for increasing transfer rate. It has programmable baud rate at even-integer division of system clock (PCLK). It supports all SPI SCPH and SCPOL, EEPROM read and up to 3 dedicated slave selects.

The SPI bus is a four-wire serial synchronous bus with a dedicated clock. To communicate with slave devices, the UT32M0R500 has three dedicated chip-select-lines (CS) to address up to three slaves. The data communication lines are master-out-slave-in (MOSI) to the slave, and master-in-slave-out (MISO) to the master.

For SPI example code and application specific interface (API), refer to request access to the UT32M0R500 Software Development Kit (SDK).

Figure 22.1 shows the SPI block diagram.

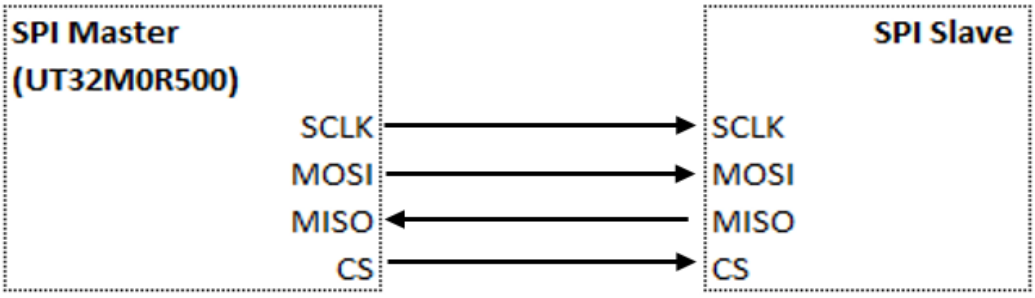


Figure 22.6: SPI Block Diagram

22.2 Clock generation

The SPI contains a 16-bit prescaler register (BAUDR) to generate the desired baud-rate. The scaler is clocked by the system clock (PCLK) and generates the SPI clock (SCLK). The frequency of the SCLK is derived from the following equation:

$$SCLK = \frac{PCLK\,(system\ clock)}{Clock\ Divider(SCKDV)}$$

22.3 Operation

22.3.1 Clocking Modes

Once the frequency is set, there are four possible clock modes available for programming the clock edge used for data sampling and data toggling, see Figure 22.2 – Figure 22.5. The clock phase (SCPH) determines whether the serial transfer begins with the falling edge of the slave select signal or the first edge of the serial clock.

With the SPI, the clock polarity (SCPOL) configuration parameter determines whether the inactive state of the serial clock is high or low. To transmit data, both SPI peripherals must have identical serial clock phase (SCPH) and clock polarity (SCPOL) values. The data frame can be 4 to 16-bits in length.

When the configuration parameter SCPH = 0, data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the MOSI and MISO lines prior to the first serial clock edge.

When the configuration parameter SCPH = 1, both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the master and slave peripherals on the leading edge of the serial clock. During continuous data frame transfers, the slave select line may be held active-low until the last bit of the last frame has been captured.

Figure 22.2 – Figure 22.5 shows the different SPI modes.

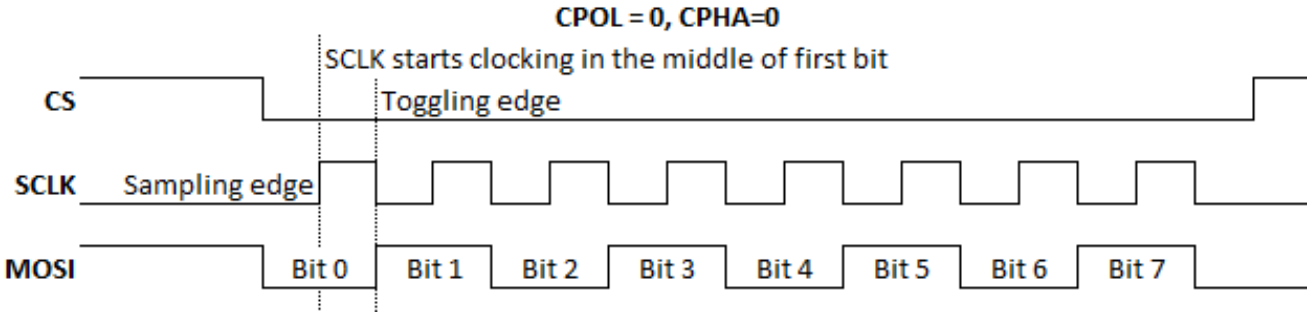


Figure 22.2: Mode 0

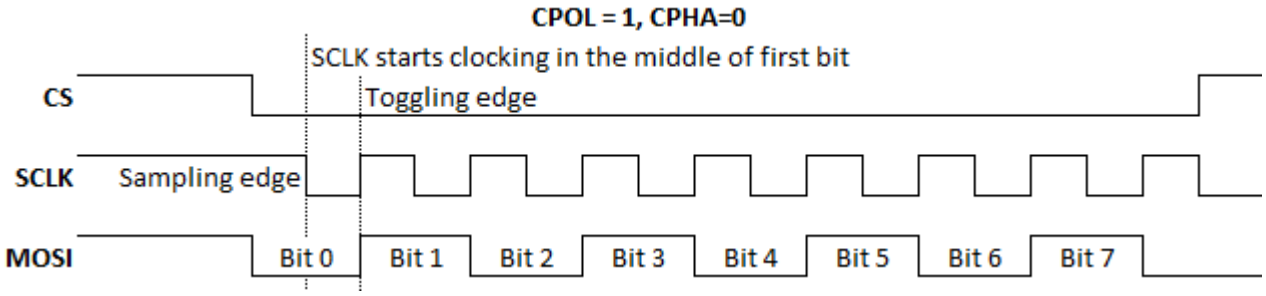


Figure 22.7: Mode 2

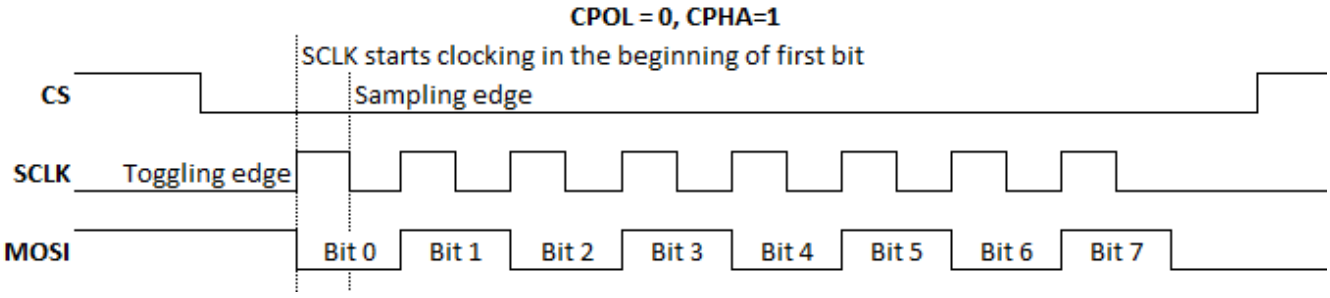


Figure 22.8: Mode 1

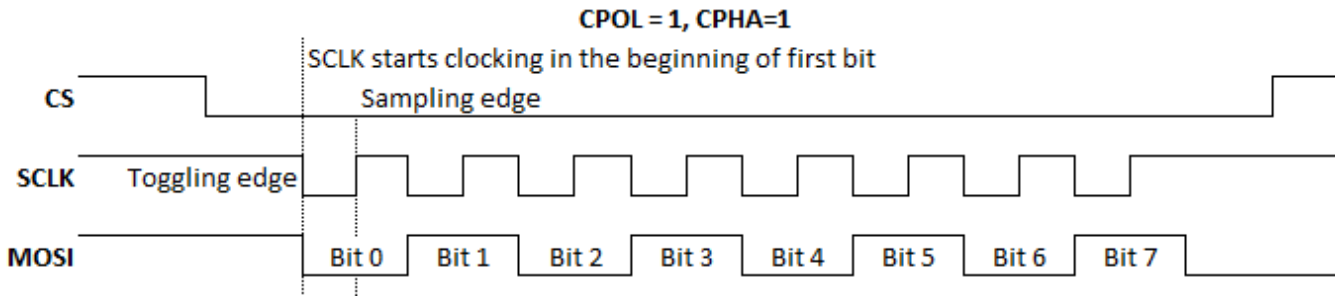


Figure 22.9: Mode 3

22.4 Transmit and Receive FIFO Buffers

The width of both transmit and receive FIFO buffers is fixed at 16, due to the serial specifications, which state that a serial transfer (data frame) can be 4 to 16 bits in length. Data frames that are less than 16-bits in size must be right-justified when written into the transmit FIFO buffer. The shift control logic automatically right-justifies receive data in the receive FIFO buffer.

Each data entry in the FIFO buffers contains a single data frame. It is impossible to store multiple data frames in a single FIFO location; for example, you may not store two 8-bit data frames in a single FIFO location. If an 8-bit data frame is required, the upper 8-bits of the FIFO entry are ignored or unused when the serial shifter transmits the data. The transmit FIFO is loaded by APB write commands to the data register (DR).

22.5 External Interrupt

The SPI module has an external Interrupt (IRQ) mapped to Interrupt Vector 20. The UTM0R500 microcontroller based on the Cortex M0+ processor contains an internal nested vector Interrupt controller (NVIC) that supports up to 32 external IRQ's and a non-maskable Interrupt (NMI). NVIC handles the nested Interrupts automatically based on priorities and Interrupt number, and if an Interrupt is accepted, it communicates with the processor to allow it to execute the correct Interrupt handler.

For more information on external Interrupts and the nested vector Interrupt (NVIC), refer to Nested Vector Interrupt Controller (NVIC).

22.6 FIFO Empty Interrupt

Data are popped (removed) from the transmit FIFO by the shift control logic into the transmit shift register. The transmit FIFO generates a FIFO empty Interrupt request by setting TXIEM flag to 1 when the number of entries in the FIFO is less than or equal to the FIFO threshold value. The threshold value, set through the programmable register TXFTLR, determines the level of FIFO entries at which an Interrupt is generated. The threshold value allows you to provide early indication to the processor that the transmit FIFO is nearly empty. A transmit FIFO overflow Interrupt (TXOIM) is generated if you attempt to write data into an already full transmit FIFO.

22.7 FIFO Full Interrupt

Data are popped from the receive FIFO by APB read commands to the data register (DR). The receive FIFO is loaded from the receive shift register by the shift control logic. The receive FIFO generates a FIFO-full Interrupt request (RXFIM) when the number of entries in the FIFO is greater than or equal to the FIFO threshold value plus 1. The threshold value, set through programmable register (RXFTLR), determines the level of FIFO entries at which an Interrupt is generated.

22.8 FIFO Overrun Interrupt

The threshold value allows you to provide early indication to the processor that the receive FIFO is nearly full. A receive FIFO overrun Interrupt (RXOIM) is generated when the receive shift logic attempts to load data into a completely full receive FIFO. However, this newly received data are lost. A receive FIFO underflow Interrupt (TXUIM) is generated if you attempt to read from an empty receive FIFO. This alerts the processor that the read data are invalid.

Note: The transmit and receive FIFO buffers are cleared when SSI_EN flag is set to 0 or when it is reset.

22.9 Transfer Modes

When transferring data on the serial bus, the SPI controller operates in the modes discussed in this section. The transfer mode (TMOD) is set by writing to control register 0 (CTRLR0).

22.9.1 Transmit and Receive, TMOD = 00

When TMOD = 2‘b00, both transmit and receive logic are valid. The data transfer occurs as normal according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the TX line to the target device, which replies with data on the RX line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

22.9.1 Transmit Only, TMOD = 01

When TMOD = 2‘b01, the receive data are invalid and should not be stored in the receive FIFO. The data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the TX line to the target device, which replies with data on the RX line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO. The data in the receive shift register is overwritten by the next transfer. You should mask Interrupts originating from the receive logic when this mode is entered.

22.9.3 Receive Only, TMOD = 10

When TMOD = 2‘b10, the transmit data are invalid. When configured as a slave, the transmit FIFO is never popped in Receive Only mode. The TX output remains at a constant logic level during the transmission. The data transfer occurs as normal according to the selected frame format (serial protocol). The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. You should mask Interrupts originating from the transmit logic when this mode is entered.

22.9.4 EEPROM Read, TMOD = 11

When TMOD = 2‘b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. Typically, this takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). During the transmission of the opcode and address, no data is captured by the receive logic (as long as the SPI master is transmitting data on its TX line, data on the RX line is ignored). The SPI master continues to transmit data until the transmit FIFO is empty. Therefore, you should ONLY have enough data frames in the transmit FIFO to supply the opcode and address to the EEPROM. If more data frames are in the transmit FIFO than are needed, then read data is lost.

When the transmit FIFO becomes empty (all control information has been sent), data on the RX line is valid and is stored in the receive FIFO; the TX output is held at a constant logic level. The serial transfer continues until the number of data frames received by the SPI master matches the value of the NDF field in the CTRLR1 register +1.

22.10 Data Transfers

Data transfers are started by the SPI master device. When the SPI controller is enabled (SPI.SPIENR.SSI_EN=1), at least one valid data entry is present in the transmit FIFO and a serial-slave device is selected. When actively transferring data, the busy flag (BUSY) in the status register (SR) is set. You must wait until the busy flag is cleared before attempting a new serial transfer.

The BUSY status is not set when the data are written into the transmit FIFO. This bit gets set only when the target slave has been selected and the transfer is underway. Before polling the BUSY status, you should first poll the TFE status flag (waiting for 1) or wait for BAUDR * SCLK clock cycles.

Note: SCLK is the clock speed that the SPI peripheral is running.

22.10.1 SPI Register Details

Table 22.1: SPI Registers

Register	Offset (Hex)
Control Register 0 (CTRLR0)	0x00
Control Register 1 (CTRLR1)	0x04
SSI Enable Register (SPIENR)	0x08
Microwire Control Register (MWCR)	0x0C
Slave Enable Register (SER)	0x10
Baud Rate Select Register (BAUDR)	0x14
Transmit FIFO Threshold Level Register (TXFTLR)	0x18
Receive FIFO Threshold Level Register (RXFTLR)	0x1C
Transmit FIFO Level Register (TXFLR)	0x20
Receive FIFO Level Register (RXFLR)	0x24
Status Register (SR)	0x28
Interrupt Mask Register (IMR)	0x2C
Interrupt Status Register (ISR)	0x30
Raw Interrupt Status Register (RISR)	0x34
Transmit FIFO Overflow Interrupt Clear Register (TXOICR)	0x38
Receive FIFO Overflow Interrupt Clear Register (RXOICR)	0x3C
Receive FIFO Underflow Interrupt Clear Register (RXUICR)	0x40
Multi-Master Interrupt Clear Register (MSTICR)	0x44
Interrupt Clear Register (ICR)	0x48
Reserved	0x4C
Reserved	0x50
Reserved	0x54
Identification Register (IDR)	0x58
CoreKit Version ID Register (SSI_COMP_VERSION)	0x5C
DR Register (DR)	0x60 – 0xEC
RXD Sample Delay Register (RX_SAMPLE_DLY)	0xF0

22.10.2 Control Register 0 (CTRLR0)

CTRLR0																Offset = 0x0000
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CFS				SRL		TMOD		SPL	SPH	FRF		DFS			
W																
Reset	[00…0]				0	0	00		0	0	00		0111			

Table 22.2: Description of Control Register 0

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-12	CFS	[00…0]	Control Frame Size: Select the length of the control word for the Microwire frame format
11	SRL	0	Shift Register Loop. Testing Purposes only. When internally active, connects the transmit shift register output to the receive shift register input 1: Test mode operation 0: Normal mode operation
10	RESERVED	0	
9-8	TMOD	00	In transmit-only mode, data received from the external device is not valid and is not stored in the receive FIFO memory; it is overwritten on the next transfer. In receive-only mode, transmitted data are not valid. After the first write to the transmit FIFO, the same word is retransmitted for the duration of the transfer. In transmit-and-receive mode, both transmit and receive data are valid. The transfer continues until the transmit FIFO is empty. Data received from the external device are stored into the receive FIFO memory, where it can be accessed by the host processor. 11: EEPROM Read 10: Receive 01: Transmit only 00: Transmit and receive
7	SCPOL (SPL)	0	1: Inactive state of serial clock is high 0: Inactive state of serial clock is low
6	SCPH (SPH)	0	1: Serial clock toggles at the start of first data bit 0: Serial clock toggles in middle of first data bit
5-4	FRF	00	00: Frame format – Read only
3-0	DFS	0111	Data Frame Size DFS+1 is the size of the SPI data field (DFS = 3 to 15 are valid). (The default value of 7 sets the data transfer to 8 bits.) When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data

22.10.3 Control Register 1 (CTRLR1)

CTRLR1																Offset = 0x0004
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NDF															
W																
Reset	[00…0]															

Table 22.3: Description of Control Register 1

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-0	NDF	[00…0]	Number of data frames. When TMOD = 10 or TMOD = 11, this register field sets the number of data frames to be continuously received by the SPI_Master. The SPI_Master continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables it to receive up to 64 KB of data in a continuous transfer

22.10.4 SSI Enable Register (SPIENR)

SPIENR																Offset = 0x0008
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																SE
W																
Reset	[00…0]															0

Table 22.4: Description of the SSI Enable Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00…0]	
0	SSI_EN (SE)	0	Enable and disables all SPI_Master operations. When Disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the SPI_Master control registers when enabled. When disabled, the SSI_Sleep output is set (after a delay) to inform the system that it is safe to remove the SSI_CLK, thus saving power consumption in the system 1: Enable SPI Master operations 0: Disables SPI Master operations

22.10.5 Microwire Control Register (MWCR)

REG NAME																**Offset = 0x000C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																
Reset	[00…0]															

Table 22.5: Description of the Microwire Control Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	RESERVED	[00…0]	Not supported

22.10.6 Slave Enable Register (SER)

SER																**Offset = 0x0010**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														SER		
W																
Reset	[00…0]													000		

Table 22.6: Description of the Slave Enable Register

Bit Number(s)	Bit Name	Reset State	Description
31-3	RESERVED	[00…0]	
2-0	SER	000	Each bit in this register corresponds to a slave select line (CSn) from the SPI_Master master. When a bit in this register is set (1), the corresponding slave select line from the master is activated when a serial transfer begins. It should be noted that setting or clearing bits in this register have no effect on the corresponding slave select outputs until a transfer is started. When not operating in broadcast mode, only one bit in this field should be set. 1: Selected 0: Not Selected

22.10.7 Baud Rate Select Register (BAUDR)

BAUDR																	Offset = 0x0014
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R																	
W																	
Reset	[00…0]																

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SCKDV															
W																
Reset	[00…0]															

Table 22.7: Description of the Baud Rate Select Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-0	SCKDV	[00…0]	SSI Clock Divider. The register derives the frequency of the serial clock that regulates the data transfer. The 16-bit field in this register defines the SCLK divider value. It is impossible to write to this register when the SPI_Master is enabled. The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (SCLK) is disabled. The frequency of the SCLK is derived from the following equation: SCLK=PCLK/SCKDV where SCKDV is any even value between 2 and 65534.

22.10.8 Transmit FIFO Threshold Level Register (TXFTLR)

TXFTLR																Offset = 0x0018
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													TFT			
W																
Reset	[00…0]												[00…0]			

Table 22.8: Description of the Transmit FIFO Threshold Level Register

Bit Number(s)	Bit Name	Reset State	Description
31-4	RESERVED	[00…0]	
3-0	TFT	[00…0]	Tx FIFO Threshold. When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty Interrupt is triggered. If you attempt to set bits [7:0] of this register to a value greater than or equal to the depth of the FIFO-1 (1), this field is not written and retains its current value.

22.10.9 Receive IFO Threshold Level Register (RXFTLR)

RXFTLR																Offset = 0x001C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													RFT			
W																
Reset	[00…0]												[00…0]			

Table 22.9: Description of the Receive FIFO Threshold Level Register

Bit Number(S)	Bit Name	Reset State	Description
31-4	RESERVED	[00…0]	
3-0	RFT	[00…0]	Rx FIFO Threshold. When the number of receive FIFO entries is greater than or equal to this value + 1, the receive FIFO full Interrupt is triggered. If you attempt to set the value of this register to a value greater than the depth of this FIFO (2), this field is not written and retains its current value.

22.10.10 Transmit FIFO Level Register (TXFLR)

TXFLR																**Offset = 0x0020**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													TXTFL			
W																
Reset	[00…0]												[00…0]			

Table 22.10: Description of the Transmit FIFO Level Register

Bit Number(s)	Bit Name	Reset State	Description
31-4	RESERVED	[00…0]	
3-0	TXTFL	[00…0]	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

22.10.11 Receive FIFO Level Register (RXFLR)

RXFLR																**Offset = 0x0024**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													RXTFL			
W																
Reset	[00…0]												[00…0]			

Table 22.11: Description of the Receive FIFO Level Register

Bit Number(s)	Bit Name	Reset State	Description
31-4	RESERVED	[00…0]	
3-0	RXTFL	[00…0]	This register contains the number of valid data entries in the receive FIFO memory. This register can be read at any time.

22.10.12 Status Register (SR)

SR																**Offset = 0x0028**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R										DC		RF	RN	TF	TN	B
W																
Reset	[00…0]									0	0	0	0	1	1	0

Table 22.12: Description of the Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-7	RESERVED	[00…0]	
6	DCOL (DC)	0	Data Collision Error. This bit is set if the SS_IN_N input is asserted by another master, while the SPI_Master is in the middle of the transfer. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. 1: Transmit data collision error 0: No error
5	RESERVED	0	
4	RFF (RF)	0	Receive FIFO Full 1: Receive FIFO is completely full 0: Receive FIFO has at least one empty location
3	RFNE (RN)	0	Receive FIFO Not Empty 1: Receive FIFO contains one or more entries 0: Receive FIFO is empty
2	TFE (TF)	1	Transmit FIFO Empty 1: Transmit FIFO is completely empty 0: Transmit FIFO contains at least one or more entries
1	TFNE (TN)	1	Transmit FIFO Not Full 1: Transmit FIFO contains one or more empty locations 0: Transmit FIFO is full
0	BUSY (B)	0	SSI Busy Flag 1: Indicates that a serial transfer is in progress 0: SPI master is idle or disabled

22.10.13 Interrupt Mask Register (IMR)

This read/write register masks or enables all Interrupts generated by the SPI_Master. Masking is accomplished by clearing a bit to value '0'.

