MAX32660 USER GUIDE

UG6659; Rev 0; 7/18

Abstract: This user guide provides application developers information on how to use the memory and peripherals of the MAX32660 microcontroller. Detailed information for all registers and fields in the device are covered. Guidance is given for managing the clocks, power and initialization of the product.

Table of Contents

List of Figures

List of Tables

1 Introduction

For ordering information, mechanical and electrical characteristics for the MAX32660 family of devices please refer to the datasheet. For information on the Arm® Cortex®-M4 with FPU core, please refer to the *Cortex*-M4 with FPU *Technical Reference Manual*.

1.1 Related Documentation

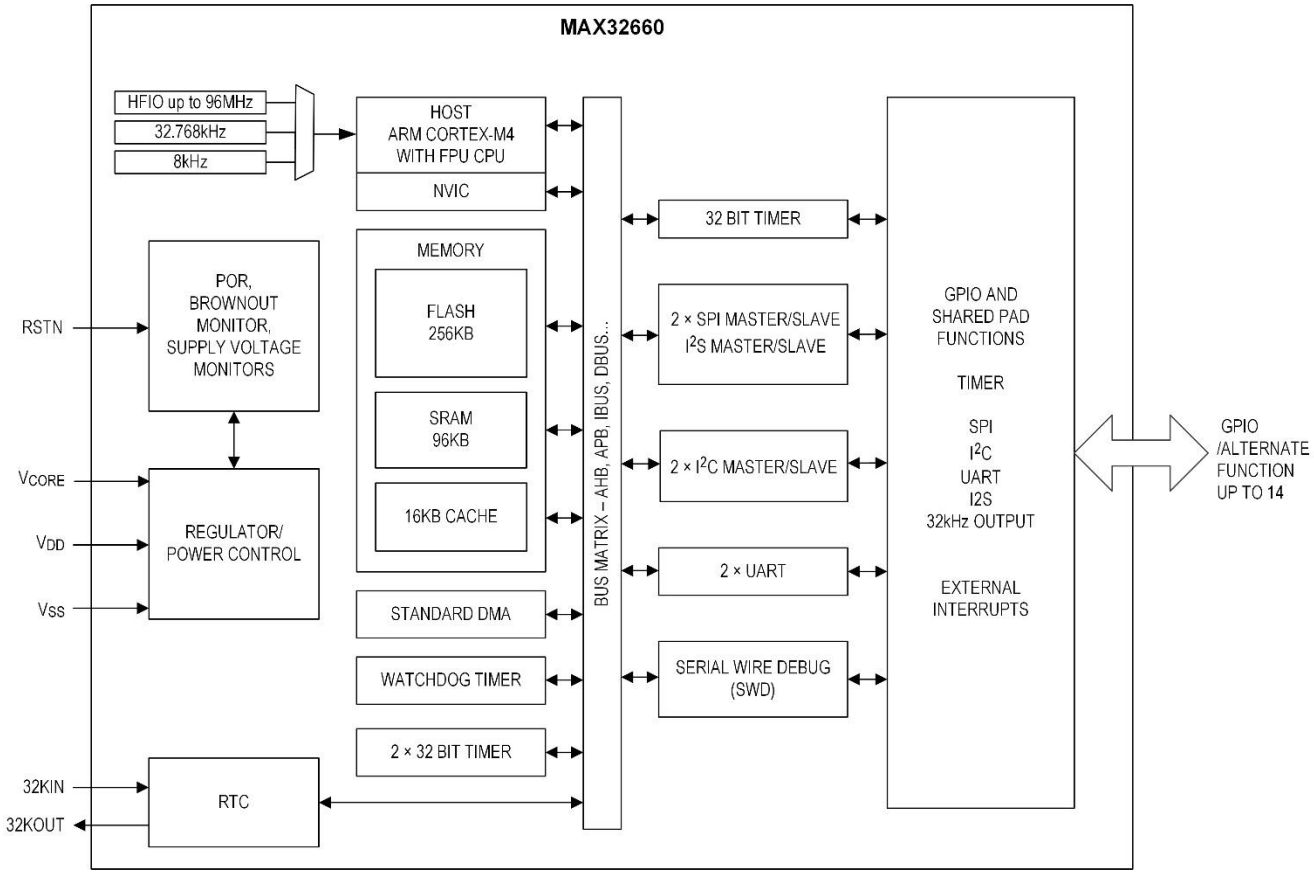
The MAX32660 datasheet and errata are available from the Maxim Integrated website, http://www.maximintegrated.com.

2 Overview

The MAX32660 is an ultra-low power, cost effective highly integrated microcontroller designed for battery-powered devices and wireless sensors. It combines a flexible and versatile power management unit with the powerful Arm Cortex-M4 with FPU (Floating Point Unit). The device enables designs with complex sensor processing without compromising battery life. It also offers legacy designs an easy and cost optimal upgrade path from 8 or 16-bit microcontrollers. The device integrates up to 256KB of flash memory and 96KB of SRAM to accommodate application and sensor code.

The device features four powerful and flexible power modes. It can operate from a single supply battery voltage, or a dual supply typically provided by a PMIC. The I^2C (Inter-Integrated Circuit) port supports standard, fast, fast-plus and high-speed modes operating up to 3400Kbps. The SPI ports can run up to 48MHz in both master and slave mode, and the UARTs can run up to 4Mbps. Three general-purpose 32-bit timers, a watchdog timer, and a real-time clock are also provided. An I^2S (Inter-IC Sound) interface provides audio streaming to or from an external audio codec.

Figure 2-1: MAX32660 High Level Block Diagram



3 Memory, Register Mapping, and Access

3.1 Overview

The Arm Cortex-M4 with FPU core defines a standard memory space for unified code and data access. This memory space is addressed in units of single bytes but is most typically accessed in 32-bit (4 byte) units. It may also be accessed, depending on the implementation, in 8-bit (1 byte) or 16-bit (2 byte) widths. The total range of the memory space is 32-bits in width (4GB addressable total), from addresses 0x0000 0000 to 0xFFFF FFFF.

It is important to note, however, that the architectural definition does not require the entire 4GB memory range to be populated with addressable memory instances.

Figure 3-1: Code Memory Mapping

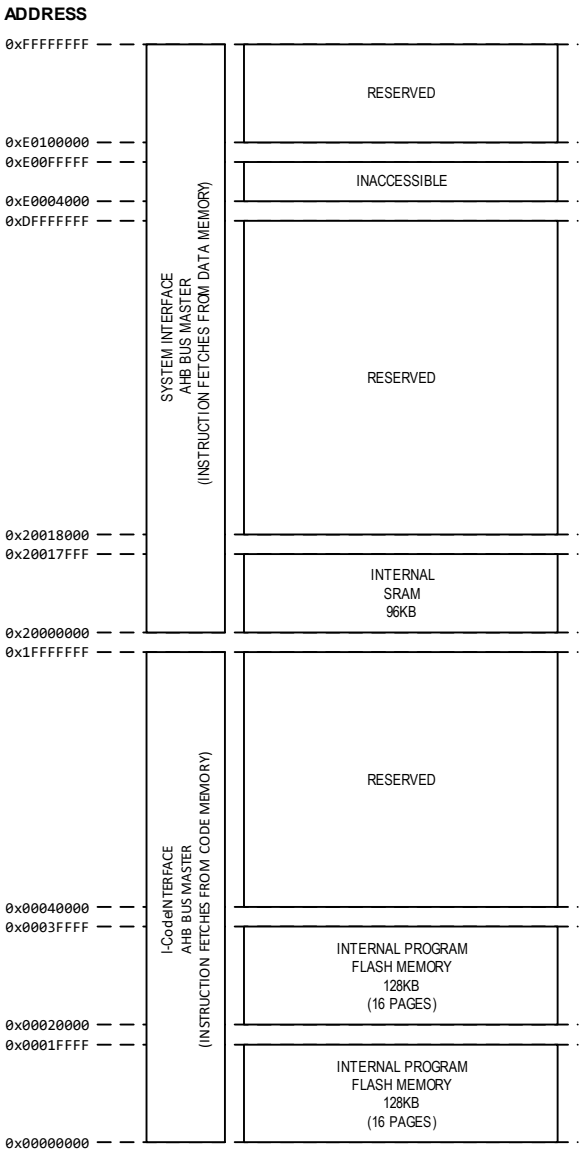
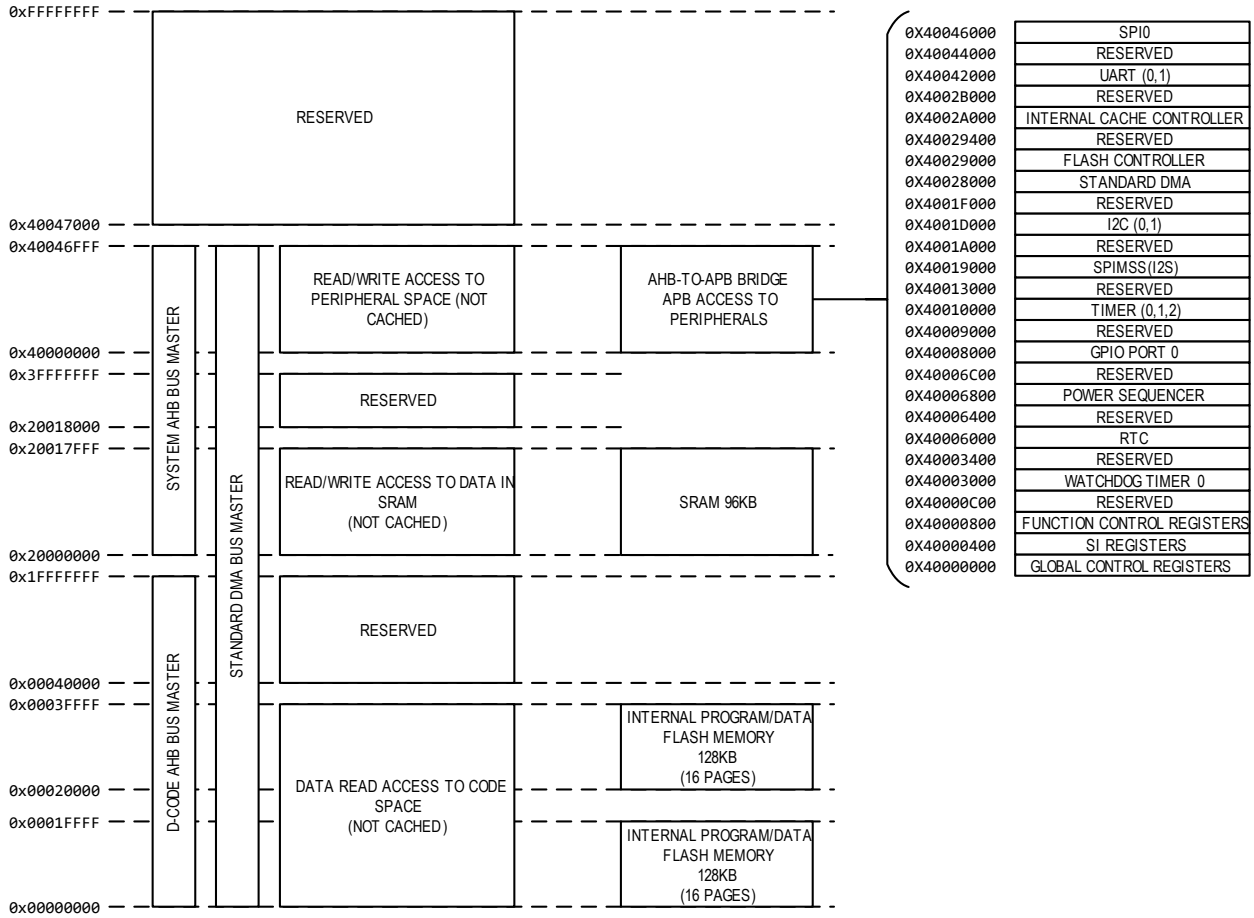


Figure 3-2: Data Memory Map



3.2 Standard Memory Regions

Many standard memory regions are defined for the Arm Cortex-M4 architecture; the use of many of these is optional for the system integrator. At a minimum, the MAX32660 must contain code and data memory for application code, stack and heap usage, as well as certain components which are part of the implemented architecture.

3.2.1 Code Space

The code space area of memory is designed to contain the primary memory used for code execution by the device. This memory area is defined from byte address range 0x0000 0000 to 0x1FFF FFFF (0.5GB maximum). Two different standard core bus masters are used by the Arm Cortex-M4 core and the Arm debugger to access this memory area. The I-Code AHB bus master is used for instruction decode fetching from code memory, while the D-Code AHB bus master is used for data fetches from code memory. This is arranged so that data fetches avoid interfering with instruction execution.

On the MAX32660, the code space memory area contains the main internal flash memory that typically contains the instruction code that will be executed on the device. The internal flash memory is mapped into both code and data space from 0x0000 0000 to 0x0003 FFFF. This program memory area must also contain the default system vector table and the initial settings for all system exception handlers and interrupt handlers. The reset vector for the device is 0x0000 0000.

3.2.2 SRAM Space

The SRAM area of memory is intended to contain the primary SRAM data memory of the device and is defined from byte address range 0x2000 0000 to 0x3FFF FFFF (0.5GB maximum). This memory can be used for general purpose variable and data storage, code execution, and the Arm Cortex-M4 stack.

On the MAX32660, this memory area contains the main system SRAM 96KB, which is mapped from 0x2000 0000 to 0x2001 7FFF.

The entirety of the SRAM memory space on the MAX32660 is contained within the dedicated Arm Cortex-M4 with FPU's SRAM bit-banding region from 0x2000 0000 to 0x200F FFFF (1MB maximum for bit-banding). This means that the CPU can access the entire SRAM either using standard byte/word/doubleword access or using bit-banding operations. The bit-banding mechanism allows any single bit of any given SRAM byte address location to be set, cleared, or read individually by reading from or writing to a corresponding doubleword (32-bit wide) location in the bit-banding alias area.

The alias area for the SRAM bit-banding is located beginning at 0x2200 0000 and is a total of 32MB maximum, which allows the entire 1MB bit banding area to be accessed. Each 32-bit (4 byte aligned) address location in the bit-banding alias area translates into a single bit access (read or write) in the bit-banding primary area. Reading from the location performs a single bit read, while writing either a 1 or 0 to the location performs a single bit set or clear.

Note: The Arm Cortex-M4 with FPU translates the access in the bit-banding alias area into the appropriate read cycle (for a single bit read) or a read-modify-write cycle (for a single bit set or clear) of the bit-banding primary area. This means that bit-banding is a core function (i.e., not a function of the SRAM memory interface layer or the AHB bus layer), and thus is only applicable to accesses generated by the core itself. Reads/writes to the bit-banding alias area by other (non-Arm -core) bus masters such as the Standard DMA AHB bus master will not trigger a bit-banding operation and will instead result in an AHB bus error.

The SRAM area on the MAX32660 is capable code execution. Code stored in the SRAM is accessed directly for execution using the system bus and is not cached. The SRAM is where the Arm Cortex-M4 stack must be located, as it is the only general-purpose SRAM memory on the device. A valid stack location inside the SRAM must be set in the vector table stored at address 0x0000 0000 in the internal flash memory. Refer to the *Cortex-M4 with FPU Technical Reference Manual*. The MAX32660 specific AHB Bus Masters can also access the SRAM to use as general storage or working space.

3.2.3 Peripheral Space

The peripheral space area of memory is intended for mapping of control registers, internal buffers/working space, and other features needed for the firmware control of non-core peripherals. It is defined from byte address range 0x4000 0000 to 0x5FFF FFFF (0.5GB maximum). On the MAX32660, all device-specific module registers are mapped to this memory area, as well as any local memory buffers or FIFOs which are required by modules.

As with the SRAM region, there is a dedicated 1MB area at the bottom of this memory region (from 0x4000 0000 to 0x400F FFFF) that is used for bit-banding operations by the Arm Cortex-M4 with FPU. Four-byte-aligned read/write operations in the peripheral bit-banding alias area (32MB in length, from 0x4200 0000 to 0x43FF FFFF) are translated by the core into read/mask/shift or read/modify/write operation sequences to the appropriate byte location in the bit-banding area.

Note: The bit-banding operation within peripheral memory space is, like bit-banding function in SRAM space, a core remapping function. As such, it is only applicable to operations performed directly by the Arm Cortex-M4 core. If another memory bus master (such as the Standard DMA AHB master) accesses the peripheral bit-banding alias region, the bit-banding remapping operation will not take place. In this case, the bit-banding alias region will appear to be a non-implemented memory area (causing an AHB bus error).

On the MAX32660, access to the region that contains most peripheral registers (0x4000 0000 to 0x400F FFFF) goes from the AHB bus through an AHB-to-APB bridge. This allows the peripheral modules to operate on the slower, easier to handle APB bus matrix. This also ensures that peripherals with slower response times do not tie up bandwidth on the AHB bus, which must necessarily have a faster response time since it handles main application instruction and data fetching.

Note: The APB bus supports 32-bit width access only. All access to the APB peripheral register area (0x4000 0000 to 0x400F FFFF) must be 32-bit width only with 32-bit (4 byte) alignment. Access using 8-bit or 16-bit width to this memory region is not supported and will result in an AHB memory fault exception (returned by the AHB-to-APB bridge interface).

3.2.4 System Area (Private Peripheral Bus)

The system area (private peripheral bus) memory space contains register areas for functions that are only accessible by the Arm Cortex-M4 core itself (and the Arm debugger, in certain instances). It is defined from byte address range 0xE000 0000 to 0xE00F FFFF. This APB bus is restricted and can only be accessed by the Arm core and core-internal functions. It cannot be accessed by other modules which implement AHB memory masters, such as the Standard DMA.

In addition to being restricted to the core, application code is only allowed to access this area when running in the privileged execution mode (as opposed to the standard user thread execution mode). This helps ensure that critical system settings controlled in this area are not altered inadvertently or by errant code that should not have access to this area.

Core functions controlled by registers mapped to this area include the SysTick timer, debug and tracing functions, the NVIC (interrupt handler) controller, and the Flash Breakpoint controller.

3.3 Device Memory Instances

This section details physical memory instances on the MAX32660 (including internal flash memory and SRAM instances) that are accessible as standalone memory regions using either the AHB or APB bus matrix. Memory areas which are only accessible via FIFO interfaces, or memory areas consisting of only a few registers for a peripheral, are not covered here.

3.3.1 Main Program Flash Memory

The main program flash memory is 256KB in size and consists of 32 logical pages of 8KB each.

3.3.2 Instruction Cache Memory

The internal flash memory instruction cache is 16KB in size and is used to cache instructions fetched using the I-Code bus. This includes instructions fetched from the internal flash memory. Note that the cache is used for instruction fetches only. Data fetches (including code literal values) from the internal flash memory do not use the instruction cache.

3.3.3 System SRAM

The system SRAM is 96KB in size and can be used for general purpose data storage, the Arm system stack, USB data transfers (endpoints), and Standard DMA operations, as well as code execution if desired.

3.3.4 AHB Bus Matrix and AHB Bus Interfaces

This section details memory accessibility on the AHB bus matrix and the organization of AHB master and slave instances.

3.3.5 Core AHB Interface

3.3.5.1 I-Code

This AHB master is used by the Arm core for instruction fetching from memory instances located in code space from byte addresses 0x0000 0000 to 0x1FFF FFFF. This bus master is used to fetch instructions from the internal flash memory. Instructions fetched by this bus master are returned by the instruction cache, which in turn triggers a cache line fill cycle to fetch instructions from the internal flash memory when a cache miss occurs.

3.3.5.2 D-Code

This AHB master is used by the Arm Cortex-M4 with FPU core for data fetches from memory instances located in code space from byte addresses 0x0000 0000 to 0x1FFF FFFF. The D-Code AHB master has access to the full internal flash memory.

3.3.5.3 System

This AHB master is used by the Arm core for all instruction fetches and data read and write operations involving the SRAM. The APB mapped peripherals (through the AHB-to-APB bridge) and AHB mapped peripheral and memory areas are also accessed using this bus master.

3.3.6 AHB Master

3.3.6.1 Standard DMA

The Standard DMA bus master has access to all off-core memory areas accessible by the System bus. It does not have access to the Arm Private Peripheral Bus area.

3.4 Peripheral Register Map

Table 3-1, *below*, contains the base address for each of the APB mapped peripherals. The base address for a given peripheral is the start of the register map for the peripheral. For a given peripheral, the address for a register within the peripheral is defined as the peripheral base address plus the register's offset.

Table 3-1: APB Peripheral Base Address Map

Peripheral	Peripheral Register Prefix	Base Address	End Address
Global Control	GCR_	0x4000 0000	0x4000 03FF
System Interface	SIR_	0x4000 0400	0x4000 07FF
Function Control	FCR_	0x4000 0800	0x4000 0BFF
Watchdog Timer 0	WDT0_	0x4000 3000	0x4000 33FF
Real-Time Clock	RTC_	0x4000 6000	0x4000 63FF
Power Sequencer	PWRSEQ_	0x4000 6800	0x4000 6BFF
GPIO Port 0	GPIO0_	0x4000 8000	0x4000 8FFF
Timer 0	TMR0_	0x4001 0000	0x4001 0FFF
Timer 1	TMR1_	0x4001 1000	0x4001 1FFF
Timer 2	TMR2_	0x4001 2000	0x4001 2FFF
SPIMSS	SPIMSS_	0x4001 9000	0x4001 9FFF

Peripheral	Peripheral Register Prefix	Base Address	End Address
I^2C 0	I2C0_	0x4001 D000	0x4001 DFFF
I^2C 1	I2C1_	0x4001 E000	0x4001 EFFF
Standard DMA	DMA_	0x4002 8000	0x4002 8FFF
Flash Controller	FLC_	0x4002 9000	0x4002 93FF
Internal Cache Controller	ICC_	0x4002 A000	0x4002 AFFF
UART 0	UART0_	0x4004 2000	0x4004 2FFF
UART 1	UART1_	0x4004 3000	0x4004 3FFF
SPI0	SPI0_	0x4004 6000	0x4004 6FFF

4 System Clocks, Reset, and Power Management

The MAX32660 includes a High-Frequency Internal Oscillator (HFIO), an 8kHz nano-ring oscillator and support for an external 32kHz crystal or external clock. Support for selectable core operating voltage is provided and the HFIO frequency is scaled based on the specific core operating voltage range selected.

4.1 Core Operating Voltage Range Selection

The MAX32660 supports three selections for the core Operating Voltage Range (OVR). In single supply operation, changing the OVR sets the output of the internal LDO regulator to the voltage shown in *Table 4-1*. For dual supply designs, setting the OVR allows the MAX32660 to use an external PMIC to provide the required V_{CORE} voltage dynamically. Changing the OVR also reduces the output frequency of the High-frequency Internal Oscillator (HFIO), further reducing power consumption.

Changes to the OVR affect the access time of the internal flash memory and the application firmware must set the flash wait states for each OVR setting as outlined in section *4.1.2 Flash Wait States* for details on minimum flash wait states for the internal flash memory.

Changing the core operating voltage immediately reduces the output frequency of the High Frequency Internal Oscillator as shown in *Table 4-1*, *below*. When operating the MAX32660 using dual external supplies requires special considerations and must be handled carefully in the application firmware.

Table 4-1: Operating Voltage Range Selection and the Effect on V_{CORE} and f_{HFIO}

PWRSEQ_LP_CTRL ***ovr***	*FLC_CTRL* ***lve***	**V_{CORE} Typical**	***f_{HFIO}***
0	1	0.9	24
1	1	1.0	48
2	0	1.1	96

4.1.1 Setting the Operating Voltage Range

The Operating Voltage Range (OVR) selection is controlled using the Power Sequencer Low Power Control Register *PWRSEQ_LP_CTRL*.*ovr* which is only reset by a Power-On Reset (POR) and these bits should be checked after every reset to determine the correct clock speed and flash wait states. Adjusting the OVR setting effects the frequency of the HFIO. Prior to changing the OVR settings, it is required to set the system clock to either the 8kHz Low-Frequency Internal Oscillator or the 32.768kHz External Oscillator. The MAX32660 coordinates OVR change between the internal LDO and the HFIO set frequency. When changing the OVR setting, the MAX32660 must be operating from the internal LDO. In a system using an external supply for V_{CORE}, firmware must transition to the internal LDO prior to changing the OVR setting.

The following steps describe how to change the OVR:

1. Ensure the part is operating from the internal LDO for V_{CORE}.
   a. Set *PWRSEQ_LP_CTRL.ldo_dis* to 0.
   b. If using an external supply for V_{CORE}, ensure the external supply is set to the same voltage as the current OVR setting. The external supply must be equal to or greater than the set OVR voltage.
2. Set either the 32.768kHz external oscillator or 8kHz nano-ring oscillator as the system clock source.
   a. Refer to section *4.3 Oscillator Sources and System Clock Selection* for details on system clock selection.
3. Set the number of Flash Wait States to the POR default value of 5.
   a. *GCR_MEM_CTRL.fw* = 5
4. Change the OVR setting to the desired range.
   a. Set *PWRSEQ_LP_CTRL.ovr* to either 0, 1, or 2 as shown in *Table 4-1*, *above*.
5. Set the Flash Low Voltage Enable according to the OVR setting set in step 4.a.
   a. Set *FLC_CTRL.lve* to either 0 or 1 as required. Reference *Table 4-1*, *above* for the required value.
6. If desired, set the system clock source to the HFIO and update the system clock prescaler to the desired value.
   a. Set *GCR_CLK_CTRL.clksel* = 0.
   b. Wait for the system clock ready bit, *GCR_CLK_CTRL.clkrdy,* to read 1.
   c. Set *GCR_CLK_CTRL.psc* to the desired prescaler value.
7. Set the number of Flash Wait States per *Table 4-2*, *below*.
   a. Set *GCR_MEM_CTRL.fws* to the minimum value shown for the selected OVR and System Clock
8. Perform a Peripheral Reset.
   a. Set *GCR_RST0.periph_rst* = 1.

On each subsequent reset event:

1. If *PWRSEQ_LP_CTRL.ovr* is set to 0, set the flash low voltage enable bit to 1 (*FLC_CTRL.lve* = 1)
2. Set the clock prescaler, *GCR_CLK_CTRL.psc*, as needed by the system.
3. Set the number of flash wait states, *GCR_MEM_CTRL.fws*, as needed based on the OVR settings using *Table 4-2*, *below*.

4.1.2 Flash Wait States

Power-On Reset, System Reset and Watchdog Reset all reset the Flash Wait State field, *GCR_MEM_CTRL.fws*, to the POR default setting of 5. The Flash Wait State field is the number of system clock cycles for accessing the internal flash memory and is dependent on the OVR settings, the system oscillator selected and the system oscillator prescaler.

The setting for the number of flash wait states effects performance and it is critical to set it correctly based on the OVR settings and the system clock frequency. Set the number of flash wait states using the field *GCR_MEM_CTRL.fws* per *Table 4-2*, *below*. The *GCR_MEM_CTRL.fws* field should always be set to the default POR reset value of 5 prior to changing the *PWRSEQ_LP_CTRL.ovr* settings.

Important: *Flash reads may fail and result in unknown instruction execution if the Flash Wait State setting is lower than the minimum required for a given OVR setting and the selected system clock frequency.*

Note: When changing the system clock prescaler to move from a slower system clock frequency to a faster system clock frequency, always set the Flash Wait State field to the minimum required for the faster system clock frequency prior to changing the system oscillator prescaler.

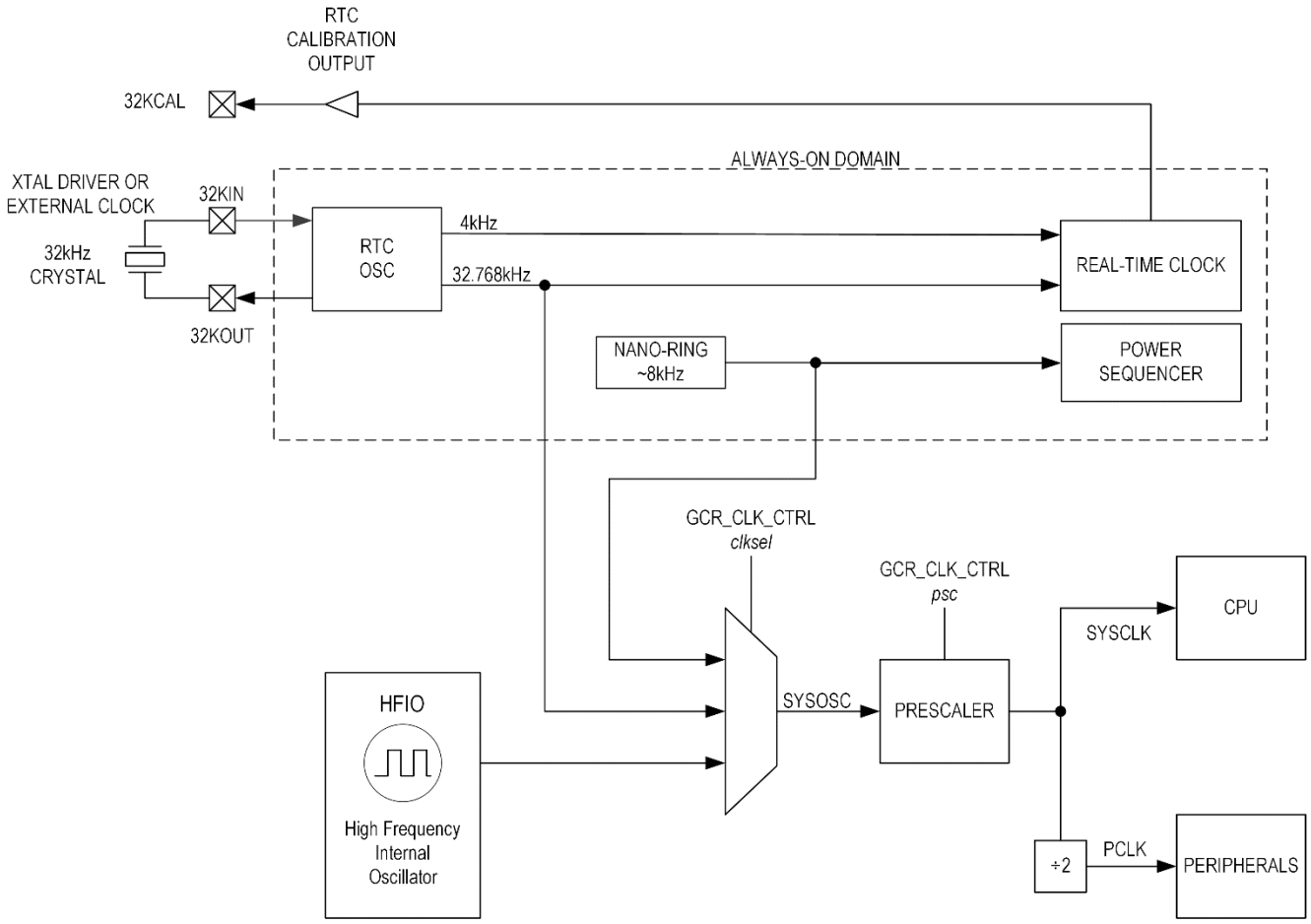
Table 4-2: Minimum Flash Wait State Setting for Each OVR Setting ($f_{SYSCLK} = f_{HFIO}$)

Core Operating Voltage Range Setting		Core Voltage Range	f_{HFIO} (MHz)	System Clock Prescaler	System Clock	Minimum Flash Wait State Setting
PWRSEQ_LP_CTRL.ovr	FLC_CTRL.lve	V_{CORE} (V)		GCR_CLK_CTRL.psc	f_{SYSCLK} (MHz)	GCR_MEM_CTRL.fws
0	1	0.9	24	0	24	2
				1	12	1
1	1	1.0	48	0	48	3
				1	24	2
2	0	1.1	96	0	96	4
				1	48	2
				2	24	1

4.2 System Clocks

Figure 4-1, *below*, shows a high-level diagram of the MAX32660 clock tree.

Figure 4-1: Clock Tree Diagram



The selected System Oscillator (SYSOSC) is the clock source for most internal blocks. Select SYSOSC from the following clock sources:

- High-Frequency Internal Oscillator (HFIO)
- 8kHz Internal Ultra-Low Power Nano-Ring Oscillator
- 32.768kHz External Crystal Oscillator

The selected SYSOSC is the input to the system oscillator prescaler, which generates the System Clock (SYSCLK). The system oscillator prescaler divides SYSOSC by a prescaler using the *GCR_CLK_CTRL.psc* field as shown in *Equation 4-1*.

Equation 4-1: System Clock Scaling

$$SYSCLK = \frac{SYSOSC}{2^{psc}}$$

GCR_CLK_CTRL.psc is selectable from 0 to 7, resulting in divisors of 1, 2, 4, 8, 16, 32, 64 or 128.

SYSCLK drives the Arm® Cortex-M4 with FPU and is used to generate the following internal clocks:

- Advanced High-Performance Bus (AHB) Clock
  - $HCLK = SYSCLK$
- Advanced Peripheral Bus (APB) Clock,
  - $PCLK = \frac{SYSCLK}{2}$

The Real-Time Clock (RTC) uses the 32.768kHz oscillator for its clock source.

All oscillators are reset to their POR reset default state during a Power-On Reset, System Reset and a Watchdog Reset. Oscillator settings are not reset during a Soft Reset or Peripheral Reset. *Table 4-3* shows each oscillator's enabled state for each type of reset source in the MAX32660. *Table 4-4* details the effect each reset source has on the System Clock selection and the System Clock prescaler settings.

Table 4-3: Reset Sources and Effect on Oscillator Status

	Reset Source				
Oscillator	**POR**	**System**	**Watchdog**	**Soft**	**Peripheral**
HFIO	Enabled	Enabled	Enabled	Retains State	Retains State
8kHz nano-ring	Enabled	Enabled	Enabled	Enabled	Enabled
32kHz oscillator	Disabled	Retains State	Retains State	Retains State	Retains State

Table 4-4: Reset Sources and Effect on System Oscillator Selection and Prescaler

	Reset Source				
Clock Field	**POR**	**System**	**Watchdog**	**Soft**	**Peripheral**
System Oscillator *GCR_CLK_CTRL.clksel*	0 (HFIO)	0 (HFIO)	0 (HFIO)	Retains State	Retains State
System Clock Prescaler *GCR_CLK_CTRL.psc*	1	1	1	Retains State	Retains State

4.3 Oscillator Sources and System Clock Selection

Before using any oscillator, the oscillator must first be enabled by setting its corresponding enable bit in the System Clock Control Register, *GCR_CLK_CTRL*. Before setting any oscillator as SYSOSC, its corresponding oscillator ready bit in the *GCR_CLK_CTRL* register must first be checked.

Once the corresponding oscillator ready bit is set, the oscillator can then be selected as SYSOSC by configuring the Clock Source Select field (*GCR_CLK_CTRL.clksel*). If an oscillator's ready bit reads 0, the hardware does not allow selecting the oscillator as the SYSOSC.

Any time firmware changes SYSOSC by changing *GCR_CLK_CTRL.clksel*, the Clock Ready bit, *GCR_CLK_CTRL.clkrdy,* is automatically cleared to indicate that a SYSOSC switchover is in progress. The application should read the clock ready field until it returns 1. While the clock ready field reads 0, the prior clock is used as the SYSOSC.

Immediately before entering any low-power mode the application must enable any oscillator needed during the low power mode.

Refer to *Table 4-3* and *Table 4-4* for details on reset sources and effects on each of the oscillators described below.

4.3.1 High-Frequency Internal Oscillator

The MAX32660 includes a High-Frequency Internal Oscillator (HFIO) that is the default system oscillator after a POR, System and Watchdog Reset.

To enter *DEEPSLEEP*, the HFIO must be powered down by setting register bit *GCR_PM.hfiopd* to 1. Failure to set this bit to 1 will inhibit the processor from entering *DEEPSLEEP*.

Refer to *Table 4-3* and *Table 4-4* for details on reset sources and the effect on the HFIO oscillator.

4.3.2 32.768kHz External Crystal or Clock

This is a very low power oscillator used for the RTC and optionally can be selected as the System Clock. This oscillator can be driven using an external clock or a 32.768kHz crystal. The 32.768kHz clock signal is available as an output on GPIO 32KCAL.

This oscillator is the dedicated clock for the Real-Time Clock (RTC). If the RTC is enabled, the 32.768kHz external oscillator must be enabled, independent of the selection of the System Clock.

When this oscillator is active, an RTC alarm can wake this device from *SLEEP* or *DEEPSLEEP* mode if the *GCR_PM.rtcwk_en* is set to 1 and the RTC alarm is configured.

The 32kHz oscillator is disabled by hardware by a Power-On Reset. All other forms of reset do not change the 32kHz oscillator enable bit. Refer to *Table 4-3* and *Table 4-4* for details on reset sources and the effect on the 32kHz oscillator.

4.3.3 8kHz Ultra-Low Power Nano-Ring Internal Oscillator

An ultra-low power internal 8kHz nano-ring oscillator is available and can be set as the System Oscillator (SYSOSC). This oscillator is enabled at device powerup by hardware and cannot be disabled by application firmware.

4.4 System Oscillators Reset

On Power-On Reset (POR) and System Reset, all oscillator states are reset to their Power-On Reset default:

- The HFIO and the 8kHz nano-ring oscillators are enabled.
- The 32.768kHz external oscillator is disabled.
- The system clock selection and prescaler are reset.

The oscillator enable fields, system clock selection field and the system clock prescaler fields are not affected by a Soft Reset or Peripheral Reset.

4.5 Operating Modes

The MAX32660 supports four operating modes:

- *ACTIVE*
- *SLEEP*
- *DEEPSLEEP*
- *BACKUP*

ACTIVE mode is the only mode is which code execution occurs.

For the low power modes, the Arm Cortex-M4 with FPU does not execute code. Each low power mode of operation supports wakeup events that result in the MAX32660 to re-enter *ACTIVE* mode operation. For *SLEEP* mode, wakeup events include any external or internal interrupt, RTC wakeup, and the Watchdog Interrupt. *DEEPSLEEP* and *BACKUP* mode wakeup events are limited to any enabled GPIO external interrupt or from an enabled RTC wakeup event.

The Arm Cortex-M family of CPUs have two built-in low power modes, designated *SLEEP* and *DEEPSLEEP*. Implementation of these low-power modes are specific to the microcontroller's design. These modes are enabled using the Arm Cortex-M4 with FPU System Control Register (SCR). Write register bit SCR.*deepsleep* to select the low power mode as shown in the pseudocode below.

```
SCR.sleepdeep = 0; // SLEEP mode enabled
SCR.sleepdeep = 1; // DEEPSLEEP mode enabled
```

Once enabled, the device enters the enabled low power mode when either a WFI (Wait for Interrupt) or WFE (Wait for Event) instruction is executed.

Immediately before entering any low-power mode, enable the SYSOSC to be used in that low-power mode. If the selected SYSOSC is disabled in the selected low power mode, it will be enabled upon returning to *ACTIVE* mode.

Refer to the *Cortex*-M4 with FPU *Technical Reference Manual* for more information on the SCR register and the WFI and WFE instructions.