IMR																**Offset = 0x002C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R											MT	RF	RO	RU	TO	TE
W																
Reset	[00…0]										1	1	1	1	1	1

Table 22.13: Description of the Interrupt Mask Register

Bit Number(s)	Bit Name	Reset State	Description
31-6	RESERVED	[00…0]	
5	MSTIM (MT)	1	Multi-Master Contention Interrupt Mask 1: SSI_MST_INTR Interrupt is enabled 0: SSI_MST_INTR Interrupt is masked
4	RXFIM (RF)	1	Receive FIFO Full Interrupt Mask 1: SSI_RXF_INTR Interrupt is enabled 0: SSI_RXF_INTR Interrupt is masked
3	RXOIM (RO)	1	Receive FIFO Overflow Interrupt Mask 1: SSI_RXO_INTR Interrupt is enabled 0: SSI_RXO_INTR Interrupt is masked
2	RXUIM (RU)	1	Receive FIFO Underflow Interrupt Mask 1: SSI_RXU_INTR Interrupt is enabled 0: SSI_RXU_INTR Interrupt is masked
1	TXOIM (TO)	1	Transmit FIFO Overflow Interrupt Mask 1: SSI_TXO_INTR Interrupt is enabled 0: SSI_TXO_INTR Interrupt is masked
0	TXIEM (TE)	1	Transmit FIFO Empty Interrupt Mask 1: SSI_TXE_INTR Interrupt is enabled 0: SSI_TXE_INTR Interrupt is masked

22.10.14 Interrupt Status Register (ISR)

ISR																Offset = 0x0030
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R											MT	RF	RO	RU	TO	TE
W																
Reset	[00…0]										0	0	0	0	0	0

Table 22.14: Description of the Interrupt Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-6	RESERVED	[00...0]	
5	MSTIS (MT)	0	Multi-Master Contention Interrupt Status 1: SSI_MST_INTR Interrupt is active after masking 0: SSI_MST_INTR Interrupt is not active after masking
4	RXFIS (RF)	0	Receive FIFO Full Interrupt Status 1: SSI_RXF_INTR Interrupt is active after masking 0: SSI_RXF_INTR Interrupt is not active after masking
3	RXOIS (RO)	0	Receive FIFO Overflow Interrupt Status 1: SSI_RXO_INTR Interrupt is active after masking 0: SSI_RXO_INTR Interrupt is not active after masking
2	RXUIS (RU)	0	Receive FIFO Underflow Interrupt Status 1: SSI_RXU_INTR Interrupt is active after masking 0: SSI_RXU_INTR Interrupt is not active after masking
1	TXOIS (TO)	0	Transmit FIFO Overflow Interrupt Status 1: SSI_TXO_INTR Interrupt is active after masking 0: SSI_TXO_INTR Interrupt is not active after masking
0	TXEIS (TE)	0	Transmit FIFO Empty Interrupt Status 1: SSI_TXE_INTR Interrupt is active after masking 0: SSI_TXE_INTR Interrupt is not active after masking

22.10.15 Raw Interrupt Status Register (RISR)

RISR															Offset = 0x0034
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R											MT	RF	RO	RU	TO	TE
W																
Reset	[00…0]										0	0	0	0	0	0

Table 22.15: Description of the Raw Interrupt Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-6	RESERVED	[00...0]	
5	MSTIS (MT)	0	Multi-Master Contention Raw Interrupt Status 1: SSI_MST_INTR Interrupt is active prior to masking 0: SSI_MST_INTR Interrupt is not active prior to masking
4	RXFIS (RF)	0	Receive FIFO Full Raw Interrupt Status 1: SSI_RXF_INTR Interrupt is active prior to masking 0: SSI_RXF_INTR Interrupt is not active prior to masking
3	RXOIS (RO)	0	Receive FIFO Overflow Raw Interrupt Status 1: SSI_RXO_INTR Interrupt is active prior to masking 0: SSI_RXO_INTR Interrupt is not active prior to masking
2	RXUIS (RU)	0	Receive FIFO Underflow Raw Interrupt Status 1: SSI_RXU_INTR Interrupt is active prior to masking 0: SSI_RXU_INTR Interrupt is not active prior to masking
1	TXOIS (TO)	0	Transmit FIFO Overflow Raw Interrupt Status 1: SSI_TXO_INTR Interrupt is active prior to masking 0: SSI_TXO_INTR Interrupt is not active prior to masking
0	TXEIS (TE)	0	Transmit FIFO Empty Raw Interrupt Status 1: SSI_TXE_INTR Interrupt is active prior to masking 0: SSI_TXE_INTR Interrupt is not active prior to masking

22.10.16 Transmit FIFO Overflow Interrupt Clear Register (TXOICR)

TXOICR																Offset = 0x0038
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																TO
W																
Reset	[00…0]															0

Table 22.16: Description of the Transmit FIFO Overflow Interrupt Clear Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00…0]	
0	TXOICR (TO)	0	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the Interrupt. A read from this register clears the SSI_TXO_INTR Interrupt. Writing has no effect.

22.10.17 Receive FIFO Overflow Interrupt Clear Register (RXOICR)

RXOICR																Offset = 0x003C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																RO
W																
Reset	[00…0]															0

Table 22.17: Description of the Receive FIFO Overflow Interrupt Clear Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	RXOICR (RO)	0	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the Interrupt. A read from this register clears the SSI_RXO_INTR Interrupt. Writing has no effect.

22.10.18 Receive FIFO Underflow Interrupt Clear Register (RXUICR)

RXUICR																**Offset = 0x0040**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																RU
W																
Reset	[00...0]															0

Table 22.18: Description of the Receive FIFO Underflow Interrupt Clear Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	RXUICR (RU)	0	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the Interrupt. A read from this register clears the SSI_RXO_INTR Interrupt. Writing has no effect.

22.10.19 Multi-Master Interrupt Clear Register (MSTICR)

REG NAME																**Offset = 0x0044**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																MT
W																
Reset	[00...0]															0

Table 22.19: Description of the Multi-Master Interrupt Clear Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	MSTICR (MT)	0	Clear Multi-Master Contention Interrupt. This register reflects the status of the Interrupt. A read from this register clears the SSI_MST_INTR Interrupt. Writing has no effect.

22.10.20 Interrupt Clear Register (ICR)

ICR																**Offset = 0x0048**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																ICR
W																
Reset	[00...0]															0

Table 22.20: Description of the Interrupt Clear Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	ICR	0	Clear Interrupts. This register is set if any of the Interrupts are active. A read clears the SSI_TXO_INTR, SSI_RXU_INTR, SSI_RXO_INTR, and the SSI_MST_INTR Interrupts. Writing to this register has no effect

22.10.21 Identification Register (IDR)

IDR																**Offset = 0x0058**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IDCODE [31:16]															
W																
Reset	[11...1]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDCODE [15:0]															
W																
Reset	[11...1]															

Table 22.21: Description of the Identification Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	IDCODE	[11…1]	This register contains the peripherals identification code, SSI_ID

22.10.22 CoreKit Version ID Register (SSI_COMP_VERSION)

SSI_COMP_VERSION																**Offset = 0x005C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SSI_COMP_VERSION [31:16]															
W																
Reset	0x3430															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SSI_COMP_VERSION [15:0]															
W																
Reset	0x302A															

Table 22.22: Description of the CoreKit Version ID Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	SSI_COMP _VERSION	0x3430 _302A	Contains the hex representation of the Synopsys component version. Consists of ASCII value for each number in the version, followed by *. For example 32_30_31_2A represents the version 2.01*.

22.10.23 DR Register (DR)

DR																**Offset = 0x0060-0x00EC**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DR [31:16]															
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DR [15:0]															
W																
Reset	[00…0]															

Table 22.23: Description of the DR Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	DR	[00…0]	The DR register in the SPI_Master occupies thirty-six 32-bit address locations of the memory map to facilitate AHB burst transfers. Writing to any of these address locations has the same effect as pushing the data from the pwdata bus into the transmit FIFO. Reading from any of these locations has the same effect as popping data from the receive FIFO onto the prdata bus. The FIFO buffers on the SPI_Master are not addressable. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

22.10.24 RXD Sample Delay Register (RX_SAMPLE_DLY)

RX_SAMPLE_DLY																**Offset = 0x00F0**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									RSD							
W																
Reset	[00…0]								[00…0]							

Table 22.24: Description of the RXD Sample Delay Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	RSD	[00…0]	Receive Data (RXD) Sample Delay This register is used to delay the sample of the RXD input signal. Each value represents a single SSI_CLK delay on the sample of the RXD signal. NOTE: If this register is programmed with a value that exceeds the depth of the internal shift registers (SSI_RX_DLY_SR_DEPTH), a zero (0) delay will be applied to the RXD sample. It is impossible to write to this register when the SPI_Master is enabled; the SPI_Master is enabled and disabled by writing to the SSIENR register.

23 Inter-Integrated Circuit (I2C)

23.1 Overview

The UT32M0R500 contains two (I2C) controllers supporting master or slave mode of operation, 7-bit or 10-bit addressing and dynamic updating of 12C address without losing the bus. It has TX and RX FIFO buffers; each buffer has a size of 8 bytes. For multi master bus operation, each I2C supports collision detection and clock synchronization. For data integrity, each I2C supports on-chip filtering to reject spikes on the bus data line and user-selectable bit filtering length. Finally, each I2C supports 14 maskable Interrupts combined to one Cortex M0+ external Interrupt.

The I2C bus is a two-wire serial interface, consisting of a serial data line (SDA) and a serial clock (SCL). These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a “transmitter” or “receiver,” depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. A master is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave. I2C is a serial, 8-bit oriented, bidirectional data transfers can be made at 100Kbits/s in the standard mode, up to 400Kbits/s in the Fast mode, or up to 1Mbits/s in fast Mode plus (high speed mode is not supported on the UT32M0R500).

For I2C bus specification and user manual details, refer to “UM10204 I2C-bus specification and user manual”.

For the UT32M0R500 I2C example code and application specific interface (API), refer to the UT32M0R500 Software Development Kit (SDK).

Figure 23.1 shows the I2C block diagram for standard mode. The value of the pull-up resistor depends on the speed of the bus: recommended resistances are 4.7K for data rates below 100Kbps, 2.2K for standard mode and 1K for fast mode.

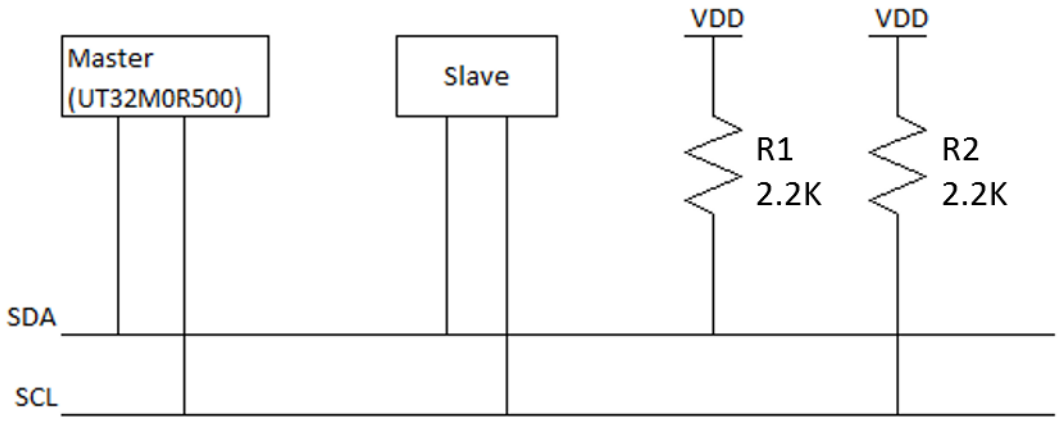


Figure 23.1: I2C Block Diagram for Standard Mode

23.2 Operation

The I2C can be controlled via software to be either:

- An I2C master only, communicating with other I2C slaves
- An I2C slave only, communicating with one more I2C masters

By default, the UT32M0R500 I2C operates in master mode.

The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. The acknowledgement of data is sent by the device that is receiving data, which can be either a master or a slave. As mentioned previously, the I2C protocol also allows multiple masters to reside on the I2 C bus and uses an arbitration procedure to determine bus ownership.

Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that is then followed by the slave's address and a control bit to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge (ACK) pulse after the address.

If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver gets one byte of data. This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with the ACK pulse. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition.

Figure 23.2. shows the data transfer of the I2C bus.

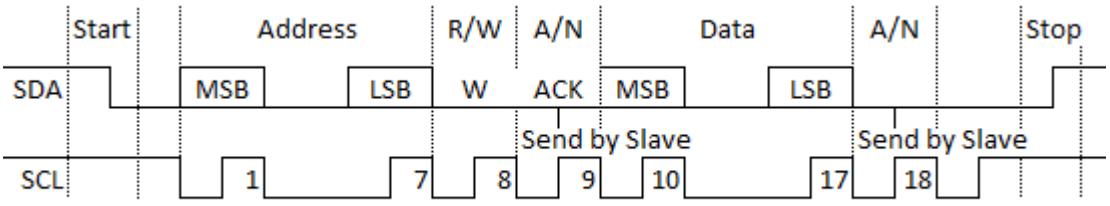


Figure 23.2: Data transfer on the I2C Bus

23.3 I2C Protocol

23.3.1 Start and Stop Conditions

When the bus is idle, both the SCL and SDA signals are pulled high through external pull-up resistors on the bus. When the master wants to start a transmission on the bus, the master issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is 1. When the master wants to terminate the transmission, the master issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1. When data is being transmitted on the bus, the SDA line must be stable when SCL is 1.

Figure 23.3. shows the timing of the START and STOP conditions.

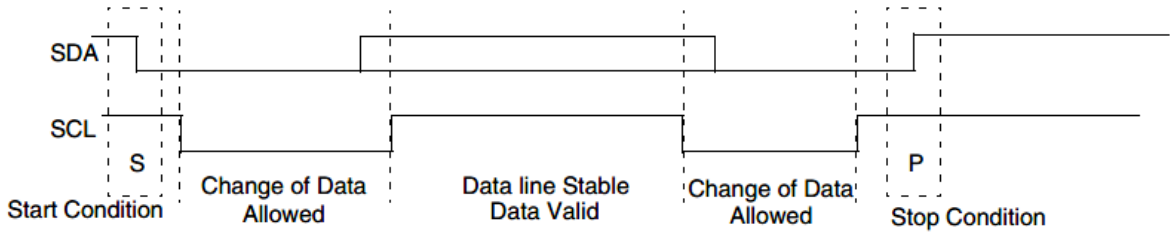


Figure 23.3: START and STOP Condition

23.3.2 7-bit Addressing

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the slave address and the LSB bit (bit 0) is the R/W bit as shown in Figure 23.4. When bit 0 (R/W) is set to 0, the master writes to the slave. When bit 0 (R/W) is set to 1, the master reads from the slave.

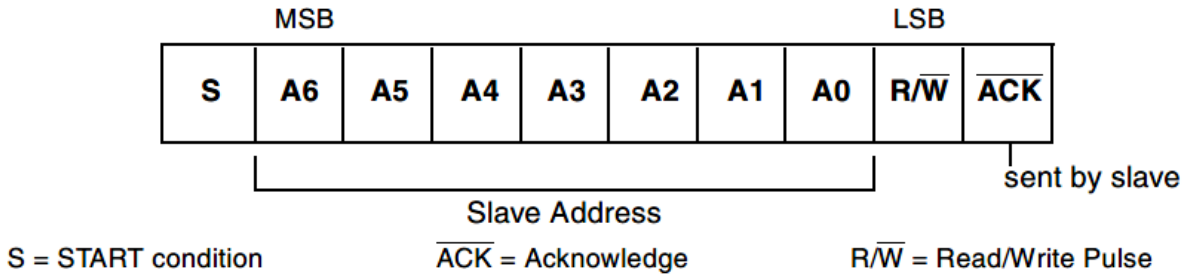


Figure 23.4: 7-bit Address Format

23.3.3 10-Bit Addressing

During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the slaves address bits 9:8, and the LSB bit (bit 0) is the R/W bit. The second byte transferred sets bits 7:0 of the slave address. Figure 23.5 shows the 10-bit address format.

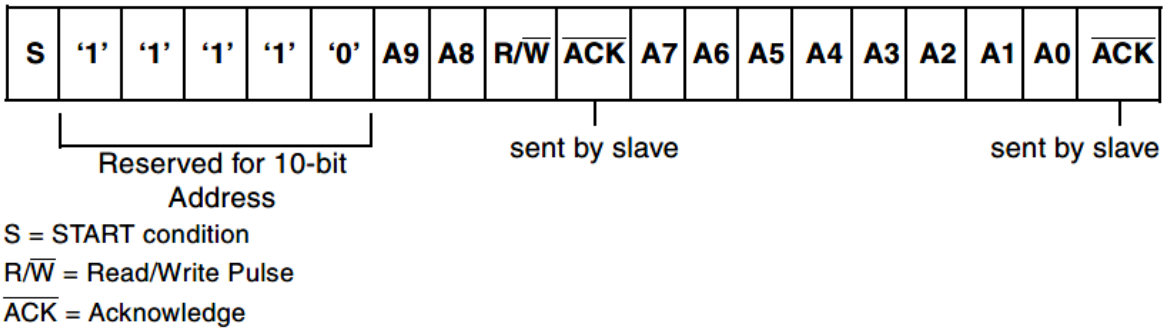


Figure 23.5: 10-bit Address Format

23.3.4 Master to Slave Transmission

The master can initiate data transmission by creating a START component followed by 7-bit address and 1-bit control. Since the master is transmitting to the slave, the control bit is low, signifying a write operation. The address and direction are shifted into all slaves, and the slave that matches the address will drive SDA low during the 9th SCL pulse to acknowledge that it got the address. The next SDA 8-bits of data are sent to the selected slave. The selected slave, again, will acknowledge that it got the data by driving SDA low. Finally, the master will generate a STOP component to signal the end of the transmission.

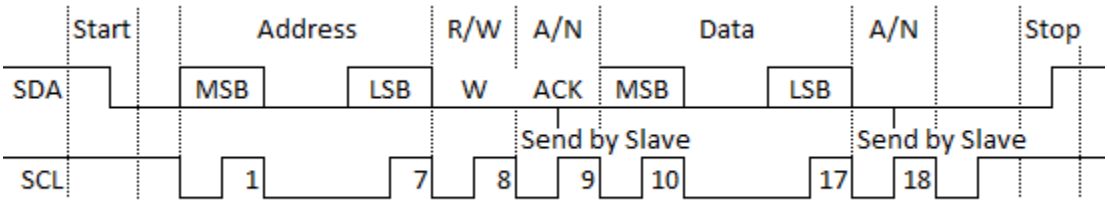


Figure 23.6: 1-Byte Master to Slave Transmission

23.3.5 Slave to Master Transmission

For slave to master transmission, again, the master begins by creating a START component followed by 7-bit address and 1-bit. Since the slave is transmitting to master, the control bit is high, signifying a read operation. The address and direction are shifted into all slaves, and the slave that matches the address will drive SDA low during the 9th SCL pulse to acknowledge that it got the address. The next SDA 8-bits of data are sent from the selected slave to the master. The master will acknowledge to the slave that it got the data by driving SDA low. Finally, the master will generate a STOP component to signal the end of the transmission.

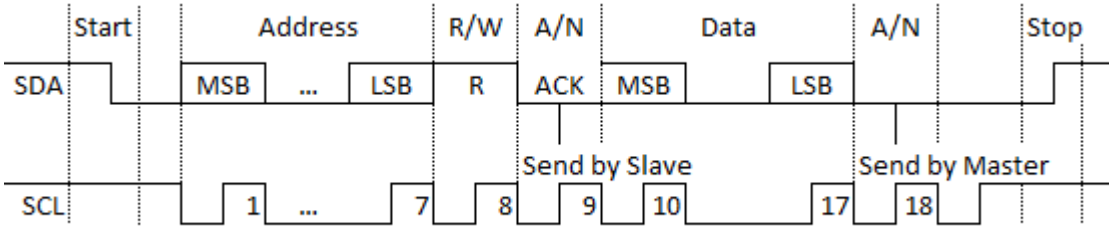


Figure 23.7: 1-Byte Slave to Master Transmission

23.3.6 Master to Multiple Slaves Transmission

The master to multiple slaves begins with the master creating a START component followed by 7-bit address and 1-bit control. During the first START, the address selects the first slave with either read/write for the control bit. The next SDA 8-bits of data are sent/received to/from the selected slave. After the data, depending on read/write, the master/slave will acknowledge respectively, and this time the master issues a repeated RESTART instead of a STOP component. After the RESATART, the 7-bit address and 1-bit control will address the next slave. The next SDA 8-bits of data are sent/received to/from the selected slave. Finally, the master will generate a STOP component to signal the end of the transmission.