4.5.1 ACTIVE Mode

This is the highest performance mode. All internal clocks, registers, memory, and peripherals are enabled. The CPU is running and executing application code. All oscillators are available for application use, if enabled.

Dynamic clocking allows firmware to selectively enable or disable clocks and power to individual peripherals, providing the optimal mix of high-performance and power conservation. Internal RAM that can be enabled, disabled, or placed in low-power RAM Retention Mode include data SRAM memory blocks, on-chip cache, and on-chip FIFOs. Refer to *RAM Low Power Modes* for details on RAM power mode control.

4.5.2 SLEEP Low Power Mode

This is a low power mode that suspends the CPU with a fast wakeup time to *ACTIVE* mode. In *SLEEP* mode, the microcontroller remains in an *ACTIVE* state with the system clock disabled for the Cortex core. Code execution stops during *SLEEP* mode. All enabled oscillators remain active and the RAM retains state if enabled. Refer to *RAM Low Power Modes* for details on enabling and disabling RAM sector data retention.

SLEEP mode wakeup events include any external or internal interrupt.

The following pseudocode places the device in *SLEEP* mode:

```
GCR_PMR.mode = 0;     // Set mode field to ACTIVE to ensure DEEPSLEEP mode is entered
SCR.sleepdeep = 0; // SLEEP mode enabled
WFI;              // Enter SLEEP mode, WFI can be replaced with WFE
```

Refer to the *Cortex*-M4 with FPU *Technical Reference Manual* for more information on the SCR register and the WFI and WFE instructions.

4.5.3 DEEPSLEEP Low Power Mode

In *DEEPSLEEP* mode all internal clocks are gated off including the system clock and the Watchdog Timer. The RTC continues operation, if enabled, during *DEEPSLEEP*. The Arm Cortex-M4 with FPU state and all system and peripheral registers retain

their state (settings) during *DEEPSLEEP* mode. RAM retention is disabled by default. The application can explicitly enable RAM retention for each RAM sector using the *PWRSEQ_LP_CTRL.ramret_sel[3:0].* Set *PWRSEQ_LP_CTRL.ramret_sel[3:0]* fields to 1 to enable data retention for each RAM sector respectively during *DEEPSLEEP* mode. GPIO also retain their configuration during *DEEPSLEEP* mode.

Wakeup from *DEEPSLEEP* can occur from an external GPIO interrupt or from a RTC alarm, both of which must be enabled separately. Refer to *Table 4-5* for Wake-up sources for each low-power mode.

The High Frequency Oscillator, HFIO, must be powered off when entering *DEEPSLEEP* mode. Set *GCR_PM.hfiopd* = 1 to enter *DEEPSLEEP* mode. The 8kHz and 32.768kHz oscillators are available. Additionally, the *GCR_PM.mode* field should be set to 0 prior to entering *DEEPSLEEP* mode.

```
GCR_PMR.hfiopd = 1;   // Set HFIO to automatic power down for DEEPSLEEP
GCR_PMR.mode = 0;     // Set mode field to ACTIVE ensure DEEPSLEEP mode is entered
SCR.sleepdeep = 1;    // DEEPSLEEP mode enabled
WFI;                 // Enter DEEPSLEEP mode, WFI can be replaced with WFE
```

4.5.4 BACKUP Low Power Mode

This is the lowest power operating mode. The HFIO is disabled in *BACKUP* mode. The 8kHz oscillator remains on. If the 32kHz oscillator is enabled, it remains enabled during *BACKUP* mode. The Arm Cortex-M4 with FPU state and all system and peripheral registers do not retain state. The RTC and AoD registers do retain state during *BACKUP* mode.

RAM retention in *BACKUP* mode requires the Retention Regulator. If RAM retention is required, enable the RAM Retention Regulator by setting the *PWRSEQ_LP_CTRL.retreg_en* field to 1. Enable each RAM sector individually for data retention during *BACKUP* mode by setting the *PWRSEQ_LP_CTRL.ramret_sel[3:0]* fields to 1. By default, the RAM sectors do not retain state during *BACKUP* mode.

Wakeup from *BACKUP* mode can occur from an external GPIO interrupt or from an RTC alarm, both of which must be enabled separately. Refer to *Table 4-5* for Wakeup sources for each low-power mode.

Set *GCR_PM.mode* = 4 to immediately enter *BACKUP* mode.

4.5.5 Wake-Up Sources

Table 4-5, *below*, lists the wake-up sources for each of the low-power modes supported by the MAX32660. Wake-up from a GPIO interrupt and the RTC alarm must be enabled by the application to wake-up the device from *SLEEP*, *DEEPSLEEP* and *BACKUP* mode.

Table 4-5: Wake-Up Sources for Each Low-Power Mode

	Low Power Mode		
Wake-Up Source	***SLEEP***	***DEEPSLEEP***	***BACKUP***
Internal Interrupt	Y	N	N
External Reset	Y	Y	Y
GPIO Interrupt	Y	Y	Y
RTC Alarm	Y	Y	Y

4.6 Shutdown State

Shutdown State is not a low-power mode. It is intended to wipe all volatile memory from the device. In the Shutdown State, internal logic gates off all internal power. There is no data, register, or RAM retention in this mode. All wakeup sources,

wakeup logic, and interrupts are disabled. The Always-on Domain (AoD) is disabled as well. The device only recovers through a Power-On Reset (POR) which re-initializes the device.

Setting *GCR_PM.mode* = 6 results in the part immediately entering Shutdown State.

4.7 Device Resets

Four device resets are available:

- Peripheral Reset
- Soft Reset
- System Reset
- Power-On Reset

On completion of any of the four reset cycles, all peripherals are reset, HCLK and PCLK are operational, the CPU core receives clocks and power, and the device enters *ACTIVE* mode. Program execution begins at the reset vector address.

Each peripheral in the MAX32660 can be reset individually by firmware using the *GCR_RST0* and *GCR_RST1* registers.

4.7.1 Peripheral Reset

This resets all peripherals. The CPU retains its state. The GPIO, Watchdog Timers, RAM Retention, and Global Control Registers (GCR), including the clock configuration, are unaffected.

Initiate a Peripheral Reset by setting *GCR_RST0.periph_rst* to 1.

4.7.2 Soft Reset

This is the same as a Peripheral Reset except that it also resets the GPIO to its Power-On Reset state. All alternate functions are tri-stated.

Initiate a Soft Reset by setting *GCR_RST0.soft* to 1.

4.7.3 System Reset

This is the same as Soft Reset and additionally resets all GCR registers, resetting the clocks to their default state. The CPU state is reset as well as the watchdog timers. The AoD and RAM Retention are unaffected.

A watchdog timer reset event initiates a System Reset. To start a System Reset from the application, set *GCR_RST0.system* = 1.

4.7.4 Power-On Reset

A POR resets everything in the device to its power-on reset default state. *Table 4-6* shows the effects of the four reset types and the four power modes supported by the MAX32660.

Table 4-6: Reset and Low Power Mode Effects

	Peripheral Reset	Soft Reset	System Reset	POR	*ACTIVE* Mode	SLEEP Mode	*DEEPSLEEP* Mode	*BACKUP* Mode
GCR Registers	No	No	Reset	Reset	N/A	N/A	N/A	N/A

	Peripheral Reset	Soft Reset	System Reset	POR	*ACTIVE* Mode	SLEEP Mode	*DEEPSLEEP* Mode	*BACKUP* Mode
8kHz Osc	On	On	On	On	On	On	On	On
High Freq Osc	-	-	On	On	Y	Y	Auto Off	Off
PCLK	On	On	On	On	On	On	Off	Off
HCLK	On	On	On	On	On	On	Off	Off
CPU Clock	On	On	On	On	On	Off	Off	Off
V_{CORE}	On	On	On	On	On	On	Off	Off
CPU State Retention	On	On	Reset	Reset	N/A	On	On	Off
RTC	-	-	Reset	Reset	Y	Y	Y	Y
Standard DMA	Reset	Reset	Reset	Reset	Y	Y	Off	Off
Watchdog Timer	-	-	Reset	Reset	Y	Y	Off	Off
GPIO	-	Reset	Reset	Reset	Y	Y	Y	Y
Flash Controller, ICC0 Cache	Reset	Reset	Reset	Reset	Y	Y	Off	Off
Other Peripherals	Reset	Reset	Reset	Reset	Y	Y	Off	Off
AoD	On	Y	Y	Y	Y	On	On	Auto Off
RAM Retention	Y	Y	Y	Reset	Y	Y	Y	Auto Off

Table key:
Y = Enabled, can be disabled by firmware
On = Enabled by hardware (Cannot be disabled)
Off = Disabled by hardware (Cannot be enabled)
Auto Off = Can either be left on, or automatically gated off when in this power mode.
- = No Effect
N/A = Not Applicable

Note: The AoD includes the oscillator trim settings, the RTC, RAM retention, and Low Power Wakeup Control Registers.

Note: Only a Power-On Reset triggers a reset of the AoD.

Note: RAM Retention applies to data SRAM, cache, and all FIFOs.

Note: Peripheral, Soft, and System Resets are initiated by firmware though the GCR_RST0 register.

Note: A Watchdog Reset initiates a System Reset.

4.8 Instruction Cache Controller

ICC0 is the Instruction Cache Controller used for the internal Flash Memory. ICC0 includes a line buffer, tag RAM and a 16KB 2-way set associative Data RAM.

4.8.1 Enabling ICC0

Perform the following steps to enable ICC0.

- Set *ICC0_CACHE_CTRL.enable* to 1
- Read *ICC0_CACHE_CTRL.ready* until it returns 1

4.8.2 Disabling ICC0

Disable ICC0 by setting *ICC0_CACHE_CTRL.enable* to 0.

4.8.3 Flushing the ICC0 Cache

The System Configuration Register (*GCR_SCON*) includes a field for flushing ICC0. Setting *GCR_SCON.ccache_flush* to 1 performs a flush of ICC0's 16KB Cache and the tag RAM. Set the *ICC0_INVALIDATE* register to 1 to invalidate the ICC0 cache and force a cache flush. Read the *ICC0_CACHE_CTRL.ready* field until it returns 1 to determine when the flush is completed.

4.9 ICC0 Registers

The ICC0 base peripheral address is 0x4002 A000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 4-7: Instruction Cache Controller Register Addresses and Descriptions

Offset	Register Name	Access	Description
[0x0000]	*ICC0_CACHE_ID*	RO	Cache ID Register
[0x0004]	*ICC0_MEM_SIZE*	R0	Cache Memory Size Register
[0x0100]	*ICC0_CACHE_CTRL*	R/W	Cache Control Register
[0x0700]	*ICC0_INVALIDATE*	R/W	Cache Invalidate Register

4.9.1 ICC0 Register Details

Table 4-8: ICC Cache ID Register

ICC Cache ID Register			ICC0_CACHE_ID	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	RO	-	**Reserved for Future Use** Do not modify this field.
15:10	cchid	RO	-	**Cache ID** Returns the Cache ID for this Cache instance.
9:6	partnum	RO	-	**Cache Part Number** Returns the part number indicator for this Cache instance.
5:0	relnum	RO	-	**Cache Release Number** Returns the release number for this Cache instance.

Table 4-9: ICC Memory Size Register

ICC Memory Size Register			ICC0_MEM_SIZE	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	memsz	RO	-	**Addressable Memory Size** Indicates the size of addressable memory by this cache controller instance in 128KB units.
15:0	cchsz	RO	-	**Cache Size** Returns the size of the cache RAM memory in 1KB units. 16: 16KB Cache RAM

Table 4-10: ICC Cache Control Register

ICC Cache Control Register			ICC0_CACHE_CTRL	[0x0100]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	-	**Reserved for Future Use** Do not modify this field.
16	ready	RO	-	**Ready** This field is cleared by hardware anytime the cache as a whole is invalidated (including a Power On Reset event). Hardware automatically sets this field to 1 when the invalidate operation is complete and the cache is ready. 0: Cache Invalidate in process. 1: Cache is ready. *Note: While this field reads 0, the cache is bypassed and reads come directly from the line fill buffer.*
15:1	-	R/W	-	**Reserved for Future Use** Do not modify this field.
0	enable	R/W	0	**Enable** Set this field to 1 to enable the cache. Setting this field to 0 automatically invalidates the cache contents. When this cache is disabled, reads are handled by the line fill buffer. 0: Disable Cache 1: Enable Cache

Table 4-11: ICC Invalidate Register

ICC Invalidate Register			ICC0_INVALIDATE	[0x0700]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	-	WO	-	**Invalidate** Any write to this register of any value invalidates the cache.

4.10 RAM Memory Management

This device has many features for managing the on-chip RAM. The on-chip RAM includes data RAM, an instruction cache (ICC0), and the peripheral FIFOs.

4.10.1 On-Chip Cache Management

The MAX32660 includes an instruction cache controller for code fetches from the flash memory. The cache can be enabled, disabled, and zeroized and the cache clock can be disabled by placing it in Light Sleep. Refer to section *4.8 Instruction Cache Controller* for details.

4.10.2 RAM Zeroization

The GCR Memory Zeroize Control Register, *GCR_MEM_ZCTRL*, allows clearing memory by the application. Zeroization writes all zeros to memory.

The following RAM memories can be zeroized:

- Data RAM
  - Zeroize the entire Data RAM by setting *GCR_MEM_ZCTRL*.*sram_zero* bit to 1
- Instruction Cache Controller Data and Tag RAM
  - Set *GCR_MEM_ZCTRL*.*icache_zero* to 1 to zeroize the entire 16KB cache RAM and tag RAM

4.10.3 RAM Low Power Modes

RAM can be placed in a low power mode, referred to as Light Sleep, using register *GCR_MEM_CTRL*, Memory Clock Control. Light Sleep gates off the clock to the RAM and makes the RAM unavailable for read/write operations. The RAM contents are retained during Light Sleep mode. Light Sleep is available for the internal Data RAM blocks as well as for the ICC0 cache RAM. Turning off Light Sleep mode for a memory enables Read/Write to that memory range.

RAM can also be shut down for power savings using the register *PWRSEQ_LPMEMSD*, RAM Shut Down Control. This conserves power by gating off the power and clock to the RAM. This invalidates the contents of the RAM and the RAM is not accessible until the RAM shutdown mode is disabled. When enabling a RAM partition from a shutdown state, the RAM contents are cleared.

4.11 Global Control Registers (GCR)

The GCR base peripheral address is 0x4000 0000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

The Global Control Registers are only reset on a System Reset, Watchdog Timer Reset and a Power-On Reset. Soft Reset and Peripheral Reset do not affect these registers.

Table 4-12: Global Control Registers, Offsets and Descriptions

Offset	Register Name	Access	Description
[0x0000]	*GCR_SCON*	R/W	System Control Register
[0x0004]	*GCR_RST0*	R/W	Reset Register 0
[0x0008]	*GCR_CLK_CTRL*	R/W	Clock Control Register
[0x000C]	*GCR_PM*	R/W	Power Management Register
[0x0024]	*GCR_PCLK_DIS0*	R/W	Peripheral Clocks Disable 0
[0x0028]	*GCR_MEM_CTRL*	R/W	Memory Clock Control
[0x002C]	*GCR_MEM_ZCTRL*	R/W	Memory Zeroize Register
[0x0040]	*GCR_SYS_STAT*	RO	System Status Flags
[0x0044]	*GCR_RST1*	R/W	Reset Register 1
[0x0048]	*GCR_PCLK_DIS1*	R/W	Peripheral Clocks Disable 1
[0x004C]	*GCR_EVTEN*	R/W	Event Enable Register
[0x0050]	*GCR_REV*	RO	Revision Register
[0x0054]	*GCR_SYS_IE*	R/W	System Status Interrupt Enable

4.11.1 Global Control Register Details

Table 4-13: System Control Register

System Control Register		GCR_SCON		[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:15	-	RO	-	**Reserved for Future Use** Do not modify this field.

System Control Register	GCR_SCON	[0x0000]

Bits	Name	Access	Reset	Description
14	swd_dis	R/W	See Description	**Serial Wire Debug Disable** 0: JTAG SWD enabled. 1: JTAG SWD disabled. *Note: If the Arm ICE is unlocked (GCR_SYS_ST.icelock=0), the POR reset value for this bit is 0. If the Arm ICE is locked (GCR_SYS_ST.icelock=1), the POR reset value for this bit is 1 and is not writable.*
13:7	-	RO	-	**Reserved for Future Use** Do not modify this field.
6	icc0_flush	R/W1O	0	**Instruction Cache Controller Flush** Write 1 to flush the internal flash cache. This bit is cleared by hardware when the flush is complete. 0: Flush not in process. 1: Write 1 to flush the code cache.
5	fpu_dis	R/W	0	**Floating Point Unit (FPU) Disable** Set this field to 1 to disable the Cortex-M4 Floating Point Unit. 0: FPU enabled. 1: FPU disabled.
4	flash_page_flip	RO	0	**Flash Page Flip Flag** Flips the bottom and top halves of the internal flash memory. This bit is controlled by hardware. Firmware should not change the state of this bit during normal operation. Any change to this bit flushes the instruction cache and the data cache 0: Physical layout matches logical layout 1: Top and Bottom halves flipped.
3:0	-	R/W	-	**Reserved for Future Use** Do not modify this field.

Table 4-14: Reset 0 Register

Reset 0 Register	GCR_RST0	[0x0004]

Bits	Name	Access	Reset	Description
31	system	R/W1O	0	**System Reset** Write 1 to perform a System Reset of the MAX32660. When set to 1, the system reset resets everything on the device except the AoD registers and the RAM retention. All other registers, peripherals, the CPU core and the watchdog timer are reset. This field is cleared by hardware when the reset is complete. 0: Reset complete. 1: Write 1 to perform a System Reset. *Refer to the Device Resets section for additional information.*
30	periph	R/W1O	0	**System Peripheral Reset** Write 1 to perform a Peripheral Reset. All peripherals are reset except for the GPIO port pin configuration and Watchdog Timer. 0: Reset complete. 1: Write 1 to perform the Peripheral reset. *Refer to the Device Resets section for additional information.*

Reset 0 Register	GCR_RST0	[0x0004]

Bits	Name	Access	Reset	Description
29	soft	R/W1O	0	**Soft Reset** Write 1 to perform a soft reset. A soft reset resets all peripherals including the GPIO port pin configuration but does not reset the Watchdog Timer peripheral. 0: Reset complete. 1: Write 1 to perform the soft reset. *Refer to the Device Resets section for additional information.*
28:18	-	RO	-	**Reserved for Future Use** Do not modify this field.
17	rtc	R/W1O	0	**RTC Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: RTC peripheral not in reset. 1: Write 1 to reset the RTC peripheral.
16	i2c0	R/W1O	0	**I2C0 Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: I2C0 peripheral not in reset. 1: Write 1 to reset the I2C0 peripheral.
15	-	RO	0	**Reserved for Future Use** Do not modify this field.
14	spi1	R/W1O	0	**SPIMSS (SPI1/I²S) Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: SPIMSS (SPI1/I²S) peripheral not in reset. 1: Write 1 to reset the SPIMSS (SPI1/I²S) peripheral.
13	spi0	R/W1O	0	**SPI0 Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: SPI0 peripheral not in reset. 1: Write 1 to reset the SPI0 peripheral.
12	uart1	R/W1O	0	**UART1 Reset** Write 1 to reset the peripheral. This field is cleared by hardware when the peripheral is reset. 0: Reset complete. 1: Write 1 to reset the UART1 peripheral.
11	uart0	R/W1O	0	**UART0 Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: UART0 peripheral not in reset. 1: Write 1 to reset the UART0 peripheral.
10:8	-	RO	0	**Reserved for Future Use** Do not modify this field.
7	timer2	R/W1O	0	**Timer2 Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: TMR2 peripheral not in reset. 1: Write 1 to reset the TMR2 peripheral.

Reset 0 Register			GCR_RST0	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
6	timer1	R/W1O	0	**Timer1 Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: TMR1 peripheral not in reset. 1: Write 1 to reset the TMR1 peripheral.
5	timer0	R/W1O	0	**Timer0 Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: TMR0 peripheral not in reset. 1: Write 1 to reset the TMR0 peripheral.
4:3	-	RO	0	**Reserved for Future Use** Do not modify this field.
2	gpio0	R/W1O	0	**GPIO0 Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: GPIO peripheral not in reset. 1: Write 1 to reset the GPIO peripheral.
1	wdt0	R/W1O	0	**Watchdog Timer 0 Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: WDT0 peripheral not in reset. 1: Write 1 to reset the WDT0 peripheral.
0	dma	R/W1O	0	**Standard DMA Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: Standard DMA peripheral not in reset. 1: Write 1 to reset the Standard DMA peripheral.

Table 4-15: System Clock Control Register

System Clock Control Register			GCR_CLK_CTRL	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:30	-	RO	0b11	**Reserved for Future Use** Do not modify this field.
29	lirc8k_rdy	RO	0	**8kHz Internal Oscillator Ready Status** On POR or System Reset this field reads 0 until the 8kHz low-frequency oscillator is ready for use. 0: Not ready or not enabled. 1: Oscillator ready.
28:27	-	RO	0	**Reserved for Future Use** Do not modify this field.
26	hirc_rdy	RO	0	**High-Frequency Internal Oscillator Ready** On POR or System Reset this field reads 0 until the HFIO oscillator is ready. If the HFIO is disabled (*GCR_CLK_CTRL*.*hirc_en* = 0) and firmware enables it (*GCR_CLK_CTRL*.*hirc_en* = 1), reading this field will return 0 until the HFIO is ready for use. 0: Oscillator not ready or not enabled. 1: Oscillator ready.

System Clock Control Register			GCR_CLK_CTRL	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
25	x32k_rdy	RO	0	**32.768kHz External Oscillator Ready Status** On POR or System Reset this field is set to 0. Setting *GCR_CLK_CTRL.x32k_en* to 1 forces this field to 0 and hardware sets this field to 1 automatically when the oscillator is ready for use. 0: Oscillator not ready. 1: Oscillator ready.
24:19	-	RO	-	**Reserved for Future Use** Do not modify this field.
18	hirc_en	R/W	1	**High-Frequency Internal Oscillator (HFIO) Enable** Write 0 to disable the internal HFIO. When this field is set to 0, hardware automatically clears the *GCR_CLK_CTRL.hirc_rdy* bit. 0: Set to 0 to disable the HFIO. 1: Set to 1 to enable the HFIO.
17	x32k_en	R/W	0	**32.768kHz External Oscillator Enable** Write 1 to enable the 32kHz external oscillator. After setting this field to 1, hardware automatically clears *GCR_CLK_CTRL.x32k_rdy*. When the *GCR_CLK_CTRL.x32k_rdy* bit reads 1, the 32.768kHz oscillator is ready. 0: Disable the 32kHz oscillator (POR default). 1: Enable the 32kHz oscillator.
16:14	-	RO	-	**Reserved for Future Use** Do not modify this field.
13	clkrdy	R/W	0	**System Oscillator Clock Source Ready** When the System Oscillator source is modified by changing the *GCR_CLK_CTRL.clksel* field, the *GCR_CLK_CTRL.clkrdy* field reads 0 until the switchover completes. This field is set to 1 by hardware when the selected clock source is ready. 0: The selected System Oscillator is not ready for use. 1: The selected clock source (*GCR_CLK_CTRL.clksel*) is the active System Oscillator.
12	-	RO	-	**Reserved for Future Use** Do not modify this field.
11:9	clksel	R/W	0	**System Oscillator Source Select** Selects the system oscillator (SYSOSC) source used to generate the system clock (SYSCLK). Modifying this field immediately clears the *GCR_CLK_CTRL.clkrdy* field. 0: High-Frequency Internal Oscillator (HFIO) 1: Reserved for Future Use 2: Reserved for Future Use 3: 8kHz Low-Frequency Internal Oscillator 4: Reserved for Future Use 5: Reserved for Future Use 6: 32.768kHz External Oscillator 7: Reserved for Future Use
8:6	psc	R/W	0	**System Oscillator Prescaler** Sets the divider for generating the System Clock (SYSCLK) from the selected System Oscillator (SYSOSC) as shown in the following equation: $f_{SYSCLK} = \frac{f_{SYSOSC}}{2^{psc}}$
5:0	-	R/W	8	**Reserved for Future Use** Do not modify this field.

Table 4-16: Power Management Register

Power Management Register			GCR_PM	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved for Future Use** Do not modify this field.
15	hfiopd	R/W	0	**HFIO *DEEPSLEEP* Auto Off** When set, the High-Frequency Internal Oscillator is automatically powered off when in *DEEPSLEEP* mode. If the HFIO is not enabled, *GCR_CLK_CTRL.hirc_en* = 1, when entering *DEEPSLEEP* mode, the value of this field is ignored. 0: HFIO oscillator is active, powered on, during *DEEPSLEEP* mode if the HFIO is enabled (*GCR_CLK_CTRL.hirc_en* = 1). 1: HFIO Oscillator is powered off in *DEEPSLEEP* mode. *Note: This field should be set to 1 prior to entering DEEPSLEEP mode to achieve the lowest power numbers for the device if the HFIO is enabled (GCR_CLK_CTRL.hirc_en = 1).*
14:6	-	R/W	0	**Reserved for Future Use** Do not modify this field.
5	rtcwk_en	R/W	0	**RTC Alarm Wakeup Enable** When this field is set to 1, If the RTC is configured to generate a wakeup alarm, an RTC wakeup event causes the MAX32660 to exit all low power modes and transition directly to *ACTIVE* mode. Refer to section *9.2.3 RTC Wakeup From DEEPSLEEP/BACKUP Power Modes* for details on enabling the RTC as a wakeup source. 0: Wakeup from RTC disabled, regardless of the RTC alarm configuration. 1: Wakeup from RTC alarm enabled.
4	gpiowk_en	R/W	0	**GPIO Wakeup Enable** When enabled, activity on any GPIO pin configured for wakeup causes an exit from *SLEEP* and *DEEPSLEEP* low power modes and transitions directly to *ACTIVE* mode. 0: Wakeup from GPIO disabled, regardless of the GPIO wakeup configuration. 1: Wakeup from GPIO enabled. *Note: Refer to section 6.4.2: Using GPIO for Wakeup from Low Power Modes for additional details.*
3	-	RO	0	**Reserved for Future Use** Do not modify this field.
2:0	mode	R/W	0	**Operating Mode** Configures the current operating mode for the device. 0: *ACTIVE* mode 1: Reserved for Future Use 2: Reserved for Future Use 3: Reserved for Future Use 4: *BACKUP* Low Power Mode 5: Reserved for Future Use 6: Shutdown Mode *Note: All other values are Reserved for Future Use.*

Table 4-17: Peripheral Clock Disable 0 Register

Peripheral Clocks Disable 0 Register			GCR_PCLK_DIS0	[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
31:29	-	R/W	0	**Reserved for Future Use** Do not modify this field.

Peripheral Clocks Disable 0 Register		GCR_PCLK_DIS0		[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
28	i2c1d	R/W	0	**I2C1 Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable, set to 0 to enable. 0: Peripheral Enabled 1: Peripheral Disabled
27:18	-	R/W	0	**Reserved for Future Use** Do not modify this field.
17	timer2d	R/W	0	**Timer2 Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable, set to 0 to enable. 0: Peripheral Enabled 1: Peripheral Disabled
16	timer1d	R/W	0	**Timer1 Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable, set to 0 to enable. 0: Peripheral Enabled 1: Peripheral Disabled
15	timer0d	R/W	0	**Timer0 Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable, set to 0 to enable. 0: Peripheral Enabled 1: Peripheral Disabled
14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13	i2c0d	R/W	0	**I2C0 Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable, set to 0 to enable. 0: Peripheral Enabled 1: Peripheral Disabled
12:11	-	RO	-	**Reserved for Future Use** Do not modify this field.
10	uart1d	R/W	0	**UART1 Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable, set to 0 to enable. 0: Peripheral Enabled 1: Peripheral Disabled

Peripheral Clocks Disable 0 Register			GCR_PCLK_DIS0	[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
9	uart0d	R/W	0	**UART0 Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable, set to 0 to enable. 0: Peripheral Enabled 1: Peripheral Disabled
8	-	R/W	0	**Reserved for Future Use** Do not modify this field.
7	spi1d	R/W	0	**SPI1 Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable, set to 0 to enable. 0: Peripheral Enabled 1: Peripheral Disabled
6	spi0d	R/W	0	**SPI0 Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable, set to 0 to enable. 0: Peripheral Enabled 1: Peripheral Disabled
5	dmad	R/W	0	**Standard DMA Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable, set to 0 to enable. 0: Peripheral Enabled 1: Peripheral Disabled
4:1	-	R/W	0	**Reserved for Future Use** Do not modify this field.
0	gpio0d	R/W	0	**GPIO0 Port and Pad Logic Clock Disable** Write 1 to disable, set to 0 to enable. Disabling the GPIO Port and Pad Logic gates off the clock from the GPIO Port and the individual GPIO pads. 0: Peripheral Enabled 1: Peripheral Disabled

Table 4-18: Memory Clock Control Register

Memory Clock Control Register			GCR_MEM_CTRL	[0x0028]
Bits	**Name**	**Access**	**Reset**	**Description**
31:13	-	R/W	0	**Reserved for Future Use** Do not modify this field.

Memory Clock Control Register			GCR_MEM_CTRL	[0x0028]
Bits	**Name**	**Access**	**Reset**	**Description**
12	icache_ret	R/W	0	**ICC0 Cache RAM Light Sleep Enable** Set this field to 1 to enable Light Sleep mode for the Internal Cache Controller's 16KB RAM. Setting this field to 1 retains the state of the cache but the cache is not accessible. This is useful if executing code from RAM or while in *SLEEP* mode operation. 0: ICC0 Cache RAM is active. 1: ICC0 Cache RAM is in Light Sleep mode. *Note: Any reset event that results in a Cache RAM reset will reset the Cache RAM regardless of the state of this field.*
11	ram3_ls		0	**System RAM 3 Light Sleep Enable** Set this field to 1 to enable Light Sleep for System RAM 3 (0x2001 0000 - 0x2001 7FFF). In Light Sleep mode, the System RAM contents are retained but the Data Memory cannot be read. 0: System RAM 3 retention is not enabled. 1: System RAM 3 is in Light Sleep mode. *Note: Any reset event that results in RAM reset will reset the RAM regardless of the state of this field.* *Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the register PWRSEQ_LPMEMSD "Low Power Mode RAM Shut Down Control".*
10	ram2_ls		0	**System RAM 2 Data Retention Enable** Set this field to 1 to enable Light Sleep mode for System RAM 2 (0x2000 8000 - 0x2000 FFFF). In Light Sleep mode, the System RAM contents are retained but the Data Memory cannot be read. 0: System RAM 2 retention is not enabled 1: System RAM 2 is in Light Sleep mode. *Note: Any reset event that results in RAM reset will reset the RAM regardless of the state of this field.* *Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the register PWRSEQ_LPMEMSD "Low Power Mode RAM Shut Down Control".*
9	ram1_ls		0	**System RAM 1 Light Sleep Enable** Set this field to 1 to enable Light Sleep for System RAM 1 (0x2000 4000 - 0x2000 7FFF). If light sleep is enabled, the System RAM contents are retained but the Data Memory cannot be read. 0: System RAM 1 retention is not enabled 1: System RAM 1 data retention is enabled for *BACKUP* mode. *Note: Any reset event that results in RAM reset will reset the RAM regardless of the state of this field.* *Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the register PWRSEQ_LPMEMSD "Low Power Mode RAM Shut Down Control".*

Memory Clock Control Register			GCR_MEM_CTRL	[0x0028]
Bits	**Name**	**Access**	**Reset**	**Description**
8	ram0_ls		0	**System RAM 0 Light Sleep Enable** Set this field to 1 to enable Light Sleep for System RAM 0 (0x2000 0000 - 0x2000 3FFF). If light sleep is enabled, the System RAM contents are retained but the Data Memory cannot be read. 0: System RAM 0 is Active. 1: System RAM 0 is in Light Sleep mode. *Note: Any reset event that results in RAM reset will reset the RAM regardless of the state of this field.* *Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the register PWRSEQ_LPMEMSD "Low Power Mode RAM Shut Down Control".*
7:3	-	R/W	0	**Reserved for Future Use** Do not modify this field.
2:0	fws	R/W	5	**Flash Wait States** Number of wait-states (system clock cycles) for internal flash read access. Refer to section *Flash Wait States* for additional information. 0: Reserved for Future Use. 1: 1 Wait State (1 System Clock) 2: 2 Wait States 3: 3 Wait States 4: 4 Wait States (Minimum value for HFIO=System Clock=96MHz) 5: 5 Wait States (Reset default) 6: 6 Wait States 7: 7 Wait States

Table 4-19: Memory Zeroization Control Register

Memory Zeroization Control Register			GCR_MEM_ZCTRL	[0x002C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved for Future Use** Do not modify this field.
1	icache_zero	R/W1O	0	**Internal Cache Data and Tag RAM Zeroization** Write 1 to clear the Internal Cache Controller's 16KB data cache and the associated Tag RAM. The bit is set to 0 by hardware when the operation is complete. 0: Operation complete 1: Zeroize memory
0	sram_zero	R/W1O	0	**System Data RAM Zeroization** Write 1 to clear the contents of the Internal Data RAM, all ranges. The bit is set to 0 by hardware when the operation is complete. 0: Operation complete 1: Zeroize memory

Table 4-20: System Status Flag Register

System Status Flag Register			GCR_SYS_STAT	[0x0040]
Bits	**Name**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved for Future Use** Do not modify this field.

System Status Flag Register			GCR_SYS_STAT	[0x0040]
Bits	**Name**	**Access**	**Reset**	**Description**
0	icelock	RO	0	**Arm Cortex-M4 with FPU ICE Lock Status Flag** This field is set in the factory and if set to 1 disables JTAG SWD access to the device. 0: Arm ICE is unlocked and JTAG debug may be enabled using *GCR_SCON.swd_dis* field. 1: Arm ICE is locked (disabled), JTAG SWD is disabled and the *GCR_SCON.swd_dis* is read-only.

Table 4-21: Reset Register 1

Reset Register 1			GCR_RST1	[0x0044]
Bits	**Name**	**Access**	**Reset**	**Description**
31:1	-	R/W1O	0	**Reserved for Future Use** Do not modify this field.
0	i2c1	R/W1O	0	**I2C1 Reset** Write 1 to reset the peripheral state to the reset default state. When complete this field will read 0. 0: I2C1 peripheral not in reset. 1: Write 1 to reset the I2C1 peripheral.

Table 4-22: Peripheral Clock Disable Register 1

Peripheral Clock Disable Register 1			GCR_PCLK_DIS1	[0x0048]
Bits	**Name**	**Access**	**Reset**	**Description**
31:12	-	R/W	0	**Reserved for Future Use** Do not modify this field.
11	iccd	R/W	0	**ICC Clock Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable the icc clock disable. 0: ICC Clock Enabled 1: ICC Clock Disabled.
3	flcd	R/W	0	**Flash Controller Disable** Setting this field disables the APB clock to this peripheral. When the clock is disabled, the peripheral power consumption is reduced, and the peripheral is disabled. Disabling the clock to the peripheral does not affect the peripheral's registers. Write 1 to disable the clock to the Flash Controller. 0: Flash Controller Clock Enabled 1: Flash Controller Clock Disabled. *Note: Disabling the clock to the Flash Controller prevents code fetches from the flash. Care should be taken to ensure code execution is performed from RAM while the Flash Controller peripheral clock is disabled.*
2:0	-	R/W	0	**Reserved for Future Use** Do not modify this field.

Table 4-23: Event Enable Register

Event Enable Register			GCR_EVTEN	[0x004C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:2	-	RO	-	**Reserved for Future Use** Do not modify this field.
1	rx_evt	R/W	0	**RX Event Enabled** Set this field to 1 to enable generation of an RXEV event to wake the CPU from a Wait for Event (WFE) sleep state. 0: RX Event is disable. 1: RX Event is enabled.
0	dmaevent	R/W	0	**DMA CTZ Event Wake-Up Enable** When set, when a DMA block transfer is completed and the DMA counter *DMAn_CNT.cnt* = 0, a CTZ DMA event occurs which generates an RXEV to wake-up the device from a low power mode entered with a WFE instruction.

Table 4-24: Revision Register

Revision Register			GCR_REV	[0x0050]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	RO	-	**Reserved for Future Use** Do not modify this field.
15:0	revision	RO	N/A	**Maxim Integrated Chip Revision** This field reads the chip revision id (A1), ascii encoded. Revision 'A1': 0x4131

Table 4-25: System Status Interrupt Enable Register

System Status Interrupt Enable			GCR_SYS_IE	[0x0054]
Bits	**Name**	**Access**	**Reset**	**Description**
31:1	-	RO	-	**Reserved for Future Use** Do not modify this field.
0	iceulie	R/W	0	**Arm ICE Unlocked Interrupt Enable** Set this bit to enable a PWRSEQ IRQ if the Arm ICE is unlocked. 0: Interrupt disabled 1: Interrupt enabled

4.12 System Initialization Registers

The SIR base peripheral address is 0x4000 0400. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 4-26: System Initialization Registers, Offsets and Descriptions

Offset	Register Name	Access	Description
[0x0000]	*SIR_STAT*	RO	System Initialization Status Register
[0x0004]	*SIR_ADDR_ER*	RO	System Initialization Address Error Register

4.12.1 System Initialization Register Details

Table 4-27: Function Control Register 0

System Initialization Status Register			SIR_STAT	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:2	-	RO	-	**Reserved for Future Use** Do not modify this field.
1	cfg_err	RO	See Description	**Configuration Error Flag** This field is set by hardware during reset if an error in the device configuration is detected. 0: Configuration valid. 1: Configuration invalid. *Note: If this field reads 1 a device error has occurred. Please contact Maxim Integrated technical support for additional assistance providing the address contained in SIR_ADDR_ER.addr.*
0	cfg_valid	RO	See Description	**Configuration Valid Flag** This field is set to 1 by hardware during reset if the device configuration is valid. 0: Configuration Invalid 1: Configuration Valid *Note: If this field reads 0 the device configuration is invalid and a device error has occurred. Please contact Maxim Integrated technical support for additional assistance.*

Table 4-28: System Initialization Address Error Register

System Initialization Status Register			SIR_ADDR_ER	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	addr	RO	0	**Configuration Error Address** If the *SIR_STAT.cfg_err* field is set to 1, the value in this register is the address of the configuration failure.

4.13 Function Control Registers

The FCR base peripheral address is 0x4000 0800. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 4-29: Function Control Registers, Offsets and Descriptions

Offset	Register Name	Access	Description
[0x0000]	*FCR_REG0*	R/W	Function Control Register 0

4.13.1 Function Control Register Details

Table 4-30: Function Control Register 0

Function Control Register 0			FCR_REG0	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:24	-	RO	0	**Reserved for Future Use** Do not modify this field.

Function Control Register 0		FCR_REG0		[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
23	i2c1_scl_filter_en	R/W	0	**I2C1 SCL Filter Enable** 0: Filter disabled 1: Filter enabled
22	i2c1_sda_filter_en	R/W	0	**I2C1 SDA Filter Enable** 0: Filter disabled 1: Filter enabled
21	i2c0_scl_filter_en	R/W	0	**I2C0 SCL Filter Enable** 0: Filter disabled 1: Filter enabled
20	i2c0_sda_filter_en	R/W	0	**I2C0 SDA Filter Enable** 0: Filter disabled 1: Filter enabled
19:0	-	R/W	-	**Reserved for Future Use** Do not modify this field.

4.14 Power Supply Monitoring

MAX32660 has a power monitor that monitors the external supply voltages during operation. The following power supplies are monitored:

- V_{CORE} (VCORE) Supply Voltage, CPU Core
- V_{DD} (VDD) Supply Voltage

Each of these supplies has a dedicated power monitor setting in the Power Sequencer Low Power Voltage Control Register, *PWRSEQ_LP_CTRL*. When the corresponding power monitor is enabled, the input voltage pin is constantly monitored. If the voltage drops below the trigger threshold, all registers and peripherals in that power domain are reset. This improves reliability and safety by guarding against a low voltage condition corrupting the contents of the registers and the device state. Disabling a power monitor risks data corruption of internal registers and corruption of the device state should the input voltage drop below the safe minimum value.

V_{CORE} has a power fail monitor. When enabled, if the power supply drops below the power fail reset voltage the entire device goes into a Power-On Reset.

Refer to the MAX32660 datasheet for the trigger threshold value and power fail reset voltage. When the power supply monitor is tripped, a Power Fail Warning Interrupt is triggered.

4.15 Power Sequencer Registers

The PWRSEQ base peripheral address is 0x4000 6800. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 4-31: Power Sequencer Low Power Control Registers, Offsets, Access and Descriptions

Offset	Register Name	Access	Reset	Description
[0x0000]	*PWRSEQ_LP_CTRL*	R/W	POR	Low Power Voltage Control Register
[0x0004]	*PWRSEQ_LP_WAKEFL*	R/W	POR	Low Power Mode Wakeup Flags for GPIO0
[0x0008]	*PWRSEQ_LPWK_EN*	R/W	POR	GPIO0 Wakeup Enable
[0x0040]	*PWRSEQ_LPMEMSD*	R/W	POR	RAM Shut Down Control

4.15.1 Power Sequencer Register Details

Table 4-32: Low Power Voltage Control Register

Low Power Voltage Control Register		PWRSEQ_LP_CTRL		[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:26	-	R/W	0	**Reserved for Future Use** Do not modify this field.
25	vddio_por_dis	R/W	0	**V_{DDIO} Power-On-Reset Monitor Disable** Set this field to 1 to disable the V_{DDIO} POR monitor. 0: V_{DDIO} POR Enabled 1: V_{DDIO} POR Disabled
24:21	-	R/W	0	**Reserved for Future Use** Do not modify this field.
20	vcore_svm_dis	R/W	0	**V_{CORE} Supply Voltage Monitor Disable** Set this field to 1 to disable the V_{CORE} SVM. 0: V_{CORE} SVM Enabled 1: V_{CORE} SVM Disabled
19:17	-	R/W	0	**Reserved for Future Use** Do not modify this field.
16	ldo_dis	R/W	See Description	**LDO Disable** This field initializes to 1 on a Power-On Reset until the hardware determines if an external power source is connected to the V_{CORE} pin. If no power supply is connected, this bit is set to 0 by the hardware. If a power supply is connected to V_{CORE}, the bit remains set to 1. Set this field to 1 to manually disable the LDO. 0: LDO Enabled. 1: LDO Disabled. Default after a POR.
15:13	-	R/W	0	**Reserved for Future Use** Do not modify this field.
12	vcore_por_dis	R/W	1	**V_{CORE} POR Disable for *DEEPSLEEP* and *BACKUP* Mode** Setting this bit to 1 blocks the Power-On-Reset signal to the core when the device is in *DEEPSLEEP* and *BACKUP* mode operation. Disconnecting the POR signal from the core during *DEEPSLEEP* and *BACKUP* modes prevents the core from detecting a POR event while the device is in *DEEPSLEEP* or *BACKUP* mode. 0: POR signal is connected to the core during *DEEPSLEEP* and *BACKUP* mode. 1: POR signal is not connected to the core during *DEEPSLEEP* and *BACKUP* mode.
11	bg_off	R/W	1	**Band Gap Disable for *DEEPSLEEP* and *BACKUP* Mode** Setting this field to 1 powers off the Bandgap during *DEEPSLEEP* and *BACKUP* mode. 0: System Bandgap (SVM) is on in *DEEPSLEEP* and *BACKUP* modes 1: System Bandgap (SVM) is off in *DEEPSLEEP* and *BACKUP* modes.
10	fast_wk_en	R/W	0	**Fast Wakeup Enable for *DEEPSLEEP* Mode** Set to 1 to enable fast wakeup from *DEEPSLEEP* mode. When enabled, the system exits *DEEPSLEEP* mode faster by: • Bypassing the 8kHz RO warmup • Reducing the warmup time for the High-Frequency Internal Oscillator. • Reducing the warmup time for the LDO. 0: Fast Wakeup Mode Disabled 1: Fast Wakeup Mode Enabled

Low Power Voltage Control Register		PWRSEQ_LP_CTRL		[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
9	-	R/W	0	**Reserved for Future Use** Do not modify this field.
8	retreg_en	R/W	1	**RAM Retention Regulator Enable for *BACKUP* Mode** This field selects the source used to retain the RAM contents during *BACKUP* mode operation. Setting this field to 0 sets the V_{DD} supply for RAM retention during *BACKUP* mode and disables the RAM retention regulator. 0: RAM retention regulator disabled, the V_{DD} supply is used to retain the state of the internal SRAM as configured by the *PWRSEQ_LP_CTRL.ramret_sel[3:0]* fields. 1: RAM retention regulator enabled. RAM retention in *BACKUP* mode is configured with the *PWRSEQ_LP_CTRL.ramret_sel[3:0]* fields.
7	-	R/W	0	**Reserved for Future Use** Do not modify this field.
6	vcore_det_bypass	R/W	0	**Bypass V_{CORE} External Supply Detection** Set this field to 1 if the system runs from a single supply only and V_{CORE} is not connected to an external supply. Bypassing the hardware detection of an external supply on V_{CORE} enables a faster wakeup time. 0: V_{CORE} External Supply Detection Enabled. 1: V_{CORE} External Supply Detection Disabled.
5-4	ovr	R/W	2	**Output Voltage Range for Internal Regulator** Set these bits to control the output voltage of the internal regulator allowing selection of the internal core operating voltage and the frequency of the internal high frequency internal oscillator. On Power-On-Reset, this field defaults to 1.1V output ± 10% with the $f_{INT\_CLK} = 96MHz$. ***Note: If V_{CORE} is connected to an external supply voltage, this field should be modified only to set it to match the input voltage on V_{CORE}.*** **Dual Supply Operation:** 0b11: Reserved for Future Use 0b10: V_{CORE} = 1.1V, $f_{INTCLK}=96MHz$ 0b01: V_{CORE} = 1.0V, $f_{INTCLK}=48MHz$ 0b00: V_{CORE} = 0.9V, $f_{INTCLK}=24MHz$ **Single Supply Operation (V_{CORE}=GND)** 0b11: Reserved for Future Use 0b10: V_{LDO} = 1.1V, $f_{INTCLK}=96MHz$ 0b01: V_{LDO} = 1.0V, $f_{INTCLK}=48MHz$ 0b00: V_{LDO} = 0.9V, $f_{INTCLK}=24MHz$
3	ramret_sel3	R/W	0	**System RAM 3 Data Retention Enable for *BACKUP* Mode** Set this field to 1 to enable Data Retention for System RAM 3, address range of 0x2001 0000 to 0x2001 7FFF. 0: Data retention for System RAM 3 address space disabled in *BACKUP* mode. 1: Data retention for System RAM 3 address space enabled in *BACKUP* mode.
2	ramret_sel2	R/W	0	**System RAM 2 Data Retention Enable for *BACKUP* Mode** Set this field to 1 to enable Data Retention for System RAM 2, address range of 0x2000 8000 to 0x2000 FFFF. 0: Data retention for System RAM 2 address space disabled in *BACKUP* mode. 1: Data retention for System RAM 2 address space enabled in *BACKUP* mode.
1	ramret_sel1	R/W	0	**System RAM 1 Data Retention Enable for *BACKUP* Mode** Set this field to 1 to enable Data Retention for System RAM 1, address range of 0x2000 4000 to 0x2000 7FFF. 0: Data retention for System RAM 1 address space disabled in *BACKUP* mode. 1: Data retention for System RAM 1 address space enabled in *BACKUP* mode.

Low Power Voltage Control Register			PWRSEQ_LP_CTRL	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
0	ramret_sel0	R/W	0	**System RAM 0 Data Retention Enable for *BACKUP* Mode** Set this field to 1 to enable Data Retention for System RAM 0, address range of 0x2000 0000 to 0x2000 3FFF. 0: Data retention for System RAM 0 address space disabled in *BACKUP* mode. 1: Data retention for System RAM 0 address space enabled in *BACKUP* mode.

Table 4-33: Low Power Mode Wakeup Flags for GPIO0

Low Power Mode GPIO Wakeup Flags Register			PWRSEQ_LP_WAKEFL	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W1C	0	**Reserved for Future Use** Do not modify this field.
13:0	wakest	R/W1C	0	**GPIO Pin Wakeup Status Flag** When a GPIO pin transitions from low-to-high or high-to-low, the corresponding bit in this field is set. If the corresponding interrupt enable bit is set in *PWRSEQ_LPWK_EN* register and *GCR_PM.gpiowk_en* bit is set to 1, a PWRSEQ IRQ is generated to wake up the device from all low power modes to *ACTIVE* mode. *Note: To enable the device to wake up from a low power mode on a GPIO pin transition, first set the GCR_PM GPIO wakeup enable field to 1 (GCR_PM.gpiowk_en = 1).*

Table 4-34: Low Power Wakeup Enable for GPIO0 Register

Low Power Mode Wakeup Enable for GPIO0			PWRSEQ_LPWK_EN	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:0	wakeen	R/W	0	**GPIO Pin Wakeup Interrupt Enable** Write 1 to a bit to enable the corresponding GPIO0 pin to generate a PWRSEQ IRQ to wake up the device from any low power mode to *ACTIVE* mode. Set the *GCR_PM.gpiowk_en* bit to 1 to enable GPIO wake up events. A wake up occurs on any low-to-high or high-to-low transition on the corresponding GPIO0 pin. *Note: To enable the device to wake up from a low power mode on a GPIO pin transition, first set the global GPIO wakeup enable field, (GCR_PM.gpiowk_en = 1).*

Table 4-35: RAM Shut Down Register

Low-Power Memory Shutdown Register			PWRSEQ_LPMEMSD	[0x0040]
Bits	**Name**	**Access**	**Reset**	**Description**
31:4	-	RO	-	**Reserved for Future Use** Do not modify this field.
3	sram3_off	R/W	0	**System RAM 3 (0x2001 0000 - 0x2001 7FFF) Shut Down** Write 1 to shut down power to System RAM 3 memory range. 0: System RAM 3 Powered On (Enabled) 1: System RAM 3 Powered Off (Disabled)

Low-Power Memory Shutdown Register			PWRSEQ_LPMEMSD	[0x0040]
Bits	**Name**	**Access**	**Reset**	**Description**
2	sram2_off	R/W	0	**System RAM 2 (0x2000 7FFF - 0x2000 FFFF) Shut Down** Write 1 to shut down power to System RAM 2 memory range. 0: System RAM 2 Powered On (Enabled) 1: System RAM 2 Powered Off (Disabled)
1	sram1_off	R/W	0	**System RAM 1 (0x2000 3FFF - 0x2000 7FFF) Shut Down** Write 1 to shut down power to System RAM 1 memory range. 0: System RAM 1 Powered On (Enabled) 1: System RAM 1 Powered Off (Disabled)
0	sram0_off	R/W	0	**System RAM 0 (0x2000 0000 – 0x2000 3FFF) Shut Down** Write 1 to shut down power to System RAM 0 memory range. 0: System RAM 0 Powered On (Enabled) 1: System RAM 0 Powered Off (Disabled)

5 Flash Controller

The MAX32660's Flash Controller is a peripheral that manages read, write, and erase accesses to the internal flash.

5.1.1 Features

- Up to 256KB total internal flash memory
  - 32 pages
  - 8,192 bytes per page
  - 2,048 words by 128 bits per page
- 128-bit data reads
- 32-bit or 128-bit write support
- Page erase and mass erase support
- Write Protection

5.2 Overview

The MAX32660 contains 256KB of internal flash memory for storing user application and data. The internal flash memory is programmable via the JTAG debug interface (in-system) or directly with user application code (in-application).

The flash is organized as an array of pages. Each page is 8,192 bytes per page. *Table 5-1*, *below*, shows the start address and end address for the internal flash memory. The internal flash memory is mapped with a start address of 0x0000 0000 and an end address of 0x0003 FFFF for a total of 256KB.

Table 5-1: Internal Flash Memory Organization

Page Number	Size in Bytes	Start Address	End Address
1	8,192	0x0000 0000	0x0000 1FFF
2	8,192	0x0000 2000	0x0000 3FFF
3	8,192	0x0000 4000	0x0000 5FFF
4	8,192	0x0000 6000	0x0000 7FFF
5	8,192	0x0000 8000	0x0000 9FFF
…	…	…	…
8	8,192	0x0000 E000	0x0000 FFFF
9	8,192	0x0001 0000	0x0001 1FFF
…	…	…	…
31	8,192	0x0003 C000	0x0003 DFFF
32	8,192	0x0003 E000	0x0003 FFFF

5.3 Usage

The Flash Controller manages write and erase operations for internal flash memory and provides a lock mechanism to prevent unintentional writes to the internal flash. In-application and in-system programming, page erase and mass erase operations are supported.

5.3.1 Clock Configuration

The Flash Controller requires a 1MHz peripheral clock for operation. The input clock to the Flash Controller block is the system clock, f_{SYSCLK}. Use the Flash Controller clock divisor to generate $f_{FLC\_CLK} = 1MHz$, as shown in *Equation 5-1*,

below. For the 96MHz Relaxation Oscillator as the system clock, the *FLC_CLKDIV.clkdiv* field should be set to 96 (0x60). If another clock source is set as the system clock, this field must be adjusted to meet the target 1MHz for f_{FLC_CLK}.

Equation 5-1: Flash Controller Clock Frequency

$$f_{FLC\_CLK} = \frac{f_{SYSCLK}}{FLC\_CLKDIV.clkdiv} = 1MHz$$

5.3.2 Lock Protection

The Flash Controller provides a locking mechanism to prevent accidental writes and erases. Each write or erase requires the *FLC_CTRL.unlock* field be set to 0x2 prior to starting the operation. Writing any other value to the *FLC_CTRL.unlock* field results in the flash remaining locked.

Note: If a write, page erase or mass erase operation is started and the unlock code was not set to 0x2, the Flash Controller hardware sets the access fail flag, FLC_INTR.access_fail, to indicate an access violation occurred.

5.3.3 Flash Write Width

The Flash Controller supports write widths of either 32-bits or 128-bits. Selection of the flash write width is controlled with the *FLC_CTRL.width* field and defaults to 128-bit width on all forms of reset. Setting *FLC_CTRL.width* to 1 selects 32-bit write widths.

In 128-bit width mode, the target address bits *FLC_ADDR*[3:0] are ignored resulting in 128-bit alignment. In 32-bit width mode, the target address bits *FLC_ADDR*[1:0] are ignored for 32-bit address alignment. If the desired target address is not 128-bit aligned (*FLC_ADDR*[3:2] ≠ 0), 32-bit width mode is required.

Table 5-2: Valid Addresses for 32-bit and 128-bit Internal Flash Writes

	FLC_ADDR[31:0]																															
Bit Number	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
32-bit Write	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0
128-bit Write	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0

5.3.4 Flash Write

Perform the following steps to write to the internal flash memory:

1. If desired, enable Flash Controller interrupts by setting the *FLC_INTR.access_fail_ie* and *FLC_INTR.done_ie* bits.
2. Set the write field, *FLC_CTRL.width*, as described in *Flash Write Width*.
3. Set the *FLC_ADDR* register to a valid target address. Reference *Table 5-2*.
4. Set the data register or registers.
5. For 32-bit write width, set *FLC_DATA0* to the data to write.
6. For 128-bit write width, set *FLC_DATA3*, *FLC_DATA2*, *FLC_DATA1*, and *FLC_DATA0* to the data to write. *FLC_DATA3* is the most significant word and *FLC_DATA0* is the least significant word.
7. Set *FLC_CTRL.unlock* to 0x2 to unlock the internal flash.
8. Read the *FLC_CTRL.busy* bit until it returns 0.
9. Start the flash write, set *FLC_CTRL.write* to 1 and this field is automatically cleared by the Flash Controller when the write operation is finished.
10. *FLC_INTR.done* is set by hardware when the write completes and if an error occurred, the *FLC_INTR.access_fail* flag is set. These bits generate a flash IRQ if the interrupt enable bits are set.

5.3.5 Page Erase

Perform the following to erase a page of internal flash memory:

1. If desired, enable Flash Controller interrupts by setting the *FLC_INTR.access_fail_ie* and *FLC_INTR.done_ie* bits.
2. Set the *FLC_ADDR* register to a page address to erase. *FLC_ADDR*[12:0] are ignored by the Flash Controller to ensure the address is page aligned. Refer to Table 5-3 for the valid page aligned addresses for the internal flash memory.
3. Set *FLC_CTRL.unlock* to 0x2 to unlock the internal flash.
4. Read the *FLC_CTRL.busy* bit until it returns 0.
5. Set *FLC_CTRL.erase_code* to 0x55 for page erase.
6. Set *FLC_CTRL.page_erase* to 1 to start the page erase operation.
7. The *FLC_CTRL.busy* bit is set by the Flash Controller while the page erase is in progress and the *FLC_CTRL.page_erase* and *FLC_CTRL.busy* are cleared by the Flash Controller when the page erase is complete.
8. *FLC_INTR.done* is set by hardware when the page erase completes and if an error occurred, the *FLC_INTR.access_fail* flag is set. These bits generate a flash IRQ if the interrupt enable bits are set.

Table 5-3: Page Boundary Address Range for Page Erase Operations

	FLC_ADDR[31:0]																															
Bit Number	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Page Aligned Address	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0	0

5.3.6 Mass Erase

Mass erase clears the internal flash memory. This operation requires the JTAG debug port to be enabled to perform the operation. If the JTAG debug port is not enabled a mass erase operation cannot be performed. Perform the following steps to mass erase the internal flash:

1. Set *FLC_CTRL.unlock* to 0x2 to unlock the internal flash.
2. Read the *FLC_CTRL.busy* bit until it returns 0.
3. Set *FLC_CTRL.erase_code* to 0xAA for mass erase.
4. Set *FLC_CTRL.mass_erase* to 1 to start the mass erase operation.
5. The *FLC_CTRL.busy* bit is set by the Flash Controller while the mass erase is in progress and the *FLC_CTRL.mass_erase* and *FLC_CTRL.busy* are cleared by the Flash Controller when the mass erase is complete.
6. *FLC_INTR.done* is set by the Flash Controller when the mass erase completes. If an error occurred, the *FLC_INTR.access_fail* flag is set. These bits generate a Flash Controller IRQ if the interrupt enable bits are set.

Note: Mass erase requires the JTAG debug port to be enabled, if the JTAG debug port is disabled on the device an access fail error is generated (FLC_INTR.access_fail = 1).

5.4 Flash Controller Registers

The FLC base peripheral address is 0x4002 9000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 5-4: Flash Controller Registers, Offsets, Access and Descriptions

Offset	Register Name	Access	Description
[0x0000]	*FLC_ADDR*	R/W	Flash Controller Address Pointer Register
[0x0004]	*FLC_CLKDIV*	R/W	Flash Controller Clock Divisor Register
[0x0008]	*FLC_CTRL*	R/W	Flash Controller Control Register
[0x0024]	*FLC_INTR*	R/W1C	Flash Controller Interrupt Register
[0x0030]	*FLC_DATA0*	R/W	Flash Controller Data Register 0
[0x0034]	*FLC_DATA1*	R/W	Flash Controller Data Register 1
[0x0038]	*FLC_DATA2*	R/W	Flash Controller Data Register 2
[0x003C]	*FLC_DATA3*	R/W	Flash Controller Data Register 3

Table 5-3. Flash Controller Address Pointer Register

Flash Address Register		FLC_ADDR		[0x00]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	addr	R/W	0	**Flash Address** This field contains the target address for a write operation. A valid internal flash memory address is required for all write operations. The reset default is always address 0x00000000.

Table 5-4. Flash Controller Clock Divisor Register

Flash Controller Clock Divisor Register		FLC_CLKDIV		[0x04]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	RO	-	**Reserved for Future Use** Do not modify this field.

Flash Controller Clock Divisor Register		FLC_CLKDIV		[0x04]
Bits	**Name**	**Access**	**Reset**	**Description**
7:0	clkdiv	R/W	0x60	**Flash Controller Clock Divisor** The system clock is divided by the value in this field to generate the FLC peripheral clock, f_{FLC_CLK}. The FLC peripheral clock must equal 1MHz. The default on all forms of reset is 96 (0x60), resulting in f_{FLC_CLK} = 1MHz. If the OVR is changed, this field must be updated to match the divisor for the HFIO oscillator to achieve f_{FLC_CLK} = 1MHz.

Table 5-5. Flash Controller Control Register

Flash Controller Control Register		FLC_CTRL		[0x08]
Bits	**Name**	**Access**	**Reset**	**Description**
31:28	unlock_code	R/W	0	**Flash Unlock** Write the unlock code, 0x2, prior to any flash write or erase operation to unlock the flash. Writing any other value to this field locks the internal flash. 0x2: flash unlock code
27:26	-	RO	-	**Reserved for Future Use** Do not modify this field.
25	lve	R/W	1	**Low Voltage Enable** Set this field to 1 to enable low voltage operation for the flash memory. Refer to section *4.1 Core Operating Voltage Range Selection* for detailed usage information on this setting. 0: Low voltage operation disabled (Default). 1: Low voltage operation enabled. *Note: The PWRSEQ_LP_CTRL.ovr field must be set to 0b00 prior to setting this field to 1.*
24	busy	RO	0	**Flash Busy Flag** When this field is set, writes to all flash registers are ignored except for the *FLC_INTR* register. 0: flash idle 1: flash busy *Note: If the Flash Controller is busy (FLC_CTRL.busy = 1), reads, writes and erase operations are not allowed and result in an access failure (FLC_INTR.access_fail = 1).*
23:16	-	RO	0	**Reserved for Future Use** Do not modify this field.
15:8	erase_code	R/W	0	**Erase Code** Prior to an erase operation this field must be set to 0x55 for a page erase or 0xAA for a mass erase. The flash must be unlocked prior to setting the erase code. This field is automatically cleared after the erase operation is complete. 0x00: Erase disabled. 0x55: Page erase code. 0xAA: Enable mass erase via the JTAG debug port.
7:5	-	R/W	0	**Reserved for Future Use** Do not modify this field.
4	width	R/W	0	**Data Width Select** This field sets the data width of a write to the flash page. The Flash Controller supports either 32-bit wide writes or 128-bit wide writes. 0: 128-bit transactions (*FLC_DATA3* - *FLC_DATA0*) 1: 32-bit transactions (*FLC_DATA0* only)

Flash Controller Control Register		FLC_CTRL		[0x08]
Bits	**Name**	**Access**	**Reset**	**Description**
3	-	R/W	0	**Reserved for Future Use** Do not modify this field.
2	page_erase	R/W1O	0	**Page Erase** Write a 1 to this field to initiate a page erase at the address in *FLC_ADDR.addr.* The flash must be unlocked prior to attempting a page erase, see *FLC_CTRL.unlock* for details. The Flash Controller hardware clears this bit when a page erase operation is complete. 0: No page erase operation in process or page erase is complete. 1: Write a 1 to initiate a page erase. If this field reads 1, a page erase operation is in progress. *Note: This field is protected and cannot be set to 0 by application code.*
1	mass_erase	R/W1O	0	**Mass Erase** Write a 1 to this field to initiate a mass erase of the internal flash memory. The flash must be unlocked prior to attempting a mass erase, see *FLC_CTRL.unlock* for details. The Flash Controller hardware clears this bit when the mass erase operation completes. 0: No operation 1: Initiate mass erase *Note: This field is protected and cannot be set to 0 by application code.*
0	write	R/W1O	0	**Write** If this field reads 0, no write operation is pending for the flash. To initiate a write operation, set this bit to 1 and the Flash Controller will write to the address set in the *FLC_ADDR* register. 0: No write operation in process or write operation complete. 1: Write 1 to initiate a write operation. If this field reads 1, a write operation is in progress. *Note: This field is protected and cannot be set to 0 by application code.*

Table 5-5: Flash Controller Interrupt Register

Flash Controller Interrupt Register		FLC_INTR		[0x24]
Bits	**Name**	**Access**	**Reset**	**Description**
31:10	-	R/W	0	**Reserved for Future Use** Do not modify this field.
9	access_fail_ie	R/W	0	**Flash Access Fail Interrupt Enable** Set this bit to 1 to enable interrupts on flash access failures. 0: Disabled 1: Enabled
8	done_ie	R/W	0	**Flash Operation Complete Interrupt Enable** Set this bit to 1 to enable interrupts on flash operations complete. 0: Disabled 1: Enabled
7:2	-	RO	0	**Reserved for Future Use** Do not modify this field.

Flash Controller Interrupt Register		FLC_INTR		[0x24]
Bits	**Name**	**Access**	**Reset**	**Description**
1	access_fail	R/W0C	0	**Flash Access Fail Interrupt Flag** This bit is set when an attempt is made to write to the flash while the flash is busy or locked. Only hardware can set this bit to 1. Writing a 1 to this bit has no effect. This bit is cleared by writing a 0. 0: No access failure has occurred. 1: Access failure occurred.
0	done	R/W0C	0	**Flash Operation Complete Interrupt Flag** This flag is automatically set by hardware after a flash write or erase operation completes. 0: Operation not complete or not in process. 1: Flash operation complete.

Table 5-6: Flash Controller Data Register 0

Flash Controller Data Register 0		FLC_DATA0		[0x30]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	data0	R/W	0	**Flash Data 0** Flash data for bits 31:0.

Table 5-7: Flash Controller Data Register 1

Flash Controller Data Register 1		FLC_DATA1		[0x34]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	data1	R/W	0	**Flash Data 1** Flash data for bits 63:32

Table 5-8: Flash Controller Data Register 2

Flash Controller Data Register 2		FLC_DATA2		[0x38]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	data2	R/W	0	**Flash Data 2** Flash data for bits 95:64

Table 5-9: Flash Controller Data Register 3

Flash Controller Data Register 3		FLC_DATA3		[0x3C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	data3	R/W	0	**Flash Data 3** Flash data for bits 127:96.

6 General-Purpose I/O and Alternate Function Pins

The general-purpose I/O (GPIO) pins share both a firmware-controlled I/O mode and up to three peripheral alternate functions. Each pin is individually enabled for GPIO or peripheral alternate function 1 (AF1), alternate function 2 (AF2) or alternate function 3 (AF3). Configuring a pin for an alternate function supersedes its use as a firmware-controlled GPIO, however the input data is always readable via the GPIO input register if the GPIO input is enabled.

Multiplexing between the alternate functions and the I/O function is often static in an application; set at initialization and dedicated as either an alternate function or GPIO. If needed, dynamic multiplexing between AF1, AF2, AF3 and I/O mode is supported. Dynamic multiplexing must be managed by the application firmware and the application must manage the AFs and GPIO to ensure each is set up properly when switching from a peripheral to the I/O function. Refer to the MAX32660 Data Sheet *Electrical Characteristics Table*, http://www.maximintegrated.com, for information on the GPIO pin behavior based on the configurations described in this document.

In GPIO mode each I/O pin supports interrupt function that can be independently enabled, and configured as a level triggered interrupt, a rising edge, falling edge or both rising and falling edge interrupt. All GPIO share the same interrupt vector. No all GPIO are available on all packages.

The GPIO are all bidirectional digital I/O that include:

- Input Mode Features
  - Standard CMOS or Schmitt Hysteresis
  - Input data from the input data register (*GPIO0_IN*) or to a peripheral (alternate function)
  - Input state selectable for floating (tri-state) or weak pull-up/pull-down
- Output Mode Features
  - Output data from the output data register (*GPIO0_OUT*) in GPIO mode
  - Output data driven from peripheral if an Alternate Function is selected
  - Standard GPIO
    - Four drive strength modes
    - Slow or Fast slew rate selection
  - GPIO with I^2C as an Alternate Function
    - Two drive strength modes
- Selectable weak pull-up resistor, weak pull-down resistor or tri-state mode for Standard GPIO pins
- Selectable weak pull-down or tri-state mode for GPIO pins with I^2C as an Alternate Function
- Wake from low power modes on rising edge, falling edge or both on the I/O pins

6.1 General Description

The MAX32660 provides up to 14 GPIO pins in the 20-TQFN package and up to 10 GPIO pins in the 16-WLP. Each GPIO pin maps to a GPIO port. For the MAX32660 all GPIO pins are grouped in GPIO port 0 (GPIO0). *Table 6-1* and *Table 6-2*, below, show the GPIO and the assigned AF1, AF2 and AF3 for the 16-WLP and 20-TQFN packages of the MAX32660.

A dedicated interrupt vector is assigned for GPIO port 0 and is detailed in the section *Interrupt*.

Table 6-1: GPIO Port, Pin Name and Alternate Function Matrix, 16-WLP

16-WLP				
GPIO Port[bit]	**GPIO**	**Alternate Function 1**	**Alternate Function 2**	**Alternate Function 3**
GPIO0[0]	P0.0	SWDIO[1]	SPI1_MISO (I2S_SDI)[2]	UART1_TX[1]
GPIO0[1]	P0.1	SWDCLK[1]	SPI1_MOSI (I2S_SDO)[2]	UART1_RX[1]
GPIO0[2][3]	P0.2	I2C1_SCL	SPI1_SCK (I2S_BCLK)[2]	32KCAL
GPIO0[3][3]	P0.3	I2C1_SDA	SPI1_SS0 (I2S_LRCLK)[2]	TMR0
GPIO0[4]	P0.4	SPI0_MISO	UART0_TX	-
GPIO0[5]	P0.5	SPI0_MOSI	UART0_RX	-
GPIO0[6]	P0.6	SPI0_SCK	UART0_CTS	UART1_TX[1]
GPIO0[7]	P0.7	SPI0_SS0	UART0_RTS	UART1_RX[1]
GPIO0[8][3]	P0.8	I2C0_SCL	SWDIO[1]	-
GPIO0[9][3]	P0.9	I2C0_SDA	SWDCLK[1]	-

Table 6-2: GPIO Port, Pin Name and Alternate Function Matrix, 20-TQFN

20-TQFN				
GPIO Port[bit]	**GPIO**	**Alternate Function 1**	**Alternate Function 2**	**Alternate Function 3**
GPIO0[0]	P0.0	SWDIO[1,]	SPI1_MISO (I2S_SDI)[1,2]	UART1_TX[1]
GPIO0[1]	P0.1	SWDCLK[1]	SPI1_MOSI (I2S_SDO)[1,2]	UART1_RX[1]
GPIO0[2][3]	P0.2	I2C1_SCL	SPI1_SCK (I2S_BCLK)[1,2]	32KCAL
GPIO0[3][3]	P0.3	I2C1_SDA	SPI1_SS0 (I2S_LRCLK)[1,2]	TMR0
GPIO0[4]	P0.4	SPI0_MISO	UART0_TX	-
GPIO0[5]	P0.5	SPI0_MOSI	UART0_RX	-
GPIO0[6]	P0.6	SPI0_SCK	UART0_CTS	UART1_TX[1]
GPIO0[7]	P0.7	SPI0_SS0	UART0_RTS	UART1_RX[1]
GPIO0[8][3]	P0.8	I2C0_SCL	SWDIO[1]	-
GPIO0[9][3]	P0.9	I2C0_SDA	SWDCLK[1]	-
GPIO0[10]	P0.10	SPI1_MISO (I2S_SDI)[1,2]	UART1_TX[1]	
GPIO0[11]	P0.11	SPI1_MOSI_ (I2S_SDO)[1,2]	UART1_RX[1]	
GPIO0[12]	P0.12	SPI1_SCK (I2S_BCLK)[1,2]	UART1_CTS	
GPIO0[13]	P0.13	SPI1_SS0 (I2S_LRCLK)[1,2]	UART1_RTS	

6.2 Power-On-Reset Configuration

During a power-on-reset event all I/O default to GPIO mode as inputs floating except the SWD JTAG pins P0.0 and P0.1. The SWD JTAG pins always default to Alternate Function 1 enabled and the SWD JTAG is enabled.

[1] This alternate function signal is mappable to more than one GPIO pin but there is only one instance of this peripheral in the MAX32660.
[2] I2S_BCLK, I2S_LRCLK, I2S_SDK, I2S_SDO when the I2S function is enabled.
[3] GPIO with I2C as an Alternate Function do not support slew rate control and only support two output drive strength modes.

Following a POR event GPIO[2:13] are configured with the following default settings:

- GPIO mode enabled
  - *GPIO0_AF0_SEL[pin]* = 1
  - *GPIO0_AF1_SEL[pin]* = 0
- Pull-up/Pull-down disabled, I/O in Hi-Z mode
  - *GPIO0_PULL_EN[pin]* = 0
- Output mode disabled
  - *GPIO0_OUT_EN[pin]* = 0
- Interrupt disabled
  - *GPIO0_INT_EN[pin]* = 0

Note: On parts without a SWD JTAG port, the SWD JTAG port is still available for boundary scan testing, however, the SWD JTAG port is hardware disabled. To use the SWD JTAG pins in I/O mode, set the desired GPIO pins for SWD alternate function and set the JTAG SWD disable field to 1 (GCR_SCON.swd_dis = 1).

6.2.1 I/O Mode and Alternate Function Selection

Each I/O pin supports standard GPIO mode or one of up to three Alternate Function modes. The alternate functions assigned to each I/O pin are shown in the pin description table for the specific package. See *Table 6-1* for the 16-WLP, and *Table 6-2* for the 20-TQFN.

6.2.2 Input mode configuration

Perform the following steps to configure a pin or pins for input mode:

1. Set the pin for I/O mode
   a. *GPIO0_AF0_SEL[pin]* = 1
   b. *GPIO0_AF1_SEL[pin]* = 0
2. Configure the pin for pull-up, pull-down, or high-impedance mode. Refer to *GPIO_PULL_SEL* register for pull-up and pull-down selection
3. GPIO pins with I^2C as an alternate function (GPIO[9:8] and GPIO[3:2]) only support high-impedance mode or a weak pull-down resistor.
4. Set *GPIO0_PULL_EN[pin]* to 1 to enable the pull resistor or clear the bit to set the input to high impedance mode.
5. Read the input state of the pin using the *GPIO0_IN[pin]* field.

6.2.3 Output Mode Configuration

Perform the following steps to configure a pin for output mode:

1. Set the pin for I/O mode.
   a. *GPIO0_AF0_SEL[pin]* = 1
   b. *GPIO0_AF1_SEL[pin]* = 0
   c. Enable the output buffer for the pin by setting *GPIO0_OUT_EN[pin]* to 1.
2. Set the output drive strength using the *GPIO0_DS1_SEL [pin]* and *GPIO0_DS0_SEL[pin]* bits.
   a. Refer to the *GPIO Drive Strength* for configuration details and the modes supported.
   b. Reference the MAX32660 datasheet for the electrical characteristics for the drive strength modes.
3. Set the output high or low using the *GPIO0_OUT[pin]* bit.

6.2.4 GPIO Drive Strength

Each I/O pin supports multiple selections for drive strength. Standard GPIO pins are configured for the supported modes using the *GPIO0_DS1_SEL* and *GPIO0_DS0_SEL* registers as shown in *Table 6-3*, *below*.

For GPIO with I²C as an Alternate Function, *Table 6-4* shows the drive strength setting options.

Table 6-3: Standard GPIO Drive Strength Selection

Drive Strength V_{DD} = 1.62V	Drive Strength V_{DD} = 3.63V	GPIO_DS1_SEL[pin]	GPIO_DS0_SEL[pin]
1mA	2mA	0	0
2mA	4mA	0	1
4mA	8mA	1	0
8mA	12mA	1	1

Table 6-4: GPIO with I²C Alternate Function Drive Strength Selection

Drive Strength V_{DD} = 1.62V	Drive Strength V_{DD} = 3.63V	GPIO_DS0_SEL[pin]
2mA	4mA	0
10mA	20mA	1

Note: The drive strength currents shown are targets only. Refer to the MAX32660 Data Sheet Electrical Characteristics table for details of the V_{OL_GPIO}, V_{OH_GPIO}, V_{OL_I2C} and V_{OH_I2C} parameters.

6.3 Alternate Function Configuration

Table 6-5, *below*, shows the alternate function selection matrix. Write the *GPIO0_AF0_SEL* and *GPIO0_AF1_SEL* fields as shown in the table to select the desired alternate function.

Table 6-5: GPIO Mode and Alternate Function Selection

GPIO MODE	GPIO0_AF1_SEL[pin]	GPIO0_AF0_SEL[pin]
I/O	0	1
Alternate Function 1	0	0
Alternate Function 2	1	0
Alternate Function 3	1	1

Note: Each Alternate Function for a given peripheral is independently selectable. Mixing functions assigned to AF1, AF2 or AF3 is supported if all of the peripheral's required functions are enabled.

6.4 Configuring GPIO (External) Interrupts

Each GPIO supports external interrupt events when the GPIO is configured for I/O mode and the input mode is enabled. If the GPIO is configured as a peripheral alternate function, the interrupts are peripheral controlled. GPIO interrupts can be

enabled for any number of GPIO on each GPIO port. The following procedure details the steps for enabling Active mode interrupt events for a GPIO pin:

1. Disable interrupts by setting the *GPIO0_INT_EN[pin]* field to 0. This will prevent any new interrupts on the pin from triggering but will not clear previously triggered (pending) interrupts. The application can disable all interrupts for GPIO by writing 0 to *GPIO0_INT_EN[13:0]*. To maintain previously enabled interrupts, read the *GPIO0_INT_EN* register and save the value to memory prior to setting the register to 0.
2. Clear pending interrupts by writing 1 to the *GPIO0_INT_FL[pin]* bit.
3. Set *GPIO0_INT_MODE[pin]* to select either level (0) or edge triggered (1) interrupts.
   a. For level triggered interrupts, the interrupt triggers on an input high or low.
      i. *GPIO0_INT_POL[pin]* = 1: Input high triggers interrupt.
      ii. *GPIO0_INT_POL[pin]* = 0: Input low triggers interrupt.
   b. For edge triggered interrupts, the interrupt triggers on an edge event.
      i. *GPIO0_INT_POL[pin]* = 0: Input rising edge triggers interrupt.
      ii. *GPIO0_INT_POL[pin]* = 1: Input falling edge triggers interrupt.
   c. Optionally set *GPIO0_INT_DUAL_EDGE[pin]* to 1 to trigger on both the rising and falling edges of the input signal.
4. Set *GPIO0_INT_EN[pin]* to 1 to enable the interrupt for the pin.