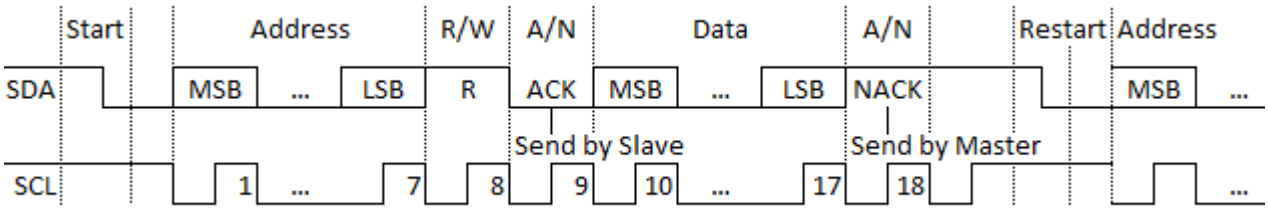


Figure 23.8: Master to Multiple Slaves Transmission

23.4 Clock Generation

23.4.1 PCLK Frequency Configuration

- When the I2C is configured as a Standard (SS), Fast (FS) or Fast-mode Plus (FM+) master, the xCNT registers must be set before any I2C bus transaction can take place in order to ensure proper I/O timing. The xCNT registers are:
- SS_SCL_HCNT
- SS_SCL_LCNT
- FS_SCL_HCNT
- FS_SCL_LCNT

Table 23.1 lists the derivation of I2C timing parameters from the xCNT programming registers.

Table 23.1: I2C Timing Parameters

Timing Parameter	Symbol	Standard Speed	Fast Speed/Fast Speed Pulse
LOW period of the SCL clock	tLOW	SS_SCL_LCNT	FS_SCL_LCNT
HIGH period of the SCL clock	tHIGH	SS_SCL_HCNT	FS_SCL_HCNT
Setup time for a repeated START condition	tSU:STA	SS_SCL_LCNT	FS_SCL_HCNT
Hold time (repeated) START condition*	tHD:STA	SS_SCL_HCNT	FS_SCL_HCNT
Setup time for STOP condition	tSU:STO	SS_SCL_HCNT	FS_SCL_HCNT
Bus free time between a and a START condition	tBUF	SS_SCL_LCNT	FS_SCL_LCNT
Spike length	tSP	FS_SPKLEN	FS_SPKLEN
Data hold time	tHD:DAT	SDA_HOLD	SDA_HOLD
Data setup time	tSU:DAT	SDA_SETUP	SDA_SETUP

23.4.2 Minimum High and Low Counts in SS, FS and FM+ Modes

When the I2C operates as an I2C master, in both transmit and receive transfers:

- SS_SCL_LCNT and FS_SCL_LCNT register values must be larger than FS_SPKLEN + 7.
- SS_SCL_HCNT and FS_SCL_HCNT register values must be larger than FS_SPKLEN + 5.

Details regarding the I2C high and low counts are as follows:

- The minimum value of x_SPKLEN + 7 for the x_LCNT registers is due to the time required for the I2C to drive SDA after a negative edge of SCL.
- The minimum value of x_SPKLEN + 5 for the x_HCNT registers is due to the time required for the I2C to sample SDA during the high period of SCL.
- The I2C adds one cycle to the programmed x_LCNT value in order to generate the low period of the SCL clock; this is due to the counting logic for SCL low counting to (x_LCNT + 1)
- The I2C adds x_SPKLEN + 7 cycles to the programmed x_HCNT value in order to generate the high period of the SCL clock; this is due to the following factors:
  - — The counting logic for SCL high counts to (x_HCNT+1).
  - — The digital filtering applied to the SCL line incurs a delay of SPKLEN + 2 PCLK cycles, where SPKLEN is:
    - FS_SPKLEN if the component is operating in SS or FS
    - HS_SPKLEN if the component is operating in HS.
    - This filtering includes metastability removal and the programmable spike suppression on SDA and SCL edges.
  - — Whenever SCL is driven 1 to 0 by I2C peripheral —that is, completing the SCL high time—an internal logic latency of three PCLK cycles is incurred. Consequently, the minimum SCL low time of which the I2C peripheral is capable is nine (9) PCLK periods (7 + 1 + 1), while the minimum SCL high time is thirteen (13) PCLK periods (6 + 1 + 3 + 3)

23.4.3 Minimum PCLK Frequency

This section describes the minimum PCLK frequencies that the I2C supports for each speed mode, and the associated high and low count values. In Slave mode, IC_SDA_HOLD (Thd:dat) and IC_SDA_SETUP (Tsu:dat) need to be programmed to satisfy the I2C protocol timing requirements.

This section details how to derive a minimum PCLK value. Although the following method shows how to do fast mode calculations, you can also use the same method in order to do calculations for standard mode and fast mode plus.

Note: The following computations do not consider the SCL_Rise_time and SCL_Fall_time.

Based on the given conditions and calculations for the minimum I2C peripheral PCLK value in fast mode:

- Fast mode has data rate of 400kb/s; implies SCL period of 1/400khz = 2.5us
- Minimum HCNT value of 14 as a seed value; HCNT_FS = 14
- Protocol minimum SCL high and low times:
  - MIN_SCL_LOWtime_FS = 1300ns
  - MIN_SCL_HIGHtime_FS = 600ns

The above results are based on the following derived equations:

$$\frac{SCL\_PERIOD\_FS}{HCNT\_FS + LCNT\_FS} = CLK\_PERIOD$$

$$LCNT\_FS * CLK\_PERIOD = MIN\_SCL\_LOWTIME\_FS$$

Combined, the previous equations produce the following:

$$LCNT\_FS * \frac{SCL\_PERIOD\_FS}{HCNT\_FS + LCNT\_FS} = CLK\_PERIOD$$

Solving for LCNT_FS:

$$LCNT\_FS * \frac{2.5\mu s}{14 + LCNT\_FS} = 1.3\mu s$$

Which gives:

$$LCNT\_FS = roundup(15.166) = 16$$

These calculations produce LCNT_FS = 16 and HCNT_FS = 14, giving a PCLK value of:

$$CLK\_PERIOD = \frac{2.5\mu s}{14 + 16} = 83.3ns$$

$$PCLK = \frac{1}{CLK\_PERIOD} = 12MHz$$

Table 23.2: Minimum PCLK Frequency for All Modes with high and low count values

Speed Mode	PCLK frequency (MHz)	Minimum Value of x_SPKLEN	SCL Low Time in PCLKs	SCL Low Program Value	SCL Low Time	SCL High Time in PCLKs	SCL High Program Value	SCL High Time
SS	2.7	1	13	12	4.7 µs	14	6	5.2 µs
FS	12	1	16	15	1.33 µs	14	6	1.16 µs
FM+	32	2	16	15	500 ns	16	7	500 ns

23.4.4 Standard Mode 100Kbps Baud-rate Generation

given PCLK = 50MHz (20ns)
given I2C mode = **std, 100K bps**
given min SCL High time = 4000ns
given min SCL Low time = 4700ns

The resulting reset values should be:
SS_SCL_HCNT = 4000/20 = 200 dec = C8h
SS_SCL_LCNT = 4700/20 = 235 dec = EBh

23.4.5 Fast Speed Mode 400Kbps Baud-rate Generation

given PCLK = 50MHz (20ns)
given I2C mode = **fast, 400K bps**
given min SCL High time = 600ns
given min SCL Low time = 1300ns

The resulting reset values should be:
FS_SCL_HCNT = 600/20 = 30 dec = **1Eh**
FS_SCL_LCNT = 1300/20 = 65 dec = **41h**

23.4.6 Fast Speed Plus Mode 1Mbps Baud-rate Generation

given PCLK = 50MHz (20ns)
given I2C mode = **fast-plus, 1M bps**
given min SCL High time = 260ns
given min SCL Low time = 500ns

The resulting reset values should be:
FS_SCL_HCNT = 260/20 = 13 dec = **0Dh**
FS_SCL_LCNT = 500/20 = 25 dec = **19h**

23.5 SDA Hold Time

The I2C protocol specification requires 300 ns of hold time on the SDA signal (tHD:DAT) in standard mode and fast mode, and a hold time long enough to bridge the undefined part between logic 1 and logic 0 of the falling edge of SCL in high-speed mode and fast mode plus.

Board delays on the SCL and SDA signals can mean that the hold-time requirement is met at the I2C master, but not at the I2C slave (or vice-versa). As each application encounters differing board delays, the I2C contains a software programmable register (I2C.SDA_HOLD) to enable dynamic adjustment of the SDA hold-time.

The bits [15:0] are used to control the hold time of SDA during transmit in both slave and master mode (after SCL goes from HIGH to LOW).

The bits [23:16] are used to extend the SDA transition (if any) whenever SCL is HIGH in the receiver (in either master or slave mode).

If different SDA hold times are required for different speed modes, the I2C.SDA_HOLD register must be reprogrammed when the speed mode is being changed. The I2C.SDA_HOLD register can be programmed only when the I2C peripheral is disabled (I2C.ENABLE[0] = 0).

23.5.1 SDA Hold Timings in Receiver

When the I2C interface acts as a receiver, according to the I2C protocol, the device should internally hold the SDA line to bridge undefined gap between logic 1 and logic 0 of SCL.

I2C.SDA_RX_HOLD register can be used to alter the internal hold time which the I2C interface applies to the incoming SDA line. Each value in the I2C.SDA_RX_HOLD register represents a unit of one PCLK period. The minimum value of I2C.SDA_RX_HOLD is 0. This hold time is applicable only when SCL is HIGH. The receiver does not extend the SDA after SCL goes LOW internally.

If I2C.SDA_RX_HOLD is greater than 3, the I2C peripheral does not hold SDA beyond 3 PCLK cycles, because SCL goes LOW internally.

The maximum values of IC_SDA_RX_HOLD that can be programmed in the register for the respective speed modes are derived from the equations show in Table 23.3.

Table 23.3: Maximum Values for I2C.SDA_RX_HOLD

Speed Mode	Maximum I2C.SDA_RX_HOLD Value
Standard Mode (SS)	2.7
Fast Mode (FS) or Fast Mode Plus (FM+)	12.0

23.5.2 SDA Hold Timings in Transmitter

The I2C.SDA_TX_HOLD register can be used to alter the timing of the generated SDA signal by the I2C interface. Each value in the I2C.SDA_TX_HOLD register represents a unit of one PCLK period.

When the I2C interface is operating in Master Mode, the minimum tHD:DAT timing is one PCLK period. Therefore, even when I2C.SDA_TX_HOLD has a value of zero, the I2C interface will drive SDA one PClK cycle after driving SCL to logic 0. For all other values of I2C.SDA_TX_HOLD, the following is true:

Drive on SDA occurs I2C.SDA_TX_HOLD PCLK cycles after driving SCL to logic 0

When the I2C interface is operating in Slave Mode, the minimum tHD:DAT timing is SPKLEN + 7 PCLK periods (due to operating in standard mode, fast mode, or fast mode plus).

This delay allows for synchronization and spike suppression on the SCL sample. Therefore, even when I2C.SDA_TX_HOLD has a value less than SPKLEN + 7, the I2C interface drives SDA SPKLEN + 7 PCLK cycles after SCL has transitioned to logic 0. For all other values of I2C.SDA_TX_HOLD, the following is true:

Drive on SDA occurs I2C.SDA_TX_HOLD PCLK cycles after SCL has transitioned to logic 0.

23.6 Synchronization

Clock Synchronization

Among the masters, clock synchronization makes the low period equal to the longest SCL/SDA low period and the high period equal to the shortest SCL/SDA high. Since I2C outputs are open collector, the signals are wired-AND.

Figure 23.8 shows the I2C clock synchronization timing diagram.

23.6.2 Data Arbitration

Data arbitration determines the priority of the contending masters. A bus master loses arbitration if it sends logic 1 while another master transmits logic 0. The losing masters immediately switch over to slave receive mode and stop driving SCL and SDA signal outputs.

Figure 23.9 shows the I2C data arbitration timing diagram in which Master #1 wins arbitration since A6 transmits a logic 0 while Master #2 A6 is still logic 1.

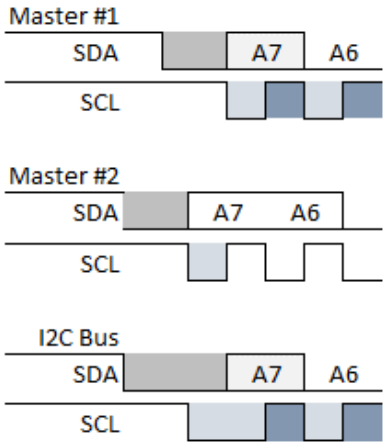


Figure 23.9: Multi Master Synchronization

23.6.3 Clock Stretching

For clock stretching, if the slave/master is not ready for the rising edge of SCL, the slave/master will hold the SCL clock low until slave/master is ready.

Figure 23.10 shows the I2C clock stretching timing diagram.

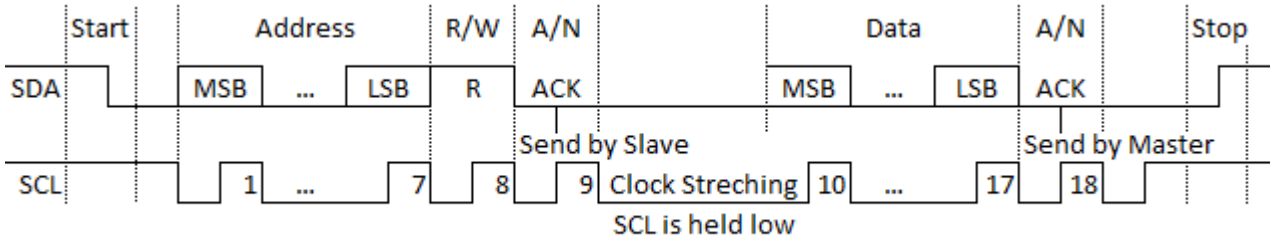


Figure 23.10: Clock Stretching

23.7 Spike Suppression

The I2C contains programmable spike suppression logic that matches requirements imposed by the I2C Bus Specification for the Standard Speed (SS) and Fast Speed (FS).

The I2C peripheral has a register for specifying the maximum spike length for the standard speed and fast speed modes called FS_SPKLEN. This register is 8 bits wide and accessible through the APB interface for read and write purposes; however, it can be written to only when the IC2 is disabled. The minimum value that can be programmed into this register is 1; attempting to program a value smaller than 1, results in the value 1 being written.

This logic is based on counters that monitor the input signals (SCL and SDA), checking if they remain stable for a predetermined amount of system clock (PCLK) cycles before they are sampled internally. There is one separate counter for each signal (SCL and SDA). The number of PCLK cycles can be programmed by the user and should be calculated taking into account the frequency of PCLK and the relevant spike length specification. Each counter is started whenever its input signal changes its value. Depending on the behavior of the input signal, one of the following scenarios occurs:

- The input signal remains unchanged until the counter reaches its count limit value. When this happens, the internal version of the signal is updated with the input value, and the counter is reset and stopped. The counter is not restarted until a new change on the input signal is detected.
- The input signal changes again before the counter reaches its count limit value. When this happens, the counter is reset and stopped, but the internal version of the signal is not updated. The counter remains stopped until a new change on the input signal is detected.

Figure 23.11 shows the timing diagram for spike suppression.

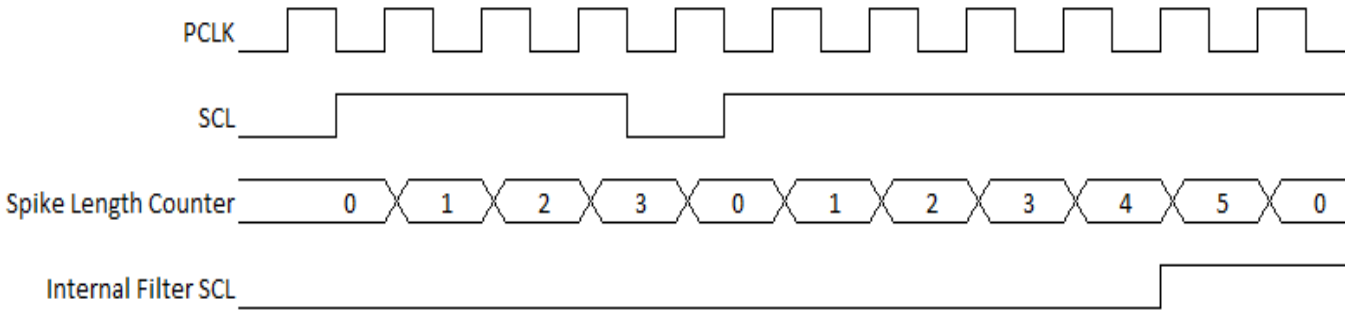


Figure 23.11: Spike Suppression Example

23.8 External Interrupt

Each I2C module, I2C0 and I2C1, has an external Interrupt (IRQ) mapped to Interrupt Vectors 18-19 respectively. The UTM0R500 microcontroller based on the Cortex M0+ processor contains an internal nested vector Interrupt controller (NVIC) that supports up to 32 external IRQ's and a non-maskable Interrupt (NMI). NVIC handles the nested Interrupts automatically based on priorities and Interrupt number, and if an Interrupt is accepted, it communicates with the processor to allow it to execute the correct Interrupt handler.

For more information on external Interrupts and the nested vector Interrupt (NVIC), refer to Nested Vector Interrupt Controller (NVIC).

23.9 FIFO Empty Interrupt

Data are popped (removed) from the transmit FIFO by the shift control logic into the transmit shift register. The transmit FIFO generates a FIFO empty Interrupt request by setting TX_EMPTY flag to 1 when the number of entries in the FIFO is less than or equal to the FIFO threshold value. The threshold value, set through the programmable register IC_TX_TL, determines the level of FIFO entries at which an Interrupt is generated. The threshold value allows to provide early indication to the processor that the transmit FIFO is nearly empty. A transmit FIFO overflow Interrupt (TX_OVER) is generated if an attempt to write data into an already full transmit FIFO.

23.10 FIFO Full Interrupt

Data are popped from the receive FIFO by APB read commands to the data register (DR). The receive FIFO is loaded from the receive shift register by the shift control logic. The receive FIFO generates a FIFO-full Interrupt request (RX_FULL) when the number of entries in the FIFO is greater than or equal to the FIFO threshold value plus 1. The threshold value, set through programmable register (IC_RX_TL), determines the level of FIFO entries at which an Interrupt is generated.

23.11 FIFO Overrun Interrupt

The threshold value allows you to provide early indication to the processor that the receive FIFO is nearly full. A receive FIFO overrun Interrupt (RX_OVER) is generated when the receive shift logic attempts to load data into a completely full receive FIFO. However, this newly received data are lost. A receive FIFO underflow Interrupt (RX_UNDER) is generated if you attempt to read from an empty receive FIFO. This alerts the processor that the read data are invalid.

Note: When IC_ENABLE[0] flag is set to 0, the transmit and receive FIFO buffers are flushed and held in reset.