6.4.1 Interrupts

The GPIO pins generate interrupts if the pin is configured for I/O mode and the interrupt is enabled for the pin (*GPIO0_INT_EN[pin]* = 1). See *Table 6-5* for details on configuring a pin for I/O mode.

Table 6-6: GPIO Port Interrupt Vector Mapping

GPIO Interrupt Source	GPIO Interrupt Flag Register	Device Specific Interrupt Vector Number	GPIO Interrupt Vector
GPIO0[13:0]	GPIO0_INT_FL	40	GPIO0_IRQHandler

To handle GPIO interrupts in your interrupt vector handler, complete the following steps:

1. Read the *GPIO0_INT_FL* register to determine the GPIO pin that triggered the interrupt. The bit position that reads 1 indicates the pin that resulted in the interrupt event. If multiple bits are set, each of them indicates an interrupt event occurred on the respective pin.
2. Complete interrupt tasks associated with the interrupt source pin (application defined).
3. Clear the interrupt flag in the *GPIO0_INT_FL* register by writing 1 to the *GPIO0_INT_FL* bit positions that triggered the interrupt. This also clears and rearms the edge detectors for edge triggered interrupts.
4. Return from the interrupt vector handler.

6.4.2 Using GPIO for Wakeup from Low Power Modes

Low power modes support wakeup from external edge triggered interrupts on the GPIO ports. Level triggered interrupts are not supported for wakeup because the system clock must be active to detect levels.

For wake-up interrupts on the GPIO a single interrupt vector, GPIOWAKE_IRQHandler, is assigned for all the GPIO pins. When the wakeup event occurs, the application software must interrogate the *GPIO0_INT_FL* register to determine which external pin caused the wake-up event.

Table 6-7: GPIO Wakeup Interrupt Vector

GPIO Wake Interrupt Source	GPIO Wake Interrupt Status Register	Device Specific Interrupt Vector Number	GPIO Wakeup Interrupt Vector
GPIO0[0:13]	GPIO0_INT_FL	70	GPIOWAKE_IRQHandler

Enable low power mode wakeup (*SLEEP*, *DEEPSLEEP* and *BACKUP*) from an external GPIO event by completing the following steps:

1. Set the polarity (rising or falling edge) by writing to the *GPIO0_INT_POL[pin]* field. The wakeup functionality uses rising and falling edge detection circuitry that operates asynchronously and does not require an active clock. Dual-edge mode is also an option to accomplish edge detection wakeup.
2. Clear pending interrupt flags by writing 0xFF to the *GPIO0_INT_FL* register.
3. Activate the GPIO wakeup function by writing 1 to *GPIO0_WAKE_EN[pin]*.
4. Configure the power manager to use the GPIO as a wakeup source by writing to the appropriate Global Control register (GCR).

6.5 GPIO Registers

The GPIO0 base peripheral address is 0x4000 8000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 6-8: GPIO Port 0 Registers

Offset	Register Name	Access	Description
[0x0000]	*GPIO0_AF0_SEL*	R/W	I/O and Alternate Function 1 Select Register
[0x000C]	*GPIO0_OUT_EN*	R/W	Output Enable Register
[0x0018]	*GPIO0_OUT*	R/W	Output Register
[0x0024]	*GPIO0_IN*	RO	Input Register
[0x0028]	*GPIO0_INT_MODE*	R/W	Interrupt Mode Register
[0x002C]	*GPIO0_INT_POL*	R/W	Interrupt Polarity Select Register
[0x0034]	*GPIO0_INT_EN*	R/W	Interrupt Enable Register
[0x0040]	*GPIO0_INT_FL*	R/W1C	Interrupt Flag Register
[0x004C]	*GPIO0_WAKE_EN*	R/W	Wakeup Enable Register
[0x005C]	*GPIO0_INT_DUAL_EDGE*	R/W	Dual Edge Select Interrupt Register
[0x0060]	*GPIO0_PULL_EN*	R/W	Input Pullup/Pulldown Select Register
[0x0068]	*GPIO0_AF1_SEL*	R/W	Alternate Function 2/3 Select Register
[0x00A8]	*GPIO0_INHYS_EN*	R/W	Input Hysteresis Enable Register
[0x00AC]	*GPIO0_SR_SEL*	R/W	Slew Rate Select Register
[0x00B0]	*GPIO0_DS0_SEL*	R/W	Drive Strength Select 0 Register
[0x00B4]	*GPIO0_DS1_SEL*	R/W	Drive Strength Select 1 Register
[0x00B8]	*GPIO_PULL_SEL*	R/W	Pullup/Pulldown Enable Register

6.5.1 GPIO Register Details

Table 6-9: GPIO Alternate Function 0 Select Register

GPIO Alternate Function 0 Select Register		GPIO0_AF0_SEL		[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	1	**Reserved for Future Use** Do not modify this field.

GPIO Alternate Function 0 Select Register		GPIO0_AF0_SEL		[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
13:2		R/W	1	**GPIO Alternate Function 0 Mode Select** If JTAG debug is available on the part, this pin defaults to the JTAG alternate function (TCK/SWCLK) on all forms of reset. 0: Alternate function JTAG TCK/SWCLK enabled (default). 1: GPIO enabled
1	-	R/W	0	**GPIO Alternate Function 0 Mode Select** If JTAG debug is available on the part, this pin defaults to the JTAG alternate function (TMS/SWDIO) on all forms of reset. 0: Alternate function JTAG TMS/SWDIO enabled (default). 1: GPIO enabled
0	-	R/W	0	**GPIO Enable** If JTAG debug is available on the part, this pin defaults to the JTAG alternate function (TDO) on all forms of reset. 0: Alternate function JTAG TDO enabled (default). 1: GPIO enabled

Table 6-10: GPIO Output Enable Register

Output Enable Register		GPIO0_OUT_EN		[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:2	-	R/W	0	**GPIO Output Enable** Setting a bit to 1 enables the output driver for the respective pin. 0: Output mode disabled, output driver disabled. 1: Output mode enabled, output driver enabled.
1	-	R/W	1	**GPIO Output Enable** This bit is set to 1 on POR and is used for the SWDIO alternate function with the output driver enabled. 0: Output mode disabled, output driver disabled. 1: Output mode enabled, output driver enabled.
0	-	R/W	0	**GPIO Output Enable** This bit is set to 0 on POR and is used for the SWDCLK alternate function with the output driver disabled. Setting this bit to 1 enables the output driver for the pin. 0: Output mode disabled, output driver disabled. 1: Output mode enabled, output driver enabled.

Table 6-11: GPIO Output Register

GPIO Output Register		GPIO0_OUT		[0x0018]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:0	-	R/W	0	**GPIO Output Level** Set the corresponding output pin high or low. 0: Drive the corresponding output pin low (logic 0). 1: Drive the corresponding output pin high (logic 1). *Note: This bit is ignored if the corresponding bit position in the GPIO0_OUT_EN register is not set or if the pin is configured for an alternate function.*

Table 6-12: GPIO Input Register

GPIO Input Register			GPIO0_IN	[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:0	-	RO	-	**GPIO Input Level** Read the state of the corresponding input pin. The input state is always readable for a pin regardless of the pin's configuration as an output or alternate function. 0: Input pin low (logic 0) 1: Input pin high (logic 1)

Table 6-13: GPIO Port Interrupt Mode Register

GPIO Port Interrupt Mode Register			GPIO0_INT_MODE	[0x0028]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:0	-	R/W	0	**GPIO Interrupt Mode** Interrupt mode selection bit for the corresponding GPIO pin. 0: Level triggered interrupt for corresponding GPIO pin. 1: Edge triggered interrupt for corresponding GPIO pin. *Note: This bit has no effect unless the corresponding bit in the GPIO0_INT_EN register is set.*

Table 6-14: GPIO Port Interrupt Polarity Registers

GPIO Interrupt Polarity Register			GPIO0_INT_POL	[0x002C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:0	-	R/W	0	**GPIO Interrupt Polarity** Interrupt polarity selection bit for the corresponding GPIO pin. Level triggered mode (*GPIO0_INT_MODE* = 0): 0: Input low (logic 0) triggers interrupt. 1: Input high (logic 1) triggers interrupt. Edge triggered mode (*GPIO0_INT_MODE* = 1): 0: Falling edge triggers interrupt 1: Rising edge triggers interrupt. *Note: This bit has no effect unless the corresponding bit in the GPIO0_INT_EN register is set.*

Table 6-15: GPIO Port Interrupt Enable Registers

GPIO Interrupt Enable Register			GPIO0_INT_EN	[0x0034]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.

GPIO Interrupt Enable Register		GPIO0_INT_EN		[0x0034]
Bits	**Name**	**Access**	**Reset**	**Description**
13:0	-	R/W	0	**GPIO Interrupt Enable** Enable or Disable the interrupt for the corresponding GPIO pin. 0: GPIO interrupt disabled. 1: GPIO interrupt enabled. *Note: Disabling a GPIO interrupt does not clear pending interrupts for the associated pin. Use the GPIO0_INT_CLR register to clear pending interrupts.*

Table 6-16: GPIO Interrupt Flag Register

GPIO Interrupt Flag Register		GPIO0_INT_FL		[0x0040]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:0	-	RO	0	**GPIO Interrupt Status** An interrupt is pending for the associated GPIO pin when this bit reads 1. 0: No interrupt pending for associated GPIO pin. 1: GPIO interrupt pending for associated GPIO pin. *Note: Write a 1 to the corresponding bit in the GPIO0_INT_CLR register to clear the interrupt pending status flag.*

Table 6-17: GPIO Wakeup Enable Registers

GPIO Wakeup Enable Register		GPIO0_WAKE_EN		[0x004C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:0	-	R/W	0	**GPIO Wakeup Enable** Enable the I/O as a wakeup from low power modes (*SLEEP*, *DEEPSLEEP*, *BACKUP*). 0: GPIO is not enabled as a wakeup source from low power modes. 1: GPIO is enabled as a wakeup source from low power modes.

Table 6-18: GPIO Interrupt Dual Edge Mode Registers

GPIO Interrupt Dual Edge Mode Register		GPIO0_INT_DUAL_EDGE		[0x005C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:0	-	R/W	0	**GPIO Interrupt Dual-Edge Mode Select** Setting this bit selects dual edge mode triggered interrupts (rising and falling edge triggered) if the associated *GPIO0_INT_MODE* bit is set to edge triggered. When dual edge mode is set, and the interrupt mode is edge-triggered, the associated polarity (*GPIO0_INT_POL*) setting has no effect. 0: Dual edge detection mode interrupts disabled. 1: Dual edge detection mode interrupts enabled.

Table 6-19: GPIO Pullup/Pulldown Enable Register

GPIO Pullup Pulldown Selection 0 Register		GPIO0_PULL_EN		[0x0060]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:10	-	R/W	0	**GPIO Pull Up/Pull Down Enable** Setting this bit to 1 enables either the weak pull-up or weak pull-down resistor on the respective pin. The selection for pull-up or pull-down resistor is set using the *GPIO_PULL_SEL* register.
9:8	-	R/W	0	**GPIO Pull Down Enable** Setting this bit to 1 enables the weak pull-down resistor on the respective I/O pin. GPIO with I^2C as an alternate function do not support a weak pull-up resistor. If either of the *GPIO_PULL_SEL*[9:8] bits are set to 1, setting the same bit in this register has no effect. 0: Pull down resistor disable. 1: Pull down resistor enabled if respective bit in *GPIO_PULL_SEL* register is set to 0. No effect if respective bit in *GPIO_PULL_SEL* register is set to 1.

Table 6-20: GPIO Alternate Function Select Register

GPIO Alternate Function Select Register		GPIO0_AF1_SEL		[0x0068]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:0	-	R/W	0	**GPIO Alternate Function 1 Mode Select** This bit combined with the corresponding bit in the *GPIO0_AF0_SEL* register set the I/O pin to GPIO mode or to Alternate Function 1, 2, or 3. *Refer to Table 6-5: GPIO Mode and Alternate Function Selection* for details on selection.

Table 6-21: GPIO Input Hysteresis Enable Register

GPIO Input Hysteresis Enable Register		GPIO0_INHYS_EN		[0x00A8]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:0	-	R/W	0	**GPIO Input Hysteresis Enable** Setting a bit to 1 enables a Schmitt input to introduce hysteresis for better noise immunity on the respective bit's port pin. 0: Input pin uses a standard CMOS input. 1: Schmitt input enabled.

Table 6-22: GPIO Slew Rate Enable Register

GPIO Slew Rate Select Register		GPIO0_SR_SEL		[0x00AC]
Bits	**Name**	**Access**	**Reset**	**Description**
31: 14	-	R/W	0	**Reserved for Future Use** Do not modify this field.

GPIO Slew Rate Select Register			GPIO0_SR_SEL	[0x00AC]
Bits	**Name**	**Access**	**Reset**	**Description**
13:10	-	R/W	0	**GPIO Slew Rate Mode** Selects between fast and slow slew rate for the respective I/O pin. Setting a bit to 1 enables slow slew rate for the respective I/O pin. 0: Fast slew rate selected. 1: Slow slew rate selected. *Note: Refer to the MAX32660 datasheet for detailed electrical characteristics of the fast and slow slew rates.*
9:8	-	R/W	0	**Reserved for Future Use** Do not modify this field.
7:4	-	R/W	0	**GPIO Slew Rate Mode** Selects between fast and slow slew rate for the respective I/O pin. Setting a bit to 1 enables slow slew rate for the respective I/O pin. 0: Fast slew rate selected. 1: Slow slew rate selected. *Note: Refer to the MAX32660 datasheet for detailed electrical characteristics of the fast and slow slew rates.*
3:2	-	R/W	0	**Reserved for Future Use** Do not modify this field.
1:0	-	R/W	0	**GPIO Slew Rate Mode** Selects between fast and slow slew rate for the respective I/O pin. Setting a bit to 1 enables slow slew rate for the respective I/O pin. 0: Fast slew rate selected. 1: Slow slew rate selected. *Note: Refer to the MAX32660 datasheet for detailed electrical characteristics of the fast and slow slew rates.*

Table 6-23: GPIO Drive Strength 0 Select Register

GPIO Drive Strength 0 Select Register			GPIO0_DS0_SEL	[0x00B0]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:10	-	R/W	0	**GPIO Drive Strength 0 Select** The output drive strength supports four modes. The mode selection is set using the combination of the *GPIO0_DS1_SEL* and *GPIO0_DS0_SEL* bits for the associated GPIO pin. Refer to the *GPIO Drive* Strength section, *above,* for the selection options on these I/O pins. *Refer to the symbols V_{OL_GPIO} and V_{OH_GPIO} in the MAX32660 Data Sheet Electrical Characteristics table for details of the drive strengths for these I/O pins.*
9:8	-	R/W	0	**GPIO Drive Strength Select** Selection of high drive strength or low drive strength for the I/O pin. Pins with I^2C as an alternate function only support two drive strength options. 0: Low output drive strength selected. 1: High output drive strength selected. *Refer to V_{OL_I2C} and V_{OH_I2C} in the MAX32660 Data Sheet Electrical Characteristics table for details of the drive strengths for these I/O pins.*

GPIO Drive Strength 0 Select Register		GPIO0_DS0_SEL		[0x00B0]
Bits	**Name**	**Access**	**Reset**	**Description**
7:4	-	R/W	0	**GPIO Drive Strength 0 Select** The output drive strength supports four modes. The mode selection is set using the combination of the *GPIO0_DS1_SEL* and *GPIO0_DS0_SEL* bits for the associated GPIO pin. Refer to the *GPIO Drive* Strength section, *above,* for the selection options on these I/O pins. *Refer to the symbols V_{OL_GPIO} and V_{OH_GPIO} in the MAX32660 Data Sheet Electrical Characteristics table for details of the drive strengths for these I/O pins.*
3:2	-	R/W	0	**GPIO Drive Strength Select** Selection of high drive strength or low drive strength for the I/O pin. Pins with I^2C as an alternate function only support two drive strength options. 0: Low output drive strength selected. 1: High output drive strength selected. *Refer to V_{OL_I2C} and V_{OH_I2C} in the MAX32660 Data Sheet Electrical Characteristics table for details of the drive strengths for these I/O pins.*
1:0	-	R/W	0	**GPIO Drive Strength 0 Select** The output drive strength supports four modes. The mode selection is set using the combination of the *GPIO0_DS1_SEL* and *GPIO0_DS0_SEL* bits for the associated GPIO pin. Refer to the *GPIO Drive* Strength section, *above,* for the selection options on these I/O pins. *Refer to the symbols V_{OL_GPIO} and V_{OH_GPIO} in the MAX32660 Data Sheet Electrical Characteristics table for details of the drive strengths for these I/O pins.*

Table 6-24: GPIO Drive Strength 1 Select Register

GPIO Drive Strength 1 Select Register		GPIO0_DS1_SEL		[0x00B4]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:10	-	R/W	0	**GPIO Drive Strength 1 Select** The output drive strength supports four modes. The mode selection is set using the combination of the *GPIO0_DS1_SEL* and *GPIO0_DS0_SEL* bits for the associated GPIO pin. Refer to the *GPIO Drive* Strength section, *above,* for details on the selection options. *Refer to the symbols V_{OL_GPIO} and V_{OH_GPIO} in the MAX32660 Data Sheet Electrical Characteristics table for details of the drive strengths for these I/O pins.*
9:8	-	R/W	0	**Reserved for Future Use** Do not modify this field.
7:4	-	R/W	0	**GPIO Drive Strength 1 Select** The output drive strength supports four modes. The mode selection is set using the combination of the *GPIO0_DS1_SEL* and *GPIO0_DS0_SEL* bits for the associated GPIO pin. Refer to the *GPIO Drive* Strength section, *above,* for the selection options on these I/O pins. *Refer to the symbols V_{OL_GPIO} and V_{OH_GPIO} in the MAX32660 Data Sheet Electrical Characteristics table for details of the drive strengths for these I/O pins.*
3:2	-	R/W	0	**Reserved for Future Use** Do not modify this field.

GPIO Drive Strength 1 Select Register		GPIO0_DS1_SEL		[0x00B4]
Bits	**Name**	**Access**	**Reset**	**Description**
1:0	-	R/W	0	**GPIO Drive Strength 1 Select** The output drive strength supports four modes. The mode selection is set using the combination of the *GPIO0_DS1_SEL* and *GPIO0_DS0_SEL* bits for the associated GPIO pin. Refer to the *GPIO Drive* Strength section, *above,* for the selection options on these I/O pins. *Refer to the symbols V_{OL_GPIO} and V_{OH_GPIO} in the MAX32660 Data Sheet Electrical Characteristics table for details of the drive strengths for these I/O pins.*

Table 6-25: GPIO Pullup/Pulldown Select Register

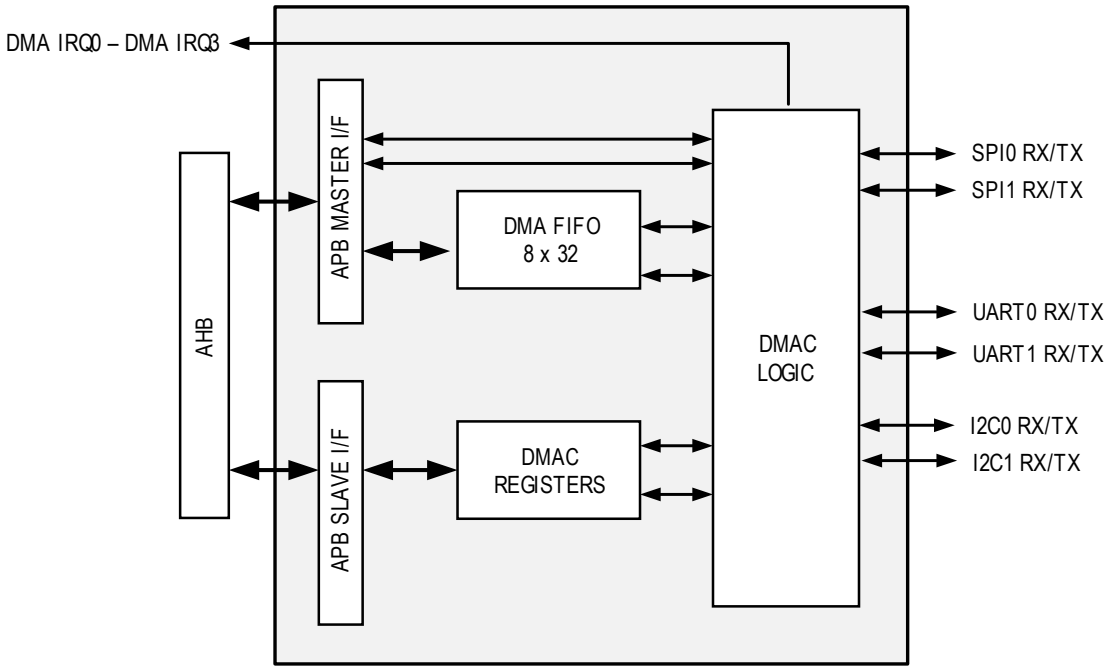
GPIO Pullup/Pulldown Select Register		GPIO_PULL_SEL		[0x00B8]
Bits	**Name**	**Access**	**Reset**	**Description**
31:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:10	-	R/W	0	**Pullup/Pulldown Resistor Select** Selects either a weak pull-up or weak pull-down resistor for the respective I/O pin. 0: Pull-down resistor selected 1: Pull-up resistor selected *Refer to the MAX32660 Data Sheet Electrical Characteristics table for details of the pull-up/pull-down resistors for the respective I/O pins.*
9:8	-	R/W	0	**Pulldown Resistor Select** This bit should always be set to 0. The I/O pins with I^2C as an alternate function only a weak pull-down resistor. 0: Pull-down resistor selected 1: Invalid *Refer to the MAX32660 Data Sheet Electrical Characteristics table for details of the pull-up/pull-down resistors for the respective I/O pins.*
7:4	-	R/W	0	**Pullup/Pulldown Resistor Select** Selects either a weak pull-up or weak pull-down resistor for the respective I/O pin. 0: Pull-down resistor selected 1: Pull-up resistor selected *Refer to the MAX32660 Data Sheet Electrical Characteristics table for details of the pull-up/pull-down resistors for the respective I/O pins.*
3:2	-	R/W	0	**Pulldown Resistor Select** This bit should always be set to 0. The I/O pins with I^2C as an alternate function only a weak pull-down resistor. 0: Pull-down resistor selected 1: Invalid *Refer to the MAX32660 Data Sheet Electrical Characteristics table for details of the pull-up/pull-down resistors for the respective I/O pins.*
1:0	-	R/W	0	**Pullup/Pulldown Resistor Select** Selects either a weak pull-up or weak pull-down resistor for the respective I/O pin. 0: Pull-down resistor selected 1: Pull-up resistor selected *Refer to the MAX32660 Data Sheet Electrical Characteristics table for details of the pull-up/pull-down resistors for the respective I/O pins.*

7 DMA Controller

The Direct Memory Access controller (DMAC) is a hardware feature that moves data blocks from peripheral to memory, memory to peripheral, and memory to memory. This data movement reduces the processor load significantly.

Figure 7-1 provides a high-level overview of the major DMA Controller components.

Figure 7-1: DMAC Block Diagram



All direct memory access (DMA) transactions consist of an advanced high-performance bus (AHB) burst read from the source into the DMA FIFO followed by an AHB burst write from the DMA FIFO to the destination.

7.1 DMA channel operation

The DMA Controller has 4 channels. Each channel is governed by the registers shown in *Table 7-1*.

Table 7-1: DMA Channel Registers

Register	Description
DMAn_DST	Destination register
DMAn_CFG	Configuration register
DMAn_STAT	Status register
DMAn_SRC	Source register
DMAn_CNT	Count register

In addition, each channel has a set of reload registers, shown in *Table 7-2*, that are used to chain DMA buffers when a count-to-zero (CTZ) condition occurs.

Table 7-2: Channel Reload Registers

Register	Description
DMAn_DST_RLD	Destination reload register
DMAn_SRC_RLD	Source reload register
DMAn_CNT_RLD	Count reload register

Using these eight registers provides each channel with the following features:

- Full 32-bit source and destination addresses with 24-bit (16 Mbytes) address increment capability
- Up to 16 Mbytes for each DMA buffer
- Programmable burst size
- Programmable priority
- Interrupt upon CTZ
- Abort on error

7.2 DMA Channel Arbitration and DMA Bursts

DMAC contains an internal arbiter that allows enabled channels to access the AHB and move data. A DMA channel is enabled using the *DMAn_CFG.chen* bit.

When disabling a channel, poll the *DMAn_STAT.ch_st* bit to determine if the channel is truly disabled. In general, *DMAn_STAT.ch_st* follows the setting of the *DMAn_CFG.chen* bit. However, the *DMAn_STAT.ch_st* bit is automatically cleared under the following conditions:

- Bus error (cleared immediately)
- CTZ when the *DMAn_CFG.rlden* = 0 (cleared at the end of the AHB R/W burst)
- *DMAn_STAT.chen* bit transitions to 0 (cleared at the end of the AHB R/W burst)

Whenever the *DMAn_STAT.ch_st* bit transitions from 1 to 0, the corresponding *DMAn_CFG.chen* bit is also cleared. During an AHB read/write burst, attempting to disable an active channel is delayed until burst completion.

Once a channel is programmed and enabled, it generates a request to the arbiter immediately (for memory-to-memory DMA) or whenever its associated peripheral requests DMA (for memory-to-peripheral or peripheral-to-memory DMA).

The arbiter grants requests to a single channel at a time. Granting is done based on priority—a higher priority request is always granted. Within a given priority level, requests are granted on a round-robin basis.

When a channel's request is granted, it runs a DMA transfer. Once the DMA transfer completes, the channel relinquishes its grant.

Only an error condition can interrupt an ongoing data transfer.

DMAn_CFG.reqsel determines which request is used to initiate a DMA burst. In the case of a memory-to-memory transfer, the channel is treated as always requesting DMA access. The *DMAn_CFG.priority* field determines the DMA channel priority.

7.3 DMA Source and Destination Addressing

For memory addresses, the *DMAn_SRC* and *DMAn_DST* registers are used to program the addresses of the source and destination. For peripherals, however, the address is fixed based on the *DMAn_CFG.reqsel* value set.

Table 7-3 shows how the source address, destination address and the address increment controls are constructed based on the *DMAn_CFG.reqsel* bit. The first column, "Request Select", of *Table 7-3* shows the values for the *DMAn_CFG.reqsel* bit.

"Programmable" in the SRCINC or DSTINC columns indicates that the bits are programmable and set according to the *DMAn_CFG.srcinc* and the *DMAn_CFG.dstinc* bits, respectively. If there is a 0 in the column, then the bit is forced to 0. For peripherals, a value of 0 for the *DMAn_CFG.srcinc* or *DMAn_CFG.dstinc* generally indicates the source or destination address is fixed to the peripheral's transmit or receive FIFO. For example, if *DMAn_CFG.reqsel = 0x2,* then the transfer is an SPI1 RX (receive). In this case, the *DMAn_CFG.srcinc* is fixed to 0 because the source for the SPI1 RX is the SPI1 receive FIFO address. However, the destination address is programmable, allowing the data from the SPI1 receive FIFO to be written to a programmable address in memory. The destination starting address is set using the *DMAn_DST* register and is incremented automatically if *DMAn_CFG.dstinc* is set to 1.

Table 7-3: Source and Destination Address Definition

DMAn_CFG reqsel	Transfer	Source Address Register	*DMAn_CFG srcinc*	Destination Address Register	*DMAn_CFG dstinc*
0x0	Mem-to-Mem	*DMAn_SRC*	Programmable	*DMAn_DST*	Programmable
0x1	SPI0 RX	*DMAn_SRC*	0	*DMAn_DST*	Programmable
0x2	SPI1 RX	*DMAn_SRC*	0	*DMAn_DST*	Programmable
0x4	UART0 RX	*DMAn_SRC*	0	*DMAn_DST*	Programmable
0x5	UART1 RX	*DMAn_SRC*	0	*DMAn_DST*	Programmable
0x7	I2C0 RX	*DMAn_SRC*	0	*DMAn_DST*	Programmable
0x8	I2C1 RX	*DMAn_SRC*	0	*DMAn_DST*	Programmable
0x21	SPI0 TX	*DMAn_SRC*	Programmable	*DMAn_DST*	0
0x22	SPI1 TX	*DMAn_SRC*	Programmable	*DMAn_DST*	0
0x24	UART0 TX	*DMAn_SRC*	Programmable	*DMAn_DST*	0
0x25	UART1 TX	*DMAn_SRC*	Programmable	*DMAn_DST*	0
0x27	I2C0 TX	*DMAn_SRC*	Programmable	*DMAn_DST*	0
0x28	I2C1 TX	*DMAn_SRC*	Programmable	*DMAn_DST*	0

7.4 Data Movement from Source to DMA FIFO

Table 7-4 shows the register and bit fields used to control the movement of data into DMA FIFO. The source is a peripheral or memory.

Table 7-4: Data movement from source to DMA FIFO

Register/Bit Field	Description	Comments
DMAn_SRC	Source address	If the increment enable is set, this increments on every read cycle of the burst.
DMAn_CNT	Number of bytes to transfer before a CTZ condition occurs	This register is decremented on each read of the burst.
DMAn_CFG.brst	Burst size (1-32)	This determines the maximum number of bytes moved during the burst read.

Register/Bit Field	Description	Comments
DMAn_CFG.srcwd	Source width	This determines the maximum data width used during each read of the AHB burst (byte, two bytes, or four bytes). The actual AHB width might be less if *DMAn_CNT* is not great enough to supply all of the needed bytes.
DMAn_CFG.srcinc	Source increment enable	Increment the *DMAn_SRC*.

7.5 Data Movement from the DMA FIFO to Destination

Table 7-5 shows the registers and bit fields used to control the burst movement of data out of the DMA FIFO. The destination is a peripheral or memory.

Table 7-5: Data movement from the DMA FIFO to destination

Register	Field	Description	Details
DMAn_DST	*src*	Destination address	If the increment enable is set, this increments on every write cycle of the burst.
DMAn_CFG	*brst*	Burst size (1-32)	This determines the maximum number of bytes moved during a single AHB read/write burst.
	dstwd	Destination width	This determines the maximum data width used during each write of the AHB burst (one byte, two bytes, or four bytes).
	dstinc	Destination increment enable	Increments *DMAn_DST*.

7.6 Count-To-Zero (CTZ) Condition

When an AHB channel burst completes, the DMAC checks whether *DMAn_CNT* is decremented to 0. If it is, then a CTZ condition is triggered.

At this point, there are two possible responses depending on the value of the *DMAn_CFG.rlden* bit.

If *DMAn_CFG.rlden* = 1, then automatic reload occurs as follows:

- The *DMAn_SRC*, *DMAn_DST*, and *DMAn_CNT* registers are loaded from the reload registers.
- The channel remains active and continues operating.
  - The channel uses the previously programmed configuration values and the reloaded source, destination and count values.

If *DMAn_CFG.rlden* = 0, then the CTZ condition indicates the operation is complete and the following occurs:

- The channel is disabled, *DMAn_CFG.chen* = 0
- The channel status is set to disabled, *DMAn_STAT.ch_st* = 0.

7.7 Chaining Buffers

Use reload registers to chain buffers. Chaining buffers reduces the DMA ISR response time and allows DMA to service requests without intermediate processing from the CPU.

To configure a channel for buffer chaining, initialize the following registers:

- *DMAn_CFG*
- *DMAn_SRC*
- *DMAn_DST*
- *DMAn_CNT*
- *DMAn_SRC_RLD*
- *DMAn_DST_RLD*
- *DMAn_CNT_RLD*

When the *DMAn_CNT_RLD* register is written, the *DMAn_CNT_RLD.rlden* bit must not be set. In addition, any writes to the *DMAn_CFG* register prior to initialization must not set the *DMAn_CFG.chen* and *DMAn_CFG.rlden* bits. After all registers are initialized, the last operation involves writing to the *DMAn_CFG.chen* and *DMAn_CFG.rlden* bits. This starts the DMA.

Set the *DMAn_CFG.ctzien* bit in the register to receive an interrupt after each buffer is accessed. In addition, set the *DMAn_CFG.chdien* bit to provide an interrupt in case of a bus error.

***Caution**: Setting the DMAn_CFG.chen and the DMAn_CFG.rlden bits separately risks a race condition. The condition occurs between a DMA completion interrupt service routine initializing the reload registers for the third buffer before the software initialization of these registers for the second buffer.*

When the first DMA transfer completes (based on the *DMAn_CNT.cnt* bit value), a CTZ interrupt occurs, and the *DMAn_SRC*, *DMAn_DST*, and *DMAn_CNT* registers are reloaded from the corresponding reload registers.

The *DMAn_STAT* register indicates that the reload and CTZ events occurred. In this case, *DMAn_STAT.ch_st* = 1 indicating that the DMA is now busy with the second DMA transfer defined in the reload registers. If *DMAn_STAT.ch_st* = 0, then the initial and second DMA transfers have completed. If there are additional buffers to chain, the interrupt service routine initializes the *DMAn_SRC_RLD*, *DMAn_DST_RLD*, and *DMAn_CNT_RLD* registers and sets the *DMAn_CNT_RLD.rlden* bit. The interrupt service routine does not write to the *DMAn_CFG*, *DMAn_SRC*, *DMAn_DST*, and *DMAn_CNT* registers, just the reload registers.

To prevent improper operation, program the address bits before setting the *DMAn_CFG.chen* and *DMAn_CNT_RLD.rlden* bits.

7.8 DMA Interrupts

Enable interrupts for each channel by setting *DMA_INT_EN.chien.* When an interrupt is pending, the corresponding *DMA_INT_FL.ipend* = 1. The *DMA_INT_FL.ipend* field is read-only, to clear the interrupt use the *DMAn_STAT* register and write a 1 to the field that indicates the cause of the interrupt.

A channel interrupt (*DMAn_STAT.ipend* = 1) is caused by:

1. *DMAn_CFG.ctzien* = 1
   a. If enabled, all CTZ occurrences set the *DMAn_STAT.ipend* bit.
2. *DMAn_CFG.chdien* = 1
   a. If enabled, any clearing of the *DMAn_STAT.ch_st* bit sets the *DMAn_STAT.ipend* bit. Examine the *DMAn_STAT* register to determine which reason caused the disable. The *DMAn_CFG.chdien* bit also enables the *DMAn_STAT.to_st* bit. The *DMAn_STAT.to_st* bit does not clear the *DMAn_STAT.ch_st* bit.

To clear the channel interrupt, write 1 to the cause of the interrupt (the *DMAn_STAT.ctz_st, DMAn_STAT.rld_st, DMAn_STAT.bus_err,* or *DMAn_STAT.to_st* bits).

When running in normal mode without buffer chaining (*DMAn_CFG.rlden* = 0), set the *DMAn_CFG.chdien* bit only. An interrupt is generated upon DMA completion or an error condition (bus error or time-out error).

When running in buffer chaining mode (*DMAn_CFG.rlden* = 1), set both the *DMAn_CFG.chdien* and *DMAn_CFG.ctzien* bits. The CTZ interrupts occur on completion of each DMA (count reaches zero and reload occurs). The setting of *DMAn_CFG.chdien* ensures that an error condition generates an interrupt. If *DMAn_CFG.ctzien* = 0, then the only interrupt occurs when the DMA completes and *DMAn_CFG.rlden* = 0 (final DMA).

7.9 Channel Time-outs

Each channel can optionally generate an interrupt when its associated request line is inactive for a given period of time. An example use of this feature is to determine an idle UART receive channel. Each channel has a dedicated 10-bit timer allowing use of a different timeout value.

7.10 10-bit Timer

Use the settings in the *DMAn_CFG* register to control each channel's 10-bit timer. Scale the input clock for the timer using the *DMAn_CFG.pssel* field. The options available are:

- $\frac{f_{HCLK}}{256}$
- $\frac{f_{HCLK}}{64K}$
- $\frac{f_{HCLK}}{16M}$

Note: HCLK is the AHB interface clock that enables the memory system to run at a different frequency than the system clock, the cache controller, and the event monitor.

The *DMAn_CFG.tosel* field sets the time the 10-bit timer counts until generating an interrupt.

The 10-bit timer resets whenever any of the following conditions occur:

- The DMA request line programmed for the channel is activated.
- The channel is disabled for any reason (*DMAn_STAT.ch_st* = 0).

To disable the 10-bit timer, set the *DMAn_CFG.pssel* field to 0.

Normally, the 10-bit timer starts when the channel is enabled and the *DMAn_CFG.pssel* field is non-zero. However, if *DMAn_CFG.reqwait* = 1, then the timer starts counting only after the first DMA request is received from the peripheral.

To calculate the time-out period, use *Equation 7-1*, *below*.

Equation 7-1: Timeout Equation for Standard DMA

$$T_{timeout} = T_{HCLK} \times N_{pssel} \times N_{tosel}$$

For example, if $T_{HCLK} = \frac{1}{90MHz}$, $N_{pssel} = 0x2$ (65,536 timer prescaler), and $N_{tosel} = 0x3$ (32 clocks), then the time-out calculation is:

$$T_{timeout} = \left(\frac{1}{90{,}000{,}000}\right) \times 65{,}536 \times 32 = 23.3ms$$

7.11 Channel and Register Access Restrictions

Writing to any register while a channel is disabled is supported, but there are certain restrictions when a channel is enabled. The *DMAn_STAT.ch_st* bit indicates whether the channel is enabled or not.

Because an active channel might be in the middle of an AHB read/write burst, do not write to the *DMAn_SRC*, *DMAn_DST*, or *DMAn_CNT* registers while a channel is active (*DMAn_STAT.ch_st* = 1).

To disable any DMA channel, clear the *DMAn_CFG.chen* bit. Then, poll the *DMAn_STAT.ch_st* bit to verify that the channel is disabled.

7.12 Memory-to-Memory DMA

Memory-to-memory transfers are completed as if the request is always active. This means that the DMA channel generates an almost constant request for the bus until its transfer is complete. For this reason, assign a lower priority to channels executing memory-to-memory transfers to prevent starvation of other DMA channels.

7.13 Standard DMA Control Registers

The DMA base peripheral address is 0x4002 8000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 7-6: Standard DMA Control Registers, Offsets, Access and Descriptions

Offset	Register	Access	Description
[0x0000]	*DMA_INT_EN*	R/W	DMA Control register
[0x0004]	*DMA_INT_FL*	RO	DMA Interrupt Status register

7.13.1 Standard DMA Control Register Details

Table 7-7: DMA Interrupt Enable Register

DMA Interrupt Enable Register			DMA_INT_EN	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved for Future Use** Do not modify this field.
3:0	chien	R/W	0	**Channel Interrupt Enable** Each bit in this field enables the corresponding channel interrupt. 0: Channel interrupt disabled 1: Channel interrupt enabled

Table 7-8: DMA Interrupt Flag Register

DMA Interrupt Flag Register			DMA_INT_FL	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved for Future Use** Do not modify this field.

DMA Interrupt Flag Register			DMA_INT_FL	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
30	ipend	RO	0	**Channel Interrupt** Each bit in this field represents an interrupt for the corresponding channel. To clear an interrupt, clear the corresponding active interrupt bit in the *DMAn_STAT* register. An interrupt bit in this field is set only if the corresponding channel interrupt enable field is set in the *DMAn_CFG* register. 0: No interrupt 1: Interrupt pending

7.14 Standard DMA Channel 0 to 3 Register Base Addresses

Each DMA channel has a set of associated Configuration Registers as shown in *Table 7-10*. Access to a channel's configuration registers requires adding the channel base address from *Table 7-9* and the offset of the desired configuration register from *Table 7-10*. For example, the address for DMA Channel 3's DMA3_DST_RLD register is DMA Channel 3 Base Address, 0x4002 0160, plus the offset of the *DMAn_DST_RLD* register, [0x0018] which gives the address 0x4002 0178 for DMA3_DST_RLD.

Table 7-9: Standard DMA Channel 0 to Channel 15 Offsets

Channel Base Address	DMA Channel	Access	Description
0x4002 8100	0	R/W	DMA Channel 0
0x4002 0120	1	R/W	DMA Channel 1
0x4002 0140	2	R/W	DMA Channel 2
0x4002 0160	3	R/W	DMA Channel 3

7.15 Standard DMA Channel Configuration Register Offsets

Table 7-10: DMAn Channel Registers, Offsets, Access and Descriptions

Address	Register	Access	Description
[0x0000]	*DMAn_CFG*	R/W	DMA Channel Configuration Register
[0x0004]	*DMAn_STAT*	R/W	DMA Channel Status Register
[0x0008]	*DMAn_SRC*	R/W	DMA Channel Source Register
[0x000C]	*DMAn_DST*	R/W	DMA Channel Destination Register
[0x0010]	*DMAn_CNT*	R/W	DMA Channel Count Register
[0x0014]	*DMAn_SRC_RLD*	R/W	DMA Channel Source Reload Register
[0x0018]	*DMAn_DST_RLD*	R/W	DMA Channel Destination Reload Register
[0x001C]	*DMAn_CNT_RLD*	R/W	DMA Channel Count Reload Register

7.15.1 Standard DMA Channel Configuration Register Details

Table 7-11: DMA Configuration Register

DMA Configuration Register			DMAn_CFG	[0x0100]
Bits	**Name**	**Access**	**Reset**	**Description**
31	ctzien	R/W	0	**CTZ Interrupt Enable** When enabled, the *DMA_INT_FL.ipend* bit is set to 1 whenever a CTZ event occurs. 0: Interrupt disabled 1: Interrupt enabled
30	chdien	R/W	0	**Channel Disable Interrupt Enable** When enabled, the *DMA_INT_FL.ipend* bit is set to 1 whenever the *DMAn_STAT.ch_st* bit changes from 1 to 0. 0: Interrupt disabled 1: Interrupt enabled
29	-	RO	0	**Reserved for Future Use** Do not modify this field.
28:24	brst	R/W	0	**Burst Size** The number of bytes transferred into and out of the DMA FIFO in a single burst. 0b00000: 1 byte 0b00001: 2 bytes 0b00010: 3 bytes ... 0b11111: 32 bytes
23	-	RO	0	**Reserved for Future Use** Do not modify this field.
22	distinc	R/W	0	**Destination Increment Enable** This bit enables the automatic increment of the *DMAn_DST* register upon every AHB transaction. This bit is forced to 0 for a DMA transmit to peripherals. 0: Increment disabled 1: Increment enabled
21:20	dstwd	R/W	0	**Destination Width** Indicates the width of each AHB transaction to the destination peripheral or memory (the actual width might be less than this if there are insufficient bytes in the DMA FIFO for the full width). 0b00: Byte 0b01: Two bytes 0b10: Four bytes 0b11: Reserved (Byte width if set)
19	-	RO	0	**Reserved for Future Use** Do not modify this field.
18	srinc	R/W	0	**Source Increment Enable** This bit enables the automatic increment of the *DMAn_SRC* register upon every AHB transaction. This bit is forced to 0 for a DMA receive from peripherals. 0: Increment disabled 1: Increment enabled

DMA Configuration Register			DMAn_CFG	[0x0100]
Bits	**Name**	**Access**	**Reset**	**Description**
17:16	srcwd	R/W	0	**Source Width** Indicates the width of each AHB transaction from the source peripheral or memory. The actual width might be less than this if the *DMAn_CNT* register indicates a smaller value. 00: Byte 01: Two bytes 10: Four bytes 11: Reserved (byte width if set)
15:14	pssel	R/W	0	**Pre-Scale Select** Selects the divider for the channel's 10-bit timer. 00: Disable timer 01: $f_{hclk} / 256$ 10: $f_{hclk} / 64K$ 11: $f_{hclk} / 16M$
13:11	tosel	R/W	0	**Time-Out Select** Selects the number of prescaler clocks seen by the channel timer before a time-out condition is generated for this channel. 000: 3-4 001: 7-8 010: 15-16 011: 31-32 100: 63-64 101: 127-128 110: 255-256 111: 511-512
10	reqwait	R/W	0	**Request Wait Enable** When enabled, delay the timeout timer start until after the first DMA transaction occurs. 0: Start timer normally 1: Delay timer start
9:4	reqsel	R/W	0	**Request Select** Select DMA request line for this channel. If memory to memory is selected, then the channel operates as if the request is always active.
3:2	pri	R/W	0	**DMA priority** 00: Highest priority 11: Lowest priority
1	rlden	R/W	0	**Reload Enable** Setting this bit to 1 allows reloading the *DMAn_SRC*, *DMAn_DST*, and *DMAn_CNT* registers with their corresponding reload registers upon CTZ. *Note: This bit is also writeable in the DMAn_CNT_RLD register.*
0	chen	R/W	0	**Channel Enable** This bit is automatically cleared when *DMAn_STAT.ch_st* changes from 1 to 0. 0: Disable this channel 1: Enable this channel

Table 7-12: DMA Status Register

DMA Status Register			DMAn_STAT	[0x0104]
Bits	**Name**	**Access**	**Reset**	**Description**
31:7	-	RO	0	**Reserved for Future Use** Do not modify this field.
6	to_st	R/W1C	0	**Time-Out Status** A time-out occurred if this field reads 1. Write 1 to clear. 0: No time out 1: A time out has occurred
5	-	RO	0	**Reserved for Future Use** Do not modify this field.
4	bus_err	R/W1C	0	**Bus Error** If this bit reads 1, an AHB abort occurred and the channel was disabled by hardware. Reading this bit indicates the following: 0: No error found 1: An AHB bus error occurred
3	rld_st	R/W1C	0	**Reload Status** This bit is set by hardware when reload is enabled and a reload occurred. Write 1 to clear. 0: Reload event has not occurred. 1: Reload event occurred.
2	ctz_st	R/W1C	0	**CTZ Status** This bit is set by hardware when a Count-to-zero (CTZ) has occurred, if enabled. Write 1 to clear. 0: CTZ has not occurred 1: CTZ has occurred
1	ipend	RO	0	**Channel Interrupt** This field is set when any enabled channel interrupt occurs. When this field is set, examine the other fields in this register, *DMAn_STAT*, to determine the cause of the Channel Interrupt. Clearing the status bits in this register clears this field. This field is read-only. 0: No channel interrupt pending. 1: Channel Interrupt pending.
0	ch_st	RO	0	**Channel Status** This bit is used to indicate when it is safe to change the configuration, address, and count registers for the channel. This field is read-only. Hardware clears this bit automatically when the channel is not active. When hardware clears this bit, hardware also clears the *DMAn_CFG.chen* bit. 0: Channel configuration can be changed. 1: Channel not safe to change configuration.

Table 7-13: DMA Source Register

DMA Source Register			DMAn_SRC	[0x0108]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	src	R/W	0	**Source Device Address** For peripheral transfers, the actual address field is either ignored or forced to zero because peripherals only have one location to read/write data based on the request select chosen. If *DMAn_CFG.srcinc* = 1: • This register is incremented on each AHB transfer cycle by one, two, or four bytes depending on the data width. If *DMAn_CFG.srcinc* = 0: • This register remains constant. If a CTZ condition occurs while *DMAn_CFG.rlden* = 1, then this register is reloaded with the contents of the *DMAn_SRC_RLD* register.

Table 7-14: DMA Destination Register

DMA Destination Register			DMAn_DST	[0x010C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	dst	R/W	0	**Destination Device Address** For peripheral transfers, the actual address field is either ignored or forced to zero because peripherals only have one location to read/write data based on the request select chosen. If *DMAn_CFG.dstinc* = 1, then this register is incremented on every AHB transfer cycle by one, two, or four bytes depending on the data width. If a CTZ condition occurs while *DMAn_CFG.rlden* = 1, then this register is reloaded with the contents of the *DMAn_DST_RLD* register.

Table 7-15: DMA Count Register

DMA Count Register			DMAn_CNT	[0x0110]
Bits	**Name**	**Access**	**Reset**	**Description**
31:24	-	RO	0	**Reserved for Future Use** Do not modify this field.
23:0	cnt	R/W	0	**DMA Counter** Load this register with the number of bytes to transfer. This counter decreases on every AHB access to DMA FIFO. The decrement is one, two, or four bytes depending on the data width. When the counter reaches 0, a CTZ condition is triggered. If a CTZ condition occurs while *DMAn_CFG.rlden* = 1, then this register is reloaded with the contents of the *DMAn_CNT_RLD* register. 0x000000: 0 Byte 0x000001: 1 Byte 0x000002: 2 Bytes … 0xFFFFFF:16,777,215 Bytes

Table 7-16: DMA Source Reload Register

DMA Source Reload Register			DMAn_SRC_RLD	[0x0114]
Bits	**Name**	**Access**	**Reset**	**Description**
31	-	RO	0	**Reserved for Future Use** Do not modify this field.
30:0	src_rld	R/W	0	**Source Address Reload Value** If *DMAn_CFG.rlden* = 1, then the value of this register is loaded into *DMAn_SRC* upon a CTZ condition.

Table 7-17: DMA Destination Reload Register

DMA Destination Reload Register			DMAn_DST_RLD	[0x0118]
Bits	**Name**	**Access**	**Reset**	**Description**
31	-	RO	0	**Reserved for Future Use** Do not modify this field.
30:0	dst_rld	R/W	0	**Destination Address Reload Value** If *DMAn_CFG.rlden* = 1, then the value of this register is loaded into *DMAn_DST* upon a CTZ condition.

Table 7-18: DMA Count Reload Register

DMA Count Reload Register			DMAn_CNT_RLD	[0x011C]
Bits	**Name**	**Access**	**Reset**	**Description**
31	rlden	R/W	0	**Reload Enable.** Enables automatic loading of the DMAn_SRC, *DMAn_DST*, and *DMAn_CNT* registers when a CTZ event occurs. Set this bit after the address reload registers are programmed. This bit is automatically cleared to 0 when reload occurs. 0: Reload disabled 1: Reload enabled *Note: This bit is also seen in the DMAn_CFG register.*
30:24	-	RO	0	**Reserved for Future Use** Do not modify this field.
23:0	cnt_rld	R/W	0	**Count Reload Value.** If *DMAn_CNT_RLD.rlden* = 1, then the value of this register is loaded into *DMAn_CNT* upon a CTZ condition.

8 UART

The MAX32660 microcontroller provides up to two industry-standard UART ports which can communicate with external devices using standard serial communications protocols. The UARTs are full-duplex Universal Asynchronous Receiver/Transmitter (UART) serial ports. Both UARTs, UART0 and UART1, support identical functionality and registers unless specifically noted otherwise. For simplicity, the UARTs are referenced in the documentation as UARTn where n = 0 or 1. The registers for each UART are documented showing an offset address, which is identical for each UART instance. To access a specific UART's control register, the UART's control register offset is added to the specific UART's base peripheral address.

8.1.1 Features:

- Flexible baud rate generation up to 4Mbps
- Programmable character size, 5, 6, 7, or 8-bits
- Stop bit settings of 1, 1.5, or 2-bits
- Parity settings of even, odd, mark (always 1), space (always 0), and no parity
- Automatic parity error detection with selectable parity bias
- Automatic framing error detection
- Separate 32-byte deep transmit and receive FIFOs
- Flexible interrupt conditions
- Hardware flow control for RTS and CTS
- Null modem support
- Break generation and detection
- Wakeup from *DEEPSLEEP* on UART edge with no character loss
- RX Timeout detection

8.2 UART Frame Characters

Character sizes of 5 to 8 bits are supported. The field *UARTn_CTRL0.charsize* is used to select the character size.

Stop bit support includes 1, 1.5, and 2 stop bits selected with the register field *UARTn_CTRL0*.

Parity support includes even, odd, mark, space or none. For no parity, set field *UARTn_CTRL0.parity_en* to 0. For all other parity options, select one of the four parity options using the *UARTn_CTRL0.parity_mode* field and enable parity (*UARTn_CTRL0.parity_en*=1). Parity can be based on the number of 1 bits or 0 bits in the receive characters as set in the register bit *UARTn_CTRL0.parity_lvl.*

Break frames are transmitted by setting the field *UARTn_CTRL0.break* to 1. A break sets all bits in the frame to 0.

When a break frame is received, two interrupts are available, *UARTn_INT_FL.break* is set to 1 when the first received break character is received and *UARTn_INT_FL.last_break* is set when the last break character is received. This prevents the system from being overloaded with multiple interrupts that could occur after the first break character and up to the Nth break character received.

Note: A break character does not set the frame error flag because breaks are not valid UART characters.

8.3 UART Interrupts

Interrupts can be generated for the following conditions:

- The Transmit FIFO level is equal or less than the set transmit threshold.
- The Receive FIFO level is equal or greater than the set receieve threshold.
- The Receive FIFO is overrun, which means the Receive FIFO is full but is still receiving data
- Any CTS state change. During Hardware Flow Control, this interrupt is generated either because:
  - CTS is deasserted, which tells the UART to pause transmitting data
  - CTS is asserted, which tells the UART to resume transmitting data
- A Receive Parity Error occurred
- A Receive Frame Error occurred, which means START or STOP bits were not detected
- A Receive Timeout condition occurred, which means the RX FIFO has not received a character for a set time
- First and Last BREAK characters

8.4 UART Bit Rate Calculation

The UART peripheral clock, f_{PCLK}, is used as the input clock to the UART bit rate generator. The following fields are used to set the target bit rate for the UART.

- *UARTn_BAUD0.clk_div* selects the bit rate clock divisor.
- *UARTn_BAUD0.ibaud* sets the integer portion of the bit rate divisor.
- *UARTn_BAUD1.dbaud* sets the decimal portion of the bit rate divisor.

Equation 8-1, *Equation 8-2*, and *Equation 8-3* are used to determine the values for each of the bit rate fields required to achieve a target bit rate for the UART.

Equation 8-1: UART Bit Rate Divisor Equation

$$DIV = \frac{f_{UART\_BIT\_RATE\_CLK}}{(Clock\ Divider \times Target\ Bit\ Rate)}$$

Note: UARTn_BAUD0.clkdiv should be set to the highest value that results in $\lfloor DIV \rfloor \geq 1$ to achieve the highest accuracy for the target bit rate.

Equation 8-2: Bit Rate Integer Calculation

$$UARTn\_BAUD0.ibaud = \lfloor DIV \rfloor$$

Equation 8-3: Bit Rate Remainder Calculation

$$UARTn\_BAUD1.dbaud = (DIV - UARTn\_BAUD0.ibaud) \times 128$$

8.4.1 Example Baud Rate Calculation:

$$Target\ Bit\ Rate = 1{,}843{,}200\ bits\ per\ second\ (1.8\ Mbps)$$

$$f_{BIT\_RATE\_CLK} = f_{PCLK} = 48\ MHz$$

$$DIV = \frac{48{,}000{,}000}{(Clock\ Divider\ \times 1{,}843{,}200)}, where\ Clock\ Divider = 2^{(7-clkdiv)}$$

Table 8-1: Example Baud Rate Calculation Results, Target Bit Rate = 1.8Mbps, f_{PCLK}=48MHz

UARTn_BAUD0 clkdiv	Clock Divider	DIV
4	8	3.256
3	16	1.628
2	32	0.814
1	64	0.407
0	128	0.203

Table 8-1, *above*, shows the DIV result for each of the *UARTn_BAUD0.clkdiv* field settings. With the Clock Divider set to 8 or 16, the resulting DIV value is greater than 1. Setting the clock divider to 16 will generate the most accurate target bit rate because it is the largest value that results in DIV ≥ 1. Using 16 for Clock Divider, *UARTn_BAUD0.clkdiv* = 3), *UARTn_BAUD0.ibaud* is 1, which is the integer portion of the 1.628 DIV calculation. The *UARTn_BAUD1.dbaud* field calculation based on *UARTn_BAUD0.clkdiv* = 3, *UARTn_BAUD0.ibaud* = 1 and DIV = 1.628 is:

Equation 8-4: UART dbaud Example Calculation

$$UARTn\_BAUD1.dbaud = (1.628 - 1) \times 128 \rightarrow 80.384$$

The resulting field settings for the example 1,843,200 bps rate are:

```
UARTn_BAUD0.clkdiv = 3
UARTn_BAUD0.ibaud = 1
UARTn_BAUD1.dbaud = 80
```

8.5 UART DMA Using the TX and RX FIFOs

Each UART has a 32-byte TX FIFO with a dedicated DMA channel and a 32-byte RX FIFO with a dedicated DMA channel. The DMA channels are configured using the DMA Configuration Register, *UARTn_DMA*. The RX FIFO DMA channel and TX FIFO DMA channels operate independently, and each can be enabled or disabled individually. Enable the RX FIFO DMA channel by setting *UARTn_DMA.rxdma_en* to 1 and enable the TX FIFO DMA channel by setting the *UARTn_DMA.txdma_en* to 1. DMA transfers are automatically triggered based on the number of bytes in the RX or TX FIFO as described in the following two sections.

8.5.1 RX FIFO DMA Operation

UARTn_DMA.rxdma_lvl configures the number of entries in the RX FIFO that triggers a DMA transfer from the RX FIFO to system RAM. If the number of entries in the RX FIFO is more than the configured value, a DMA transfer is triggered from the RX FIFO to system RAM. If *UARTn_DMA.rxdma_lvl*=0 then a transfer is triggered when there is one byte in the FIFO.

Note: The RX DMA level must be set to a value less than 32 to avoid an RX FIFO overrun condition that results in loss of received data.

8.5.2 TX FIFO DMA Operation

UARTn_DMA.txdma_lvl sets the number of entries (level) in the TX FIFO that will trigger a DMA transfer from system RAM to the TX FIFO. If the number of entries (level) in the TX FIFO falls below this value a TX DMA transfer is automatically triggered from System RAM to the TX FIFO.

Note: The TX DMA level must be set to a value greater than 1 to avoid stalling the UART transfer.

8.6 Flushing the UART FIFOs

The FIFOs can be flushed independently by setting *UARTn_CTRL0.rxflush* to 1 for the RX FIFO and *UARTn_CTRL0.txflush.* to 1 for the TX FIFO. The TX FIFO and RX FIFO are automatically flushed if the UART is disabled by clearing the *UARTn_CTRL0.enable* field (*UARTn_CTRL0.enable* = 0).

8.7 Hardware Flow Control

When hardware flow control is enabled, the CTS (Clear-to-send) and RTS (Request-to-Send) external signals are directly managed by hardware without CPU intervention. RTS and CTS are active when flow control is enabled by setting the register bit *UARTn_CTRL0*.flowctl=1. The polarity of the CTS/RTS signals are configured with register bit *UARTn_CTRL0.flowpol* and can be active low or active high.

In operation, the host UART that wants to transmit data asserts the RTS output pin and waits for the CTS input pin to be asserted. If CTS is asserted, then the host UART begins transmitting data to the slave UART. If during the transmission the host UART notices CTS is deasserted, the host UART finishes transmitting the current character and then pauses to wait for CTS to return to an asserted level before transmitting more data.

If this UART is receiving data, and the RX FIFO reaches the level set in the 6-bit register field *UARTn_CTRL1.rts_fifo_lvl,* then the RTS signal of this UART is deasserted, informing the transmitting UART to stop sending data to this UART to prevent data overflow. Transmission resumes when the level of the RX FIFO drops below *UARTn_CTRL1.rts_fifo_lvl,* which automatically asserts RTS.

8.8 UART Registers

The UART0 base peripheral address is 0x4004 2000 and the UART1 base peripheral address is 0x4004 3000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 8-2: UART Registers, Offset Addresses and Descriptions

Offset	Register Name	Access	Description
[0x0000]	*UARTn_CTRL0*	R/W	UARTn Control 0 Register
[0x0004]	*UARTn_CTRL1*	R/W	UARTn Control 1 Register
[0x0008]	*UARTn_STAT*	RO	UARTn Status Register
[0x000C]	*UARTn_INT_EN*	R/W	UARTn Interrupt Enable Register
[0x0010]	*UARTn_INT_FL*	R/1	UARTn Interrupt Flag Register
[0x0014]	*UARTn_BAUD0*	R/W	UARTn Baud Rate Integer Register
[0x0018]	*UARTn_BAUD1*	R/W	UARTn Baud Rate Decimal Register
[0x001C]	*UARTn_FIFO*	R/W	UARTn FIFO Read/Write Register
[0x0020]	*UARTn_DMA*	R/W	UARTn DMA Configuration Register
[0x0024]	*UARTn_TXFIFO*	RO	UARTn TX FIFO Register

8.8.1 UART Register Details

Table 8-3: UART Control 0 Register

UART Control 0 Register			UARTn_CTRL0	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:24	-	R/W	0	**Reserved for Future Use** Do not modify this field.

UART Control 0 Register			UARTn_CTRL0	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
23:16	to_cnt	R/W	0	**RX Timeout Frame Count** Set this field to the number of frames before a Receive Timeout occurs. If the RX FIFO contains data, a RX Timeout condition occurs if the time for the number of frames in this register passes without the FIFO receiving any new data. If a timeout occurs, the hardware sets the receive timeout flag to 1 (*UARTn_INT_FL.rxto* = 1). 0: 256 1: 1 2: 2 3: 3 … 254: 254 255: 255
15	clk_sel	R/W	0	**Bit Rate Clock Source Select** Selects the bit rate clock, $f_{\mathrm{UART\_BIT\_RATE\_CLK}}$ 0: Peripheral Clock, $f_{UART\_BIT\_RATE\_CLK} = f_{PCLK}$ 1: Reserved for Future Use
14	break	R/W	0	**Transmit BREAK Frame** Set this field to 1 to send a BREAK frame. A BREAK frame transmits a character with all bits set to 0. 0: Normal UART operation. 1: Transmit BREAK frame.
13	nullmod	R/W	0	**Null Modem Support** 0: Normal operation for RTS/CTS and TXD/RXD 1: Null Modem Mode: RTS/CTS swapped, TXD/RXD swapped
12	flowpol	R/W	0	**RTS/CTS Polarity** 0: RTS/CTS asserted is 0 1: RTS/CTS asserted is 1
11	flow	R/W	0	**Hardware Flow Control Enable** 0: Hardware flow control disabled. 1: Hardware RTS/CTS flow control enabled.
10	stop	R/W	0	**STOP Bit Mode Select** 0: 1 STOP bit. 1: 1.5 STOP bits for 5-bit character size or 2 STOP bits for all other character sizes
9:8	size	R/W	0	**Character Size** Set the number of data bits per frame. 0: 5 data bits 1: 6 data bits 2: 7 data bits 3: 8 data bits
7	bitacc	R/W	0	**Frame or Bit Accuracy Select** This field selects between either Frame Accuracy or Bit Accuracy for transmitting data. Frame Accuracy: Individual frame bit durations may be varied by hardware to meet the target frame period. Bit accuracy: Bit width is fixed by hardware. The frame accuracy of data transmitted may be reduced if bit accuracy is prioritized. 0: Frame accuracy. 1: Bit accuracy. *Note: A frame includes the start, stop, all data bits, and parity bit/bits for the character being transmitted.*

UART Control 0 Register			UARTn_CTRL0	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
6	rxflush	R/W1O	0	**Receive FIFO Flush** Write 1 to flush the receive FIFO Cleared to 0 by hardware when flush is completed
5	txflush	R/W1O	0	**Transmit FIFO Flush** Write 1 to flush the Transmit FIFO Cleared to 0 by hardware when flush is completed
4	parity_lvl	R/W	0	**Parity Level Select** 0: Parity is based on number of 0 bits in the character. 1: Parity is based on number of 1 bits in the character.
3:2	parity_mode	R/W	0	**Parity Mode Select** 0: Even parity 1: Odd Parity 2: Mark parity 3: Space parity
1	parity_en	R/W	0	**Parity Enable** 0: No parity 1: Parity enabled as charsize+1 bit
0	enable	R/W	0	**UART Enable** 0: UART disabled. FIFOs are flushed, bit rate generator is off. 1: UART Enabled, bit rate generator is active.

Table 8-4: UART Control 1 Register

UART Control 1 Register			UARTn_CTRL1	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:22	0	R/W	0	**Reserved for Future Use** Do not modify this field.
21:16	rts_fifo_lvl	R/W	0	**RTS RX FIFO Threshold Level** When the RX FIFO level is equal to or greater than this level, assert RTS output signal to inform the transmitting UART to stop sending data to this UART. Valid values are from 0 to 32.
15:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:8	tx_fifo_lvl	R/W	0	**TX FIFO Threshold Level** When the TX FIFO level is less than or equal to this level the *UARTn_INT_FL.tx_fifo_lvl* interrupt flag is set. Valid values are from 0 to 32.
7:6	-	R/W	0	**Reserved for Future Use** Do not modify this field.
5:0	rx_fifo_lvl	R/W	0	**RX FIFO Threshold Level** When the RX FIFO reaches this level or higher the *UARTn_INT_FL.rx_fifo_lvl* interrupt flag is set. Valid values are from 0 to 32.

Table 8-5: UART Status Register

UART Status Register			UARTn_STAT	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:25	-	RO	0	**Reserved for Future Use** Do not modify this field.
24	rx_to	RO	0	**RX Timeout** This field is set to 1 when a receive timeout occurs. This field is set by hardware when the condition occurs and is automatically cleared when the condition is no longer valid.
23:22	-	RO	0	**Reserved for Future Use** Do not modify this field.
21:16	tx_num	RO	0	**Number of Bytes in the TX FIFO** Read this field to determine the number of bytes in the transmit FIFO.
15:14	-	RO	0	**Reserved for Future Use** Do not modify this field.
13:8	rx_num	RO	0	**Number of Byes in RX FIFO** Read this field to determine the number of bytes in the receive FIFO.
7	tx_full	RO	0	**TX FIFO Full Status Flag** This field reads 1 when the TX FIFO is full. This field is set by hardware when the condition occurs and is automatically cleared when the condition is no longer valid. 0: TX FIFO is not full. 1: TX FIFO is full.
6	tx_empty	RO	1	**TX FIFO Empty Flag** This field reads 1 when the TX FIFO is empty. This field is set by hardware when the condition occurs and is automatically cleared when the condition is no longer valid. 0: TX FIFO is not empty, *UARTn_STAT*.*tx_num* > 0. 1: TX FIFO is empty.
5	rx_full	RO	0	**RX FIFO Full Flag** This field reads 1 when then RX FIFO is full. This field is set by hardware when the condition occurs and is automatically cleared when the condition is no longer valid. 0: RX FIFO is not full. 1: RX FIFO is full.
4	rx_empty	RO	1	**RX FIFO Empty Flag** This flag reads 1 when the RX FIFO is empty.
3	break	RO	0	**Break Flag** This field is set when a break condition occurs. 0: BREAK not received. 1: BREAK condition received.
2	parity	RO	0	**Parity Bit State** This field returns the state of the parity bit. 0: Parity bit is 0. 1: Parity bit is 1.
1	rx_busy	RO	0	**RX Busy** This field reads 1 when the UART is receiving data. 0: UART is not actively receiving data. 1: UART is actively receiving data.

UART Status Register			UARTn_STAT	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
0	tx_busy	RO	0	**TX Busy** This field reads 1 when the UART is transmitting data. 0: UART is not actively transmitting data. 1: UART is transmitting data.

Table 8-6: UART Interrupt Enable Register

UART Interrupt Enable Register			UARTn_INT_EN	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:10	-	R/W	0	**Reserved for Future Use** Do not modify this field.
9	last_break	R/W	0	**Last Break Interrupt Enable** When the UART receives a series of BREAK frames, this enables an interrupt when the last BREAK frame is received. 0: Interrupt disabled. 1: Interrupt enabled.
8	rx_to	R/W	0	**RX Timeout Interrupt Enable** Enable the receive timeout interrupt. 0: Interrupt disabled. 1: Interrupt enabled.
7	break	R/W	0	**Received BREAK Interrupt Enable** Enables the BREAK interrupt for the first BREAK received on the UART. 0: Interrupt disabled. 1: Interrupt enabled.
6	tx_fifo_lvl	R/W	0	**TX FIFO Threshold Level Interrupt Enable** Set this field to 1 to enable an interrupt when the number of entries in the transmit FIFO is less than or equal to the *UARTn_CTRL1.tx_fifo_lvl* field. 0: Interrupt disabled. 1: Interrupt enabled.
5	tx_fifo_ae	R/W	0	**TX FIFO Almost Empty Interrupt Enable** Enables an interrupt indicating the transmit FIFO is almost empty, one byte remaining. 0: Interrupt disabled. 1: Interrupt enabled.
4	rx_fifo_lvl	R/W	0	**RX FIFO Threshold Level Interrupt Enable** Set this field to 1 to enable an interrupt when the number of entries in the receive FIFO is greater than or equal to the *UARTn_CTRL1.rx_fifo_lvl* field. 0: Interrupt disabled. 1: Interrupt enabled.
3	rx_overrun	R/W	0	**RX FIFO Overrun Interrupt Enable** Enables an interrupt when a write is made to a full RX FIFO 0: Interrupt disabled. 1: Interrupt enabled.
2	cts	R/W	0	**CTS State Change Interrupt Enable** Enable the CTS level change interrupt event. 0: Interrupt disabled. 1: Interrupt enabled.

UART Interrupt Enable Register			UARTn_INT_EN	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
1	rx_parity_error	R/W	0	**RX Parity Error Interrupt Enable** Enables an interrupt when a receive parity error is detected. 0: Interrupt disabled. 1: Interrupt enabled.
0	rx_frame_error	R/W	0	**RX Frame Error Interrupt Enable** Enables an interrupt when a receive frame error is detected. 0: Interrupt disabled. 1: Interrupt enabled.

Table 8-7: UART Interrupt Flags Register

UART Interrupt Flags Register			UARTn_INT_FL	[0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
31:10	-	RO	0	**Reserved for Future Use** Do not modify this field.
9	last_break	R/W1C	0	**Last Break Interrupt Flag** When the UART receives a series of BREAK frames, this flag is set when the last BREAK frame is received. Write 1 to clear this field. 0: Last BREAK condition has not occurred. 1: Last BREAK condition has occurred.
8	rx_to	R/W1C	0	**Receive Frame Timeout Interrupt Flag** This field is set when a receive frame timeout occurs. Write 1 to clear this field. 0: Receive frame timeout has not occurred. 1: A receive frame timeout was detected by the UART.
7	break	R/W1C	0	**Received Break Interrupt Flag** When the UART receives a series of BREAK frames, this flag is set when the first BREAK frame is received. Write 1 to clear this field.
6	tx_fifo_lvl	R/W1C	0	**Transmit FIFO Threshold Interrupt Flag** This interrupt flag is set when number of entries in in the Transmit FIFO is less than or equal to the Transmit FIFO level set in *UARTn_CTRL1.tx_fifo_lvl.* Write 1 to clear. 0: TX FIFO level is greater than the *UARTn_CTRL1.tx_fifo_lvl*. 1: TX FIFO level is equal to or less than the *UARTn_CTRL1.tx_fifo_lvl*. *Note: This flag is set immediately by hardware when the condition exists. To prevent additional interrupts from occurring after clearing this flag, write data to the TX FIFO or decrease the TX FIFO threshold level.*
5	tx_fifo_ae	R/W1C	0	**Transmit FIFO Almost Empty Interrupt Flag** This field is set when there is one byte remaining in the Transmit FIFO. Write 1 to clear.
4	rx_fifo_lvl	R/W1C	0	**RX FIFO Threshold Interrupt Flag** Set when number of entries in the RX FIFO is equal to or greater than the RX FIFO threshold level as set in the *UARTn_CTRL1.rx_fifo_lvl* field. Data must be read from the RX FIFO to reduce the level below the threshold to guarantee this interrupt does not occur again after clearing the flag. Write 1 to clear this field. 0: The number of bytes in the RX FIFO is below the threshold level. 1: The number of bytes in the RX FIFO is equal to or greater than the threshold level.

UART Interrupt Flags Register		UARTn_INT_FL		[0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
3	rx_ovr	R/W1C	0	**RX FIFO Overrun Interrupt Flag** This field is set if the receive FIFO is full and an additional byte is received resulting in a FIFO overrun condition. If this field is set at least one byte of received data has been lost. Write 1 to clear. 0: RX FIFO overrun has not occurred. 1: RX FIFO overrun occurred.
2	cts	R/W1C	0	**CTS Interrupt Flag** CTS, also referred to as Modem Status Interrupt, flag. Write 1 to clear.
1	parity	R/W1C	0	**Receive Parity Error Status Flag** Set if a parity error is detected. This flag applies to data received only. Write 1 to clear. 0: Parity error has not been detected. 1: Parity error detected.
0	frame	R/W1C	0	**Frame Error Status Flag** Set if a frame error occurs while receiving data. Write 1 to clear.

Table 8-8: UART Rate Integer Register

UART Baud Rate Integer Register			UARTn_BAUD0	[0x0014]
Bits	**Name**	**Access**	**Reset**	**Description**
31:17	-	R/W	0	**Reserved for Future Use** Do not modify this field.
18:16	clkdiv	R/W	0	**Bit Rate Clock Divisor** This field is used to divide the bit rate clock by the selected Clock Divider value. Refer to the *UART Bit Rate Calculation* section for details of determining this field's value for a given UART bit rate. **clkdiv** / **Clock Divider Value** 0 / 128 1 / 64 2 / 32 3 / 16 4 / 8 5 - 7 / Reserved for Future Use
15:12	-	R/W	0	**Reserved for Future Use** Do not modify this field.
11:0	ibaud	R/W	0	**Integer Portion of Baud Rate Divisor** This field contains the integer value of the bit rate divisor. Refer to the *UART Bit Rate Calculation* section for details of determining this field's value for a given UART bit rate.

Table 8-9: UART Baud Rate Decimal Register

UART Baud Rate Decimal Register			UARTn_BAUD1	[0x0018]
Bits	**Name**	**Access**	**Reset**	**Description**
31:12	-	R/W	0	**Reserved for Future Use** Do not modify this field.

UART Baud Rate Decimal Register		UARTn_BAUD1		[0x0018]
Bits	**Name**	**Access**	**Reset**	**Description**
11:0	dbaud	R/W	0	**Decimal Portion of Baud Rate Divisor** This field contains the remainder portion of the bit rate divisor. Refer to the *UART Bit Rate Calculation* section for details of determining this field's value for a given UART bit rate.

Table 8-10: UART FIFO Register

UART FIFO Register		UARTn_FIFO		[0x001C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	R/W	0	**Reserved for Future Use** Do not modify this field.
7:0	fifo	R/W	N/A	**UART FIFO Register** Reading this field reads data from the RX FIFO and writes to this field write to the TX FIFO.

Table 8-11: UART DMA Configuration Register

UART DMA Configuration Register		UARTn_DMA		[0x0020]
Bits	**Name**	**Access**	**Reset**	**Description**
31:22	-	R/W	0	**Reserved for Future Use** Do not modify this field.
21:16	rxdma_lvl	R/W	0	**RX FIFO Level DMA Trigger** If the RX FIFO level is greater than this value, the DMA channel transfers data from the RX FIFO into memory. DMA transfers continue until the RX FIFO is empty. To avoid an RX FIFO overrun, do not set this value to 32. Values above 32 are reserved for future use.
15:14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:8	txdma_lvl	R/W	0	**TX FIFO Level DMA Trigger** If the TX FIFO level is less than this value, the DMA channel transfers data from memory into the TX FIFO. DMA transfers continue until the TX FIFO is full. To avoid stalling a UART transmission, do not set this value to 1 or 0. Values above 32 are reserved for future use.
7:2	-	R/W	0	**Reserved for Future Use** Do not modify this field.
1	rxdma_en	R/W	0	**RX FIFO DMA Channel Enable** 0: RX DMA is disabled 1: RX DMA is enabled
0	txdma_en	R/W	0	**TX FIFO DMA Channel Enable** 0: TX DMA is disabled 1: TX DMA is enabled

Table 8-12: UART TX FIFO Data Output Register

UART TX FIFO Data Output Register		UARTn_TXFIFO		[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	R/W	0	**Reserved for Future Use** Do not modify this field.

UART TX FIFO Data Output Register			UARTn_TXFIFO	[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
7:0	data	RO	0	**TX FIFO Data Output Peek Register** Reads from this register return the next character available for transmission at the end of the TX FIFO. If no data is available, 0x00 is returned. Reads from this register do not affect the TX FIFO state.

9 Real-Time Clock (RTC)

9.1 Overview

The Real-Time Clock (RTC) is a binary timer that keeps the time of day and provides time-of-day and sub-second alarm functionality in the form of RTC system interrupts. The RTC time base is created using a 32.768kHz crystal connected between the 32KIN and 32KOUT pins on the MAX32660. See the MAX32660 datasheet for detailed connection and pin information related to the 32KIN and 32KOUT pins.