23.12 I2C Register Details

Table 23.4: I2C Registers

Register	Offset (Hex)
Control Register (CON)	0x0000
Target Address Register (TAR)	0x0004
Slave Address Register (SAR)	0x0008
RESERVED	0x000C
RX/TX Data Buffers and Command Register (DATA_CMD)	0x0010
Standard-Speed Clock Hi-Count Register (SS_SCL_HCNT)	0x0014
Standard Speed Clock Lo-Count Register (SS_SCL_LCNT)	0x0018
Fast-Speed Clock Hi-Count Register (FS_SCL_HCNT)	0x001C
Fast-Speed Clock Lo-Count Register (FS_SCL_LCNT)	0x0020
RESERVED	0x0024-0x0028
Interrupt Status Register (INTR_STAT)	0x002C
Interrupt Mask Register (INTR_MASK)	0x0030
Raw Interrupt Status Register (RAW_INTR_STAT)	0x0034
Receive FIFO Threshold Register (RX_TL)	0x0038
Transmit FIFO Threshold Register (TX_TL)	0x003C
Clear All Software Interrupts Register (CLR_INTR)	0x0040
Clear Receive Underflow Interrupt Register (CLR_RX_UNDER)	0x0044
Clear Receive Overflow Interrupt Register (CLR_RX_OVER)	0x0048
Clear Transmit Overflow Interrupt Register (CLR_TX_OVER)	0x004C
Clear Read Request Interrupt Register (CLR_RD_REQ)	0x0050
Clear Transmit Abort Interrupt Register (CLR_TX_ABRT)	0x0054
Clear Receive Complete Interrupt Register (CLR_RX_DONE)	0x0058

Register	Offset (Hex)
Clear Bus Activity Interrupt Register (CLR_ACTIVITY)	0x005C
Clear Stop Condition Interrupt Register (CLR_STOP_DET)	0x0060
Clear Start Condition Interrupt Register (CLR_START_DET)	0x0064
Clear General Call Received Interrupt Register (CLR_GEN_CALL)	0x0068
Enable Register (ENABLE)	0x006C
Status Register (STATUS)	0x0070
Transmit FIFO Level Register (TXFLR)	0x0074
Receive FIFO Level Register (RXFLR)	0x0078
SDA Hold Time Register (SDA_HOLD)	0x007C
Transmit Abort Source Register (TX_ABORT_SOURCE)	0x0080
Generate Slave Data NACK Register (SLV_DATA_NACK_ONLY)	0x0084
RESERVED	0x0088-0x0090
SDA Setup Register (SDA_SETUP)	0x0094
ACK General Call Register (ACK_GENERAL_CALL)	0x0098
Enable Status Register (ENABLE_STATUS)	0x009C
SS/FS Spike Suppression Limit Register (FS_SPKLEN)	0x00A0
HS Spike Suppression Limit Register (HS_SPKLEN)	0x00A4
Clear Restart Detect Interrupt Register (CLR_RESTART_DET)	0x00A8
RESERVED	0x00AC-0x00F0
Component Parameter Register 1 (COMP_PARAM_1)	0x00F4
Component Version Register (COMP_VERSION)	0x00F8
Component Type Register (COMP_TYPE)	0x00FC

23.12.1 Control Register (CON)

CON																Offset = 0x0000
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R						SIM	RF	TE	SIA	ID	IR	IM	IS	SPEED		MM
W																
Reset	[00…0]					0	0	0	1	1	1	0	0	10		0

Table 23.5: Description of the Control Register

Bit Number(s)	Bit Name	Reset State	Description
31-11	RESERVED	[00…0]	
10	STOP_DET_IF _MASTER _ACTIVE (SIM)	0	1: The STOP_DET Interrupt is asserted only in master mode 0: No action
9	RX_FIFO_FULL_HLD_ CTRL (RF)	0	1: Holds the bus when the RX FIFO is full 0: No action
8	TX_EMPTY _CTRL (TE)	0	1: Controls the trigger criteria for the TX_EMPTY Interrupt (Described in the RAW_INTR_STAT register) 0: No action
7	STOP_DET _IFADDRESSED (SIA)	1	Only applicable if in slave mode (MASTER_MODE=0) 1: Issues the STOP_DET Interrupt only when it is addressed 0: Issues the STOP_DET irrespective of whether it is addressed or not
6	IC_SLAVE _DISABLE (ID)	1	This bit controls whether the I2C has its slave disabled. If the bit is set (slave is disabled), the I2C peripheral functions only as a master and does not perform any action that requires a slave. 1: Slave is enabled 0: Slave is disabled NOTE: Software should ensure that if this bit is written with ‘0’, then bit 0 should also be written with a ‘0’
5	IC_RESTART_EN (IR)	1	Determines whether RESTART conditions may be sent when acting as a master. Some older slaves do not support handling RESTART conditions; however, RESTART conditions are used in several of the I2C operations. 1: Enable 0: Disable When RESTART is disabled, the I2C master is incapable of performing the following functions: -Sending a START BYTE -Performing any high-speed mode operation -Performing direction changes in combined format mode -Performing a read operation with a 10-bit address

Bit Number(s)	Bit Name	Reset State	Description
			By replacing RESTART condition followed by a STOP and a subsequent START condition, split operations are broken down into multiple I2C transfers. If the above operations are performed, it will result in setting bit 6 (TX_ABRT) of the I2C.RAW_INTR_STAT register.
4	IC_10BITADDR _MASTER_RD _ONLY (IM)	0	1: 10-bit addressing 0: 7-bit addressing
3	IC_10BITADDR_SLAV E (IS)	0	When acting as a slave, this bit controls whether the I2C responds to 7- or 10-bit addresses 1: 10-bit addressing. The I2C responds to only 10-bit addressing transfers that match the full 10 bits of the I2C.SAR register 0: 7-bit addressing. The I2C responds to only 7-bit addressing transfers that match the lower 7 bytes of the I2C.SAR register
2-1	SPEED	10	These bits control the speed at which the I2C operates; its setting is relevant only if one is operating the I2C peripheral in master mode. Hardware protects against illegal values being programmed by software. This register should be programmed only with a value in the range of 1 or 2; otherwise, hardware updates this register with the value of 2. 10: Fast mode (≤ 400 Kb/s) or fast mode plus (≤ 1000 Kb/s) 01: Standard mode (0 to 100Kb/s)
0	MASTER_MODE (MM)	0	This bit controls whether the I2C master is enabled. 1: Master enabled 0: Master disabled NOTE: Software should ensure that if this bit is written with '1', then bit 6 should also be written with a '1'

Certain combinations of the IC_SLAVE_DISABLE (bit 6) and MASTER_MODE (bit 0) result in a configuration error. The table below lists the states that result from the combinations of these two bits.

IC_SLAVE_DISABLE (I2.CON[6])	MASTER_MODE I2C.CON[0]	State
0	0	Slave device
0	1	Config Error
1	0	Config Error
1	1	Master device

23.12.2 Target Address Register (TAR)

REG NAME															Offset = 0x0004
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R				IM	SP	GC	IC_TAR									
W																
Reset	000			0	0	0	0x055									

Table 23.6: Description of the Target Address Register

Bit Number(s)	Bit Name	Reset State	Description
31-13	RESERVED	[00…0]	
12	IC_10BITADDR_MASTER (IM)	0	This bit controls whether the I2C starts its transfers in 7- or 10-bit addressing mode when acting as a master. This bit exists in this register only if the I2C_DYNAMIC_TAR_UPDATE configuration parameter is set to Yes(‘1’) 1: 10-bit addressing 0: 7-bit addressing
11	SPECIAL (SP)	0	This bit indicates whether software performs a General Call or START BYTE command. 1: Perform special I2C command as specified in GC_OR_START bit 0: Ignore bit 10 GC_OR_START and use IC_TAR normally
10	GC_OR_START (GC)	0	If bit 11 (SPECIAL) is set to ‘1’, then this bit indicates whether a General Call or START byte command is to be performed by the I2C. 1: START byte 0: General Call Address – after issuing a General Call, only writes may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the I2C.RAW_INTR_STAT register. The I2C remains in General Call mode until the SPECIAL bit value (bit 11) is cleared
9-0	IC_TAR	0x055	This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits. If the I2C.TAR and I2C.SAR are the same, loopback exists but the FIFOs are shared between master and slave, so full loopback is not feasible. Only one direction loopback mode is supported (simplex), not duplex. A master cannot transmit to itself; it can transmit to only a slave.

23.12.3 Slave Address Register (SAR)

SAR																Offset = 0x0008
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							IC_SAR									
W																
Reset	[00…0]						0x055									

Table 23.7: Description of the Slave Address Register

Bit Number(s)	Bit Name	Reset State	Description
31-10	RESERVED	[00...0]	
9-0	IC_SAR	0x055	The IC_SAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only IC_SAR[6:0] is used. When the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect. NOTE: The default values cannot be any of the reserved address locations: that is, 0x00 to 0x07, or 0x78 to 0x7f. The correct operation of the device is not guaranteed if you program the I2C.SAR or I2C.TAR to a reserved value. NOTE: Changing the SAR after a RD_REQ interrupt flag is set will cause the RD_REQ flag to re-set once the I2C peripheral is re-enabled. See the I2C RD_REQ Interrupt errata in Error! Unknown switch argument.

23.12.4 Data Command Register (DATA_CMD)

DATA_CMD																Offset = 0x0010
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					FDB				DAT							
W						RS	SP	CMD								
Reset	[00…0]				0	0	0	0	[00…0]							

Table 23.8: Description of the Data Command Register

Bit Number(s)	Bit Name	Reset State	Description
31-12	RESERVED	[00...0]	
11	FIRST_DATA_BYTE (FDB)	0	This bit field holds the value of the I2C HS mode master code. HSmode master codes are reserved 8-bit codes (00001xxx) that are not used for slave addressing or other purposes. Each master has its unique master code; up to eight high speed mode masters can be present on the same I2C bus system. Valid values are from 0 to 7.
10	RESTART (RS)	0	This bit is used to cause a RESTART condition. 1: A RESTART condition is issued before the next data byte is sent/received (depending on the value of the CMD bit), regardless of whether or not the transfer direction is changing from the previous command. If IC_RESTART_EN is '0', a STOP followed by a START is issued instead. 0: A RESTART condition will only be issued if the transfer direction has changed from the previous command and there is no delay between the two commands. If there is a delay, a RESTART is not sent. If IC_RESTART_EN is '0', a STOP followed by a START is issued instead.
9	STOP (SP)	0	This bit is used to cause a STOP condition. 1: A STOP condition is issued after the current byte, regardless of whether or not the TX FIFO is empty. If the TX FIFO is not empty, the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus. 0: A STOP condition is not issued after this byte, regardless of whether or not the TX FIFO is empty. If the TX FIFO is not empty, the master continues the current transfer by sending/receiving data bytes according to the value of the CMD bit. If the TX FIFO is empty, the master holds the SCL line low and stalls the bus until a new command is available in the TX FIFO.
8	CMD	0	This bit controls whether a read or a write is performed. This bit does not control the direction when the I2C acts as a slave. It controls only the direction when it acts as a master. 1: Read 0: Write When a command is entered in the TX FIFO, this bit distinguishes the write and read commands. In slave-receiver mode, this bit is a "don't care" because writes to this register are not required. In slave-transmitter mode, a '0' indicates that the data in I2C.DATA_CMD is to be transmitted. When programming this bit, you should remember the following: attempting to perform a read operation after a General Call command has been sent results in a TX_ABRT Interrupt (bit 6 of the I2C.RAW_INTR_STAT register), unless bit 11 (SPECIAL) in the I2C.TAR register has been cleared. If a '1' is written to this bit after receiving a RD_REQ Interrupt, then a TX_ABRT Interrupt occurs.
7-0	DAT	[00...0]	This register contains the data to be transmitted or received on the I2C bus. If you are writing to this register and want to perform a read, bits 7:0 (DAT) are ignored by the I2C. However, when you read this register, these bits return the value of data received on the I2C interface.

23.12.5 Standard Speed Clock SCL High Count Register (SS_SCL_HCNT)

SS_SCL_HCNT																Offset = 0x0014
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SS_SCL_HCNT															
W																
Reset	[00…0]															

Table 23.9: Description of the Standard Speed Clock SCL High Count Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-0	SS_SCL_HCNT	[00…0]	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed. For more information, refer to “PCLK Frequency Configuration” section. This register can be written only when the I2C interface is disabled which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. For designs with APB_DATA_WIDTH = 8, the order of programming is important to ensure the correct operation of the I2C. The lower byte must be programmed first. Then the upper byte is programmed. NOTE: This register must not be programmed to a value higher than 65525, because I2C uses a 16-bit counter to flag an I2C bus idle condition when this counter reaches a value of IC_SS_SCL_HCNT + 10.

23.12.6 Standard Speed Clock SCL Low Count Register (SS_SCL_LCNT)

SS_SCL_LCNT																Offset = 0x0018
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SS_SCL_LCNT															
W																
Reset	[00…0]															

Table 23.10: Description of the Standard Speed Clock SCL Low Count Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-0	SS_SCL_LCNT	[00…0]	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for standard speed. For more information, refer to "PCLK Frequency Configuration" section. This register can be written only when the I2C interface is disabled which corresponds to IC_ENABLE[0] being set to '0'. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted, results in 8 being set.

23.12.7 Fast Mode or Fast Mode Plus Clock SCL High Count Register (FS_SCL_HCNT)

FS_SCL_HCNT																**Offset = 0x001C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FS_SCL_HCNT															
W																
Reset	[00…0]															

Table 23.11: Description of the Fast Mode or Fast Mode Plus Clock SCL High Count Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-0	FS_SCL_HCNT	[00…0]	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast mode or fast mode plus. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. For more information, refer to "PCLK Frequency Configuration" section. This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set.

23.12.8 Fast Mode or Fast Mode Plus Clock SCL Low Count Register (FS_SCL_LCNT)

FS_SCL_LCNT																Offset = 0x0020
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FS_SCL_LCNT															
W																
Reset	[00…0]															

Table 23.12: Description of the Fast Mode or Fast Mode Plus Clock SCL Lowe Count Register

Bit Number(s)	Bit Name	Reset State	Description
31-16	RESERVED	[00…0]	
15-0	FS_SCL_LCNT	[00…0]	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for fast mode or fast mode plus. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. For more information, refer to “PCLK Frequency Configuration” section. This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set. If the value is less than 8 then the count value gets changed to 8.

23.12.9 Interrupt Status Register (INTR_STAT)

INTR_STAT																Offset = 0x002C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			MS	RD	GC	ST	SP	AC	RX	TX	RR	TE	TO	RF	RO	RU
W																
Reset	00		0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23.13: Description of the Interrupt Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-14	RESERVED	[00...0]	
13	R_MST_ON _HOLD (MS)	0	See “I2C.RAW_INTR_STAT” for a detailed description of these bits
12	R_RESTART _DET (RD)	0	
11	R_GEN_CALL (GC)	0	
10	R_START_DET (ST)	0	
9	R_STOP_DET (SP)	0	
8	R_ACTIVITY (AC)	0	
7	R_RX_DONE (RX)	0	
6	R_TX_ABRT (TX)	0	
5	R_RD_REQ (RR)	0	
4	R_TX_EMPTY (TE)	0	
3	R_TX_OVER (TO)	0	
2	R_RX_FULL (RF)	0	
1	R_RX_OVER (RO)	0	
0	R_RX_UNDER (RU)	0	

23.12.10 Interrupt Mask Register (INTR_MASK)

INTR_MASK																**Offset = 0x0030**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			MS	RD	GC	ST	SP	AC	RX	TX	RR	TE	TO	RF	RO	RU
W																
Reset	01		0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23.14: Description of the Interrupt Mask Register

Bit Number(s)	Bit Name	Reset State	Description
31-14	RESERVED	[00...0]	
13	M_MST_ON _HOLD (MS)	0	This bit masks the R_MST_ON_HOLD Interrupt bit in the IC_INTR_STAT register. Dependencies: If I2C_DYNAMIC_TAR_UPDATE = 1 and IC_EMPTYFIFO_HOLD_MASTER_EN = 1, then M_MST_ON_HOLD is read/write. Otherwise M_MST_ON_HOLD is read-only
12	M_RESTART _DET (RD)	0	This bit masks the R_RESTART_DET Interrupt status bit in the IC_INTR_STAT register. Dependencies: If IC_SLV_RESTART_DET_EN = 1, then M_RESTART_DET is read/write. Otherwise M_RESTART_DET is read-only.
11	M_GEN_CALL (GC)	0	These bits mask their corresponding Interrupt status bits in the IC_INTR_STAT register
10	M_START_DET (ST)	0	
9	M_STOP_DET (SP)	0	
8	M_ACTIVITY (AC)	0	
7	M_RX_DONE (RX)	0	
6	M_TX_ABRT (TX)	0	
5	M_RD_REQ (RR)	0	
4	M_TX_EMPTY (TE)	0	
3	M_TX_OVER (TO)	0	
2	M_RX_FULL (RF)	0	
1	M_RX_OVER (RO)	0	
0	M_RX_UNDER (RU)	0	

23.12.11 Raw Interrupt Status Register (RAW_INTR_STAT)

RAW_INTR_STAT																Offset = 0x0034
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			MS	RD	GC	ST	SP	AC	RX	TX	RR	TE	TO	RF	RO	RU
W																
Reset	00		0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23.15: Description of the Raw Interrupt Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-14	RESERVED	[00...0]	
13	MST_ON_HOLD (MS)	0	Not used.
12	RESTART_DET (RD)	0	Indicates whether a RESTART condition has occurred on the I2C interface when I2C is operating in slave mode and the slave is the addressed slave. NOTE: However, in high-speed mode or during a START BYTE transfer, the RESTART comes before the address field as per the I2C protocol. In this case, the slave is not the addressed slave when the RESTART is issued, therefore I2C does not generate the RESTART_DET Interrupt
11	GEN_CALL (GC)	0	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling I2C or when the CPU reads bit 0 of the I2C.CLR_GEN_CALL register. The I2C stores the received data in the Rx buffer.
10	START_DET (ST)	0	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether is operating in slave or master mode.
9	STOP_DET (SP)	0	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether is operating in slave or master mode. In Slave Mode: ■ If I2C.CON[7]=1'b1 (STOP_DET_IFADDRESSED), the STOP_DET Interrupt is generated only if the slave is addressed. NOTE: During a general call address, this slave does not issue a STOP_DET Interrupt if STOP_DET_IF_ADDRESSED = 1'b1, even if the slave responds to the general call address by generating ACK. The STOP_DET Interrupt is generated only when the transmitted address matches the slave address (SAR). ■ If I2C.CON[7]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET Interrupt is issued irrespective of whether it is being addressed. In Master Mode: ■ If I2C.CON[10]=1'b1 (STOP_DET_IF_MASTER_ACTIVE), the STOP_DET Interrupt is issued only if the master is active. ■ If I2C.CON[10]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET Interrupt is issued irrespective of whether the master is active.

Bit Number(s)	Bit Name	Reset State	Description
8	ACTIVITY (AC)	0	This bit captures I2C activity and stays set until it is cleared. There are four ways to clear it: ■ Disabling the I2C ■ Reading the I2C.CLR_ACTIVITY register ■ Reading the I2C.CLR_INTR register ■ System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the I2C module is idle, this bit remains set until cleared, indicating that there was activity on the bus.
7	RX_DONE (RX)	0	When the I2C is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.
6	TX_ABRT (TX)	0	This bit indicates if I2C peripheral, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a “transmit abort”. When this bit is set to 1, the IC_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes places.
5	RD_REQ (RR)	0	This bit is set to 1 when I2C peripheral is acting as a slave and another I2C master is attempting to read data from I2C peripheral. The I2C peripheral holds the I2C bus in a wait state (SCL=0) until this Interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this Interrupt and then write the requested data to the IC_DATA_CMD register. This bit is set to 0 just after the processor reads the IC_CLR_RD_REQ register. NOTE: Any function that receives a RD_REQ bit and then changes the Slave Address Register (SAR) must have a delay between these two functions. For more information, see Error! Unknown switch argument.
4	TX_EMPTY (TE)	0	The behavior of the TX_EMPTY Interrupt status differs based on the TX_EMPTY_CTRL selection in the I2C.CON register. ■ When TX_EMPTY_CTRL = 0: This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register. ■ When TX_EMPTY_CTRL = 1: This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register and the transmission of the address/data from the internal shift register for the most recently popped command is completed. It is automatically cleared by hardware when the buffer level goes above the threshold. When I2C.ENABLE[0] is set to 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer any activity, then with I2C.ENABLE_STATUS.IC_EN=0, this bit is set to 0.
3	TX_OVER (TO)	0	Set during transmit if the transmit buffer is filled to the depth of 8 and the processor attempts to issue another I2C command by writing to the I2C.DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when I2C.ENABLE_STATUS.IC_EN goes to 0, this Interrupt is cleared.
2	RX_FULL	0	Set when the receive buffer reaches or goes above the RX_TL threshold in

Bit Number(s)	Bit Name	Reset State	Description
	(RF)		the IC_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (I2C.ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once I2C.ENABLE[0] is set to 0, regardless of the activity that continues.
1	RX_OVER (RO)	0	Set if the receive buffer is completely filled to a depth of 8 and an additional byte is received from an external I2C device. The I2C acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (I2C.ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when I2C.ENABLE_STATUS.IC_EN goes to 0, this Interrupt is cleared.
0	RX_UNDER (RU)	0	Set if the processor attempts to read the receive buffer when it is empty by reading from the I2CDATA_CMD register. If the module is disabled (I2C.ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when I2C.ENABLE_STATUS.IC_EN goes to 0, this Interrupt is cleared.

23.12.12 Receive FIFO Threshold Register (RX_TL)

RX_TL																**Offset = 0x0038**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									RX_TL							
W																
Reset	[00…0]								0x02							

Table 23.16: Description of the Receive FIFO Threshold Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00...0]	
7-0	RX_TL	0x02	Receive FIFO Threshold Level Controls the level of entries (or above) that triggers the RX_FULL Interrupt (bit 2 in I2C.RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries.

23.12.13 Transmit FIFO Threshold Register (TX_TL)

TX_TL																Offset = 0x003C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									TX_TL							
W																
Reset	[00...0]								0x02							

Table 23.17: Description of the Transmit FIFO Threshold Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00...0]	
7-0	TX_TL	0x02	Transmit FIFO Threshold Level Controls the level of entries (or below) that trigger the TX_EMPTY Interrupt (bit 4 in I2C.RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that it may not be set to value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 0 entries, and a value of 255 sets the threshold for 255 entries.

23.12.14 Clear Combined and Individual Interrupt Register (CLR_INTR)

CLR_INTR																Offset = 0x0040
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																CI
W																
Reset	[00...0]															0

Table 23.18: Description of the Clear Combined and Individual Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_INTR (CI)	0	Read this register to clear the combined Interrupt, all individual Interrupts, and the I2C.TX_ABRT_SOURCE register. This bit does not clear hardware clearable Interrupts but software clearable Interrupts. Refer to Bit 9 of the I2C.TX_ABRT_SOURCE register for an exception to clearing I2C. TX_ABRT_SOURCE.

23.12.15 Clear RX_UNDER Interrupt Register (CLR_RX_UNDER)

CLR_RX_UNDER																**Offset = 0x0044**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																CU
W																
Reset	[00...0]															0

Table 23.19: Description of the Clear RX_UNDER Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_RX_UNDER (CU)	0	Read this register to clear the RX_UNDER Interrupt (bit 0) of the I2C.RAW_INTR_STAT register.

23.12.16 Clear RX_OVER Interrupt Register (CLR_RX_UNDER)

CLR_RX_UNDER																**Offset = 0x0048**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																CO
W																
Reset	[00...0]															0

Table 23.20: Description of the Clear RX_OVER Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLEAR_RX _OVER (CO)	0	Read this register to clear the RX_OVER Interrupt (bit 1) of the I2C.RAW_INTR_STAT register

23.12.16 Clear TX_OVER Interrupt Register (CLR_TX_OVER)

CLR_TX_OVER						**Offset = 0x004C**									
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																CO
W																
Reset	[00...0]															0

Table 23.21: Description of the Clear TX_OVER Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_TX_OVER (CO)	0	Read this register to clear the TX_OVER Interrupt (bit 3) of the I2C.RAW_INTR_STAT register

23.12.18 lear RD_REQ Interrupt Register (CLR_RD_REQ)

CLR_RD_REQ																**Offset = 0x0050**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																RR
W																
Reset	[00...0]															0

Table 23.22: Description of the Clear RD_REQ Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_RD_REQ (RR)	0	Read this register to clear the RD_REQ Interrupt (bit 5) of the I2C.RAW_INTR_STAT register.

23.12.19 Clear TX_ABRT Interrupt Register (CLR_TX_ABRT)

CLR_TX_ABRT																Offset = 0x0054
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																TA
W																
Reset	[00...0]															0

Table 23.23: Description of the Clear TX_ABRT Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_TX_ABRT	0	Read this register to clear the TX_ABRT Interrupt (bit 6) of the I2C.RAW_INTR_STAT register, and the I2C.TX_ABRT_SOURCE register. This also releases the Tx FIFO from the flushed/reset state, allowing more writes to the Tx FIFO. Refer to Bit 9 of the I2C.TX_ABRT_SOURCE register for an exception to clearing I2C.TX_ABRT_SOURCE

23.12.20 Clear RX_DONE Interrupt Register (CLR_RX_DONE)

CLR_RX_DONE																Offset = 0x0058
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																RD
W																
Reset	[00...0]															0

Table 23.24: Description of the Clear RX_DONE Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_RX_DONE	0	Read this register to clear the RX_DONE Interrupt (bit 7) of the I2C.RAW_INTR_STAT register

23.12.21 Clear ACTIVITY Interrupt Register (CLR_ACTIVITY)

CLR_ACTIVITY																**Offset = 0x005C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																AC
W																
Reset	[00...0]															0

Table 23.25: Description of the Clear ACTIVITY Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_ACTIVITY (AC)	0	Reading this register clears the ACTIVITY Interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY Interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register has the status of the ACTIVITY Interrupt (bit 8) of the I2C.RAW_INTR_STAT register.

23.12.22 Clear STOP_DET Interrupt Register (CLR_STOP_DET)

CLR_STOP_DET																Offset = 0x0060
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																SP
W																
Reset	[00...0]															0

Table 23.26: Description of the Clear STOP_DET Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_STOP_DET (SP)	0	Read this register to clear the STOP_DET Interrupt (bit 9) of the I2C.RAW_INTR_STAT register.

23.12.23 Clear START_DET Interrupt Register (CLR_START_DET)

CLR_START_DET																Offset = 0x0064
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																ST
W																
Reset	[00...0]															0

Table 23.27: Description of the Clear START_DET Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_START _DET (ST)	0	Read this register to clear the START_DET Interrupt (bit 10) of the I2C.RAW_INTR_STAT register.

23.12.24 Clear GEN_CALL Interrupt Register (CLR_GEN_CALL)

CLR_GEN_CALL																Offset = 0x0068
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																GC
W																
Reset	[00…0]															0

Table 23.28: Description of the Clear GEN_CALL Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_GEN_CALL	0	Read this register to clear the GEN_CALL Interrupt (bit 11) of I2C.RAW_INTR_STAT

23.12.25 Enable Register (ENABLE)

ENABLE																Offset = 0x006C
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														CB	AB	EN
W																
Reset	[00…0]													0	0	0

Table 23.29: Description of the Enable Register

Bit Number(s)	Bit Name	Reset State	Description
31-3	RESERVED	[00...0]	
2	TX_CMD_BLOCK (CB)	0	In Master mode: 1: Blocks the transmission of data on I2C bus even if TX FIFO has data to transmit 0: The transmission of data starts on I2C bus automatically, as soon as the first data is available in the TX FIFO
1	ABORT (AB)	0	When set, the controller initiates the transfer abort. 1: ABORT operation in progress 0: ABORT not initiated or ABORT done The software can abort the I2C transfer in master mode by setting this bit. The software can set this bit only when ENABLE is already set; otherwise, the controller ignores any write to ABORT bit. The software cannot clear the ABORT bit once set. In response to an ABORT, the controller issues a STOP and flushes the TX FIFO after completing the current transfer, then sets the TX_ABORT Interrupt after the abort operation. The ABORT bit is cleared automatically after the abort operation. For a detailed description on how to abort I2C transfers, refer to “Aborting I2C Transfers”.
0	ENABLE (EN)	0	Controls whether the I2C is enabled. 1: Enables I2C 0: Disables I2C (TX and RX FIFOs are held in an erased state) Software can disable I2C while it is active. However, it is important that care be taken to ensure that I2C is disabled properly. A recommended procedure is described in “Disabling I2C”. When I2C is disabled, the following occurs: ■ The TX FIFO and RX FIFO get flushed. ■ Status bits in the I2C.INTR_STAT register are still active until I2C goes into IDLE state. If the module is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. If the module is receiving, the I2C stops the current transfer at the end of the current byte and does not acknowledge the transfer. For a detailed description on how to disable I2C, refer to “Disabling I2C”.

23.12.26 Status Register (STATUS)

STATUS	Offset = 0x0070															
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R										SA	MA	RF	RF	TE	TN	AC
W																
Reset	[00…0]									0	0	0	0	0	0	0

Table 23.20: Description of the Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-7	RESERVED	[00...0]	
6	SLV_ACTIVITY (SA)	0	Slave FSM (Finite State Machine) Activity Status. 1: Slave FSM is not in IDLE state so the Slave part of the I2C is active 0: Slave FSM is in IDLE state so the Slave part of the I2C is not active
5	MST_ACTIVITY (MA)	0	Master FSM Activity Status. 1: Master FSM is not in IDLE state so the Master part of I2C is Active 0: Master FSM is in IDLE state so the Master part of I2C is not Active NOTE: IC2.STATUS[0]—that is, ACTIVITY bit—is the OR of SLV_ACTIVITY and MST_ACTIVITY bits.
4	RFF (RF)	0	Receive FIFO Completely Full. 1: Receive FIFO is completely full 0: Receive FIFO is not full (one or more empty locations)
3	RFNE (RF)	0	Receive FIFO Not Empty. 1: Receive FIFO is not empty (one or more valid entries) 0: Receive FIFO is completely empty
2	TFE (TE)	0	Transmit FIFO Completely Empty. This bit field does not request an Interrupt. 1: Transmit FIFO is completely empty 0: Transmit FIFO is not empty (one or more valid entries)
1	TFNF (TN)	0	Transmit FIFO Not Full. 1: Transmit FIFO is not full (one or more empty locations) 0: Transmit FIFO is full

23.12.27 Transmit FIFO Level Register (TXFLR)

TXFLR																Offset = 0x0074
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									TXFLR							
W																
Reset	[00…0]								[00…0]							

Table 23.31: Description of the Transmit FIFO Level Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	TXFLR	[00…0]	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO

23.12.28 Receive FIFO Level Register (RXFLR)

RXFLR																Offset = 0x0078
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									RXFLR							
W																
Reset	[00…0]								[00…0]							

Table 23.32: Description of the Receive FIFO Level Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	RXFLR	[00…0]	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO

23.12.29 SDA Hold Time Length Register (SDA_HOLD)

Writes to this register succeed only when I2C.ENABLE[0]=0.

The values in this register are in units of PCLK period. The value programmed in I2C.SDA_TX_HOLD must be greater than the minimum hold time in each mode —one cycle in master mode, seven cycles in slave mode —for the value to be implemented.

The programmed SDA hold time during transmit (I2C.SDA_TX_HOLD) cannot exceed at any time the duration of the low part of SCL. Therefore, the programmed value cannot be larger than N_SCL_LOW-2, where N_SCL_LOW is the duration of the low part of the SCL period measured in PCLK cycles

SDA_HOLD																**Offset = 0x007C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									SDA_RX_HOLD							
W																
Reset	[00…0]								[00…0]							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SDA_TX_HOLD															
W																
Reset	[00…0]															

Table 23.33: Description of the SDA Hold Time Length Register

Bit Number(s)	Bit Name	Reset State	Description
31-24	RESERVED	[00…0]	
23-16	SDA_RX_HOLD	[00…0]	Sets the required SDA hold time (while SCL is HIGH in the receiver) in units of PCLK period, when I2C acts as a receiver. This applies to both slave and master mode.
15-0	SDA_TX_HOLD	[00…0]	Sets the required SDA hold time (after SCL goes from HIGH to LOW) in units of PCLK period, when I2C acts as a transmitter. This applies to both slave and master mode.

23.12.30 Transmit Abort Source Register (TX_ABORT_SOURCE)

TX_ABORT_SOURCE																Offset = 0x0080
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TX_FLUSH_CNT															UA
W																
Reset	[00…0]									[00…0]						0

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SI	SA	ST	AL	MD	RN	SN	HN	SB	HA	GR	GN	TN	A2N	A1N	7AN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23.34: Description of the Transmit Abort Source Register

Bit Number(s)	Bit Name	Reset State	Description
31-23	TX_FLUSH_CNT	[00...0]	This field indicates the number of TX FIFO data commands that are flushed due to TX_ABRT Interrupt. It is cleared whenever I2C is disabled. Master or Slave Transmitter
22-17	RESERVED	[00…0]	
16	ABRT_USER _ABRT (UA)	0	1: Master has detected the transfer abort (I2C.ENABLE[1]). Master Transmitter
15	ABRT_SLVRD _INTX (SI)	0	1: When the processor side responds to a slave mode request for data to be transmitted to a remote master AND the user wrote a 1 in CMD (bit 8) of the I2C.DATA_CMD register. See I2C.DATA for more information. Slave Transmitter
14	ABRT_SLV_ ARBLST (SA)	0	1: Slave lost the bus while transmitting data to a remote master. I2C.TX_ABRT_SOURCE[12] is set at the same time Slave Transmitter
13	ABRT_SLVFLUSH_T XFIFO (ST)	0	1: Slave has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT Interrupt to flush old data in TX FIFO. Slave Transmitter
12	ARB_LOST (AL)	0	1: Master has lost arbitration, or if I2C.TX_ABRT_SOURCE[14] is also set, then the slave transmitter has lost arbitration. Master or Slave Transmitter
11	ABRT_MASTER _DIS (MD)	0	1: User tries to initiate a Master operation with the Master mode disabled Master Transmitter or Receiver
10	ABRT_10B_RD _NORSTRT (RN)	0	1: The restart is disabled (IC_RESTART_EN bit (I2C.CON[5]) = 0) and the master sends a read command in 10-bit addressing mode Master Receiver
9	ABRT_SBYTE _NORSTRT (SN)	0	To clear bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; restart must be enabled (I2C.CON[5]=1), the SPECIAL bit must be cleared (I2C.TAR[11]), or the GC_OR_START bit must be cleared (I2C.TAR[10]). Once the source of the ABRT_SBYTE_NORSTRT is fixed,

Bit Number(s)	Bit Name	Reset State	Description
			then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, bit 9 clears for one cycle and then gets re-asserted. 1: The restart is disabled (IC_RESTART_EN bit (I2C.CON[5])=0) and the user is trying to send a START byte. Master
8	ABRT_HS _NORSTRT (HN)	0	1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) = 0) and the user is trying to use the master to transfer data in High Speed mode. Master Transmitter or Receiver
7	ABRT_SBYTE _ACKDET (SB)	0	1: Master has sent a START Byte and the START Byte was acknowledged (wrong behavior). Master
6	ABRT_HS _ACKDET (HA)	0	1: Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior). Master
5	ABRT_GCALL _READ (GR)	0	1: I2C in master mode sent a General Call but the user programmed the byte following the General Call to be a read from the bus (I2C.DATA_CMD[9] is set to 1). Master Transmitter
4	ABRT_GCALL _NOACK (GN)	0	1: I2C in master mode sent a General Call and no slave on the bus acknowledged the General Call. Master Transmitter
3	ABRT_TXDATA _NOACK (TN)	0	1: This is a master-mode only bit. Master has received an acknowledgement for the address, but when it sent data byte(s) following the address, it did not receive an acknowledge from the remote slave(s). Master Transmitter
2	ABRT_10ADDR2 _NOACK (A2N)	0	1: Master is in 10-bit address mode and the second address byte of the 10-bit address was not acknowledged by any slave. Master Transmitter or Receiver
1	ABRT_10ADDR1 _NOACK (A1N)	0	1: Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave. Master Transmitter or Receive
0	ABRT_7B_ADDR _NOACK (7AN)	0	1: Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave. Master Transmitter or Receiver

23.12.31 Generate Slave Data NACK Register (SLV_DATA_NACK_ONLY)

The register is used to generate a NACK for the data part of a transfer when I2C is acting as a slave-receiver. This register only exists when the IC_SLV_DATA_NACK_ONLY parameter is set to 1. When this parameter disabled, this register does not exist and writing to the register's address has no effect.

A write can occur on this register if both of the following conditions are met:

- I2C is disabled (I2C.ENABLE[0] = 0)
- Slave part is inactive (I2C.STATUS[6] = 0)

SLV_DATA_NACK_ONLY																**Offset = 0x0084**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																NA
W																
Reset	[00…0]															0

Table 23.35: Description of the Generate Slave Data NACK Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	NACK (NA)	0	Generate NACK. This NACK generation only occurs when I2C is a slave receiver. If this register is set to a value of 1, it can only generate a NACK after a data byte is received; hence the data transfer is aborted and the data received is not pushed to the receive buffer. When the register is set to a value of 0, it generates NACK/ACK, depending on normal criteria. 1: Generate NACK after data byte received 0: Generate NACK/ACK normally

23.12.32 SDA Setup Register (SDA_SETUP)

This register controls the amount of time delay (in terms of number of PCLK clock periods) introduced in the rising edge of SCL—relative to SDA changing—by holding SCL low when I2C services a read request while operating as a slave-transmitter. The relevant I2C requirement is tSU:DAT (note 4) as detailed in the I2C Bus Specification. This register must be programmed with a value equal to or greater than 2.

Writes to this register succeed only when I2C.ENABLE[0] = 0.

SDA_SETUP																Offset = 0x0094
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									SDA_SETUP							
W																
Reset	[00…0]								[00…0]							

Table 23.36: Description of the SDA Setup Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00…0]	
7-0	SDA_SETUP	[00…0]	SDA Setup. It is recommended that if the required delay is 1000ns, then for an I2C clock frequency of 10 MHz, I2C.SDA_SETUP should be programmed to a value of 11. I2C.SDA_SETUP must be programmed with a minimum value of 2.

23.12.33 ACK General Call Register (ACK_GENERAL_CALL)

The register controls whether I2C responds with an ACK or NACK when it receives an I2C General Call address. This register is applicable only when the I2C is in the slave mode.

ACK_GENERAL_CALL																Offset = 0x0098
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																AG
W																
Reset	[00…0]															0

Table 23.37: Description of the ACK General Call Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	ACK_GEN_CALL (AG)	0	ACK General Call. 1: The I2C responds with an ACK (by asserting IC_DATA_OE) when it receives a General Call. 0: The I2C does not generate General Call Interrupts

23.12.34 Enable Status Register (ENABLE_STATUS)

The register is used to report the I2C hardware status when I2C.ENABLE[0] is set from 1 to 0; that is, when I2C is disabled.

- If I2C.ENABLE[0] has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1.
- If I2C.ENABLE[0] has been set to 0, bits 2:1 is only be valid as soon as bit 0 is read as '0'.

ENABLE_STATUS																**Offset = 0x009C**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														DL	DB	EN
W																
Reset	[00...0]													0	0	0

Table 23.38: Description of the Enable Status Register

Bit Number(s)	Bit Name	Reset State	Description
31-3	RESERVED	[00...0]	
2	SLV_RX_DATA _LOST (DL)	0	Slave Receiver Data Lost. This bit indicates if a Slave Receiver operation has been aborted with at least one data byte received from an I2C transfer due to setting IC_ENABLE[0] from 1 to 0. 1: I2C is deemed to have been actively engaged in an aborted I2C transfer (with matching address) and the data phase of the I2C transfer has been entered, even though a data byte has been responded with a NACK. NOTE: If the remote I2C master terminates the transfer with a STOP condition before the I2C has a chance to NACK a transfer, and I2C.ENABLE[0] has been set to 0, then this bit is also set to 1. 0: I2C is deemed to have been disabled without being actively involved in the data phase of a Slave Receiver transfer. NOTE: the CPU can safely read this bit when IC_EN (bit 0) is read as 0.
1	SLV_DISABLE _WHILE_BUSY (DB)	0	Slave Disabled While Busy. This bit indicates if a potential or active Slave operation has been aborted due to setting I2C.ENABLE[0] from 0 to 1. This bit is set when the CPU writes a 0 to I2C.ENABLE[0] while: (a) I2C is receiving the address byte of the Slave Transmitter operations from a remote master, OR, (b) address and data bytes of the Slave Receiver operation from a remote master. 1: The I2C is deemed to have forced a NACK during any part of an I2C transfer, irrespective of whether the I2C address matches the slave address set in I2C (I2C.SAR register) OR if the transfer is completed before bit 0 of IC_ENABLE is set to 0, but has not taken effect. NOTE: If the remote I2C master terminates the transfer with a STOP condition before the I2C peripheral has a chance to NACK a transfer, and of I2C.ENABLE[0] has been set to 0, then this bit will also be set to 1. 0: The I2C is deemed to have been disabled when there is master activity, or when the I2C bus is idle. NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0
0	IC_EN	0	IC_EN Status 1: I2C is deemed to be in an enabled state 0: I2C is deemed to be completely inactive NOTE: The CPU can safely read this bit anytime. When this bit is read as 0, the CPU can safely read SLV_RX_DATA_LOST[2] and SLV_DISABLED_WHILE_BUSY[1]

23.12.35 SS and FS Spike Suppression Limit Register (FS_SPKLEN)

FS_SPKLEN																Offset = 0x00A0
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									C_FS_SPKLEN							
W																
Reset	[00…0]								[00…0]							

Table 23.39: Description of the SS and FS Spike Suppression Limit Register

Bit Number(s)	Bit Name	Reset State	Description
31-8	RESERVED	[00...0]	
7-0	C_FS_SPKLEN	[00…0]	This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in IC_CLK, of the longest spike in the SCL or SDA lines that are filtered out by the spike suppression logic; for more information, refer to “Spike Suppression”. This register can be written only when the I2C interface is disabled, which corresponds to I2C.ENABLE[0] being set to 0. Writes at other times have no effect. The minimum valid value is 1; hardware prevents values less than this being written, and if attempted, results in 1 being set. The relevant I2C requirement is tSP (Table 4) as detailed in the I2C Bus Specification.

23.12.36 HS Spike Suppression Limit Register (HS_SPKLEN)

HS_SPKLEN																Offset = 0x00A4
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00…0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																
Reset	[00…0]															

Table 23.40: Description of the HS Spike Suppression Limit Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	RESERVED	[00...0]	

23.12.37 Clear RESTART_DET Interrupt Register (CLR_RESTART_DET)

CLR_RESTART_DET																Offset = 0x00A8
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																CR
W																
Reset	[00...0]															0

Table 23.41: Description of the Clear RESTART_DET Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
31-1	RESERVED	[00...0]	
0	CLR_RESTART _DET (CR)	0	Read this register to clear the RESTART_DET Interrupt (I2C.RAW_INTR_STAT[12])

23.12.38 Component Parameter Register 1 (COMP_PARAM_1)

COMP_PARAM_1								Offset = 0x00F4								
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									TX_BUFFER_DEPTH							
W																
Reset	[00...0]								0x07							

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RX_BUFFER_DEPTH								AE	HD	IO	CV	MSM		ADW	
W																
Reset	0x07								1	0	1	0	10		10	

Table 23.42: Description of the Component Parameter Register 1

Bit Number(s)	Bit Name	Reset State	Description
31-24	RESERVED	[00...0]	
23-16	TX_BUFFER _DEPTH	0x07	TX FIFO depth. 0x00 = Reserved 0x01 = 2 0x02 = 3 ... 0xFF = 256
15-8	RX_BUFFER _DEPTH	0X07	RX FIFO depth 0x00 = Reserved 0x01 = 2 words 0x02 = 3 words ... 0xFF = 256 words
7	ADD_ENCODED _PARAMS (AE)	1	1: The capability of reading these encoded parameters via software has been included. 0: The entire register is 0 regardless of the setting of any other bits
6	HAS_DMA (HD)	0	1: The device has DMA 0: The device does not have DMA
5	INTR_IO (IO)	1	1: Combined Interrupts 0: Individual Interrupts
4	HC_COUNT _VALUES (CV)	0	1: True 0: False
3-2	MAX_SPEED _MODE (MSM)	10	11: High 10: Fast 01: Standard 00: Reserved
1-0	APD_DATA _WIDTH (ADW)	10	11: Reserved 10: 32 bits 01: 16 bits 00: 8 bits

23.12.39 Component Version Register (COMP_VERSION)

COMP_VERSION																**Offset = 0x00F8**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IC_COMP_VERSION															
W																
Reset	[00...0]															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IC_COMP_VERSION															
W																
Reset	[00...0]															

Table 23.43: Description of the Component Version Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	IC_COMP _VERSION	[00...0]	Specific values for this register are described in the Releases Table in the AMBA 2 release notes

23.12.40 Component Type Register (COMP_TYPE)

COMP_TYPE																**Offset = 0x00FC**
Bit#	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IC_COMP_TYPE															
W																
Reset	0x4457															

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IC_COMP_TYPE															
W																
Reset	0x0140															

Table 23.44: Description of the Component Type Register

Bit Number(s)	Bit Name	Reset State	Description
31-0	IC_COMP_TYPE	0x4457 _0140	I2C IP type number = 0x4457_0140

24 Controller Area Network (CAN-2.0)

24.1 Overview

The CAN-2.0 interface in the UT32M0R500 is based on the CAN core from Frontgrade with an AHB slave interface for accessing all CAN core registers. The CAN core is a derivative of the Philips SJA1000 and has a compatible register map with a few exceptions. Each CAN core is capable of up to 1 Mb/s Baud rate. These exceptions are indicated in the register description tables and in the Function Differences for Design section. The CAN core supports both BasicCAN and PeliCAN modes. In PeliCAN mode the extended features of CAN 2.0B are supported. The mode of operation is chosen through the clock divider register.

This chapter lists the CAN core registers and their functionality. The Philips SJA1000 data sheet can be used as an additional reference, except as noted in the Function Differences for Design section. The register map and functionality is different depending upon which mode of operation is selected. BasicCAN mode will be described in BasicCAN Mode of Operation, followed by PeliCAN in PeliCAN Mode of Operation. The common registers (Clock Divisor and Bus Timing) are described in Common Registers. The register map also differs depending on whether the core is in operating mode or in reset mode. After reset, the CAN core starts up in reset mode awaiting configuration. Operating mode is entered by clearing the Reset Request (CR.0) bit in the Control Register. Set the bit to re-enter reset mode.