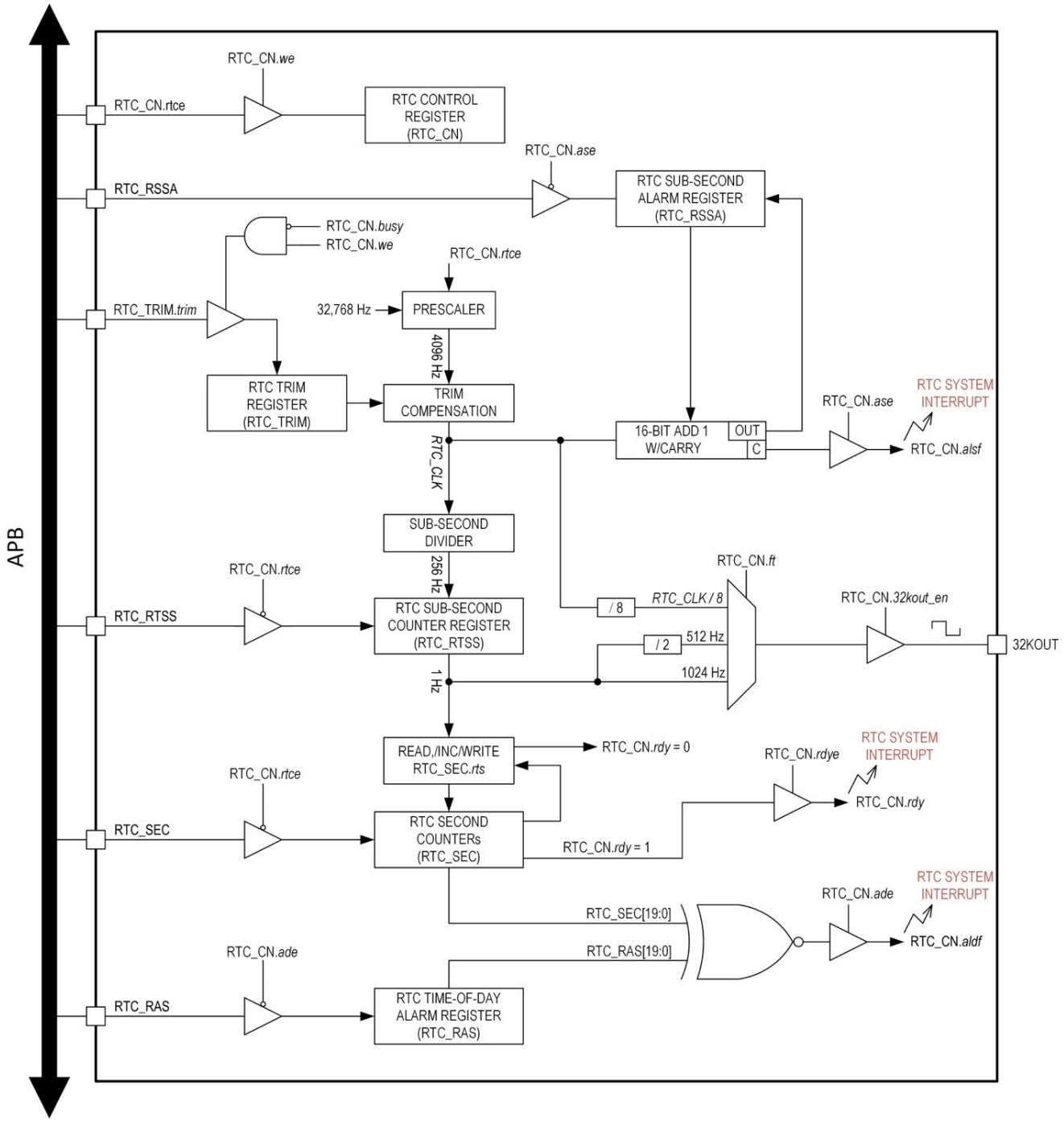
In the RTC, two registers combine to create a 40-bit counter representing time with 1/256 second resolution. The *RTC_SSEC.rtss* field contains the least significant 8 bits and represents the sub-second count. The *RTC_SEC.rts* field contains the most significant 32 bits and represents the seconds count. The *RTC_SEC.rts* field increments on each rollover of the *RTC_SSEC.rtss* field. Together the 40 bits represent time in seconds up to approximately 136 years.

A programmable time-of-day alarm is usable with the 32-bit seconds counter to provide a single event/alarm timer. You must disable the RTC to write the counter registers. When the RTC counter is started, the RTC counts continuously unless it is disabled, and reads of the counter registers do not affect the count. Digital trim is available for applications requiring higher accuracy.

A separate 32-bit auto-reload sub-second alarm counter register (*RTC_RSSA*) generates interval alarms. Incremented at 256Hz, this counter has a granularity of 3.9 msec, with a maximum interval of approximately 16,777,216 seconds.

The RTC operates in the always-on domain. Once enabled, it continues counting as long as the RTC is enabled and the VRTC supply remains within the acceptable range given in the datasheet. The RTC increments the *RTC_TRIM.vrtc_tmr* field every 32 seconds when the RTC is enabled and operating.

Figure 9-1. RTC Block Diagram



9.2 RTC Alarm Functions

The RTC provides time-of–day and sub-second interval alarm functions. The time-of-day alarm is implemented by matching the count values in the counter register with the value stored in the alarm register. The sub-second interval alarm provides an auto-reload timer that is driven by the trimmed RTC clock source.

9.2.1 Time-of-Day Alarm

Program the RTC Time-of-Day Alarm register (*RTC_RAS*) to configure the time-of-day-alarm. The alarm triggers when the value stored in *RTC_RAS* matches the lower 20 bits of the *RTC_SEC.rts* seconds count register. This allows programming the time-of-day-alarm to any future value between 1 second and 12 days relative to the current time with a resolution of 1 second. You must disable the time-of-day alarm before changing the *RTC_RAS.tod* field.

When the alarm occurs, hardware sets the Time-of-Day Alarm Interrupt Flag (*RTC_CTRL.alarm_tod_fl*) to 1.

Setting the *RTC_CTRL.alarm_tod_fl* bit to 1 in software results in an interrupt request to the processor if the Alarm Time-of-Day Interrupt Enable (*RTC_CTRL.alarm_tod_en*) bit is set to 1, and the RTC's system interrupt enable is set.

9.2.2 Sub-Second Alarm

The *RTC_RSSA* and *RTC_CTRL.alarm_ss_en* field control the sub-second alarm. Writing RTC_RSSA sets the starting value for the sub-second alarm counter. Writing the Sub-Second Alarm Enable (*RTC_CTRL.alarm_ss_en*) bit to 1 enables the sub-second alarm. Once enabled, the sub-second alarm begins up-counting from the RTC_RSSA value. When the counter rolls over from 0xFFFF FFFF to 0x0000 0000, hardware sets the *RTC_CTRL.alarm_ss_fl* bit triggering the alarm. At the same time, hardware also reloads the counter with the value previously written to *RTC_RSSA.rssa.* A 256Hz clock drives the sub-second alarm allowing a maximum interval of 16,777,216 seconds with a resolution of approximately 3.9 msec.

You must disable the sub-second interval alarm, *RTC_CTRL.alarm_ss_en,* prior to changing the interval alarm value, *RTC_RSSA*.

The delay (uncertainty) associated with enabling the sub-second alarm is up to one period of the sub-second clock, approximately 3.9 msec based on 256Hz RTC clock input to the register. This uncertainty is propagated to the first interval alarm. Thereafter, if the interval alarm remains enabled, the alarm triggers after each sub-second interval as defined without the first alarm uncertainty because the sub-second alarm is an auto-reload timer. Enabling the sub-second alarm with with the sub-second alarm register set to 0 (*RTC_RSSA.rssa* = 0) results in the maximum sub-second alarm interval.

9.2.3 RTC Wakeup From DEEPSLEEP/BACKUP Power Modes

The RTC alarms are an optional wakeup source for the MAX32660 during *DEEPSLEEP*/*BACKUP* mode. Perform the following steps to use the RTC as a *DEEPSLEEP*/*BACKUP* wakeup source:

1. Configure the RTC Time-of-Day Alarm for the required number of seconds.
2. Create a RTC IRQ handler function and register the address of the RTC IRQ handler using the NVIC.
3. Enable the RTC time of day interrupt enable, (*RTC_CTRL.alarm_tod_en* = 1).
4. Enable the System wakeup for the RTC by setting the *GCR_PM.rtcwk_en* field to 1.
5. Enter the desired low power mode. Refer to section *Operating Modes* for details on entering *DEEPSLEEP* or *BACKUP* mode.

9.3 RTC Register Access

Restricted access to specific registers prevents software reading from or writing to the RTC registers while they are updated by the RTC hardware.

9.3.1 RTC Register Write Protection

The *RTC_CTRL.busy* bit is a read-only status bit controlled by hardware and set when any of the following conditions occur:

- System Reset.
- Software writes to the *RTC_SEC* register or RTC trim registers.
- Software modifies the *RTC_CTRL.enable, RTC_CTRL.alarm_tod_en,* or *RTC_CTRL.alarm_ss_en* bits.

When the *RTC_CTRL.busy* bit is set by hardware, writes to the above RTC control bits and count registers are blocked by hardware. The *RTC_CTRL.busy* bit remains active until the register or bit is synchronized by hardware. The synchronization by hardware occurs on the next rising edge of the 32kHz clock. The *RTC_CTRL.busy* bit is set for a maximum of one 4kHz clock, approximately 250µs. Therefore, a software write is not complete until hardware clears the *RTC_CTRL.busy* bit indicating that a 32kHz synchronized version of the registers and bit are in place.

Once the *RTC_CTRL.busy* bit is cleared to 0, additional writes are completed as permitted by individual count or alarm-enable bits.

9.3.2 RTC Register Read Protection

The Ready (*RTC_CTRL*.ready) bit indicates when the RTC count registers contain valid data. Hardware clears the *RTC_CTRL.ready* bit approximately one 4kHz clock before the ripple occurs through the RTC counter registers (*RTC_SEC* and *RTC_SSEC*) and is set once again immediately after the ripple occurs. The period of the *RTC_CTRL.ready* bit set/clear activity is approximately 3.9 msec, providing a large window during which the RTC count registers are readable. Software can clear the *RTC_CTRL.ready* bit at any time and the bit remains clear until set by hardware when the next ripple occurs. A separate Ready Enable (*RTC_CTRL*.ready_int_en) bit is provided to generate an interrupt when hardware sets the *RTC_CTRL.ready* bit. You can use this interrupt to signal the start of a new RTC read window.

9.3.3 RTC Count Register Access

The RTC count registers (*RTC_SEC* and *RTC_SSEC*) are readable when the *RTC_CTRL.ready* bit is set to 1. Data read from these registers when *RTC_CTRL.ready* is 0 is invalid. To write the RTC count registers, set the RTC Enable (*RTC_CTRL*.enable) bit to 0. Clearing the *RTC_CTRL.enable* bit is permitted only when the Write Enable (*RTC_CTRL*.write_en) bit is set to 1 and is governed by the *RTC_CTRL.busy* bit signaling process (that is, the *RTC_CTRL.busy* bit is 0). Writes to each RTC count register must occur only when the *RTC_CTRL.busy* bit reads 0.

9.3.4 RTC Alarm Register Access

The RTC alarm registers (*RTC_RAS* and *RTC_RSSA*) are readable at any time. To write to an alarm register, disable the corresponding alarm enable first (*RTC_CTRL.alarm_ss_en* = 0 or *RTC_CTRL.alarm_tod_en* = 0). Clearing these bits requires monitoring the *RTC_CTRL.busy* bit to assess completion of the write. Once the alarm is disabled, update the associated RTC alarm registers using software.

9.3.5 RTC Trim Register Access

The RTC Trim register (*RTC_TRIM*) is readable at any time. To write to this register, set the Write Enable (*RTC_CTRL*.write_en) bit to 1 and check the *RTC_CTRL.busy* bit until it reads 0 and then write the *RTC_TRIM* register.

9.3.6 RTC Oscillator Control Register Access

The RTC oscillator control register (*RTC_OSCCTRL*) is readable at any time. To write to this register, set the Write Enable (*RTC_CTRL*.write_en) bit to 1 and check the *RTC_CTRL.busy* bit until it reads 0 and then write to the *RTC_OSCCTRL* register.

9.4 RTC Output Pin

The RTC is capable of outputting the raw 4KHz signal or a trim compensated 1KHz or 512Hz signal to the 32KCAL alternate pin function. On both the 16 WLP package and the 20 TQFN package for the MAX32660 the 32KCAL is alternate function 2 on pin P0.2. P0.2 corresponds to GPIO0[2].

9.5 RTC Calibration

The uncompensated accuracy of the RTC is a function of the attached crystal. A digital trim facility allows the device to compensate for up to ± 127ppm (parts per million) as designated by the *RTC_TRIM.trim* register field.

Complete the following steps to measure a square wave output on the 32KCAL alternate function pin and determine the accuracy of the RTC:

1. Enable the 32KCAL alternate function. Refer to the *RTC Output Pin* section for details.
2. Set the *RTC_CTRL.freq_sel* field to the desired output frequency.
3. Set *RTC_CTRL.32kout_en* to 1, enabling the square wave output on the 32KCAL alternate pin function.
4. Measure the square wave output and compare it to an accurate reference clock.
5. Set *RTC_CTRL.write_en* to 1 and adjust the *RTC_TRIM* register.
6. Repeat steps 1 through 5 as necessary until optimum accuracy is achieved.

9.6 RTC Registers

The RTC base peripheral address is 0x4000 6000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 9-1. RTC Registers, Offsets and Descriptions

Offset	Register	Access	Description
[0x0000]	*RTC_SEC*	R/W	Seconds Counter Register
[0x0004]	*RTC_SSEC*	R/W	Sub-Seconds Counter Register
[0x0008]	*RTC_RAS*	R/W	Alarm Time-of-Day Register
[0x000C]	*RTC_RSSA*	R/W	Sub-Second Alarm Register
[0x0010]	*RTC_CTRL*	R/W	Control Register
[0x0014]	*RTC_TRIM*	R/W	Trim Register
[0x0018]	*RTC_OSCCTRL*	R/W	Oscillator Control Register

9.6.1 RTC Register Details

Table 9-2: RTC Seconds Counter Register

RTC Seconds Counter Register			RTC_SEC	[0x00]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	rts	R/W	-	**Seconds Counter** This register is the 32-bit count of seconds.

Table 9-3: RTC Sub-Seconds Counter Register

RTC Sub-Seconds Counter Register			RTC_SSEC	[0x04]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	R/W	0	**Reserved for Future Use** Do not modify this field.
7:0	rtss	R/W	-	**Sub-Seconds Counter** This field represents sub-seconds and increments at 256Hz. When this field rolls from 0xFF to 0x00, the *RTC_SEC.count* increments.

Table 9-4: RTC Sub-Seconds Counter Register

RTC Alarm Time-of-Day Register			RTC_RAS	[0x08]
Bits	**Name**	**Access**	**Reset**	**Description**
31:20	-	R/W	0	**Reserved for Future Use** Do not modify this field.
19:0	ras	R/W	0	**Time-of-Day Alarm** Sets the time-of-day alarm from 1 second up to 12-days. When this field matches *RTC_SEC[19:0]*, an RTC system interrupt is generated.

Table 9-5: RTC Sub-Second Alarm Register

RTC Sub-Second Alarm Register			RTC_RSSA	[0x0C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	rssa	R/W	0	**Sub-second Alarm** Sets the starting value for the sub-second alarm. The sub-second alarm increments at 256Hz providing an alarm interval of up to 16,777,216 seconds in increments of 3.9 msec. An alarm is generated when the counter rolls from 0xFFFF FFFF to 0x0000 0000.

Table 9-6: RTC Control Register

RTC Control Register			RTC_CTRL	[0x10]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.
15	write_en	R/W	0	**Write Enable** Set this field to 1 to write to the *RTC_TRIM* register, the RTC enable (*RTC_CTRL.enable*) bit, or both. 1: Writes to the *RTC_TRIM* register and the *RTC_CTRL.enable* bit are allowed. 0: Writes to the *RTC_TRIM* register and the *RTC_CTRL.enable* bit are ignored.
14:13	-	R/W	0	**Reserved for Future Use** Do not modify this field.

RTC Control Register			RTC_CTRL	[0x10]
Bits	**Name**	**Access**	**Reset**	**Description**
12:11	x32k_mode	R/W	0	**32kHz Oscillator Mode Select** Selects the operating mode for the 32kHz oscillator. 0: Operates in noise immunity mode 1: Operates in quiet mode. Oscillator warm-up is not required. 2: Operates in noise immunity mode when the processor is in active modes and switches to quiet mode when the processor enters *DEEPSLEEP*. The system waits for the 32kHz oscillator to warm-up prior to the processor exiting stop mode. 3: Operates in noise immunity mode when the processor is in active modes and switches to quiet mode when the processor enters stop mode. The system does not wait for the 32kHz oscillator to warm-up prior to the processor exiting stop mode and beginning code execution.
10:9	freq_sel	R/W	0	**Frequency Output Select** Selects the output frequency to output on the 32KCAL alternate function output pin if the *RTC_CTRL.32kout_en* bit is set to 1 and the GPIO is enabled for the alternate pin function 32KCAL; unused otherwise. 0b00: 1Hz (Compensated) 0b01: 512Hz (Compensated) 0b1x: 4kHz
8	32kout_en	R/W	-	**Square Wave Output Enable** 0: Square wave output disabled. 1: Square wave is output on the 32KCAL alternate function pin with the frequency determined by the *RTC_CTRL.freq_sel* field. *Note: This bit is set to 0 on a POR and is not affected by other resets.*
7	alarm_ss_fl	R/W	0	**Sub-second Alarm Interrupt Flag** This interrupt flag is set when a sub-second alarm condition occurs. This flag is a wake-up source for the processor. 0: No sub-second alarm pending. 1: Sub-second interrupt pending.
6	alarm_tod_fl	R/W	0	**Time-of-Day Alarm Interrupt Flag** This interrupt flag is set by hardware when a time-of-day alarm occurs. This flag is a wake-up source for the processor. 0: No Time-of-Day alarm interrupt pending. 1: Time-of-day interrupt pending.
5	ready_int_en	R/W	0	**RTC Ready Interrupt Enable** This interrupt flag is set when the RTC ready bit is set by hardware. 0: Interrupt disabled. 1: Interrupt enabled.
4	ready	R/W0O	0	**RTC Ready** This bit is set to 1 by hardware when the *RTC_SEC* register is updated. Software can clear this bit at any time. Hardware automatically clears this bit just prior to updating the *RTC_SEC* register, indicating the RTC is busy. 0: *RTC_SEC* register not updated. 1: *RTC_SEC* register updated.
3	busy	RO	0	**RTC Busy Flag** This bit is set by hardware when changes to the RTC registers are synchronized. The bit is automatically cleared by hardware when the synchronization is complete. Software should poll this field for 0 after changing RTC registers to ensure the change is complete prior to making any other RTC register changes. 0: RTC not busy. 1: RTC busy.

RTC Control Register			RTC_CTRL	[0x10]
Bits	**Name**	**Access**	**Reset**	**Description**
2	alarm_ss_en	R/W	0	**Sub-Second Alarm Interrupt Enable** Set this bit to 1 to enable the RTC sub-second alarm interrupt. Check the *RTC_CTRL.busy* flag after writing this bit to determine when the RTC synchronization is complete. 0: Alarm sub-second interrupt disabled. 1: Enable alarm sub-second interrupt.
1	alarm_tod_en	R/W	0	**Time-of-Day Alarm Interrupt Enable** Set this bit to 1 to enable the RTC time-of-day alarm interrupt. Check the *RTC_CTRL.busy* flag after writing to this bit to determine when the RTC synchronization is complete. 0: Time-of-day alarm interrupt is disabled. 1: Enable the time-of-day alarm interrupt.
0	enable	R/W	0	**Real-Time Clock Enable** Enables and disables the RTC. The RTC write enable (*RTC_CTRL.write_en*) bit must be set before changing this field. RTC Busy (*RTC_CTRL.busy*) must read 0 before writing to this bit (*RTC_CTRL.write_en*). After writing to this bit, check the *RTC_CTRL.busy* flag for 0 to determine when the RTC synchronization is complete. 0: RTC disabled. 1: RTC enabled.

Table 9-7: RTC Trim Register

RTC Trim Register			RTC_TRIM	[0x14]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	vrtc_tmr	R/W	0	**VRTC Time Counter** This field is used to show the number of seconds the RTC has since the RTC was enabled. Hardware increments this field every 32 seconds. *Note: This field is reset on a Power On Reset (POR).*
7:0	trim	R/W	0	**RTC Trim** This field specifies the 2s complement value of the trim resolution. Each increment or decrement of the field adds or subtracts 1ppm at each 4kHz clock value with a maximum correction of ± 127ppm.

Table 9-8: RTC Oscillator Control Register

RTC Oscillator Control Register			RTC_OSCCTRL	[0x18]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.
5	32kout	R/W	0	**RTC Square Wave Output** 0: 32kHz signal is not output to port pin (POR default). 1: Outputs the raw 32kHz clock to the 32KCAL alternate function if the alternate function is enabled. *Note: This field is only reset on POR and not effected by other forms of reset.*

RTC Oscillator Control Register			RTC_OSCCTRL	[0x18]
Bits	**Name**	**Access**	**Reset**	**Description**
4	bypass	R/W	0	**RTC Crystal Bypass** Bypass the crystal oscillator to allow a digital square wave to be driven on the 32KIN pin. 0: Disable bypass (POR default) 1: Enable bypass *Note: This field is only reset on POR and not effected by other forms of reset.*
3	ibias_en	R/W	1	**RTC Oscillator Bias Current Enable** Enables 4× or 2× bias current selected by *RTC_OSCCTRL.ibias_sel* in noise immunity mode 0: Disable 1: Enable (POR default) *Note: This field is only reset on POR and not effected by other forms of reset.*
2	hyst_en	R/W	0	**RTC Oscillator Hysteresis Buffer Enable** Enables the RTC hysteresis buffer in noise immunity mode. This increases DC current consumption by ~144nA. 0: Disable (POR default) 1: Enable *Note: This field is only reset on POR and not effected by other forms of reset.*
1	ibias_sel	R/W	0	**RTC Oscillator 4× Bias Current Select** 0: selects 2× bias current for RTC oscillator (POR default). 1: selects 4× bias current for RTC oscillator. *Note: This field is only reset on POR and not effected by other forms of reset.*
0	filter_en	R/W	1	**RTC Oscillator Filter Enable** 0: Disable RTC oscillator filter 1: Enable RTC oscillator filter (POR default) *Note: This field is only reset on POR and not effected by other forms of reset.*

10 Timers

The MAX32660 contains three 32-bit, reloadable timers. Each timer provides multiple operating modes. Timer modes supported include:

- One-Shot: Timer counts up to terminal value then halts.
- Continuous: Timer counts up to terminal value then repeats.
- Counter: Timer counts input edges received on timer input pin (Timer 0 only).
- Capture: Captures a snapshot of the current timer count when timer input edge transitions (Timer 0 only).
- Compare: Timer pin toggles when timer exceeds terminal count (Timer 0 only).
- Gated: Timer increments only when timer input pin is asserted (Timer 0).
- Capture/Compare: Timer counts when timer input is asserted, captures timer count when input is deasserted (Timer 0 only).

10.1 Features

- 32-bit reload counter
- Programmable prescaler with values from 1 to 4096
- Independent interrupt
- Timer 0 supports a Timer I/O alternate function pin for:
  - Capture, compare, and capture/compare capability
  - Timer pin available as alternate function
  - Configurable Input pin for event triggering, clock gating, or capture signal
  - Configurable output pin for event output and PWM signal generation

10.2 Basic Operation

The timer modes operate by incrementing the *TMRn_CNT* register, driven by either the timer clock, an external stimulus on the timer pin, or a combination of both. The *TMRn_CNT* register is always readable, even while the timer is enabled and counting.

Each timer mode has a user-configurable timer period, which terminates on the timer clock cycle following the end of timer period condition. Each timer mode has a different response at the end of a timer period, which can include changing the state of the timer pin, capturing a timer value, reloading *TMRn_CNT* with a new starting value, or disabling the counter. The end of a timer period will always set the corresponding interrupt bit and can generate an interrupt, if enabled.

In most modes the timer peripheral automatically sets *TMRn_CNT* to 0x0000 0001 at the end of a timer period, but *TMRn_CNT* is set to 0x0000 0000 following a system reset. This means the first timer period following a system reset will be one timer clock longer than subsequent timer periods if *TMRn_CNT* is not initialized to 0x0000 0001 during the timer configuration step.

Clocking of timer functions is driven by the timer clock frequency, f_{CNT_CLK}. The timer clock frequency is a user-configurable, division of the system peripheral clock, PCLK. Each timer has an independent prescaler, allowing timers to operate at different frequencies. The prescaler can be set from 1 to 4096 using the *TMRn_CN.pres3*:*TMRn_CN.pres* fields. Unless otherwise mentioned, the timer clock is generated as follows:

Equation 10-1: Timer Clock Frequency Calculation

$$f_{CNT\_CLK} = \frac{f_{PCLK}}{prescaler}$$

Application firmware writes to the timer registers and external events on timer pins will be asynchronous events to the slower timer clock frequency. These events are latched on the next rising edge of the timer clock. Since it is not possible to observe the timer clock directly, input events may have up to 0.5 timer clock delay before being recognized.

10.3 Timer Pin Functionality

On the MAX32660, only Timer 0 supports a Timer Pin. The timer pin functionality is mapped as an alternate function that is shared with a GPIO. Timer pin assignments are detailed in the data sheet for the specific device.

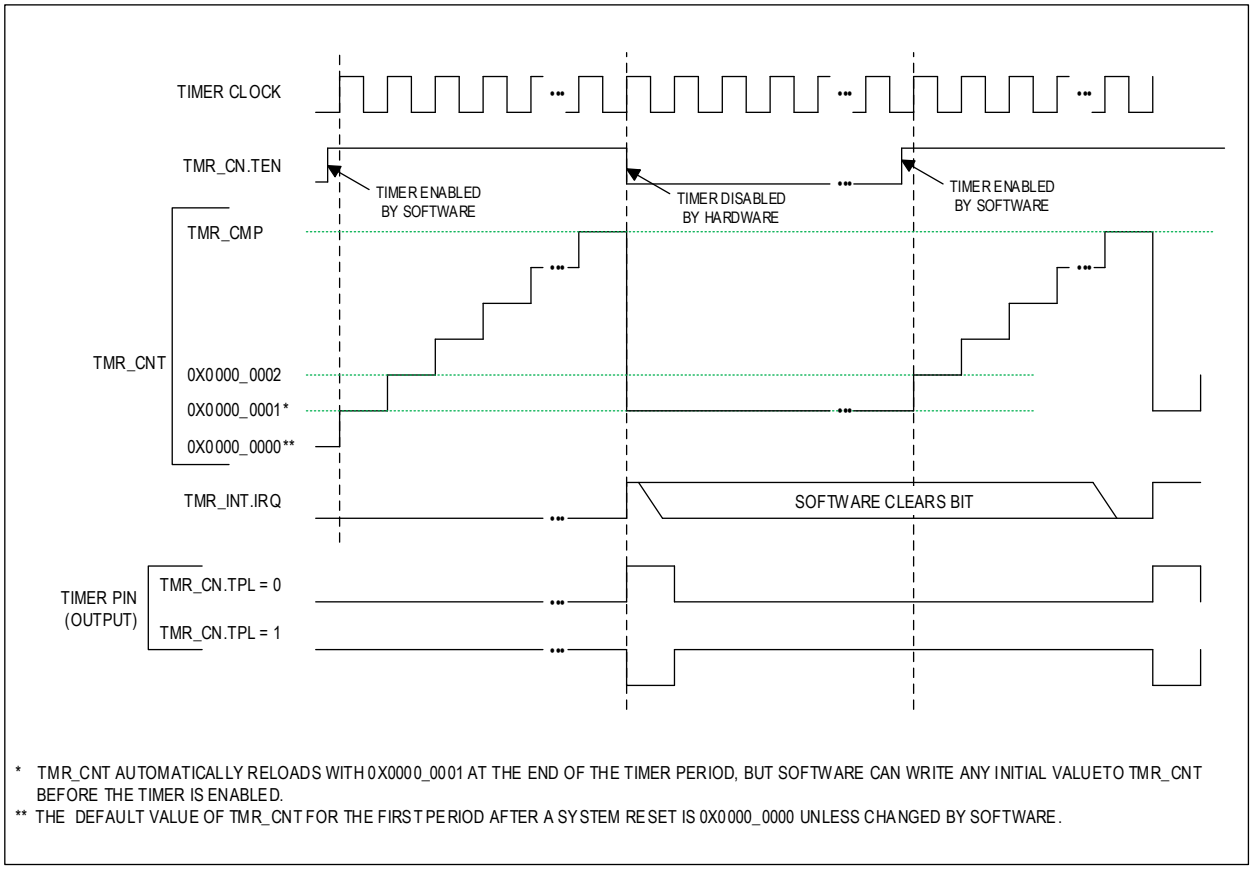
When the timer pin alternate function is enabled, the timer pin will have the same electrical characteristics, such as pullup/pulldown strength, drive strength, etc. as the GPIO mode settings for that pin. When configured as an output, the corresponding bit in the GPIO_OUT register should be configured to match the inactive state of the timer pin for that mode. The pin characteristics must be configured before enabling the timer. Consult the GPIO section for details on how to configure the electrical characteristics for the pin.

Each timer has a dedicated interrupt flag, *TMRn_INT.irq*, which is set at the end of a timer period. If enabled, an interrupt will be generated. The interrupt flag can be cleared by writing any value to *TMRn_INT.irq*.

10.4 One-Shot Mode (000b)

In One-shot mode the timer peripheral increments *TMRn_CNT* until it matches *TMRn_CMP* and then stops incrementing and disables the timer. The timer can optionally output a pulse on the timer pin at the end of the timer period. In this mode, the timer must be re-enabled to start another one-shot mode event.

Figure 10-1: One-Shot Mode Diagram



10.4.1 Timer Period

The timer period ends on the timer clock following *TMRn_CNT* = *TMRn_CMP*.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. *TMRn_CNT* is reset to 0x0000 0001.
2. The timer is disabled by setting *TMRn_CN.ten* = 0.
3. If the timer output is enabled, the timer pin is driven to its active state for one timer clock. It then returns to its inactive state.
4. The timer interrupt bit *TMRn_INT.irq* will be set. An interrupt will be generated if enabled.

10.4.2 Configuration

Configure the timer for One-Shot mode by doing the following:

1. Set *TMRn_CN.ten* = 0 to disable the timer
2. Set *TMRn_CN.tmode* to 000b to select One-shot mode.
3. Set *TMRn_CN.pres3*:*TMRn_CN.pres* to set the prescaler that determines the timer frequency.
4. If using the timer pin:
   a. Configure the pin as a timer output and configure the electrical characteristics as needed.
   b. Set *TMRn_CN.tpol* to match the desired (inactive) state.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write an initial value to *TMRn_CNT*, if desired. This effects only the first period; subsequent timer periods always reset *TMRn_CNT*= 0x0000 0001.
7. Write the compare value to *TMRn_CMP*.
8. Set *TMRn_CN.ten* = 1 to enable the timer.

The timer period is calculated using the following equation:

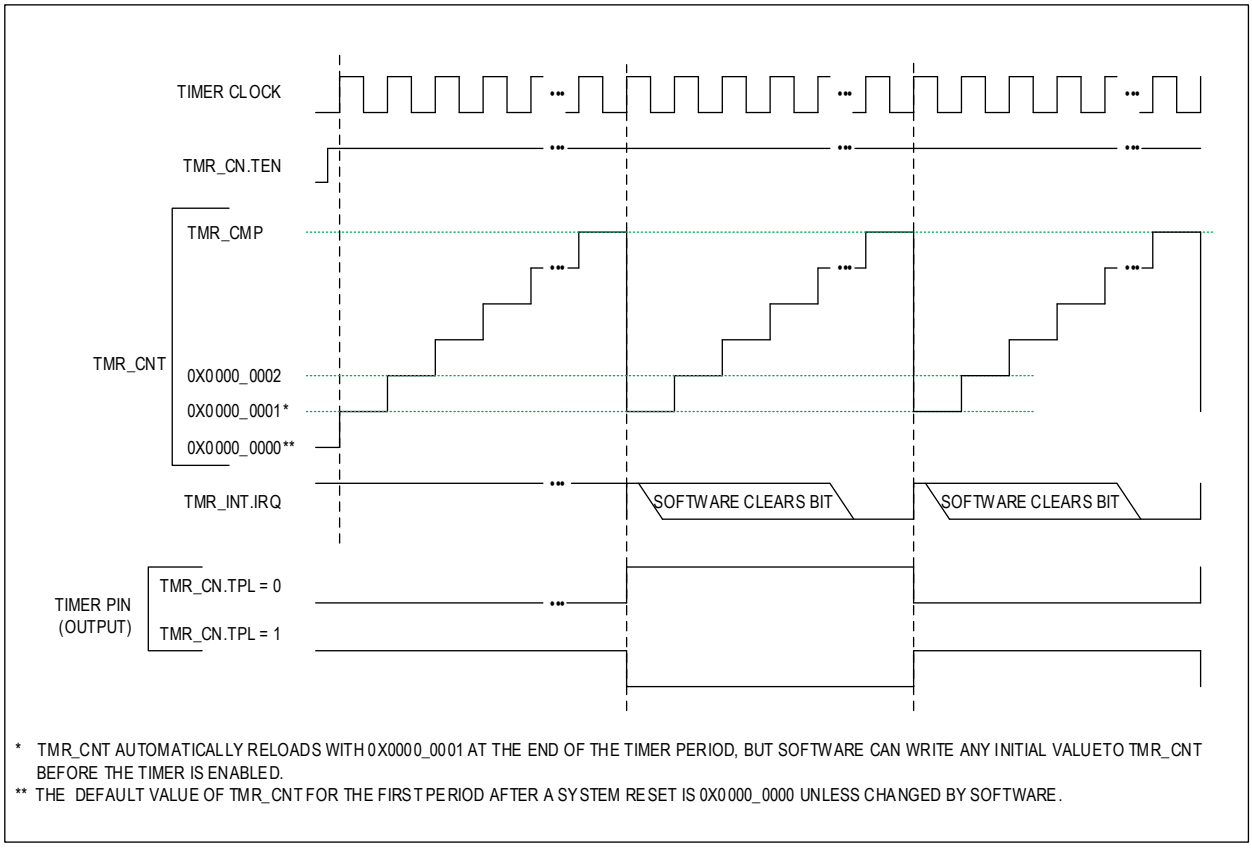
Equation 10-2: One-shot Mode Timer Period

$$One\text{-}shot\ mode\ timer\ period\ in\ seconds = \frac{TMR\_CMP - TMR\_CNT_{INITIAL\_VALUE} + 1}{f_{CNT\_CLK}\ (Hz)}$$

10.5 Continuous Mode (001b)

In Continuous mode, the timer peripheral increments *TMRn_CNT* until it matches *TMRn_CMP*, resets *TMRn_CNT* to 0x0000 0001, and continues incrementing. The timer peripheral can optionally toggle the state of the timer pin at the end of the timer period.

Figure 10-2: Continuous Mode Diagram



10.5.1 Timer Period

The timer period ends on the timer clock following *TMRn_CNT* = *TMRn_CMP*.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. *TMRn_CNT* is reset to 0x0000 0001. The timer remains enabled and continues incrementing.
2. If the timer output is enabled, the timer pin toggles state (low to high or high to low).
3. The timer interrupt bit *TMRn_INT.irq* will be set. An interrupt will be generated if enabled.

10.5.2 Configuration

Configure the timer for Continuous mode by performing the following steps:

1. Set *TMRn_CN.ten* = 0 to disable the timer.
2. Set *TMRn_CN.tmode* to 001b to select Continuous mode.
3. Set *TMRn_CN.pres3*:*TMRn_CN.pres* to set the prescaler that determines the timer frequency.
4. If using the timer pin:
   a. Configure the pin as a timer output and configure the electrical characteristics as needed.
   b. Set *TMRn_CN.tpol* to match the desired (inactive) state.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write an initial value to *TMRn_CNT*, if desired. This effects only the first period; subsequent timer periods always reset *TMRn_CNT*= 0x0000 0001.
7. Write the compare value to *TMRn_CMP*.
8. Set *TMRn_CN.ten* = 1 to enable the timer.

The timer period is calculated using the following equation.

Equation 10-3: Continuous Mode Timer Period

$$Continuous\ mode\ timer\ period\ in\ seconds = \frac{TMR\_CMP - TMR\_CNT_{INITIAL\_VALUE} + 1}{f_{CNT\_CLK}\ (Hz)}$$

10.6 Counter Mode (010b)

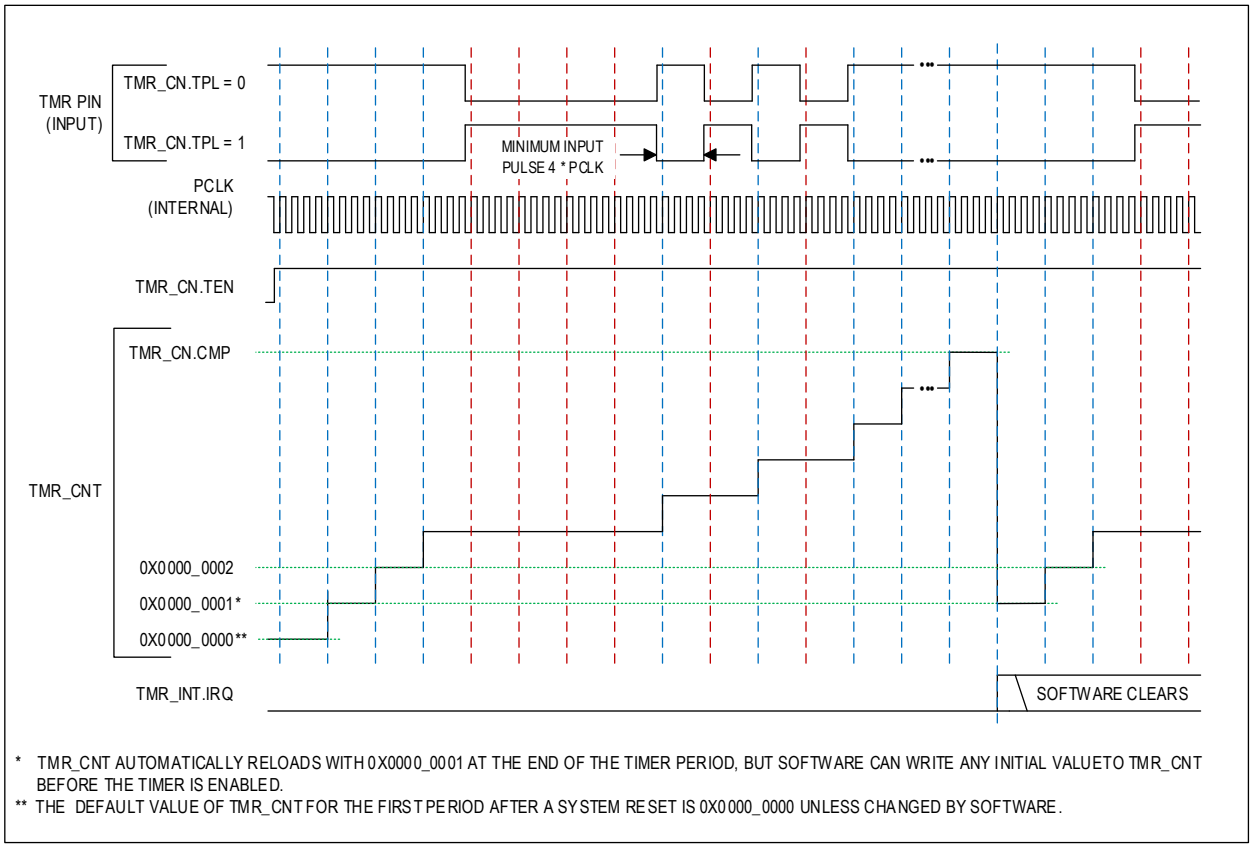
In Counter mode, the timer peripheral increments *TMRn_CNT* when a transition occurs on the timer pin. When *TMRn_CNT* = TMR.CMP, the interrupt bit is set, *TMRn_CNT* is set to 0x0000 0001, and continues incrementing. The timer can be configured to increment on either the rising edge or the falling edge, but not both.

The timer prescaler setting has no effect in this mode. The frequency of the timer's input signal (f_{CNT_CLK}) must not exceed 25 percent of the PCLK frequency as shown in the following equation:

Equation 10-4: Counter Mode Maximum Clock Frequency

$$f_{CNT\_CLK} \leq \frac{PCLK\ (Hz)}{4}$$

Figure 10-3: Counter Mode Diagram



10.6.1 Timer Period

The timer period ends on the rising edge of PCLK following *TMRn_CNT* = *TMRn_CMP*.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. *TMRn_CNT* is reset to 0x0000 0001. The timer remains enabled and continues incrementing on selected transitions of the timer pin.
2. The timer interrupt bit *TMRn_INT.irq* will be set. An interrupt will be generated if enabled.