The UT32M0R500 implements two identical instances of the CAN-2.0 core. Both operate completely independent of each other. The AHB register mapping for each core is indicated in the Arm Cortex M0+ Memory Map section.

The user is referenced to the ISO 11898 Road Vehicles – Controller Area Network (CAN) Specification and BOSCH CAN Specification version 2.0 for further details.

24.2 BasicCAN Mode of Operation

24.2.1 BasiCAN Register Listing

The register map also differs depending on whether the core is in operating mode or in reset mode. When reset the core starts in reset mode awaiting configuration. Operating mode is entered by clearing the reset request bit in the command register. To re-enter reset mode set this bit high again.

Table 24.1: BasiCAN Registers

Register Offset (Hex)	OPERATING MODE		RESET MODE	
	Read	Write	Read	Write
00	Control	Control	Control	Control
01	(0xFF)	Command	(0xFF)	Command
02	Status	-	Status	-
03	Interrupt	-	Interrupt	-
04	(0xFF)	-	Acceptance code	Acceptance code
05	(0xFF)	-	Acceptance mask	Acceptance mask
06	(0xFF)	-	Bus timing 0	Bus timing 0
07	(0xFF)	-	Bus timing 1	Bus timing 1

Register Offset (Hex)	OPERATING MODE		RESET MODE	
	Read	Write	Read	Write
08	(0x00)	n/a	(0x00)	n/a
09	(0x00)	n/a	(0x00)	n/a
0A	TX ID1	TX ID1	(0xFF)	-
0B	TX ID2, rtr, dlc	TX ID2, rtr, dlc	(0xFF)	-
0C	TX data byte 1	TX data byte 1	(0xFF)	-
0D	TX data byte 2	TX data byte 2	(0xFF)	-
0E	TX data byte 3	TX data byte 3	(0xFF)	-
0F	TX data byte 4	TX data byte 4	(0xFF)	-
10	TX data byte 5	TX data byte 5	(0xFF)	-
11	TX data byte 6	TX data byte 6	(0xFF)	-
12	TX data byte 7	TX data byte 7	(0xFF)	-
13	TX data byte 8	TX data byte 8	(0xFF)	-
14	RX ID1	-	RX ID1	-
15	RX ID2, rtr, dlc	-	RX ID2, rtr, dlc	-
16	RX data byte 1	-	RX data byte 1	-
17	RX data byte 2	-	RX data byte 2	-
18	RX data byte 3	-	RX data byte 3	-
19	RX data byte 4	-	RX data byte 4	-
1A	RX data byte 5	-	RX data byte 5	-
1B	RX data byte 6	-	RX data byte 6	-
1C	RX data byte 7	-	RX data byte 7	-
1D	RX data byte 7	-	RX data byte 7	-
1E	(0x00)	(0x00)	(0x00)	(0x00)
1F	Clock divider	Clock divider	Clock divider	Clock divider

24.2.2 Control Register (CONTROL)

The control register contains Interrupt enable bits as well as the reset request bit. Bit interpretation of control register (CONTROL) (address 0).

CONTROL								**Offset = 0x0000**
Bit#	7	6	5	4	3	2	1	0
R				OIE	EIE	TIE	RIE	RR
W								
Reset	001			0	0	0	0	1

Table 24.2: Description of BasiCAN Control Register

Bit Number(s)	Bit Name	Reset State	Description
7-5	RESERVED	001	
4	OIE	0	Overrun Interrupt Enable 1: Enabled 0: Disabled
3	EIE	0	Error Interrupt Enable 1: Enabled 0: Disabled
2	TIE	0	Transmit Interrupt Enable 1: Enabled 0: Disabled
1	RIE	0	Receive Interrupt Enable 1: Enabled 0: Disabled
0	RR	1	Reset Request 1: Writing 1 aborts any ongoing transfer and enters reset mode 0: Writing 0 returns to operating mode

24.2.3 Command Register (COMMAND)

A transmission is started by writing 1 to BASI_CAN.COMMAND.0 bit. It can only be aborted by writing 1 to BASI_CAN.COMMAND.1 and only if the transfer has not yet started. If the transmission has started it will not be aborted when setting BASI_CAN.COMMAND.1 but it will not be retransmitted if an error occurs.

Giving the release receive buffer command should be done after reading the contents of the receive buffer in order to release this memory. If there is another message waiting in the FIFO a new receive Interrupt will be generated (if enabled) and the receive buffer status bit will be set again. To clear the Data overrun status bit BASI_CAN.COMMAND.3 must be written with 1.

COMMAND								Offset = 0x0001
Bit#	7	6	5	4	3	2	1	0
R								
W				GTS	CDO	RRB	AT	TR
Reset	111			1	1	1	1	1

Table 24.3: Description of BasiCAN Command Register

Bit Number(s)	Bit Name	Reset State	Description
7-5	RESERVED	111	
4	GTS	1	Not Used (go to sleep in SJA1000)
3	CDO	1	Clear Data Overrun 1: Data overrun stat bit is cleared 0: No action
2	RRB	1	Release Receive Buffer 1: Free the current receive buffer for new reception 0: No action
1	AT	1	Abort Transmission 1: Aborts a not yet started transmission 0: No action
0	TR	1	Transmission Request 1: Start the transfer of the message in the TX buffer 0: No action

24.2.4 Status Register (STATUS)

The status register is read-only and reflects the current status of the core. Receive buffer status is cleared when the Release receive buffer command is given and set high if there are more messages available in the FIFO. When the transmit buffer status is high the transmit buffer is available to be written into by the CPU. During an on-going transmission the buffer is locked and this bit is 0.

The transmission complete bit is set to 0 when a transmission request has been issued and will not be set to 1 again until a message has successfully been transmitted. The data overrun status signals that a message which was accepted could not be placed in the FIFO because not enough space left.

STATUS								Offset = 0x0002
Bit#	7	6	5	4	3	2	1	0
R	BS	ES	TS	RS	TCS	TBS	DOS	RBS
W								
Reset	0	0	0	0	1	1	0	0

Table 24.4: Description of BasiCAN Status Register

Bit Number(s)	Bit Name	Reset State	Description
7	BS	0	Bus Status 1: Core is in the bus-off and not involved in bus activities 0: The controller is involved in bus activities
6	ES	0	Error Status 1: One of the error counters has reached or exceeded the CPU warning limit of 96 0: Both error counters are below the warning limit
5	TS	0	Transmit Status 1: When transmitting a message 0: No transmit message is in progress
4	RS	0	Receive Status 1: When receiving a message 0: No receive message is in progress
3	TCS	1	Transmission Complete Status 1: The last requested transmission was successfully transferred 0: The previously requested transmission has not transferred
2	TBS	1	Transmit Buffer Status 1: The CPU can write into the transmit buffer 0: The CPU cannot access the transmit buffer
1	DOS	0	Data Overrun Status 1: Message was lost because there was not enough space in the FIFO 0: No data overrun has occurred
0	RBS	0	Receive Buffer Status 1: One or more messages are available in the FIFO 0: No message is available

24.2.5 Interrupt Register (INTERRUPT)

The *Interrupt register signals to CPU* what caused the Interrupt. The Interrupt bits are only set if the corresponding Interrupt enable bit is set in the control register.

This register is reset on read with the exception of BASI_CAN.INTERRUPT.0. Note that this differs from the SJA1000 behavior where all bits are reset on read in BasicCAN mode. This core resets the receive Interrupt bit when the release receive buffer command is given (like in PeliCAN mode). Also note that bit BASI_CAN.INTERRUPT.5 through BASI_CAN.INTERRUPT.7 reads as 1 but BASI_CAN.INTERRUPT.4 is 0.

INTERRUPT								**Offset = 0x0003**
Bit#	7	6	5	4	3	2	1	0
R					DOI	EI	TI	RI
W								
Reset	1110				0	0	0	0

Table 24.5: Description of BasiCAN Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
7-4	RESERVED	1110	
3	DOI	0	Data Overrun Interrupt 1: Set on a '0-to-1' transition of the STATUS.1 (DOS) bit 0: Cleared by any read access
2	EI	0	Error Interrupt 1: Set on change of either the error status or bus status bits 0: This bit is cleared by any read access
1	TI	0	Transmit Interrupt 1: Set on a '0-to-1' transition of the STATUS.2 (TBS) bit 0: Cleared by any read access
0	RI	0	Receive Interrupt 1: Set while the receive FIFO is not empty 0: This bit is cleared when the receive FIFO has been emptied

24.2.6 Transmit Buffer Layout

Address Offset: 0x0A – 0x13
Reset Value (Address 0x0A – 0x13): 0xFF

The table below shows the layout of the *transmit buffer.* In BasicCAN only standard frame messages can be transmitted and received (EFF messages on the bus are ignored).

If the RTR bit (ID byte2.4) is set no data bytes will be sent but DLC is still part of the frame and must be specified according to the requested frame. Note that it is possible to specify a DLC larger than 8 bytes but should not be done for compatibility reasons. If DLC > 8, then only 8 bytes are sent.

Offset	Name	Bits							
		7	6	5	4	3	2	1	0
0A	ID byte 1	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
0B	ID byte 2	ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
0C	TX data 1	TX byte 1							
0D	TX data 2	TX byte 2							
0E	TX data 3	TX byte 3							
0F	TX data 4	TX byte 4							
10	TX data 5	TX byte 5							
11	TX data 6	TX byte 6							
12	TX data 7	TX byte 7							
13	TX data 8	TX byte 8							

24.2.7 Receive Buffer Layout

Address Offset: 0x14 – 0x1D
Reset Value (Address 0x1F – 0x1D): 0xUU (unknown random value, requires received data transfers)

The receive buffer on address 0x14 through 0x1D is the visible part of the 64-byte RX FIFO. Its layout is identical to that of the transmit buffer.

24.2.8 Acceptance Filter

Messages can be filtered based on their identifiers using the acceptance code and acceptance mask registers. The top 8 bits of the 11-bit identifier(ID.10 to ID.0 as shown transmit buffer above) are compared with the acceptance code register only comparing the bits set to zero(0's) in the acceptance mask register. If a match is detected the message is stored to the FIFO.

24.2.9 ACCEPTANCE CODE Register (ACCEPT_CODE)

Reset value unknown, requires received data transfer

ACCEPT_CODE								**Offset = 0x0004**
Bit#	7	6	5	4	3	2	1	0
R								
W								
Reset	[UU…U]							

24.2.10 ACCEPTANCE MASK Register (ACCEPT_MASK)

Reset value unknown, requires received data transfer

ACCEPT_MASK								**Offset = 0x0005**
Bit#	7	6	5	4	3	2	1	0
R								
W								
Reset	[UU…U]							

24.4 PeliCAN Mode of Operation

The PeliCAN mode of operation is selected using the Clock Divide Register (CLOCK_DIVER), bit 7.

24.4.1 PeliCAN Register Listing

The transmit and receive buffers have different layout depending on if standard frame format (SFF) or extended frame format (EFF) is to be transmitted/received. SFF or EFF is selected using the Transmit Buffer Frame Info, bit 7.

Table 24.6: PeliCAN Registers

Register Offset (Hex)	Operating Mode				Reset Mode	
	Read		Write		Read	Write
00	Mode		Mode		Mode	Mode
01	(0x00)		Command		(0x00)	Command
02	Status		-		Status	NA
03	Interrupt		-		Interrupt	NA
04	Interrupt enable		Interrupt enable		Interrupt enable	Interrupt enable
05	Reserved(0x00)		-		Reserved(0x00)	NA
06	Bus Timing 0		-		Bus Timing 0	Bus Timing 0
07	Bus Timing 1		-		Bus Timing 1	Bus Timing 1
08	(0x00)		-		(0x00)	-
09	(0x00)		-		(0x00)	-
0A	Reserved(0x00)		-		Reserved(0x00)	-
0B	Arbitration lost capture		-		Arbitration lost capture	-
0C	Error code capture		-		Error code capture	-
0D	Error warning limit		-		Error warning limit	Error warning limit
0E	RX error counter		-		RX error counter	RX error counter
0F	TX error counter		-		TX error counter	TX error counter
10	RX FI SFF	RX FI EFF	TX FI SFF	TX FI EFF	Acceptance code 0	Acceptance code 0
11	RX ID 1	RX ID 1	TX ID 1	TX ID 1	Acceptance code 1	Acceptance code 1
12	RX ID 2	RX ID 2	TX ID 2	TX ID 2	Acceptance code 2	Acceptance code 2
13	RX data 1	RX ID 3	TX data 1	TX ID 3	Acceptance code 3	Acceptance code 3
14	RX data 2	RX ID 4	TX data 2	TX ID 4	Acceptance mask 0	Acceptance mask 0
15	RX data 3	RX data 1	TX data 3	TX data 1	Acceptance mask 1	Acceptance mask 1
16	RX data 4	RX data 2	TX data 4	TX data 2	Acceptance mask 2	Acceptance mask 2
17	RX data 5	RX data 3	TX data 5	TX data 3	Acceptance mask 3	Acceptance mask 3
18	RX data 6	RX data 4	TX data 6	TX data 4	reserved (0x00)	-
19	RX data 7	RX data 5	TX data 7	TX data 5	reserved (0x00)	-
1A	RX data 8	RX data 6	TX data 8	TX data 6	reserved (0x00)	-
1B	FIFO	RX data 7	-	TX data 7	reserved (0x00)	-
1C	FIFO	RX data 8	-	TX data 8	reserved (0x00)	-
1D	RX message counter		-		RX message counter	-
1E	(0x00)		-		(0x00)	-
1F	Clock divider		Clock divider		Clock divider	Clock divider

24.4.2 Mode Register (MODE)

Writing to MODE.1-3 can only be done when reset mode has been entered previously. In Listen only mode, the core will not send any acknowledgements. *Note that unlike the SJA1000 the core does not become error passive and active error frames are still sent.*

When in Self-test mode the core can complete a successful transmission without getting an acknowledgement

if given the Self reception request command. *Note that the core must still be connected to a real bus, it does not do an internal loopback.*

MODE								**Offset = 0x0000**
Bit#	7	6	5	4	3	2	1	0
R				SM	AFM	STM	LOM	RM
W								
Reset	000			0	0	0	0	1

Table 24.7: Description of PeliCAN Mode Register

Bit Number(s)	Bit Name	Reset State	Description
7-5	RESERVED	000	
4	SM	0	Not used (SJA1000 sleep mode)
3	AFM	0	Acceptance Filter Mode 1: The single acceptance filter option is enabled 0: The dual acceptance filter option is enabled
2	STM	0	Self-Test Mode 1: This mode allows for a full node test without any other active node on the bus using the self-reception request command; the CAN controller will perform a successful transmission, even if there is no acknowledge received 0: An acknowledge is required for successful transmission
1	LOM	0	Listen-Only Mode 1: Set the controller in listen-only mode 0: Normal
0	RM	1	Reset Mode 1: Detection of a set reset mode bit results in aborting the current transmission/reception of a message and entering the reset mode 0: On the '1-to-0' transition of the reset mode bit, the CAN controller returns to the operating mode

24.4.3 Command Register (COMMAND)

A transmission is started by writing 1 to COMMAND.0. It can only be aborted by writing 1 to COMMAND.1 and only if the transfer has not yet started. Setting COMMAND.0 and COMMAND.1 simultaneously will result in a so called single shot transfer, i.e. the core will not try to retransmit the message if not successful the first time.

Giving the Release receive buffer command should be done after reading the contents of the receive buffer in order to release this memory. If there is another message waiting in the FIFO a new receive Interrupt will be generated (if enabled) and the receive buffer status bit will be set again.

The Self reception request bit together with the self-test mode makes it possible to do a self-test of the core without any other cores on the bus. A message will simultaneously be transmitted and received and both receive and transmit Interrupt will be generated.

COMMAND								**Offset = 0x0001**
Bit#	7	6	5	4	3	2	1	0
R				SRR	CDO	RRB	AT	TR
W								
Reset	000			0	0	0	0	0

Table 24.8: Description of PeliCAN Command Register

Bit Number(s)	Bit Name	Reset State	Description
7-5	RESERVED	000	
4	SSR	0	Self-Reception Request 1: A message shall be transmitted and received simultaneously 0: No action
3	CDO	0	Clear Data Overrun 1: The data overrun status bit (DOS/SR.1) is cleared 0: No action
2	RRB	0	Release Receive Buffer 1: The received buffer is released and available for new reception 0: No action
1	AT	0	Abort Transmission 1: If not already in progress, a pending transmission request is cancelled 0: No action
0	TR	0	Transmission Request 1: Starts the transfer of the message in the TX buffer 0: No action

24.4.4 Status Register (STATUS)

The status register is read only and reflects the current status of the core. Receive buffer status is cleared when there are no more messages in the FIFO. The data overrun status signals that a message which was accepted could not be placed in the FIFO because not enough space left. When the transmit buffer status is high the transmit buffer is available to be written into by the CPU. During an on-going transmission the buffer is locked and this bit is 0.

The transmission complete bit is set to 0 when a transmission request or self-reception request has been issued and will not be set to 1 again until a message has successfully been transmitted.

STATUS								**Offset = 0x0002**
Bit#	7	6	5	4	3	2	1	0
R	BS	ES	TS	RS	TCS	TBS	DOS	RBS
W								
Reset	0	0	1	1	1	1	0	0

Table 24.9: Description of PeliCAN Status Register

Bit Number(S)	Bit Name	Reset State	Description
7	BS	0	Bus Status 1: CAN controller is not involved in bus activities 0: CAN controller is involved in bus activities
6	ES	0	Error Status 1: At least one of the error counters have reached or exceeded the error warning limit defined by the Error Warning Limit register (EWLR) 0: Both error counters are below the error warning limit
5	TS	1	Transmit Status(1) 1: The CAN controller is transmitting a message 0: The CAN controller is not transmitting a message
4	RS	1	Receive Status(1) 1: The CAN controller is receiving a message 0: The CAN controller is not receiving a message
3	TCS	1	Transmission Complete Status 1: Last message requested transmission has completed successfully 0: Previous request transmission has not yet completed
2	TBS	1	Transmit Buffer Status 1: The CPU may write a message into the transmit buffer 0: The CPU cannot access the transmit buffer
1	DOS	0	Data Overrun Status 1: A message was lost because there was not enough space in the FIFO 0: No message was lost
0	RBS	0	Receiver Buffer Status 1: One or more messages are available in the receive FIFO 0: no message is available

Note 1: The '1' value after reset of Transmit Status and Receive Status have been observed, which are not indicative of the message activity until the start of the first bus transaction.

24.4.5 Interrupt Register (INTERRUPT)

The Interrupt register signals to the CPU what caused the Interrupt. The Interrupt bits are only set if the corresponding Interrupt enable bit is set in the Interrupt enable register.

This register is reset on read with the exception of PELI_CAN.INTERRUPT.0 which is reset when the FIFO has been emptied.

INTERRUPT								**Offset = 0x0003**
Bit#	7	6	5	4	3	2	1	0
R	BEI	ALI	EPI		DOI	EI	TI	RI
W								
Reset	0	0	0	0	0	0	0	0

Table 24.10: Description of PeliCAN Interrupt Register

Bit Number(s)	Bit Name	Reset State	Description
7	BEI	0	Bus Error Interrupt 1: This bit is set when the CAN controller detects an error on the CAN bus and the BEIE bit is set in the Interrupt enable register 0: This bit is cleared by any read access
6	ALI	0	Arbitration Lost Interrupt 1: This bit is set when the CAN controller lost the arbitration and becomes a receiver and the ALIE bit is set in the Interrupt enable register 0: This bit is cleared by any read access
5	EPI	0	Error Passive Interrupt 1: This bit is set whenever the CAN controller has reached the error passive status (at least one error counter exceeds the protocol defined level of 127) or if the CAN controller is in the error passive status and enter error active status again and the EPIE bit is set in the Interrupt enable register 0: This bit is cleared by any read access
4	RESERVED	0	Not used (SJA1000 wake-up Interrupt)
3	DOI	0	Data Overrun Interrupt 1: This bit is set on a '0-to-1' transition of the data overrun status (DOS) bit and the DOIE bit is set in the Interrupt enable register 0: This bit is cleared by any read access
2	EI	0	Error Warning Interrupt 1: This be is set on every change (set and clear) of either the error status or bus status bits and the EIE bit is set in the Interrupt enable register 0: This bit is cleared by any read access
1	TI	0	Transmit Interrupt 1: This bit is set whenever the transmit buffer status changes from '0-to-1' (released) and the TIE bit is set in the Interrupt enable register 0: This bit is cleared by any read access
0	RI	0	Receiver Interrupt 1: This bit is set while the receive FIFO is not empty and the RIE bit is set in the Interrupt enable register 0: This bit is cleared when the receive FIFO is emptied

24.4.6 Interrupt Enable Register (INTERRUPT_ENABLE)

In the Interrupt enable register the separate Interrupt sources can be enabled/disabled. If enabled the corresponding bit in the Interrupt register can be set and an Interrupt generated.

INTERRUPT_ENABLE								**Offset = 0x0004**
Bit#	7	6	5	4	3	2	1	0
R W	BEIE	ALIE	EPIE		DOIE	EIE	TIE	RIE
Reset	U	U	U	U	U	U	U	U

Table 24.11: Description of PeliCAN Interrupt Enable Register

Bit Number(s)	Bit Name	Reset State	Description
7	BEIE	U	Bus Error Interrupt Enable 1: Enabled 0: Disabled
6	ALIE	U	Arbitration Lost Interrupt Enable 1: Enabled 0: Disabled
5	EPIE	U	Error Passive Interrupt Enable 1: Enabled 0: Disabled
4	RESERVED	U	Not Used (SJA1000 wake-up Interrupt enabled)
3	DOIE	U	Data Overrun Interrupt Enable 1: Enabled 0: Disabled
2	EIE	U	Error Warning Interrupt Enable 1: Enabled 0: Disabled
1	TIE	U	Transmit Interrupt Enable 1: Enabled 0: Disabled
0	RIE	U	Receiver Interrupt Enable 1: Enabled 0: Disabled

24.4.7 Arbitration Lost Capture Register (ARB_LOST_CAPTURE)

When the core loses arbitration the bit position of the bit stream processor is captured into arbitration lost capture register. The register will not change content again until read out.

ARB_LOST_CAPTURE								**Offset = 0x000B**
Bit#	7	6	5	4	3	2	1	0
R				BITNO4	BITNO3	BITNO2	BITNO1	BITNO0
W								
Reset	000			0	0	0	0	0

Table 24.12: Description of PeliCAN Arbitration Lost Capture Register

Bit Number(s)	Bit Name	Reset State	Description
7-5	RESERVED	000	
4-0	BITNOx	[00…0]	Bit number x; See Table 6.11 for function

Table 24.13: Function of bits 4:0 of the Arbitration Lost Capture Register

BIT					Decimal Value	Description
4	3	2	1	0		
0	0	0	0	0	00	Arbitration lost in bit 1 of identifier
0	0	0	0	1	01	Arbitration lost in bit 2 of identifier
0	0	0	1	0	02	Arbitration lost in bit 3 of identifier
0	0	0	1	1	03	Arbitration lost in bit 4 of identifier
0	0	1	0	0	04	Arbitration lost in bit 5 of identifier
0	0	1	0	1	05	Arbitration lost in bit 6 of identifier
0	0	1	1	0	06	Arbitration lost in bit 7 of identifier
0	0	1	1	1	07	Arbitration lost in bit 8 of identifier
0	1	0	0	0	08	Arbitration lost in bit 9 of identifier
0	1	0	0	1	09	Arbitration lost in bit 10 of identifier
0	1	0	1	0	10	Arbitration lost in bit 11 of identifier
0	1	0	1	1	11	Arbitration lost in bit 12 of identifier
0	1	1	0	0	12	Arbitration lost in bit 13 of identifier
0	1	1	0	1	13	Arbitration lost in bit 14 of identifier
0	1	1	1	0	14	Arbitration lost in bit 15 of identifier
0	1	1	1	1	15	Arbitration lost in bit 16 of identifier
1	0	0	0	0	16	Arbitration lost in bit 17 of identifier
1	0	0	0	1	17	Arbitration lost in bit 18 of identifier
1	0	0	1	0	18	Arbitration lost in bit 19 of identifier
1	0	0	1	1	19	Arbitration lost in bit 20 of identifier
1	0	1	0	0	20	Arbitration lost in bit 21 of identifier
1	0	1	0	1	21	Arbitration lost in bit 22 of identifier
1	0	1	1	0	22	Arbitration lost in bit 23 of identifier
1	0	1	1	1	23	Arbitration lost in bit 24 of identifier
1	1	0	0	0	24	Arbitration lost in bit 25 of identifier
1	1	0	0	1	25	Arbitration lost in bit 26 of identifier
1	1	0	1	0	26	Arbitration lost in bit 27 of identifier
1	1	0	1	1	27	Arbitration lost in bit 28 of identifier
1	1	1	0	0	28	Arbitration lost in bit 29 of identifier
1	1	1	0	1	29	Arbitration lost in bit 30 of identifier
1	1	1	1	0	30	Arbitration lost in bit 31 of identifier
1	1	1	1	1	31	Arbitration lost in bit 32 of identifier

24.4.8 Error Code Capture Register (ERROR_CODE_CAPTURE)

This register contains information about the type and location of errors on the bus.

ERROR_CODE_CAPTURE								Offset = 0x000C
Bit#	7	6	5	4	3	2	1	0
R	ERRC1	ERRC0	DIR	SEG4	SEG3	SEG2	SEG1	SEG0
W								
Reset	0	0	0	0	0	0	0	0

Table 24.14: Description of PeliCAN Error Code Capture Register

Bit Number(s)	Bit Name	Reset State	Description
7-6	ERRCx	00	Error Code Number: See Table 6.13
5	DIR	0	Direction 1: Error occurred during reception 0: Error occurred during transmission
4-0	SEGx	[00…0]	Segment: See Table 6.14

Table 24.15: Bit interpretation of bits ECC.7 and ECC.6

Bit ECC.7	Bit ECC.6	Description
0	0	Bit error
0	1	Form error
1	0	Stuff error
1	1	Other type of error

Table 24.46: Bit interpretation of bits ECC.4 to ECC.0

ECC.4 – ECC.0	Description
0x03	Start of frame
0x02	ID.28 - ID.21
0x06	ID.20 - ID.18
0x04	Bit SRTR
0x05	Bit IDE
0x07	ID.17 - ID.13
0x0F	ID.12 - ID.5
0x0E	ID.4 - ID.0
0x0C	Bit RTR
0x0D	Reserved bit 1
0x09	Reserved bit 0
0x0B	Data length code
0x0A	Data field
0x08	CRC sequence
0x18	CRC delimiter
0x19	Acknowledge slot
0x1B	Acknowledge delimiter
0x1A	End of frame
0x12	Intermission
0x11	Active error flag
0x16	Passive error flag
0x13	Tolerate dominant bits
0x17	Error delimiter
0x1C	Overload flag

24.4.9 Error Warning Limit Register (ERROR_WARNING_LIMIT)

This register allows for setting the CPU error warning limit. It defaults to 96d. Note that this register is read/write in reset mode only. In operating mode it is read-only.

ERROR_WARNING_LIMIT								**Offset = 0x000D**
Bit#	7	6	5	4	3	2	1	0
R								
W								
Reset	0110_0000							

24.4.10 RX Error Counter Register (RX_ERROR_COUNTER)

This register reflects the current value of the receive error counter. After a hard reset this register is initialized to logic 0. In operating mode this register is read only. A write access to this register is possible in reset mode.

If a bus-off event occurs, the RX error counter is initialized to logic 0.

RX_ERROR_COUNTER								**Offset = 0x000E**
Bit#	7	6	5	4	3	2	1	0
R								
W								
Reset	[00…0]							

24.4.11 TX Error Counter Register (TX_ERROR_COUNTER)

A write access to this register is possible only in reset mode. After hardware reset this register is initialized to logic 0. If a bus-off event occurs, the TX error counter is initialized to 127 to count the minimum protocol defined time (128 occurrences of the bus-free signal). Reading the TX error counter during this time gives information about the status of the bus-off recovery.

The CPU can force a bus-off by writing 255 to this register. Note that unlike the SJA1000 this core will signal bus-off immediately and not first when entering operating mode. The bus-off recovery sequence starts when entering operating mode after writing 255 to this register in reset mode.

TX_ERROR_COUNTER								**Offset = 0x000F**
Bit#	7	6	5	4	3	2	1	0
R								
W								
Reset	[00…0]							

24.4.12 Transmit Buffer

Address Offset: 0x10 – 0x1C
Reset Value: 0xUU (unknown random value; a write is required after reset)

The transmit buffer is write-only and mapped on address 0x10 to 0x1C. Reading of this area is mapped to the receive buffer described in the next section. The layout of the transmit buffer depends on whether a standard frame (SFF) or an extended frame (EFF) is to be sent as seen below. The transmit buffer allows the definition of one transmit message with up to eight data bytes.

The transmit buffer layout is subdivided into descriptor and data fields where the first byte of the description field is the frame information byte (frame information). The transmit buffer has a length of 13 bytes and is located in the CAN address range from 0x10 to 0x1C.

Table 24.17: Transmit buffer layout for standard and extended frame formats

CAN Address (hex)	Standard Frame Format	Extended Frame Format
10	TX frame information	TX frame information
11	TX ID 1	TX ID 1
12	TX ID 2	TX ID 2
13	TX data 1	TX ID 3
14	TX data 2	TX ID 4
15	TX data 3	TX data 1
16	TX data 4	TX data 2
17	TX data 5	TX data 3
18	TX data 6	TX data 4
19	TX data 7	TX data 5
1A	TX data 8	TX data 6
1B	-	TX data 7
1C	-	TX data 8

TX_FRAME_INFORMATION (Same for SFF and EFF)								**Offset = 0x0010**
Bit#	7	6	5	4	3	2	1	0
W	FF	RTR			DLC3	DLC2	DLC1	DLC0
Reset	U	U	U	U	U	U	U	U

Table 24.18: Description of TX Frame Information

Bit Number(s)	Bit Name	Reset State	Description
7	FF	U	Frame Format 1: EFF 0: SFF
6	RTR	U	Remote Transmission Request 1: Initiate request 0: Data frame
5-4	RESERVED	UU	
3-0	DLCx	[UU…U]	Data length code bit and should be a value between 0 and 8. If a value greater than 8 is used, 8 bytes will be transmitted

TX IDentifier 1 (SFF)								**Offset = 0x0011**
Bit#	7	6	5	4	3	2	1	0
W	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Reset	U	U	U	U	U	U	U	U

Table 24.19: Description of TX IDentifier 1 (SFF)

Bit Number(s)	Bit Name	Reset State	Description
7-0	IDx	[UU…U]	Top eight bits of the 11 bit SFF identifier

TX IDentifier 2 (SFF)								**Offset = 0x0012**
Bit#	7	6	5	4	3	2	1	0
W	ID20	ID19	ID18					
Reset	U	U	U	[UU…U]				

Table 24.20: Description of TX IDentifier 2 (SFF)

Bit Number(s)	Bit Name	Reset State	Description
7-5	IDx	[UU…U]	Bottom three bits of the 11 bit SFF identifier

TX IDentifier 1 (EFF)								**Offset = 0x0011**
Bit#	7	6	5	4	3	2	1	0
W	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Reset	U	U	U	U	U	U	U	U

Table 24.21: Description of TX IDentifier 1 (EFF)

Bit Number(s)	Bit Name	Reset State	Description
7-0	IDx	[UU…U]	Bits 28-21 of the 29-bit EFF identifier

TX IDentifier 2 (EFF)								Offset = 0x0012
Bit#	7	6	5	4	3	2	1	0
W	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
Reset	U	U	U	U	U	U	U	U

Table 24.22: Description of TX IDentifier 2 (EFF)

Bit Number(s)	Bit Name	Reset State	Description
7-0	IDx	[UU…U]	Bits 20-13 of the 29 bit EFF identifier

TX IDentifier 3 (EFF)								Offset = 0x0013
Bit#	7	6	5	4	3	2	1	0
W	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
Reset	U	U	U	U	U	U	U	U

Table 24.23: Description of TX IDentifier 3 (EFF)

Bit Number(s)	Bit Name	Reset State	Description
7-0	IDx	[UU…U]	Bits 12-5 of the 29 bit EFF identifier

TX IDentifier 4 (EFF)								Offset = 0x0014
Bit#	7	6	5	4	3	2	1	0
W	ID4	ID3	ID2	ID1	ID0			
Reset	U	U	U	U	U	UUU		

Table 24.24: Description of TX IDentifier 4 (EFF) Register

Bit Number(s)	Bit Name	Reset State	Description
7-3	IDx	[UU…U]	Bits 4-0 of the 29 bit EFF identifier

Data Field

The number of transferred data bytes is defined by the data length code (DLC). The first bit transmitted is the most significant bit of data byte 1 at the lowest address. For SFF frames the data field is located at address 0x13 to 0x1A and for EFF frames at 0x15 to 0x1C.

24.4.13 Receive Buffer

Reset Value: 0xUU (unknown random value; a write is required after reset)

Can Address Offset (Hex)	Standard Frame Format	Extended Frame Format
10	RX frame information	RX frame information
11	RX ID 1	RX ID 1
12	RX ID 2	RX ID 2
13	RX data 1	RX ID 3
14	RX data 2	RX ID 4
15	RX data 3	RX data 1
16	RX data 4	RX data 2
17	RX data 5	RX data 3
18	RX data 6	RX data 4
19	RX data 7	RX data 5
1A	RX data 8	RX data 6
1B	RX FI of next message in FIFO	RX data 7
1C	RX ID1 of next message in FIFO	RX data 8

RX_FRAME_INFORMATION (Same for SFF and EFF)								**Offset = 0x0010**
Bit#	7	6	5	4	3	2	1	0
R	FF	RTR			DLC3	DLC2	DLC1	DLC0
Reset	U	U	U	U	U	U	U	U

Table 24.25: Description of RX Frame Information

Bit Number(s)	Bit Name	Reset State	Description
7	FF	U	Frame Format 1: EFF 0: SFF
6	RTR	U	Remote Transmission Request 1: Initiate request 0: Data frame
5-4	RESERVED	UU	Always 00
3-0	DLCx	[UU…U]	Data length code

RX IDentifier 1 (SFF)								**Offset = 0x0011**
Bit#	7	6	5	4	3	2	1	0
R	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Reset	U	U	U	U	U	U	U	U

Table 24.26: Description of RX IDentifier 1 (SFF)

Bit Number(s)	Bit Name	Reset State	Description
7-0	IDx	[UU…U]	Top eight bits of the 11 bit SFF identifier

RX IDentifier 2 (SFF)								**Offset = 0x0012**
Bit#	7	6	5	4	3	2	1	0
R	ID20	ID19	ID18	RTR	0	0	0	0
Reset	U	U	U	U	U	U	U	U

Table 24.27: Description of RX IDentifier 2 (SFF)

Bit Number(s)	Bit Name	Reset State	Description
7-5	IDx	[UU…U]	Bottom three bits of the 11 bit SFF identifier
4	RTR	U	1: RTR frame 0: Data frame
3-0	RESERVED	[UU…U]	Always 0

RX IDentifier 1 (EFF)								**Offset = 0x0011**
Bit#	7	6	5	4	3	2	1	0
R	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Reset	U	U	U	U	U	U	U	U

Table 24.28: Description of RX IDentifier 1 (EFF)

Bit Number(s)	Bit Name	Reset State	Description
7-0	IDx	[UU…U]	Bits 28-21 of the 29 bit EFF identifier

RX IDentifier 2 (EFF)								Offset = 0x0012
Bit#	7	6	5	4	3	2	1	0
R	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
Reset	U	U	U	U	U	U	U	U

Table 24.29: Description of RX IDentifier 2 (EFF)

Bit Number(s)	Bit Name	Reset State	Description
7-0	IDx	[UU…U]	Bits 20-13 of the 29 bit EFF identifier

RX IDentifier 3 (EFF)								Offset = 0x0013
Bit#	7	6	5	4	3	2	1	0
R	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
Reset	U	U	U	U	U	U	U	U

Table 24.30: Description of RX IDentifier 3 (EFF)

Bit Number(s)	Bit Name	Reset State	Description
7-0	IDx	[UU…U]	Bits 12-5 of the 29 bit EFF identifier

RX IDentifier 4 (EFF)								Offset = 0x0014
Bit#	7	6	5	4	3	2	1	0
R	ID4	ID3	ID2	ID1	ID0	RTR		
Reset	U	U	U	U	U	U	UU	

Table 24.31: Description of RX IDentifier 4 (EFF) Register

Bit Number(s)	Bit Name	Reset State	Description
7-3	IDx	[UU…U]	Bits 4-0 of the 29 bit EFF identifier
2	RTR	U	1: RTR Frame 0: Data Frame
1-0	RESERVED	UU	

Data Field

For received SFF frames the data field is located at address 0x13 to 0x1A and for EFF frames at 0x15 to 0x1C.

24.4.14 Acceptance Filter (PELI_CAN_ACCEPT.ACCEPT_CODE_0: ACCEPT_CODE_3, PELI_CAN_ACCEPT.ACCEPT_MASK_0: ACCEPT_MASK_3)

Address Offset: 0x10 – 0x17

The acceptance filter can be used to filter out messages not meeting certain demands. If a message is filtered out it will not be put into the receive FIFO and the CPU will not have to deal with it.

There are two different filtering modes, single and dual filter. Which one is used is controlled by bit 3 in the PELI_CAN.MODE register. In single filter mode only one 4-byte filter is used. In dual filter two smaller filters are used and if either of these signals a match the message is accepted. Each filter consists of two parts the acceptance code and the acceptance mask. The code registers are used for specifying the pattern to match and the mask registers specify don't care bits. In total eight registers are used for the acceptance filter as shown in the table below. Note that they are only read/writable in reset mode.

ADDRESS OFFSET (hex)	Description	Reset Value
10	Acceptance code 0 (ACR0)	0x51
11	Acceptance code 1 (ACR1)	0xUU (1)
12	Acceptance code 2 (ACR2)	0xUU (1)
13	Acceptance code 3 (ACR3)	0xUU (1)
14	Acceptance mask 0 (AMR0)	0xFF
15	Acceptance mask 1 (AMR1)	0xUU (1)
16	Acceptance mask 2 (AMR2)	0xUU (1)
17	Acceptance mask 3 (AMR3)	0xUU (1)

Note 1: unknown random value: a write is required after the reset

Single filter mode, standard frame

When receiving a standard frame in single filter mode the registers ACR0: 3 (only ACR0 is used in BasiCAN mode) are compared against the incoming message in the following way:

- ACR0[7:0] are compared to ID[10:3] or ID[28:21]
- ACR1[7:5] are compared to ID[2:0] or ID[20:18]
- ACR1[4] is compared to the RTR bit
- ACR1[3-0] are unused.
- ACR2 and ACR3 are compared to data byte 0 & 1

The corresponding bits in the PELI_CAN.ACCEPT_MASK_x registers selects if the results of the comparison doesn't matter. A set bit in the mask register means "don't care".

Single filter mode, extended frame

When receiving an extended frame in single filter mode the registers ACR0: 3 are compared against the incoming message in the following way:

- ACR0[7:0] and ACR1[7:0] are compared to ID[28:13]
- ACR2[7:0] and ACR3[7:3] are compared to ID[12:0]
- ACR3[2] is compared to the RTR bit
- ACR3[1:0] are unused

The corresponding bits in the AMRx registers selects if the results of the comparison doesn't matter. A set bit in the mask register means "don't care".

Dual filter mode, standard frame

When receiving a standard frame in dual filter mode the registers ACR0-3 are compared against the incoming message in the following way:

- Filter 1
  - ACR0[7:0] and ACR1[7:5] are compared to ID[28:21]
  - ACR1[4] is compared to the RTR bit
  - ACR1[3:0] are compared against upper nibble of data byte 0
  - ACR3[3:0] are compared against lower nibble of data byte 0
- Filter 2
  - ACR2[7:0] and ACR3[7:5] are compared to ID[28:21]
  - ACR3[4] is compared to the RTR bit.

Dual filter mode, extended frame

When receiving a standard frame in dual filter mode the registers ACR0: 3 are compared against the incoming message in the following way:

- Filter 1
  - ACR0[7:0] and ACR1[7:0] are compared to ID[28:13]
- Filter 2
  - ACR2[7:0] and ACR3[7:0] are compared to ID[28:13]

The corresponding bits in the AMR registers selects if the results of the comparison doesn't matter. A set bit in the mask register means "don't care".

24.4.15 RX Message Counter (RMC)

The RX message counter register holds the number of messages currently stored in the receive FIFO. The top three bits are always 0.

RMC								**Offset = 0x001D**
Bit#	7	6	5	4	3	2	1	0
R	0	0	0	NM4	NM3	NM2	NM1	NM0
Reset	0	0	0	0	0	0	0	0

Table 24.32: Description of _ Register

Bit Number(s)	Bit Name	Reset State	Description
7-5	RESERVED	000	Always 0
4-0	NMx	[00…0]	Number of messages currently in the receive FIFO

24.5 Common Registers

24.5.1 Clock Divider Register (CLOCK_DIVIDER)

This register allows selection to choose between PeliCAN and BasiCAN mode of operation.

CLOCK_DIVIDER								**Offset = 0x001F**
Bit#	7	6	5	4	3	2	1	0
R W	Mode							
Reset	0	[00…0]						

Table 24.33: Description of CAN Clock Divider Register

Bit Number(s)	Bit Name	Reset State	Description
7	Mode	0	CAN Mode selection bit 1: PeliCAN Mode 0: BasiCAN Mode
6-0	RESERVED	[00…0]	

24.5.2 BUS TIMING 0 Register (BUS_TIMING_0)

BUS_TIMING_0								**Offset = 0x0006**
Bit#	7	6	5	4	3	2	1	0
R W	SJW		BRP					
Reset	00		[00…0]					

Table 24.34: Description of CAN Bus Timing 0 Register

Bit Number(s)	Bit Name	Reset State	Description
7-6	SJW	00	Synchronization jump width
5-0	BRP	[00…0]	Baud rate pre-scalar Final pre-scalar value = 2*(BRP+1) x 2

24.5.3 BUS TIMING 1 Register (BUS_TIMING_1)

BUS_TIMING_1								Offset = 0x0007
Bit#	7	6	5	4	3	2	1	0
R	SAM	TSEG[2:0]			TSEG1[3:0]			
W								
Reset	0	000			[00…0]			

Table 24.35: Description of CAN Bus Timing 1 Register

Bit Number(s)	Bit Name	Reset State	Description
7	SAM	0	Sample Mode: 1: Bus is sampled three times 0: Bus is sample once
6-4	TSEG2	000	Time segment 2
3-0	TSEG1	[00…0]	Time segment 1

The CAN bus bit period is determined by the CAN system clock and time segment 1 and 2 as shown in the equations below:

ttseg1= tscl * (TSEG1+1)
ttseg2 = tscl * (TSEG2+1)
tbit = ttseg1 + ttseg2 + tscl

The additional tscl term comes from the initial sync segment. Sampling is done between TSEG1 and TSEG2 in the bit period.