10.6.2 Configuration

Configure the timer for Counter mode by doing the following:

1. Set *TMRn_CN.ten* = 0 to disable the timer.
2. Set *TMRn_CN.tmode* to 010b to select Counter mode.
3. Configure the timer pin:
   a. Configure the pin as a timer input and configure the electrical characteristics as needed.
   b. Set *TMRn_CN.tpol* to match the desired initial (inactive) state.
4. If using the timer interrupt, enable the interrupt and set the interrupt priority.
5. Write an initial value to *TMRn_CNT*, if desired. This effects only the first period; subsequent timer periods always reset *TMRn_CNT*= 0x0000 0001.
6. Write the compare value to *TMRn_CMP*.
7. Set *TMRn_CN.ten* = 1 to enable the timer.

In Counter mode, the number of timer input transitions since timer start is calculated using the following equation:

Equation 10-5: Counter Mode Timer Input Transitions

$$Counter\ mode\ timer\ input\ transitions = TMR\_CNT_{CURRENT\_COUNT\_VALUE} - TMR\_CNT_{START\_VALUE}$$

10.7 PWM Mode (011b)

In PWM mode, the timer sends a Pulse-Width Modulated (PWM) output using the timer's output signal. The timer first counts up to the match value stored in the *TMRn_PWM* register. At the end of the cycle where the *TMRn_CNT* value matches the *TMRn_PWM* value, the timer's output toggles state. The timer continues counting until it reaches the *TMRn_CMP* value.

10.7.1 Timer Period

The timer period ends on the rising edge of PCLK following *TMRn_CNT* = *TMRn_CMP*.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. The *TMRn_CNT* is reset to 0x0000 0001, and the timer resumes counting.
2. The timer output signal is toggled.
3. The timer interrupt bit *TMRn_INT.irq* will be set. An interrupt will be generated if enabled.

When *TMRn_CN.tpol* = 0, the timer output signal starts low and then transitions to high when the *TMRn_CNT* value matches the *TMRn_PWM* value. The timer output signal remains high until the *TMRn_CNT* value reaches the *TMRn_CMP* value, resulting in the timer output signal transitioning low, and the *TMRn_CNT* value resetting to 0x0000 0001.

When *TMRn_CN.tpol* = 1, the Timer output signal starts high and transitions low when the *TMRn_CNT* value matches the *TMRn_PWM* value. The timer output signal remains low until the *TMRn_CNT* value reaches the *TMRn_CMP* value, resulting in the timer output signal transitioning high, and the *TMRn_CNT* value resetting to 0x0000 0001.

10.7.2 PWM Mode Configuration

Complete the following steps to configure a timer for PWM mode and initiate the PWM operation:

1. Set *TMRn_CN.ten* = 0 to disable the timer.
2. Set *TMRn_CN.tmode* to 011b to select PWM mode.
3. Set *TMRn_CN.pres3*:*TMRn_CN.pres* to set the prescaler that determines the timer frequency.
4. Configure the timer pin:
5. Configure the pin as a timer input and configure the electrical characteristics as needed.
6. Set *TMRn_CN.tpol* to match the desired initial (inactive) state.
   a. Set *TMRn_CN.tpol* to select the initial logic level (high or low) and PWM transition state for the timer's output.
   b. Set *TMRn_CNT* to the starting count, typically 0x0000 0001. The initial *TMRn_CNT* value only effects the initial period in PWM mode with subsequent periods always setting *TMRn_CNT* to 0x0000 0001.
   c. Set the *TMRn_PWM* value to the transition period count.
7. Set the *TMRn_CMP* value for the PWM second transition period. Note: *TMRn_CMP* must be greater than the *TMRn_PWM* value.
8. Optionally enable the timer's interrupt in the Interrupt Controller and set the timer's interrupt priority.
9. Set *TMRn_CN.ten* to 1 to enable the timer and start the PWM.

The PWM period is calculated using the following equation:

Equation 10-6: Timer PWM Period

$$PWM\ period\ in\ seconds = \frac{TMR\_CMP}{f_{CNT\_CLK}\ (Hz)}$$

If an initial starting value other than 0x0000 0001 is loaded into the *TMRn_CNT* register, use the One-Shot mode equation to determine the initial PWM period.

If *TMRn_CN.tpol* is 0, the ratio of the PWM output high time to the total period is calculated using the following equation:

$$PWM\ output\ high\ time\ ratio\ (\%) = \frac{(TMR\_CMP - TMR\_PWM)}{TMR\_CMP} \times 100$$

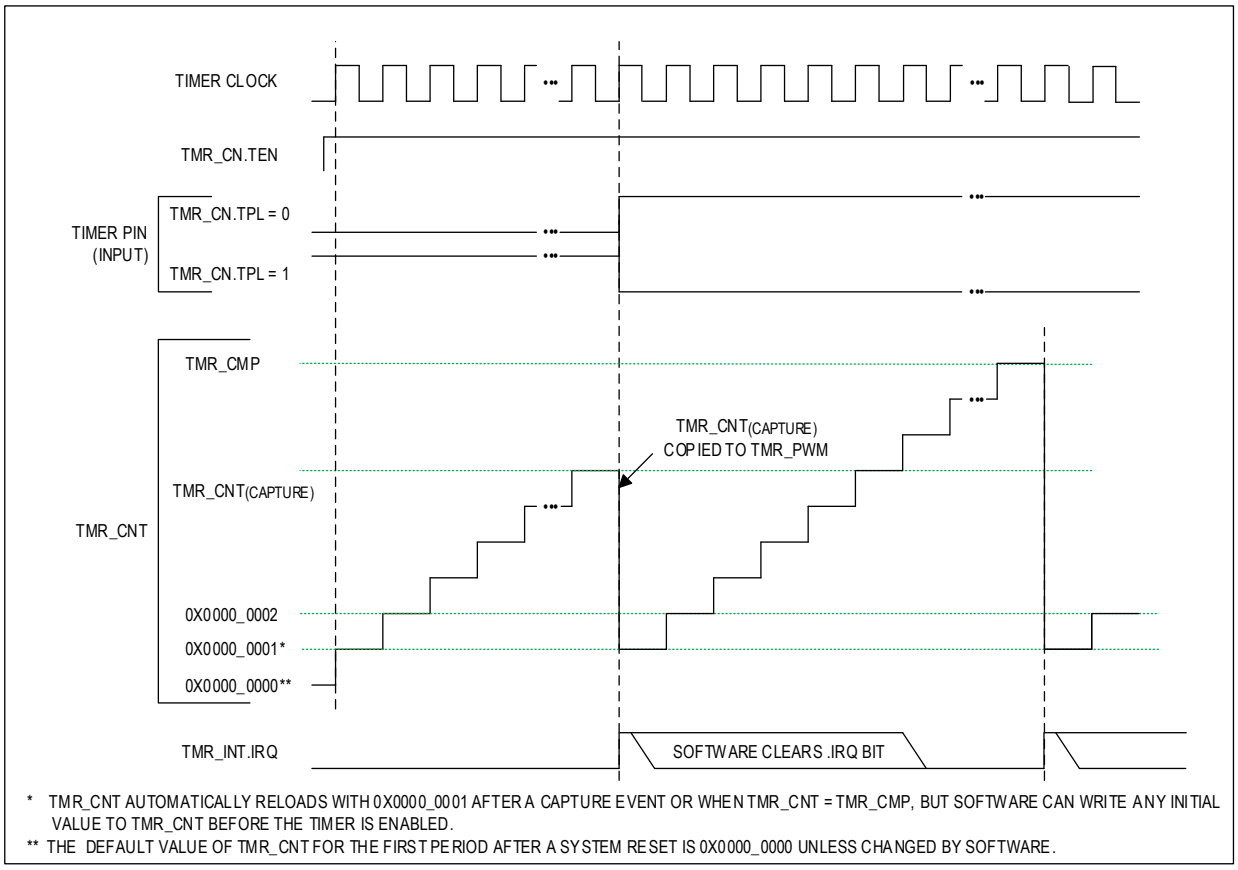
If *TMRn_CN.tpol* is set to 1, the ratio of the PWM output high time to the total period is calculated using the following equation:

$$PWM\ output\ high\ time\ ratio\ (\%) = \frac{TMR\_PWM}{TMR\_CMP} \times 100$$

10.8 Capture Mode (100b)

Capture mode most often used to measure the time between events. The timer increments from an initial value until an edge transition occurs on the timer pin. This triggers the 'capture' event which copies *TMRn_CNT* to the *TMRn_PWM.pwm* register, resets *TMRn_CNT* = 0x0000 0001, and continues incrementing. Also, if the timer pin does not go active before *TMRn_CNT* = *TMRn_CMP*, the timer will reset *TMRn_CNT* = 0x0000 0001, and continue incrementing. Either event will set the timer interrupt bit.

Figure 10-4: Capture Mode Diagram



10.8.1 Timer Period

Two timer period events are possible in Capture Mode:

The Capture event occurs on the timer clock following the selected transition on the timer pin. The timer peripheral automatically performs the following actions:

1. The value in *TMRn_CNT* is copied to *TMRn_PWM*
2. The timer interrupt bit *TMRn_INT.irq* will be set. An interrupt will be generated if enabled.
3. The timer remains enabled and continues incrementing.
4. The timer period ends on the timer clock following *TMRn_CNT* = *TMRn_CMP*.

The timer period event occurs on the timer clock *TMRn_CNT* = *TMRn_CMP*. The timer peripheral automatically performs the following actions when an end of timer period event occurs:

1. The value in *TMRn_CNT* is reset to 0x0000 00001. The timer remains enabled and continues incrementing.
2. The timer interrupt bit *TMRn_INT.irq* will be set. An interrupt is generated if the IRQ is enabled.

10.8.2 Configuration

Configure the timer for Capture mode by doing the following:

1. Disable the timer by setting *TMRn_CN.ten* to 0.
2. Select Counter mode by setting *TMRn_CN.tmode* to 010b.
3. Set *TMRn_CN.pres3*:*TMRn_CN.pres* to set the prescaler that determines the timer frequency.
4. If using the timer pin:
   a. Configure the pin as a timer output and configure the electrical characteristics as needed.
   b. Set *TMRn_CN.tpol* to match the desired (inactive) state.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write the initial value to *TMRn_CNT*. This effects only the first period; subsequent periods always begin with 0x0000 0001.
7. Write the compare value to *TMRn_CMP*.
8. Set *TMRn_CN.ten* = 1 to enable the timer.

The timer period is calculated using the following equation:

Equation 10-7: Capture Mode Elapsed Time

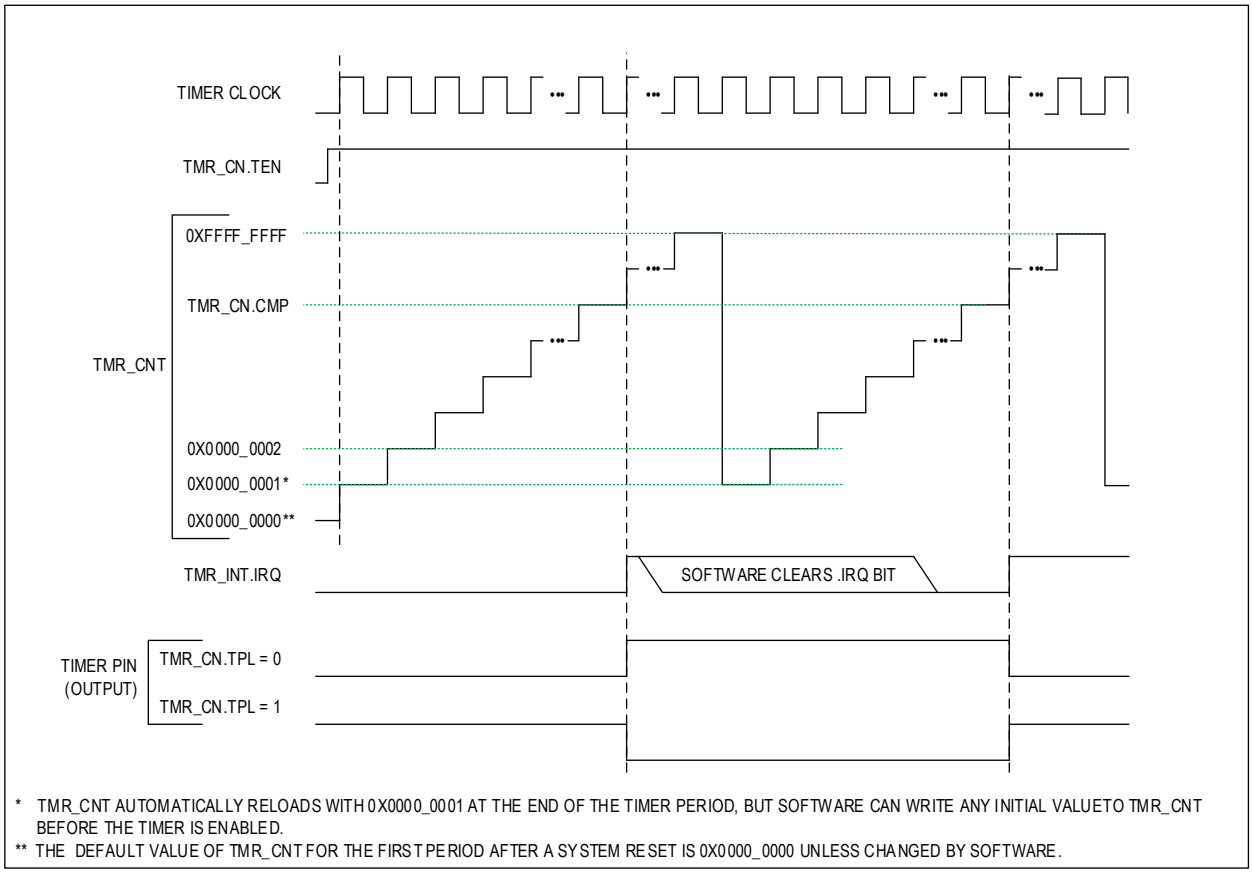
$$Capture\ elapsed\ time\ in\ seconds = \frac{TMR\_PWM - TMR\_CNT_{INITIAL\_COUNT\_VALUE}}{f_{CNT\_CLK}}$$

Note: The capture elapsed time calculation is only valid after the capture event occurs, and the timer stores the captured count in the TMRn_PWM register.

10.9 Compare Mode (101b)

In Compare mode the timer peripheral increments continually, allowing the timer to be a programmable 32-bit programmable period timer. The end of timer period event occurs when the timer value matches the compare value, but the timer continues to increment until the count reaches 0xFFFF FFFF. The timer counter then rolls over and continues counting from 0x0000 0000.

Figure 10-5: Counter Mode Diagram



10.9.1 Timer Period

The timer period ends on the timer clock following *TMRn_CNT* = *TMRn_CMP*.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. The timer remains enabled and continues incrementing. Unlike other modes, *TMRn_CNT* is not reset to 0x0000 0001 at the end of the timer period.
2. If the timer output is enabled, then the timer pin toggles state (low to high or high to low).
3. The timer interrupt bit *TMRn_INT.irq* will be set. An interrupt will be generated if enabled.

10.9.2 Configuration

Configure the timer for Compare mode by doing the following:

1. Set *TMRn_CN.ten* = 0 to disable the timer.
2. Set *TMRn_CN.tmode* to 011b to select Compare mode.
3. Set *TMRn_CN.pres3*:*TMRn_CN.pres* to set the prescaler that determines the timer frequency.
4. If using the timer pin:
   a. Configure the pin as a timer output and configure the electrical characteristics as needed.
   b. Set *TMRn_CN.tpol* to match the desired (inactive) state.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write the initial value to *TMRn_CNT*. This effects only the first period as the counter increments continuously, rolling over to 0x0000 0000 and continuing.
7. Write the compare value to *TMRn_CMP*.
8. Set *TMRn_CN.ten* = 1 to enable the timer.

The timer period is calculated using the following equation:

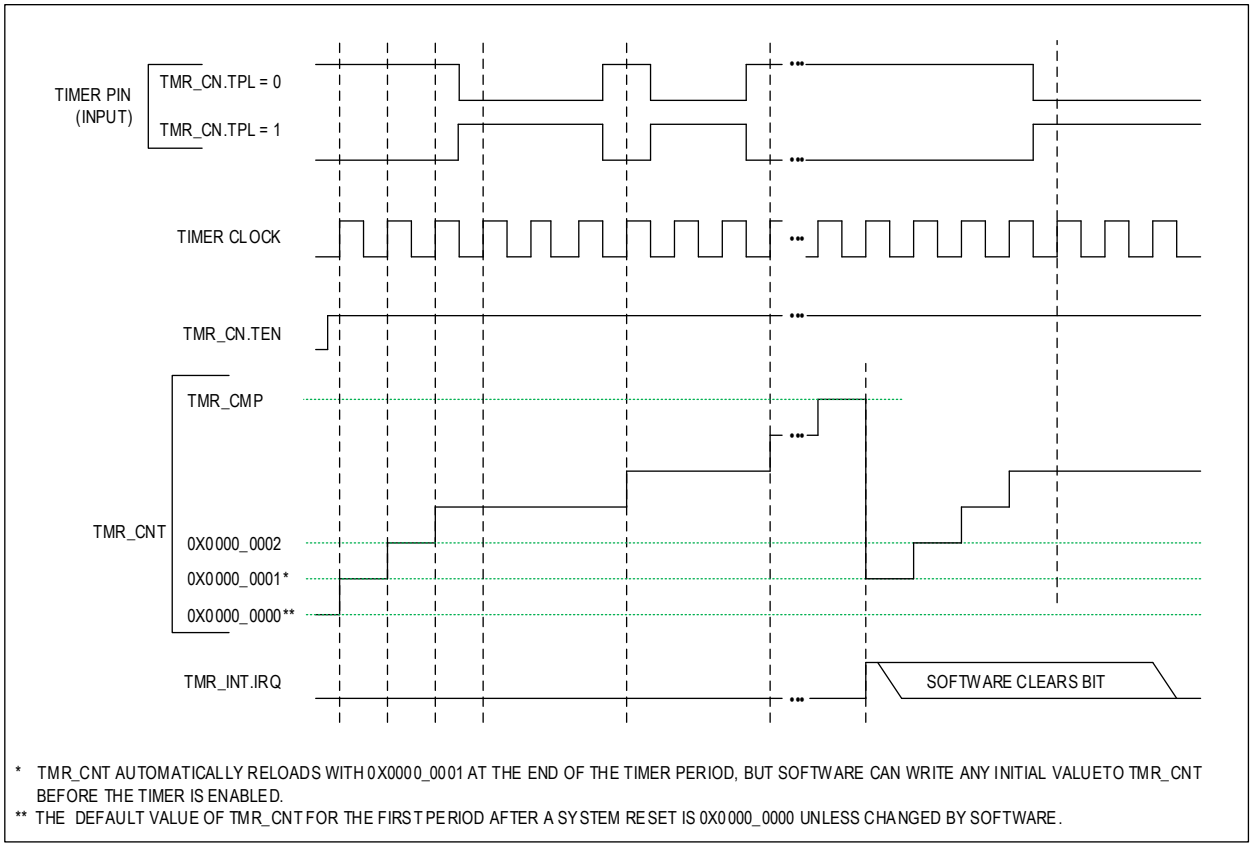
Equation 10-8: Compare Mode Timer Period

$$Compare\ mode\ timer\ period\ in\ seconds = \frac{TMR\_CMP - TMR\_CNT_{INITIAL\_COUNT\_VALUE} + 1}{f_{CNT\_CLK}\ (Hz)}$$

10.10 Gated Mode (110b)

Gated mode is like continuous mode, except that *TMRn_CNT* only increments when the timer pin is in its active state.

Figure 10-6: Gated Mode Diagram



10.10.1 Timer Period

The timer period ends when *TMRn_CNT* = *TMRn_CMP*.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. *TMRn_CNT* is reset to 0x0000 0001. The timer remains enabled and continues incrementing.
2. The timer interrupt bit *TMRn_INT.irq* will be set. An interrupt will be generated if enabled.

10.10.2 Configuration

Configure the timer for Gated mode by doing the following:

1. Set *TMRn_CN.ten* = 0 to disable the timer.
2. Set *TMRn_CN.tmode* to 110b to select Gated mode.
3. Set *TMRn_CN.pres3*:*TMRn_CN.pres* to set the prescaler that determines the timer frequency.
4. Configure the timer pin:
   a. Configure the pin as a timer input and configure the electrical characteristics as needed.
   b. Set *TMRn_CN.tpol* to match the desired initial (inactive) state.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write an initial value to *TMRn_CNT*, if desired. This effects only the first period; subsequent timer periods always reset *TMRn_CNT*= 0x0000 0001.
7. Write the compare value to *TMRn_CMP*.
8. Set *TMRn_CN.ten* = 1 to enable the timer.

10.11 Capture/Compare Mode (111b)

In Capture/Compare mode, the timer starts counting after the first external timer input transition occurs. The transition, a rising edge or falling edge on the timer's input signal, is set using the *TMRn_CN.tpol* bit.

Each subsequent transition, after the first transition of the timer input signal, captures the *TMRn_CNT* value, writing it to the *TMRn_PWM* register (capture event). When a capture event occurs, a timer interrupt is generated, the *TMRn_CNT* value is reset to 0x0000 0001, and the timer resumes counting.

If no capture event occurs, the timer counts up to the *TMRn_CMP* value. At the end of the cycle where the *TMRn_CNT* equals the *TMRn_CMP* value, a timer interrupt is generated, the *TMRn_CN*T value is reset to 0x0000 0001, and the timer resumes counting.

10.11.1 Timer Period

The timer period ends when the selected transition occurs on the timer pin, or on the clock cycle following *TMRn_CNT* = *TMRn_CMP*.

The timer peripheral automatically performs the following actions at the end of the timer period:

If the end of the timer period was caused by a transition on the timer pin:

1. The value in *TMRn_CNT* is copied to *TMRn_PWM*.
2. *TMRn_CNT* is reset to 0x0000 0001. The timer remains enabled and continues incrementing.
3. The timer interrupt bit, *TMRn_INT.irq*, is set. A Timer IRQ is generated, if enabled.
4. If the end of the timer period was caused by a transition on the timer pin:
   a. *TMRn_CNT* is reset to 0x0000 0001.
   b. The timer remains enabled and continues incrementing.
5. The timer interrupt bit *TMRn_INT.irq* will be set. An interrupt is generated, if enabled.

10.11.2 Configuration

Configure the timer for Capture/Compare mode by doing the following:

1. Set *TMRn_CN.ten* = 0 to disable the timer.
2. Set *TMRn_CN.tmode* to 111b to select Capture/Compare mode.
3. Set *TMRn_CN.pres3*:*TMRn_CN.pres* to set the prescaler that determines the timer frequency.
4. Configure the timer pin:
   a. Configure the pin as a timer input and configure the electrical characteristics as needed.
   b. Set *TMRn_CN.tpol* to select the positive edge (*TMRn_CN.tpol* = 0) or negative edge (*TMRn_CN.tpol* = 0) transition causes the capture event.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write an initial value to *TMRn_CNT*, if desired. This effects only the first period; subsequent timer periods always reset *TMRn_CNT*= 0x0000 0001.
7. Set *TMRn_CN.ten* to 1 to enable the timer. Counting starts after the first transition of the timer's input signal. No interrupt is generated by the first transition of the input signal.

In Capture/Compare mode, the elapsed time from the timer start to the capture event is calculated using the following equation:

Equation 10-9: Capture Mode Elapsed Time

$$Capture\ elapsed\ time\ in\ seconds = \frac{TMR\_PWM - TMR\_CNT_{INITIAL\_CNT\_VALUE}}{f_{CNT\_CLK}\ (Hz)}$$

10.12 Timer Registers

Each timer is controlled by a block of registers assigned to that specific timer. All timers contain an identical set of registers.

Register names for a specific instance are defined by appending the instance number to the peripheral name. For example, the Timer Count Register for Timer 0 is TMR0_CNT while the Timer Count Register for Timer 1 is TMR1_CNT, and so on. The MAX32660 includes three timer instances, TMR0, TMR1 and TMR2.

The base address for TMR0, TMR1 and TMR2 are as follows:

- TMR0: 0x4001 0000
- TMR1: 0x4001 1000
- TMR2: 0x4001 2000

Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 10-1: Timer Register Offsets, Names, Access and Descriptions

Offset	Register Name	Access	Description
[0x0000]	*TMRn_CNT*	R/W	Timer Counter Register
[0x0004]	*TMRn_CMP*	R/W	Timer Compare Register
[0x0008]	*TMRn_PWM*	R/W	Timer PWM Register
[0x000C]	*TMRn_INT*	R/W	Timer Interrupt Register
[0x0010]	*TMRn_CN*	R/W	Timer Control Register

10.12.1 Timer Register Details

Table 10-2: Timer Count Registers

Timer Count Register			TMRn_CNT	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	count	R/W	0	**Count** The current count value for the timer. This field increments as the timer counts. Reads to this register are always valid. In general, disable the timer by clearing bit *TMRn_CN.ten,* prior to writing the *TMRn_CNT* field.

Table 10-3: Timer Compare Registers

Timer Compare Register			TMRn_CMP	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	compare	R/W	0	**Timer Compare Value** The value in this register is used as the compare value for the timer's count value. The compare field meaning is determined by the specific mode of the timer. Refer to the mode's detailed configuration section for compare usage and meaning.

Table 10-4: Timer PWM Registers

Timer PWM Register			TMRn_PWM	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	pwm	R/W	0	**Timer PWM Match or Timer Capture Value** *PWM Mode:* In PWM mode, this field sets the count value for the first transition period of the PWM cycle. At the end of the cycle where *TMRn_CNT* equals *TMRn_CMP*, the PWM output transitions to the second period of the PWM cycle. The second PWM period count is stored in the *TMRn_CMP* register. The value set for *TMRn_PWM.pwm* must be less than the value set in *TMRn_CMP* for PWM mode operation. *Capture or Capture/Compare Mode:* In Capture, Compare, and Capture/Compare modes, this field is used to store the *TMRn_CNT* value when a Capture, Compare, or Capture/Compare event occurs.

Table 10-5: Timer Interrupt Registers

Timer Interrupt Register			TMRn_INT	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:1	-	R	0	**Reserved for Future Use** Do not modify this field.
0	irq	RW	0	**Timer Interrupt** Writing any value to this bit clears the timer's interrupt.

Table 10-6: Timer Control Registers

Timer Control Register			TMRn_CN	[0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
	-	R	0	**Reserved for Future Use** Do not modify this field.
12	pwmckbd	R/W	1	**PWM Output $\phi_{A'}$ Disable** 1: Disable PWM Output $\phi_{A'}$ 0: Enable PWM Output $\phi_{A'}$
11	nollpol	R/W	0	**PWM Output $\phi_{A'}$ Polarity Bit** 1: Output $\phi_{A'}$ inverted 0: Output $\phi_{A'}$ non-inverted
10	nolhpol	R/W	0	**PWM Output ϕ_A Polarity Bit** 1: Output ϕ_A inverted 0: Output ϕ_A non-inverted
9	pwmsync	R/W	0	**PWM Synchronization Mode** 1: PWM synchronization mode enabled 0: PWM synchronization mode disabled
8	pres3	R/W	0	**Timer Prescale Select MSB** See *TMRn_CN.pres* for details about this bit.
7	ten	R/W	0	**Timer Enable** 1: Timer enabled 0: Timer disabled

Timer Control Register			TMRn_CN	[0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
6	tpol	R/W	0	**Timer Polarity** Selects the polarity of the timer's input and output signal. This setting is not used if the GPIO Port Pin for the timer's input, output, or both is not configured for the timer alternate function in GPIO. The tpol field meaning is determined by the specific mode of the timer. Refer to the mode's detailed configuration section for tpol usage and meaning.
5:3	pres	R/W	0	**Timer Prescaler Select** Sets the timer's prescaler value. The prescaler divides the PCLK input to the timer and sets the timer's count clock, $$f_{CNT\_CLK} = {PCLK\ (Hz)}/{Prescaler}$$ The timer's prescaler setting is a 4-bit value with pres3 as the most significant bit and pres as the three least significant bits. The table below shows the prescaler values based on *pres3*:*pres*.

pres3	pres	Prescaler	f_{CNT_CLK}
0	0b000	0	${f_{PCLK}(Hz)}/{1}$
0	0b001	1	${f_{PCLK}\ (Hz)}/{2}$
0	0b010	2	${f_{PCLK}\ (Hz)}/{4}$
0	0b011	3	${f_{PCLK}(Hz)}/{8}$
0	0b100	4	${f_{PCLK}(Hz)}/{16}$
0	0b101	5	${f_{PCLK}(Hz)}/{32}$
0	0b110	6	${f_{PCLK}(Hz)}/{64}$
0	0b111	7	${f_{PCLK}(Hz)}/{128}$
1	0b000	8	${f_{PCLK}(Hz)}/{256}$
1	0b001	9	${f_{PCLK}(Hz)}/{512}$
1	0b010	10	${f_{PCLK}(Hz)}/{1024}$
1	0b011	11	${f_{PCLK}(Hz)}/{2048}$
1	0b100	12	${f_{PCLK}(Hz)}/{4096}$
1	0b101	13	${f_{PCLK}(Hz)}/{8192}$
1	0b110	Reserved	Reserved
1	0b111	Reserved	Reserved

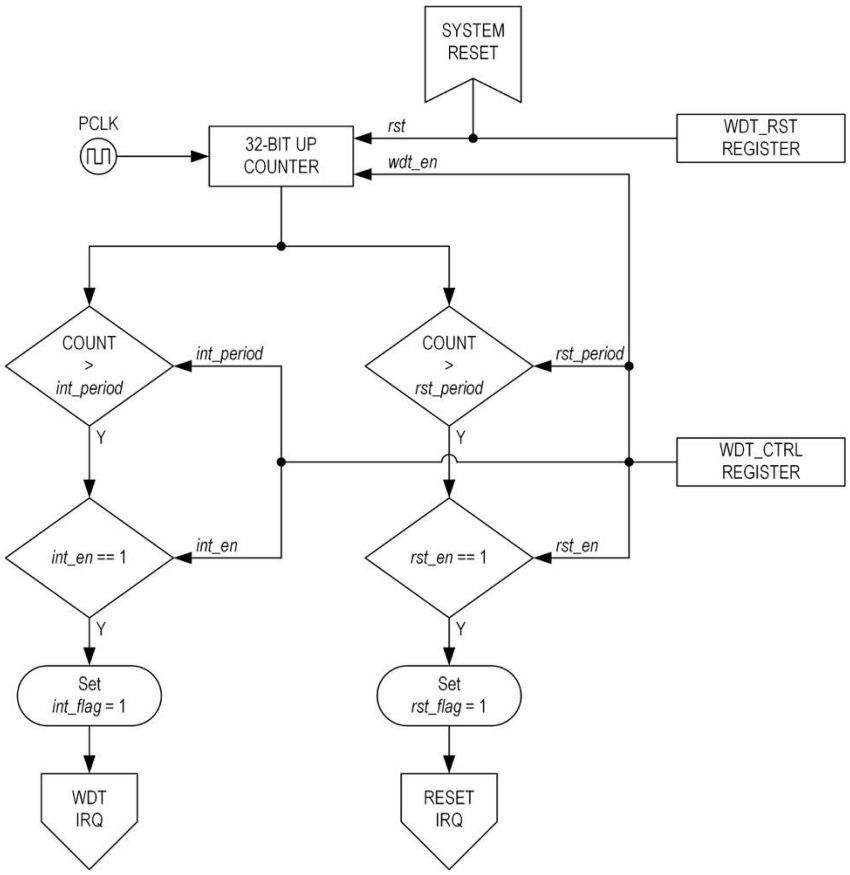
Timer Control Register			TMRn_CN	[0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
2:0	tmode	R/W	0	**Timer Mode Select** Sets the timer's operating mode. **tmode** / **Selected Timer Mode** 000b – One-Shot 001b – Continuous 010b – Counter 011b – PWM 100b – Capture 101b – Compare 110b – Gated 111b – Capture/Compare

11 Watchdog Timer (WDT)

The watchdog timer protects against corrupt or unreliable software, power faults, and other system-level problems, which may place the microcontroller into an improper operating state. When the application is executing properly, application software periodically resets the watchdog counter. If the watchdog timer interrupt is enabled and the software does not reset the counter within the interrupt time period (*WDT0_CTRL*.*int_period*), the watchdog timer generates a watchdog timer interrupt. If the watchdog timer reset is enabled and the software does not reset the counter within the reset time period (*WDT0_CTRL*.*rst_period*), the watchdog timer generates a system reset.

Figure 11-1 shows the block diagram of the watchdog timers.

Figure 11-1: Watchdog Timer Block Diagram



11.1 Features

- Sixteen programmable time periods for the watchdog interrupt
  - 2^{16} through 2^{31} PCLK cycles
- Sixteen programmable time periods for the watchdog reset
  - 2^{16} through 2^{31} PCLK cycles
- The watchdog timer counter is reset on all forms of reset

11.2 Usage

Utilizing the watchdog timer in the application software is straightforward. As early as possible in the application software, enable the watchdog timer interrupt and watchdog timer reset. Periodically the application software must write to the *WDT0_RST* register to reset the watchdog counter. If program execution becomes lost, the watchdog timer interrupt will occur, giving the system a "last chance" to recover from whatever circumstance caused the improper code execution. The interrupt routine may either attempt to repair the situation or allow the watchdog timer reset to occur. In the event of a system software failure, the interrupt will not be executed, and the watchdog system reset will recover operation.

As soon as possible after a reset, the application software should interrogate the *WDT0_CTRL.rst_flag* to determine if the reset event resulted from a watchdog timer reset. If so, application software should assume that there was a program execution error and take whatever steps necessary to guard against a software corruption issue.

11.3 Interrupt and Reset Period Timeout Configuration

Each watchdog timer supports two independent timeout periods, the interrupt period timeout and reset period timeout.

Interrupt Period Timeout (WDT_CTRL0.int_period) - Sets the number of PCLK cycles until a watchdog timer interrupt is generated. This period must be less than the Reset Period Timeout for the watchdog timer interrupt to occur.

Reset Period Timeout (WDT_CTRL0.rst_period) – Sets the number of PCLK cycles until a system reset event occurs.

The interrupt and reset period timeouts are calculated using *Equation 11-1* and *Equation 11-2* respectively, where $f_{PCLK} = {f_{SYSCLK}}/{2}$. *Table 11-1* shows example interrupt period timeout calculations for several *WDT0_CTRL.int_period* settings with the System Clock set as the 96MHz High-Frequency Internal Oscillator.

Equation 11-1: Watchdog Timer Interrupt Period

$$T_{INT\_PERIOD} = \left(\frac{1}{f_{PCLK}}\right) \times 2^{(31-WDT0\_CTRL.int\_period)}$$

Equation 11-2. Watchdog Timer Reset Period

$$T_{RST\_PERIOD} = \left(\frac{1}{f_{PCLK}}\right) \times 2^{(31-WDT0\_CTRL.rst\_period)}$$

Table 11-1: Watchdog Timer Interrupt Period with f_{SYSCLK} = 96MHz and f_{PCLK} = 48MHz

WDT0_CTRL int_period	T_{INT_PERIOD} (seconds)
15	0.001
14	0.002
13	0.004
12	0.009
11	0.018
10	0.035
9	0.070
8	0.140

WDT0_CTRL int_period	T_{INT_PERIOD} (seconds)
7	0.280
6	0.560
5	1.12
4	2.24
3	4.47
2	8.95
1	17.9
0	Disabled

11.4 Enabling the Watchdog Timer

The watchdog timers are free running and require a protected sequence of writes to enable the watchdog timers to prevent an unintended reset during the enable process.

11.4.1 Enable sequence

1. Write *WDT0_RST*.*wdt_rst*: 0x000000A5
2. Write *WDT0_RST*.*wdt_rst*: 0x0000005A
3. Set *WDT0_CTRL*.*wdt_en* to 1

11.5 Disabling the Watchdog Timer

The watchdog timers can be disabled by the application code manually or by the microcontroller automatically as shown below.

11.5.1 Manual Disable

Setting *WDT0_CTRL*.*wdt_en* to 0 disables the watchdog timer.

11.5.2 Automatic Disable

A power-on-reset (POR) event automatically disables the watchdog timers by setting *WDT0_CTRL*.*wdt_en* to 0.

Note: The watchdog timer remains enabled during all other types of reset.

11.6 Resetting the Watchdog Timer

To prevent a watchdog interrupt or a watchdog reset or both, application software must write the reset sequence, shown below, to the *WDT0_RST* register prior to an interrupt or reset timeout occurring.

11.6.1 Reset Sequence

1. Write *WDT0_RST*: 0x0000 00A5
2. Write *WDT0_RST* 0x0000 005A

11.7 Detection of a Watchdog Reset Event

During system start-up, system software should check the *WDT0_CTRL*.*rst_flag* to determine if the reset was the result of a watchdog reset. Application software is responsible for taking appropriate actions if a watchdog reset occurred.