24.6 Function Differences for Design

24.6.1 SJA1000 Functional Differences

- All bits related to sleep mode are unavailable
- Output control and test registers do not exist (reads 0x00)
- Clock divisor register bit 6 (CBP) and 5 (RXINTEN) are not implemented
- Overrun IRQ and status not set until FIFO is read out

24.6.2 BasicCAN Functional Differences

- The receive IRG bit is not reset on read, works like in PeliCAN mode
- Bit CR.6 always reads 0 and is not a flip flop with no effect as in SJA1000

24.6.3 PeliCAN Functional Differences

- Writing 256 to TX Error Counter gives immediate bus-off when still in reset mode
- Read Buffer Start Address register does not exist
- Addresses above 31 are not implemented (i.e. the internal RAM/FIFO access)
- The core transmits active error frames in listen only mode

Appendix A: Errata

ADC Continuous Mode Wrap Around Hardware Inaccurate Delay

In Continuous mode, the ADC performs continuous conversions sequentially on all enabled channels. From one channel to the next, the hardware has a predefined delay for setup time that allows for an accurate ADC conversion. Once the last channel finishes an ADC conversion, the hardware wraps around to the first enabled channel in the sequence to begin a new conversion. When the hardware wraps around from last to first channel in the sequence, **the predefined delay for setup time is reduced and causes an inaccurate ADC conversion reading in the first enabled channel.**

Note: the “first channel” is not physical channel #1, but the first enabled channel in the sequence, i.e., channel #2 in enabled channel 2-3, channel #5 in enabled channel 5-10, channel #1 in enabled channel 1-15 or any other sequence combination.

Workarounds

1. Run the ADC in Single Sweep mode, and add a software delay of 20us minimum between each sweep completion (CONV_COMPL_COMB and INTR_PEND bit of highest enabled channel) and the triggering of the next sweep
2. In continuous mode, configure the first enabled channel as a "dummy" channel, and only keep data from the second enabled channel and beyond.

ADC External vs Internal Oscillator CONV_COMPL Conflict

When the UT32M0R500 uses an external clock as its system clock (PCLK), the ADC peripheral still uses the internal 50MHz oscillator to perform conversions (at a clock divided rate determined by the ADC_OSCDIV[1:0] bits in the ADC.TIM_CTRL register). When using an external system clock, the ADC_CONV_COMPL bit, which is supposed to set at the end of a conversion, will sometimes not set. The DATA_VALUE_STALE bit located in the ADC->DATA register, bit 15, used the ADC_CONV_COMPL bit to determine if data was stale. Both bits have been marked as having this chance for error. This is due to a hardware error that occurs a small fraction of the time caused by using two different clocks. When using the internal 50MHz oscillator as the system clock (PCLK), this is not an issue.

Workarounds

Bits 31 and [24:0] found in ADC->INT_STATUS are still accurate and can be used to determine which channels have completed a conversion. Once an enabled channel has been sampled, its respective INTR_PEND bit will set along with the CONV_COMPL_COMB bit. Once the INTR_PEND bit for the highest enabled channel is set along with the CONV_COMPL_COMB, the full ADC sweep is complete. This method does not require users to enable ADC interrupts, regardless of the bit name.

There is no workaround for the DATA_VALUE_STALE bit.

ADC COI3_OVER Low Voltage False Flag

The COI3_OVER flag signals to the user when the input voltage from the most recent measurement is out of range. Although it is called COI3_OVER, the flag signals both over-voltage and under-voltage measurements. When the input into a Single-Ended channel is at or below 0.003V (3mV), the COI3_OVER flag has a chance to set. This chance increases as the input voltage approaches 0V, to the point where the COI3_OVER flag is always set. This false flag only occurs when all of the data bits in the respective channel’s ADC.DATA register are equal to zero (both the DATA_OUT[11:0] bits and the DATA_OUT_LSB[3:0] bits. If any of these data bits are set in conjunction with the COI3_OVER flag, the flag is operating as intended.

Note that the 3mV value was measured using a high-precision voltage supply and noise-reducing practices. Noise on the voltage input will cause the COI3_OVER false flags to occur at a higher voltage threshold.

Workarounds

If this flag is set and the DATA_OUT and DATA_OUT_LSB bits of the ADC.DATA register are zero, users can ignore the COI3_OVER flag.

ADC Single-Ended Even/Odd Channel Offset Error

When ADC channels are configured as single-ended inputs, even channels (ADC_SE_CHAN_[0,2,4,6,8,10,12,14]) and odd channels (ADC_SE_CHAN_[1,3,5,7,9,11,13,15]) have an offset error that becomes noticeable when the channels are measuring the same signal, but is always there in single-ended mode. The offset error affects the singe-ended channels by the same magnitude, but opposite polarity. Even channel measurements are increased by the offset, odd channel measurements are decreased by the offset. Even channel codes are typically 20 counts higher than the odd channel codes (~7.3mV with a gain of 1V/V). The offset is uniform when measured on a single part but may vary between parts.

When a channel is configured as a differential input, the offset cancels itself out, and has not been observed on any measurements or characterization data.

Workarounds

Users that intend to use ADC channels as Single-Ended inputs should characterize the even/odd offset by comparing the output codes of an even and odd channel measuring the same voltage, and then addressing that offset in their software. For additional accuracy, users can screen individual parts to find a more accurate part-specific offset for their software.

I2C RD_REQ Interrupt Re-Asserts Itself After a Slave Address Register Change

Some I2C systems will require the I2C slaves to change or modify their slave address. The UT32M0R500 does this by changing the Slave Address Register (I2C.SAR). To change a peripherals' SAR, first clear the I2C.ENABLE register's Enable bit (bit 0). Then write the new Slave Address to the I2C.SAR register. Finally, set the I2C.ENABLE Enable bit. When an I2C master wants to read data from an I2C slave, the RD_REQ interrupt will set when the I2C slave detects an I2C read condition that targets its current SAR. Directly after this interrupt signal is cleared and the read transaction completed, if the slave device attempts to change its Slave Address, upon re-enabling the I2C peripheral the RD_REQ interrupt will set itself, even though there is no such transaction on the I2C bus for the new Slave Address Register value.

Workarounds

To avoid this flag from setting, the most reliable option is to implement a small delay before changing the SAR register after a RD_REQ flag was set.

CAN RX Message Counter to Receiver Interrupt One Clock Cycle Delay

This errata is relevant to users that have enabled the CAN Receiver Interrupt and one or more other interrupt signals during normal operation. There is a single clock cycle where reading the INTERRUPT register will indicate the Receiver Interrupt is not set when the RX FIFO has a frame ready to read. This could result in user code not servicing the Receiver Interrupt, which would result in the CAN NVIC vector not changing to pending.

The CAN 2.0 peripheral interrupts are combined into a single signal from the peripheral to the Nested Vector Interrupt Controller (NVIC). The combined interrupt signal will only send a pulse from the CAN to the NVIC when the value of the INTERRUPT register changes from 0x00 to a valid nonzero value, and there is no level-based functionality. Excluding the Receiver Interrupt, a set interrupt can be cleared by reading the INTERRUPT register. The Receiver Interrupt is only cleared by setting the Release Receive Buffer bit in the COMMAND register.

When a valid CAN frame is received by UT32M0R500, it will update the value of the RX Message Counter register. If the Receiver Interrupt is enabled and not already set, it will always change from '0' to '1' exactly one clock cycle after the RX Message Counter value does. This means there is a single clock cycle where reading the INTERRUPT register would falsely indicate there is not any data in the RX Message FIFO.

This creates a possible condition where reading the INTERRUPT register can happen as the RX Message Counter is being updated, which is right before the Receiver Interrupt (RI) bit gets set. In this single clock cycle, reading the INTERRUPT register would report that the RI bit isn't set, but reading the RX Message Counter would show a nonzero value. In this condition, user code would believe there is not a Receive Interrupt, so it doesn't need to read from the RX FIFO. When the RI bit sets a clock cycle later the combined interrupt signal appears to not have changed, and no new interrupt pulse is generated from the CAN to the NVIC.

There are a few ways to recover from this state, and start seeing interrupts in the NVIC again. User code written to detect when the NVIC CANx vector stops pending, but the CAN INTERRUPT register still has the RI bit set can attempt to recover by:

- Using the ARM NVIC_SetPending() function to manually set the CANx vector to pending. This will result in the processor entering the CAN ISR. Inside the CAN ISR, reading the INTERRUPT register would clear any enabled interrupts (except RI), and then the RX FIFO can be read from and cleared.
- Reading from the RX FIFO outside of the CAN ISR function. The RX FIFO read function should set the RRB bit at the end in order to clear the FIFO. This option only works if the other bits in the INTERRUPT register are already cleared, but user code could ensure that is the case prior to the RX FIFO read.
- Disabling and re-enabling the CAN peripheral. This action resets the CAN peripheral, and would be destructive to the data in the CAN FIFO.

For the above condition to occur, the Receiver Interrupt needs to be enabled, the RX Message Counter empty, and one other CAN interrupt signal needs to be set (which means the NVIC CANx vector is pending). The below six signals are capable of causing an interrupt pulse and thus need to be considered.

Table 36: CAN Interrupt Bits Capable of Causing No-Interrupt Errata

Bit	Name	Condition
7	Bus Error Interrupt (BEI)	When the CAN peripheral detects a BUS Error, devices detecting the error simultaneously would send out 6 dominant bits, plus at least 8 bits of Error Delimiter. Following those 14 bit times, the RI signal will not set until a complete and valid CAN frame is received. If the UT32M0R500 can service the CAN ISR before a valid CAN frame can arrive, there is no risk to normal operation. If the ISR is not serviced before a complete frame can arrive, there is a very small probability that the UT32M0R500 reads the INTERRUPT register in the exact clock the RX Message Counter changes from 0 to 1.
6	Arbitration Lost Interrupt (ALI)	When arbitration is lost, the CAN Controller becomes a receiver. If the message is valid, reading the message that lost the arbitration or subsequent valid messages has a small probability of occurring at the same time the INTERRUPT register is read.
5	Error Passive Interrupt (EPI)	The Error Passive Interrupt occurs when either Error Counter exceeds 127. If this occurred due to transmitting, the RMC register would not increment, so the transmitted frame could not cause the condition. If the error occurred due to receiving, the frame would not be accepted because of the error. After a frame causes the Error Passive Interrupt, if the UT32M0R500 can service the interrupt (or at minimum read the INTERRUPT register before the next valid frame is received there is no chance of the condition. If the UT32M0R500 does not read the INTERRUPT register before another valid frame arrives, there is a small probability a received frame increments the RX Message Counter from 0->1 at the same time as an INTERRUPT read.
3	Data Overrun Interrupt (DOI)	Data Overrun occurs when there is not enough space in the RX FIFO. For this to occur, the RX Message Counter has to be a non-zero value, and the RI bit would already be set when the INTERRUPT register is read.
2	Error Warning Interrupt (EI)	The device can still receive valid frames when there is an Error Warning Interrupt, so there is a low probability a valid frame arrives when the INTERRUPT register is read.
1	Transmit Interrupt (TI)	After the UT32M0R500 successfully transmits a frame, it has the period of before another frame arriving to read INTERRUPT. After, there is a small probability that the UT32M0R500 reads the INTERRUPT register in the exact clock the RX Message Counter changes from 0 to 1.

Workarounds

This workaround keeps the RI bit enabled – which is needed to generate interrupts when frames are received – but doesn't make decisions based upon the status of the RI bit inside the ISR. In the ISR, the INTERRUPT register is still read at the start as normal. However, instead of using the RI bit from the INTERUPT read to determine if a message should be read, users should read the STATUS Receive Buffer Status (RBS) bit.

If the low probability event happens when the INTERRUPT register is read, the RBS bit will correctly reflect that there is data that needs to be read, and a proper read will set the Release Receive Buffer bit. If the low probability event happens when the STATUS register is read, the first INTERRUPT read will already have cleared the INTERRUPT register for at least one clock cycle, which will result in a pulse to the NVIC, making the vector pending and active, and a second run of the ISR will occur to address the RI bit.

Pseudocode:

```
CANx_IRQHandler(){
      uint8 Interrupt;
      Interrupt = CANx->INTERRUPT;
      //user logic responding to all interrupt signals other than RI
      //end response logic
      if( (CANx->STATUS & CAN_BC_STATUS_RECEIVE_MSG_READY) != 0){
            CANx_ReceiveData();
      }
}
```

Revision History

Date	Rev. #	Author	Change Description	Page #
30/5/2018	1.0.0		Initial Release	
8/22/2019	1.1.0	JA	Edits to Chapter 1: Introduction Added Paragraph to the Scope to introduce Document Structure Added the Related Resources section with links to useful resources Moved UT32M0R500 Configuration Table from the ARM Cortex M0+ Processor Chapters' Overview section to this chapters' Features section Modified Generic Register Format to improve clarity Edits to Chapter 2: ARM Cortex M0+ Processor Added additional information about processor architecture and components to the Overview Added the Processor Core, 2-Stage Pipeline, Wake-Up Interrupt Controller, and Bus Interface sections NVIC now has its own Chapter Flushed out the Debug System section Added Chapter 3: ARM Cortex M0+ Registers Added Chapter 4: ARM Cortex M0+ Instruction Set Moved ARM Cortex M0+ Memory Map to Chapter 5 (previously Ch. 3) Split Memory information into the Overview, Boot ROM, and Embedded SRAM sections, added additional information Added Chapter 6: Nested Vector Interrupt Controller (NVIC) Added the Interrupts, Reset Vector, Non Maskable Interrupt (NMI), HardFault, SVCall (SuperVisor Call), PendSV (Pendable Service Call), SysTick, External Interrupts, CMSIS Functions for NVIC Registers, and Wake-Up Interrupt Controller sections, detailing more information about the NVIC operation, priority levels, interrupts, ect. Added Chapter 7: Power Management Moved System Controller (SYSCON) to Chapter 8 (previously Ch. 5) Keep Alive Functionality: Now only lists the DAC and GPIO functionality. The internal general purpose registers are still listed in the General Purpose Register 0-2 sections Added Chapter 9: SRAM controller with EDAC and Scrubbing Added Scrub Rate Calculation Information CONTROL register: Removed bit 1, MBEA_HRESP_EN – simulation bit only Added Chapter 10: NOR Flash Controller (NFC) CONTROL register: Added reset values for bits 11-8 STATUS register: Removed Table 14.1 – Information maintained in UT8QNF8M8 datasheet PERIPH_ID2 register: Bits [7:0] now say PERIPH_ID2 instead of PERIPH_ID1 Added Chapter 11: ARM Keil µVision Tools Moved Boot Configuration Chapter 12 (previously Ch. 4) Moved GPIO to Chapter 13 (previously Ch. 8) Overview section: Re-wrote overview to better describe GPIO configuration Added the Functional Description section Operation section: Moved AHB Addresses to the Registers Section External Interrupt Generation Section: Re-wrote subsection to better describe interrupt functionality Soft Reset Mode Section: Re-wrote section with additional details Added the Input Configuration and Output Configuration sections DATA register: Gave bits [15:0] a name DATAOUT register: Gave bits [15:0] a name ALTFUNCSET register: Changed description of read value ALTFUNCCLR register: Changed description of read value INTENSET register: Changed description to make sense	

Date	Rev. #	Author	Change Description	Page #
			INTENCLR register: Changed description to make sense LB_MASKED register: Removed upper byte that couldn't be read by this register UB_MASKED register: Removed lower byte that couldn't be read by this register Moved ADC to Chapter 14 (previously Ch. 9) SPB_CFG_0 register: Combined bits 15 and 14 to become OBD[1:0] SPB_CFG_1 register: Added Note 1, describing errata found in Appendix A TEMPCHAN_CFG register: Reserved bits [9:8], only bit 0 should be used to enable the temperature channel DATA register: Added Note 1 Moved DAC to Chapter 15 (previously Ch. 10) Moved Comparators to Chapter 16 (previously Ch.15) Moved Timers to Chapter 17 (previously Ch.18) Overview section: Improved the timer explanation and added a block diagram Added the Functional Description and Operation sections Timer Register Detail section: Non-Register information moved to the Operation section TIMERxMIS register: Re-named bit 0 to properly reflect being a mask of the interrupt bit Moved PWM to Chapter 18 (previously Ch.17) Overview Section: Moved information into the Functional Description and Operation sections, updated the Block Diagram Added the Functional Description section Operation section: Added information previously in the Overview Prescaler section: Re-worded clock scaling information Single Output Signal section: Added detailed information about single output signals from the Asymmetric and Symmetric PWM Generation section Paried Output Signal section: Added detailed information about paired output signals from the Asymmetric and Symmetric PWM Generation section Dead Band Time section: Made this section easier to understand Interrupt section: Made this section easier to understand Moved Watchdog Timer to Chapter 19 (previously Ch.11) INT_CLR register: Added description to bit 0 MASK_INT_STAT register: Added description to bit 0 Moved Real Time Clock to Chapter 20 (previously Ch.19) Added the Overview, Functional Description, and Operation sections Moved UART to Chapter 21 (previously Ch.13) Overview section: Added new information Protocol section: Added new information Added the Character-Encoding Scheme section Operation Section: Transmitter and receiver operation subsections were the same, now are specific to tx or rx operation Moved SPI to Chapter 22 (previously Ch.12) Overview section: Re-worded section Added the Clock Generation and Operation sections Transmit and Receive FIFO Buffers section: Moved some information into the FIFO Empty/Full/Overrun Interrupt sections Added the External Interrupt, FIFO Empty Interrupt, FIFO Full Interrupt, and FIFO Overrun Interrupt sections Data Transfers section: Removed Motorola Serial Peripheral Interface (SPI) section, information now covered in the Clocking Modes subsection Moved I2C to Chapter 23 (previously Ch.16) Overview Section: Re-worded section I2C Protocol Section: Added the Master to Slave Transmission, Slave to Master	

Date	Rev. #	Author	Change Description	Page #
			Transmission, and Master to Multiple Slaves subsections, Removed START BYTE Transfer Protocol Added the Synchronization, External Interrupt, FIFO Empty Interrupt, FIFO Full Interrupt, and FIFO Overrun Interrupt sections CLR_START_DET register: Fixed bit name ENABLE register: Re-named bit to properly reflect UT32M0R500.h header file RXFLR register: Fixed bit 2 reset value SDA_SETUP register: Removed redundant information Moved CAN to Chapter 24 (previously Ch.6) PELI_CAN_ACCEPT.ACCEPT_CODE_0:ACCEPT_CODE_3 and PELI_CAN_ACCEPT.ACCEPT_MASK_0:ACCEPT_MASK_3 registers: Removed Single Filter Mode Extended Frame ASCII Graphic, above bullet list has same information RMC register: Added bit names to bits [4:0] Added Appendix A: Errata Every chapter with register tables are replaced by the new register tables	
3/4/2021	1.2.0	OW	Updated Template Clarified Revision History 1.1.0 Description to better detail changes made between 1.0.0 and 1.1.0 New Information (by Section): ADC_SEQDLY Requirements section: Made the requirements for setting ADC_SEQDLY clearer Added the Calculating Channel Conversion Time and Conversion Error Flags and Their Meanings to the ADC Chapter ADC SPB_CFG_1 register: Reserved the CONV_COMPL bit, due to new errata; See Errata for workaround UART STATUS register: Added maximum FIFO size to table description I2C DATA_CMD register: Added two bits, RESTART and STOP for additional user control Appendix A: ADC External vs Internal Oscillator CONV_COMPL Conflict ADC COI3_OVER Low Voltage False Flag ADC Even/Odd Offset Error I2C RD_REQ Interrupt Re-Asserts Itself After a Slave Address Register Change Fixed Errors (by Section): Various: Updated Inter-Document References NVIC Interrupts section: NFC Vector added to interrupt table NFC CONTROL register: Un-highlighted bits GPIO ALTFUNCSET and ALTFUNCCLR reset values ADC TEMPCHAN_CFG register: Fixed Table Formatting ADC STAB_CTRL register: Bit 16 fixed to be R only, not R/W ADC DATA register: Bit 19 now R only, not R/W Bit 15 now correctly RESERVED Fixed table formatting PWM CAPREG2 register: Added abbreviated bit names PWM PWM_CTRL_X register: Bit 6 now R only, not R/W Added abbreviated bit names WDT CONTROL register: Fixed table WDT RAW_INT_STAT register: Fixed Bit Name Discrepancy SPI SR register: Fixed reset values I2C Register Details: Removed all registers marked as “RESERVED” in the Register table (functionality not supported) I2C CLR_STOP_DET register: Updated bit name	

Date	Rev. #	Author	Change Description	Page #
			I2C CLR_START_DET register: Fixed bit abbreviation I2C ACK_GENERAL_CALL register: Added bit 0 name back in BasiCAN CONTROL register: Fixed bits [7:5] reset value in table PeliCAN Receive Buffer register: Reserve bits [5:4], updated register description Updated Errata: ADC Continuous Mode Wrap Around Hardware Inaccurate Delay Workarounds to match new errata	
1/19/2023	1.3.0	OW	New Information (by Section) • OSC_SHUTDOWN register now lists all possible external clock configuration options, added the External Clock Configuration table • Appendix A: Errata new section for the CAN RX Message Counter to Receiver Interrupt One Clock Delay errata Fixed Errors (by Section): • Table 5.2 APB Peripheral Memory Map Table now shows Trim Control as RESERVED • NFC.PERIPH_ID[15:0] registers are now RESERVED • ADC DIFFCHAN_CFG[7:0] now correctly describes AIN pin relation to DIFF channel (was counting from 1 to 16, now counting from 0 to 15) • RTC CMR, CLR, and CCR are now listed as R/W, not just R • UART FIFO_DEBUG bit name corrected • CAN Table 24.1: BasiCAN Registers fixed offset 0x1D name • CAN PELICAN Mode of Operation fixed typo CAN.BUS_TIMING_0 clarified that the final value of the Sync Jump Width is equal to SJW+1, and that SJW must not be set greater than TSEG2 or TSEG1, whichever is lower	
6/2/2025	1.3.1	JA	Updated Table 5.1 0x4002 F000 - 0x4002 FFFF for System Controller	