11.8 Watchdog Timer Registers

The WDT0 base peripheral address is 0x4000 3000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 11-2: Watchdog Timer Registers

Address	Register Name	Access	Description
[0x0000]	*WDT0_CTRL*	R/W	Watchdog Timer 0 Control Register
[0x0004]	*WDT0_RST*	R/W	Watchdog Timer 0 Reset Register

1.1.1 Watchdog Timer Register Details

Table 11-3: Watchdog Timer Control Register

Watchdog Timer 0 Control Register			WDT0_CTRL	0x0000 [0x00]
Bits	**Name**	**Access**	**Reset**	**Description**
31	rst_flag	R/W	See description	**Reset Flag** If set a watchdog system reset occurred. 0: Watchdog did not cause reset event. 1: Watchdog reset occurred.
30:12	-	RO	0	**Reserved for Future Use** Do not modify this field.
11	rst_en	R/W	0	**Reset Enable** Enable/Disable system reset if the *WDT0_CTRL*.*rst_period* expires. Only reset by power on reset. 0: Disabled 1: Enabled
10	int_en	R/W	0	**Interrupt Enable** Enable or Disable the watchdog interrupt. 0: Disabled 1: Enabled
9	int_flag	R/W1C	0	**Interrupt Flag** If set, the watchdog interrupt period has occurred. 0: IRQ not pending 1: Interrupt period expired. Generates a WDT IRQ if *WDT0_CTRL*.*int_en* = 1.
8	wdt_en	R/W	See Description	**Enable** Enable or disable the watchdog timer. Only reset by a power on reset. To enable the watchdog timer, the following sequence of writes must be performed. 1) Write *WDT0_RST*: 0x0000 00A5 2) Write *WDT0_RST*: 0x0000 005A 3) Write *WDT0_RST*.*wdt_en*: 0x1 0: Disabled 1: Watchdog Timer Enabled. To set this field to 1, perform the sequence shown above. *Note: This field is reset by a Power-On Reset event only. Other forms of reset do not reset this field.*

Watchdog Timer 0 Control Register			WDT0_CTRL	0x0000 [0x00]
Bits	**Name**	**Access**	**Reset**	**Description**
7:4	rst_period	R/W	0	**Reset Period** Sets the number of PCLK cycles until a system reset occurs if the watchdog timer is not reset. 0xF: $2^{16} \times t_{PCLK}$ 0xE: $2^{17} \times t_{PCLK}$ 0xD: $2^{18} \times t_{PCLK}$ 0xC: $2^{19} \times t_{PCLK}$ 0xB: $2^{20} \times t_{PCLK}$ 0xA: $2^{21} \times t_{PCLK}$ 0x9: $2^{22} \times t_{PCLK}$ 0x8: $2^{23} \times t_{PCLK}$ 0x7: $2^{24} \times t_{PCLK}$ 0x6: $2^{25} \times t_{PCLK}$ 0x5: $2^{26} \times t_{PCLK}$ 0x4: $2^{27} \times t_{PCLK}$ 0x3: $2^{28} \times t_{PCLK}$ 0x2: $2^{29} \times t_{PCLK}$ 0x1: $2^{30} \times t_{PCLK}$ 0x0: $2^{31} \times t_{PCLK}$
3:0	int_period	R/W	0	**Interrupt Period** Sets the number of PCLK cycles until a watchdog timer interrupt is generated. 0xF: $2^{16} \times t_{PCLK}$ 0xE: $2^{17} \times t_{PCLK}$ 0xD: $2^{18} \times t_{PCLK}$ 0xC: $2^{19} \times t_{PCLK}$ 0xB: $2^{20} \times t_{PCLK}$ 0xA: $2^{21} \times t_{PCLK}$ 0x9: $2^{22} \times t_{PCLK}$ 0x8: $2^{23} \times t_{PCLK}$ 0x7: $2^{24} \times t_{PCLK}$ 0x6: $2^{25} \times t_{PCLK}$ 0x5: $2^{26} \times t_{PCLK}$ 0x4: $2^{27} \times t_{PCLK}$ 0x3: $2^{28} \times t_{PCLK}$ 0x2: $2^{29} \times t_{PCLK}$ 0x1: $2^{30} \times t_{PCLK}$ 0x0: $2^{31} \times t_{PCLK}$

Table 11-4: Watchdog Timer Reset Register

Watchdog Timer 0 Reset Register			WDT0_RST	0x0004 [0x04]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved for Future Use** Do not modify this field.
7:0	wdt_rst	R/W	0	**Reset Register** Writing the watchdog counter reset sequence to this register resets the watchdog counter. The following is the required reset sequence to reset the watchdog and prevent a watchdog timer interrupt or watchdog system reset. • Write *WDT0_RST*: 0x0000 00A5 • Write *WDT0_RST*: 0x0000 005A

12 I²C Master/Slave Serial Controller

The MAX32660 integrates two I²C peripherals, designated I2C0 and I2C1. The register interfaces for I2C0 and I2C1 are identical with the same offset addresses for each register. For simplicity, I2Cn is used throughout this section to refer to both I²C ports (where *n* = [0,1]). The I2Cn peripheral supports High-speed mode (Hs-mode), Fast-mode and Standard mode communication speeds. Each I2Cn peripheral can operate as a master or a slave device.

The I²C bus is a standardized two-wire, bidirectional serial bus. It uses only two bus lines, a Serial Data Access (SDA) line for data, and a Serial Clock line (SCL) for the clock. SDA and SCL idle high with external pullup resistors. They are pulled low by open-drain drivers in the peripherals. Internal pullup circuits in the GPIO pins are capable of holding the SDA line and SCL line at a logic high state when all devices are idle, but external pullup resistors are highly recommended for all but the simplest, lowest-capacitance systems.

An I²C master owns the I²C bus for the duration of a transfer, driving the clock (SCL) and generating START and STOP signals. In slave mode, the I²C Controller relies on an external master to generate the clock on SCL. An I²C slave responds to data and commands only when an I²C master device addresses it.

For detailed information on I²C bus operation refer to Maxim Application Note 4024: SPI/I²C Bus Lines Control Multiple Peripherals.

12.1.1 Related Documentation

For details of the I²C-bus, please refer to the *I²C-bus specification and user manual, Rev. 6 - 4 April 2014*.

12.1.2 I²C Bus Terminology

Table 12-1, *below*, contains terms and definitions used in this chapter for the I²C Bus Terminology.

Table 12-1: I²C Bus Terminology

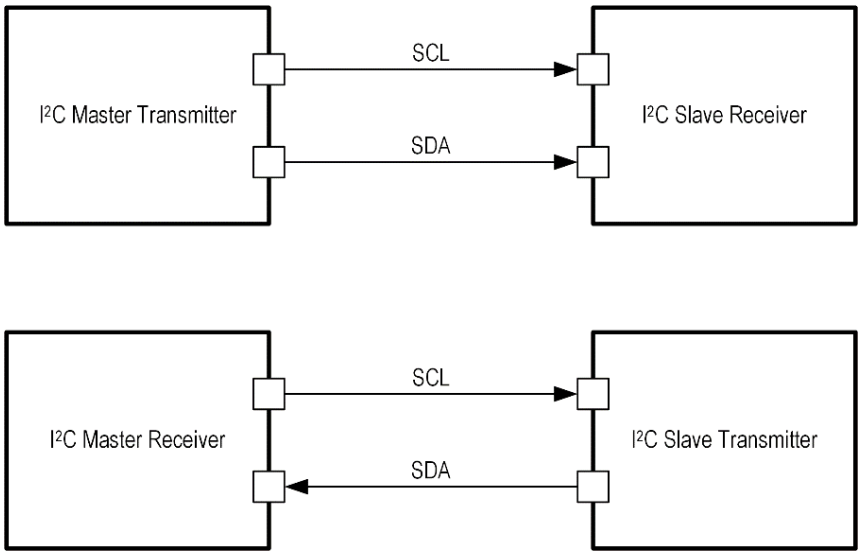
Term	Definition
Transmitter	The device that sends data to the bus.
Receiver	The device that receives data from the bus.
Master	The device that initiates a transfer, generates clock signals and terminates a transfer.
Slave	The device addressed by a master.
Multi-master	More than one master can attempt to control the bus at the same time without corrupting the message.
Arbitration	Procedure to ensure that, if more than one master simultaneously tries to control the bus, only one can do so and the resulting message is not corrupted.
Synchronization	Procedure to synchronize the clock signals of two or more devices.
Clock Stretching	When a slave device holds SCL low to pause a transfer until it is ready. This feature is optional according to the I²C Specification; thus, a master does not have to support slave clock stretching if none of the slaves in the system are capable of clock stretching.

A device attached to the I²C-bus assumes one of the following roles and can change between each transaction.

- Master-Transmitter
- Master-Receiver
- Slave-Transmitter
- Slave-Receiver

For all of the above roles, the Master is responsible for initiating the transfer and generating the clock signal on the SCL line. *Figure 12-1*, *below*, shows the four roles and the direction of the communication on the SCL and SDA lines.

Figure 12-1: The Roles of I²C Devices and the Direction the I²C Signals



12.2 I²C Master/Slave Features

Each I²C Master/Slave is compliant with the I²C Bus Specification and include the following features

- Compliant with I²C bus specification revision 06 (2014)
- Information transferred via a serial data bus (SDA) and a serial clock line (SCL)
- Operates as either a master or slave device as a transmitter or receiver
- Supports I²C Standard Mode, Fast Mode, Fast Mode Plus and High Speed (Hs) mode
- Data transfer rates up to:
  - 100 kbps in Standard Mode
  - 400 kbps in Fast Mode
  - 1 Mbps in Fast Mode Plus
  - 3.4 Mbps in Hs Mode
- Multi-master capable, including support for arbitration and clock synchronization for Standard, Fast and Fast Plus modes
- Supports the maximum bus capacitance as defined by the I²C bus specification
- Supports 7- and 10-bit addressing
- Supports RESTART condition
- Supports clock stretching
- Support for 7- and 10-bit device addressing
- Transfer status interrupts and flags
- DMA data transfer support
- I²C timing parameters fully controllable via firmware
- Glitch filter and Schmitt trigger hysteresis on SDA and SCL
- Control, status, and interrupt events are available for maximum flexibility
- Independent 8-byte RX FIFO and 8-byte TX FIFO
- TX FIFO preloading
- Programmable interrupt threshold levels for the TX and RX FIFO

12.3 I²C Overview

12.3.1 I²C Bus Speeds

The I²C peripherals support standard mode, fast mode, fast-plus mode and high-speed mode for I²C communications. All modes are downward compatible and operate at a lower bus speed as necessary.

12.3.2 I²C Transfer Protocol Operation

The I²C protocol operates over a two-wire bus: a clock circuit (SCL) and a data circuit (SDA). I²C is a half-duplex protocol: only one device is permitted to transmit on the bus at a time. The data rate is not fixed and can dynamically operate up to 100kHz in Standard Mode and up to 400kHz in Fast Mode.

Each transfer is initiated when the bus master sends a START or repeated START condition followed by the address of the slave peripheral. Information is sent most significant bit (MSB) first. Following the slave address, the master exchanges data with the addressed slave. The master can transmit data to the slave (a 'write' operation) or receive data from the slave (a 'read' operation). An acknowledge bit is sent by the receiving device after each byte is transferred. When all necessary data bytes have been transferred, a STOP or RESTART condition is sent by the bus master to indicate the end of the transaction. After the STOP condition has been sent, the bus is idle and ready for the next transaction. After a RESTART condition is sent, the same master begins a new transmission. The number of bytes that can be transmitted per transfer is unrestricted.

12.3.3 START and STOP Conditions

A START condition occurs when a bus master pulls SDA from high to low while SCL is high, and a STOP condition occurs when a bus master allows SDA to be pulled from low to high while SCL is high. Because these are unique conditions that cannot occur during normal data transfer, they are used to denote the beginning and end of the data transfer.

12.3.4 Master and Slave Overview

I²C transmit and receive data transfer operations are initiated by first loading the data to be sent in the I²C FIFO by writing data to the *I2Cn_FIFO* register. Once the transaction has completed, the data received can be read from the FIFO by reading data from the *I2Cn_FIFO* register. If a slave sends a NACK in response to a write operation, the I²C master generates an interrupt to the core. The I²C controller can be configured to issue a STOP condition to free the bus.

The receive FIFO contains the received data. If the receive FIFO is full or the transmit FIFO is empty, the I²C master stretches the clock to allow time to read bytes from the receive FIFO or load bytes into the transmit FIFO.

12.3.5 Slave Addressing

The first byte transmitted after a START condition is the slave address byte. If seven-bit addressing is used, the address byte consists of seven address bits and one R/W bit.

The I²C peripheral in the MAX32660 supports both 7-bit and 10-bit addressing. However, some slave addresses are reserved for special purposes by the I²C specification, including, but not limited to the following:

- Address 0b0000 0000 is a General Call Address and all slave devices recognize this address.
- Address 0b0000 0001 is used as a START condition for older devices.
- Address 0b1111 $1x_1x_2$1 is a request for a slave device's device ID.
- Address 0b1111 $0x_3x_2x_1$ indicates that the I²C master is initiating 10-bit addressing mode. The 3 most significant bits of the 10-bit address are $x_3x_2x_1$.

- Address 0b0000 $x_4x_3x_2x_1$ is reserved by the I²C specification for special purposes and should not be used for a slave address.
- Address 0b1111 1 $x_3x_2x_1$ is reserved by the I²C specification for special purposes and should not be used for a slave address.

12.3.6 Acknowledge and Not Acknowledge

An acknowledge bit (ACK) is generated by the receiver, whether I²C master or slave, after every byte received. The ACK bit is how the receiver tells the transmitter that the byte was successfully received, and another byte might be sent.

A Not Acknowledge (NACK) occurs if the receiver does not generate an ACK when the transmitter releases SDA. A NACK allows SDA to float high during the acknowledge time slot. The I²C master can then either generate a STOP condition to abort the transfer, or it can generate a repeated START condition (that is, send a START condition without an intervening STOP condition) to start a new transfer.

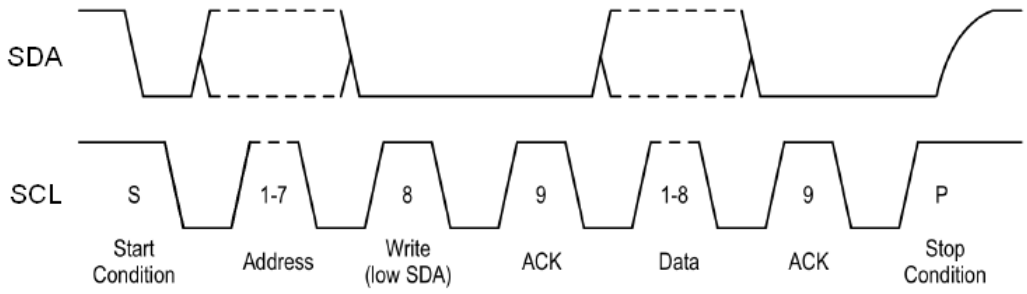
A receiver can generate a NACK after a byte transfer if any of the following conditions occur:

- No receiver is present on the bus with the transmitted address. In that case, no device will respond with an acknowledge signal.
- The receiver is unable to receive or transmit because it is busy and is not ready to start communication with the master.
- During the transfer, the receiver receives data or commands it does not understand.
- During the transfer, the receiver is unable to receive any more data.
- If an I²C master has requested data from a slave, it signals the slave to stop transmitting by sending a NACK following the last byte it requires.

12.3.7 Bit Transfer Process

Both SDA and SCL circuits are open-drain, bidirectional circuits. Each has an external pullup resistor that ensures each circuit is high when idle. The I²C specification states that during data transfer, the SDA line can change state only when SCL is low, and that SDA is stable and able to be read when SCL is high as shown in *Figure 12-2*, *below*.

Figure 12-2: I²C Write Data Transfer



The process for an I²C data transfer is as follows:

1. A bus master indicates a data transfer to a slave with a START condition.
2. The master then transmits one byte with a 7-bit slave address and a single read-write bit: a zero for a write or a one for a read.
3. During the next SCL clock following the read-write bit, the master releases SDA. During this clock period, the addressed slave responds with an ACK by pulling SDA low.
4. The master senses the ACK condition and begins transferring data. If reading from the slave, it floats SDA and allows the slave to drive SDA to send data. After each byte, the master drives SDA low to acknowledge the byte. If writing to the slave, the master drives data on the SDA circuit for each of the eight bits of the byte, and then floats SDA during the ninth bit to allow the slave to reply with the ACK indication.
5. After the last byte is transferred, the master indicates the transfer is complete by generating a STOP condition. A STOP condition is generated when the master pulls SDA from a low to high while SCL is high.

12.3.8 SCL and SDA Bus Drivers

The I²C bus expects SCL and SDA to be open-drain signals. In the MAX32660, once the I²C peripheral is enabled and the proper GPIO alternate function is selected, the corresponding pad circuits are automatically configured as open-drain outputs. However, SCL can also be optionally configured as a push-pull driver to conserve power and avoid the need for any pull-up resistor. This should only be used in systems where no I²C slave device can hold the SCL line low. Push-pull operation is enabled by setting *I2Cn_CTRL0.sclppm* to 1. (SDA always operates in open-drain mode.)

12.3.9 I²C Interrupt Sources

The I²C Controller has a very flexible interrupt generator that generates an interrupt signal to the Interrupt Controller on any of several events. On recognizing the I²C interrupt, firmware determines the cause of the interrupt by reading the I²C Interrupt Flags registers *I2Cn_INTFL0* and *I2Cn_INTFL1*. Interrupts can be generated for the following events:

- Transaction Complete (Master/Slave)
- Address NACK received from slave (Master)
- Data NACK received from slave (Master)
- Lost arbitration (Master)
- Transaction timeout (Master/Slave)
- FIFO is empty, not empty, full to configurable threshold level (Master/Slave)
- TX FIFO locked (Master/Slave)
- Out of sequence START and STOP conditions (Master/Slave)
- Sent a NACK to an external master because the TX or RX FIFO was not ready (Slave)
- Address ACK or NACK received (Master)
- Incoming address match (Slave)
- TX Underflow or RX Overflow (Slave)

Interrupts for each event can be enabled or disabled by setting or clearing the corresponding bit in the *I2Cn_INTEN0* or *I2Cn_INTEN1* interrupt enable register.

Note: Disabling the interrupt does not prevent the corresponding flag from being set by hardware but does prevent an IRQ when the interrupt flag is set.

Note: Prior to enabling an interrupt, the status of the corresponding interrupt flag should be checked and, if necessary, serviced or cleared. This prevents a previous interrupt event from interfering with a new I²C communications session.

12.3.10 SCL Clock Configurations

The SCL frequency is dependent upon the values of I²C peripheral clock and the values of the external resistor and capacitor on the SCL clock line.

Note: An external RC load on the SCL line will affect the target SCL frequency calculation.

Figure 12-3: I²C Specification Minimum and Maximum Clock Parameters for Standard and Fast Mode

Parameter	Standard Mode		Fast Mode	
	Min	Max	Min	Max
SCL Clock Freq.	0	100 kHz	0	400 kHz
I²C Hold Time	4.0 µs	-	0.6 µs	-
SCL High	4.0 µs	-	0.6 µs	-
SCL Low	4.7 µs	-	1.3 µs	-
t_{RC} Rise Time	-	1000 ns	20 ns	300 ns

12.3.11 Clock Synchronization

The I²C specification allows for more than one bus master. When more than one master is on the same bus, clock synchronization between different master's clocks is necessary. The I²C Master mode supports automatic clock synchronization and is compliant with the clock synchronization requirements of the I²C Specification. Clock synchronization is automatically handled in the I²C controller.

12.3.12 Transmit and Receive FIFOs

Each I²C master/slave has one 8-byte deep transmit FIFO (TX FIFO) and one 8-byte deep receive FIFO (RX FIFO) that reduces processor overhead. To further speed transfers, the DMA can read and write to each FIFO. When the DMA is used to read and write to the FIFOs, no additional I²C configuration is required and interrupts are still sent to the core. See the DMA section for more details.

When the receive FIFO is enabled, received bytes are automatically written to it. If the receive FIFO is full, no more data is written and any newly received bytes are lost.

When the transmit FIFO is enabled, either user firmware or the DMA can provide data to be transmitted. The oldest byte in the FIFO is sent out over SDA only when an ACK signal is received from an addressed slave.

Interrupts can be generated for the following FIFO status:

- TX FIFO level less than or equal to threshold
- RX FIFO level greater than or equal to threshold
- TX FIFO underflow
- RX FIFO overflow
- TX FIFO locked for writing

12.4 Clock Stretching

If a slave cannot receive or transmit a complete byte of data, it can force the master into a wait state by clock stretching. Clock stretching is when a slave holds SCL low after an ACK is on the bus. When the slave is ready, it releases the SCL line from low and then resumes the data transfer.

The I²C controller can hold SCL low in both master and slave modes after an ACK bit transmission. However, the term 'clock stretching' as defined in the I²C Bus Specification only applies if performed by a slave device. When an I²C master holds the SCL line low, the master is technically varying the clock speed. The master can vary the clock speed from DC (0Hz) up to the maximum f_{SCL}. For simplicity, this section describes situations where either an external slave or external master holds the SCL line low.

For clock stretching, SCL is held low after an ACK bit and before the first data bit. This is often done when a receiver cannot receive more data (usually from a full RX FIFO), or a transmitter needs to send more data but is not ready (usually from an empty TX FIFO).

However, during Interactive Receive Mode (IRXM), the receiver begins clock stretching before the ACK bit, allowing firmware time to decide whether to send an ACK or NACK. If operating in IRXM (*I2Cn_CTRL0.irxm*=1) as a slave with clock stretching disabled (*I2Cn_CTRL0.sclstrd*=1), SCL is not held low. Thus, the burden is on firmware to respond quickly enough to meet the data setup timing requirements as a late ACK could cause a transition on SDA while SCL is high, resulting in an unwanted STOP or RESTART. For these reasons, it is not recommended to use interactive receive mode with slave clock stretching disabled.

For a transmit operation as either master or slave, when the TX FIFO is empty after the last byte is shifted out, SCL is automatically held low until data is written to the TX FIFO. Master transmitters can stop clock stretching in this situation to end the transaction by sending a START or RESTART condition. When a slave transmitter sees an external master end the transaction by sending a NACK, it can then release SDA.

12.5 I²C Bus Timeout

The Timeout register, *I2Cn_TIMEOUT.to,* is used to detect bus errors. *Equation 12-1* and *Equation 12-2* show equations for calculating the maximum and minimum timeout values based on the value loaded into the *I2Cn_TIMEOUT.to* field.

Equation 12-1: I²C Timeout Maximum

$$t_{TIMEOUT} \leq \left(\frac{1}{f_{I2C\_CLK}}\right) \times \left((I2Cn\_TIMEOUT.to \times 8) + 3\right)$$

Due to clock synchronization, the timeout is guaranteed to meet the following minimum time calculation shown in *Equation 12-2*.

Equation 12-2: I²C Timeout Minimum

$$t_{TIMEOUT} \leq \left(\frac{1}{f_{I2C\_CLK}}\right) \times \left((I2Cn\_TIMEOUT.to \times 8) + 2\right)$$

The timeout feature is disabled when *I2Cn_TIMEOUT.to* = 0 and is enabled for any non-zero value. When the timeout is enabled, the timeout timer starts counting when the I²C peripheral hardware drives SCL low and is reset by the I²C peripheral hardware when the SCL line is released.

The timeout counter only monitors if the I²C peripheral hardware is driving the SCL line low. It does not monitor if an external I²C device is actively holding the SCL line low. The I²C peripheral does not monitor the status of the SDA line.

If the timeout timer expires, a bus error condition has occurred. When a timeout error occurs, the I²C peripheral hardware release the SCL and SDA lines and sets the timeout error interrupt flag to 1 (*I2Cn_INTFL0.toeri* = 1).

For applications where an external device may hold the SCL line low longer than the maximum timeout supported, the timeout can be disabled by setting the timeout field to 0 (*I2Cn_TIMEOUT.to* = 0).

12.6 I²C Addressing

After a START or RESTART condition, an address byte is transmitted where the first seven bits are the address, and the last bit indicates to the slave if the operation is a read or a write.

Table 12-2: I²C Address Byte Format
X=Don't Care

Slave Address Bits		R/W Bit	Description
0000	000	0	General Call Address
0000	000	1	START Condition
0000	001	X	CBUS Address
0000	010	X	Reserved for different bus format
0000	011	X	Reserved for future purposes
0000	1XX	X	HS-mode master code
1111	1XX	X	Reserved for future purposes
1111	0XX	X	10-bit slave addressing

In 7-bit addressing mode, the master sends one address byte. To address a 7-bit address slave, first clear *I2Cn_MSTR_MODE.sea* = 0, then write the address to the TX FIFO formatted as follows:

Master write to Slave : 7-bit address : [A6A5A4A3A2A1A0 0]
Master read from Slave : 7-bit address : [A6A5A4A3A2A1A0 1]

In 10-bit addressing mode (*I2Cn_MSTR_MODE.sea* = 1), the first byte the master sends is the 10-bit Slave Addressing byte which includes the first two bits of the 10-bit address, followed by a 0 for the R/W bit. The master then sends a second byte representing the remainder of the 10-bit address. If the operation is a write as indicated by the R/W bit in the second byte, the data is transmitted to the slave for the write. If the operation is a read, as indicated by the R/W bit in the second byte, the I²C master performs a REPEATED START and transmits the 10-bit slave address again with the R/W bit set to 1. The I²C peripheral hardware then begins receiving data from the slave device.

If the RX FIFO is not empty and an I²C write occurs, the I²C peripheral hardware automatically sends a NACK.

The setting of the Do Not Respond bit (*I2Cn_RXCTRL0.dnr*) controls when a NACK is sent as follows:

- *I2Cn_RXCTRL0.dnr* = 1
  - A NACK is sent on the first address byte received and the hardware sets the Do Not Respond Interrupt Flag (*I2Cn_INTFL0.dnreri = 1)*
- *I2Cn_RXCTRL0.dnr* = 0
  - Sends an ACK for each address byte, but NACKs subsequent data received.

If the TX FIFO is not ready (*I2Cn_TXCTRL1.txrdy* = 0) and the I²C controller receives a data read, the hardware automatically sends a NACK during the first address byte. The setting of the Do Not Respond field is ignored by the hardware for this condition because it is the only opportunity to send a NACK for an I²C read transaction.

12.7 I²C TX FIFO and RX FIFO Management

There are separate transmit and receive FIFOs, TX FIFO and RX FIFO. Both are accessed using the FIFO Data register *I2Cn_FIFO*. Writes to this register enqueue data into the TX FIFO. Writes to a full TX FIFO have no effect. Reads from *I2Cn_FIFO* dequeue data from the RX FIFO. Reads from an empty RX FIFO returns 0xFF.

The TX and RX FIFO will only read or write one byte at a time. Transactions larger than 8 bits can still be performed, however. A 16- or 32-bit write to the TX FIFO stores just the lowest 8 bits of the write data. A 16- or 32-bit read from the RX FIFO will have the valid data in the lowest 8 bits and 0's in the upper bits. In any case, the TX and RX FIFOs will only accept 8 bits for a read or a write.

Both the RX FIFO and TX FIFO are flushed when the I²C port is disabled by clearing *I2Cn_CTRL0.i2cen* = 0.

The TX FIFO and RX FIFO can be flushed by setting the Transmit FIFO Flush bit (*I2Cn_TXCTRL0.txfsh* = 1) or the Receive FIFO Flush bit (*I2Cn_RXCTRL0.rxfsh* = 1), respectively.

12.7.1 Transmit Lockout

Under certain conditions the TX FIFO is automatically locked by hardware and flushed so stale data is not unintentionally transmitted. The TX FIFO is automatically flushed, and writes are locked out under the following conditions:

- General Call Address match and TX FIFO Preloading is disabled
- Slave Address match and TX FIFO Preloading is disabled
- Operating as a slave transmitter, and a NACK is received.
- Any of the following interrupts: Arbitration Error, Timeout Error, Master Mode Address NACK, Data NACK Error, Start Error, and STOP Condition Detected.

When the above conditions occur, the TX FIFO is flushed so stale data is not unintentionally transmitted. In addition, the Transmit Lockout Flag is set (*I2Cn_INTFL0.txloi* = 1) and writes to the TX FIFO are ignored until firmware acknowledges the external event by clearing *I2Cn_INTFL0.txloi.*

Flushing the TX FIFO on Slave Address Match or General Call Match can be disabled using the Transmit FIFO Preload bit (*I2Cn_TXCTRL0.txpreld*). Setting this bit allows applications to preload the Transmit FIFO prior to a Slave Address Match. This can be combined with Slave Clock Stretching disabled (*I2Cn_CTRL0.sclstrd* = 0) to maximize the chance of completing a transmit operation without a transmit underflow error.

12.8 Interactive Receive Mode (IRXM)

In some situations, this I²C might want to inspect and respond to each byte of received data. In this case, Interactive Receive Mode can be used. Interactive Receive Mode is enabled by setting *I2Cn_CTRL0.irxm* = 1. If Interactive Receive Mode is enabled, it must occur before any I²C transfer is initiated.

When Interactive Receive Mode (IRXM) is enabled, after every data byte received the I²C peripheral automatically holds SCL low before the ACK bit. Additionally, after the 8th SCL falling edge, the I²C peripheral sets the IRXM Interrupt Status Flag (*I2Cn_INTFL0.irxmi* = 1). Application firmware must read the data and generate a response (ACK or NACK) by setting the IRXM Acknowledge (*I2Cn_CTRL0.ack*) bit accordingly. Send an ACK by clearing the *I2Cn_CTRL0.ack* bit to 0. Send a NACK by setting the *I2Cn_CTRL0.ack* bit to 1.

After setting the *I2Cn_CTRL0.ack* bit, clear the IRXM interrupt flag. Write 1 to *I2Cn_INTFL0.irxmi* to clear the interrupt flag. When the IRXM interrupt flag is cleared, the I²C peripheral hardware releases the SCL line and sends the *I2Cn_CTRL0.ack* on the SDA line. For both master and slave operations, SCL is released only after the necessary SCL low time requirement is satisfied to conform with $t_{su;dat}$ timing.

While I²C peripheral is waiting for the application firmware to clear the *I2Cn_INTFL0.irxmi* flag, firmware can disable Interactive Receive Mode and, if operating as a master, load the remaining number of bytes to be received for the transaction. This allows firmware to examine the initial bytes of a transaction, which might be a command, and then disable Interactive Receive Mode to receive the remaining bytes in normal operation.

During IRXM, received data is not placed in the RX FIFO. Instead, the *I2Cn_FIFO* address is repurposed to directly read the receive shift register, bypassing the RX FIFO. Therefore, before disabling Interactive Receive Mode, firmware must first read the data byte from *I2Cn_FIFO.data.* If the IRXM byte is not read, the next read from the RX FIFO returns 0xFF.

Note: Interactive Receive Mode does not apply to address bytes, only to data bytes.

Note: Interactive Receive Mode does not apply to general call address responses or START byte responses.

12.9 I²C DMA Control

There are independent DMA channels for each TX FIFO and each RX FIFO. DMA activity is triggered by the TX FIFO (*I2Cn_TXCTRL0.txth*) and RX FIFO (*I2Cn_RXCTRL0.rxth*) threshold levels.

12.9.1 I²C Transmit DMA Burst Size

When the TX FIFO byte count (*I2Cn_TXCTRL1.txfifo*) is less than or equal to the TX FIFO Threshold Level *I2Cn_TXCTRL0.txth*, then the DMA transfers data into the TX FIFO according to the DMA configuration. To ensure the DMA does not overflow the TX FIFO, the DMA burst size should be set as follows:

Equation 12-3: DMA Burst Size Calculation for I²C Transmit

$$DMA\ Burst\ Size = TX\ FIFO\ Depth - I2Cn\_TXCTRL0.txth \equiv 8 - I2Cn\_TXCTRL0.txth$$

$$\text{where } 0 \leq I2Cn\_TXCTRL0 \leq 7$$

Applications trying to avoid transmit underflow and/or clock stretching should use a smaller burst size and higher *I2Cn_TXCTRL0.txth* setting. This fills up the FIFO more frequently but increases internal bus traffic.

12.9.2 I²C Receive DMA Burst Size

When the RX FIFO count (*I2Cn_RXCTRL1.rxfifo*) is greater than or equal to the RX FIFO Threshold Level *I2Cn_RXCTRL0.rxth*, the DMA transfers data out of the RX FIFO according to the DMA configuration. To ensure the DMA does not underflow the RX FIFO, the DMA burst size should be set as follows:

Equation 12-4: DMA Burst Size Calculation for I²C Receive

$$DMA\ Burst\ Size = I2Cn\_RXCTRL0.rxth$$

$$\text{where } 1 \leq I2Cn\_RXCTRL0.rxth \leq 8$$

Applications trying to avoid receive overflow and/or clock stretching should use a smaller burst size and lower *I2Cn_RXCTRL0.rxth*. This results in reading from the Receive FIFO more frequently but increases internal bus traffic.

Note for receive operations, the length of the DMA transaction (in bytes) must be an integer multiple of I2Cn_RXCTRL0.rxth. Otherwise, the receive transaction will end with some data still in the RX FIFO, but not enough to trigger an interrupt to the DMA, leaving the DMA transaction incomplete. One easy way to ensure this for all transaction lengths is to set burst size to 1 (I2Cn_RXCTRL0.rxth = 1).

To enable DMA transfers, enable the TX DMA channel (*I2Cn_DMA.txen*) and/or the RX DMA channel (*I2Cn_DMA.rxen*). Refer to section *I2C DMA Control* for more information on configuring the DMA.

12.10 I²C Master Mode Transmit Operation

The peripheral operates in master mode when Master Mode Enable *I2Cn_CTRL0.mst* = 1. To initiate a transfer, the master generates a START condition by setting *I2Cn_MSTR_MODE.start*=1. If the bus is busy, it does not generate a START condition until the bus is available.

A master can communicate with two slave devices without relinquishing the bus. Instead of generating a STOP condition after communicating with the first slave, the master generates a Repeated START condition, or RESTART, by setting *I2Cn_MSTR_MODE.restart* = 1. If a transaction is in progress, the master finishes the transaction before generating a RESTART. The controller then transmits the slave address stored in the TX FIFO. The *I2Cn_MSTR_MODE.restart* bit is automatically cleared to 0 as soon as the master begins a RESTART condition. The reception of a STOP condition clears any pending RESTART.

I2Cn_MSTR_MODE.start is automatically cleared to 0 after the master has completed a transaction and sent a STOP condition.

The master can also generate a STOP condition by setting *I2Cn_MSTR_MODE.stop* = 1.

If both START and RESTART conditions are enabled at the same time, a START condition is generated first. Then, at the end of the first transaction, a RESTART condition is generated.

If both RESTART and STOP conditions are enabled at the same time, a STOP condition is not generated. Instead, a RESTART condition is generated. After the RESTART condition is generated, both bits are cleared.

If START, RESTART, and STOP are all enabled at the same time, a START condition is first generated. At the end of the first transaction, a RESTART condition is generated. The *I2Cn_MSTR_MODE.stop* bit is cleared and ignored.

A slave cannot generate START, RESTART, or STOP conditions. Therefore, when Master Mode is disabled, the *I2Cn_MSTR_MODE.start*, *I2Cn_MSTR_MODE.restart*, and *I2Cn_MSTR_MODE.stop* bits are all cleared to 0.

Note: After starting a transfer, I2Cn_MSTR_MODE.start = 1, changing the I²C configuration results in unpredictable behavior.

12.11 I²C Master Mode Transmit Bus Arbitration

The I²C protocol supports multiple masters on the same bus. When the bus is free, it is possible that two masters might try to initiate communication at the same time. This is a valid bus condition. If this occurs, only one master can remain in master mode and complete its transaction. The other master must back off transmission and wait until the bus is idle. This process is called bus arbitration.

To determine which master wins the arbitration, each master compares the data being transmitted on SDA to the value observed on SDA. If the master attempting to transmit a 1 on SDA (that is, the master wants SDA to float) senses a 0 instead, that master concludes that it has lost arbitration because another master is transmitting a 0 onto SDA. It then cedes the bus by switching off its SDA driver.

Note that this arbitration scheme works with any number of bus masters: if more than two masters begin transmitting simultaneously, the arbitration continues as each master cedes the bus until only one master remains transmitting. Data is not corrupted because as soon as each master realizes it has lost arbitration it stops transmitting, leaving the data on SDA intact.

Once a master has lost arbitration it stops generating SCL, sets *I2Cn_INTFL0.areri*, and clears *I2Cn_MSTR_MODE.start*, *I2Cn_MSTR_MODE.restart*, and *I2Cn_MSTR_MODE.stop* to 0.

The I²C master peripheral is compliant with the bus arbitration requirements of the I²C specification. I²C bus arbitration is handled by the peripheral hardware and requires no additional configuration.

12.12 SCL Clock Generation for Standard, Fast and Fast-Plus Modes

The master generates the I²C clock on the SCL line. Application code must configure the *I2Cn_CLKHI* and *I2Cn_CLKLO* registers for the desired I²C operating frequency.

The I²C peripheral clock is supplied directly by the system peripheral clock, f_{PCLK}.

The SCL high time is configured in the I²C Clock High Time register field *I2Cn_CLKHI.scl_hi.* The SCL low time is configured in the I²C Clock Low Time register field *I2Cn_CLKLO.scl_lo*. Each of these fields is 8-bits.

Equation 12-5: I²C Clock High Time Calculation

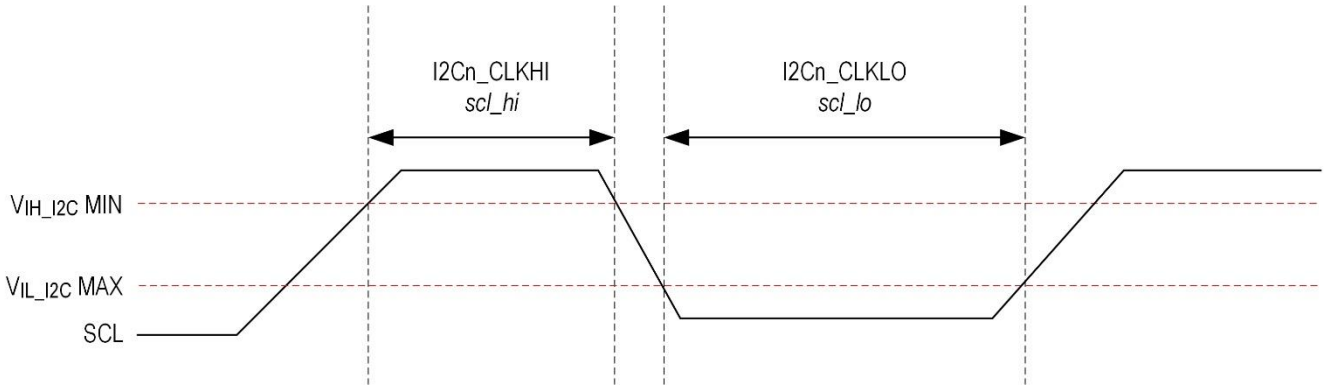
$$t_{SCL\_HI} = t_{I2C\_CLK} \times (I2Cn\_CLKHI.scl\_hi + 1)$$

Equation 12-6: I²C Clock Low Time Calculation

$$t_{SCL\_LO} = t_{I2C\_CLK} \times (I2Cn\_CLKLO.scl\_lo + 1)$$

Figure 12-4 shows the association between the SCL clock low and high times for Standard, Fast and Fast-Plus I²C frequencies.

Figure 12-4: I²C Clock Period



During synchronization, external masters or external slaves may be driving SCL simultaneously. This affects the SCL duty cycle. By monitoring SCL, the controller can determine whether an external master or slave is holding SCL low. In either case, the controller waits until SCL is high before starting to count the number of SCL high cycles. Similarly, if an external master pulls SCL low before the controller has finished counting SCL high cycles, then the controller starts counting SCL low cycles and releases SCL once the time period, *I2Cn_CLKLO.scl_lo*, has expired.

Because the controller does not start counting the high/low time until the input buffer detects the new value, the actual clock behavior is based on many factors. These include bus loading, other devices on the bus holding SCL low, and the filter delay time of this device.

12.13 SCL Clock Generation for Hs-Mode

In Hs-Mode, the I²C-bus specification requires a SCL clock HIGH to LOW ratio of 1 to 2, $t_{SCL\_LO} = 2 \times t_{SCL\_HI}$. The *I2Cn_HS_CLK* register is used for configuring the Hs-Mode clock high and low times. Calculate the number of peripheral clocks for the Hs-Mode bit rate of 3.4 Mbps using *Equation 12-7*. Using the result, *pclk_count*, calculate the SCL high time using *Equation 12-8*, and the SCL low time using *Equation 12-9*.

Equation 12-7: I²C High Speed Mode Clock Count

$$pclk\_count = \frac{f_{PCLK}}{3{,}400{,}000}$$

Equation 12-8: Hs-Mode SCL High Time Calculation

$$I2Cn\_HS\_CLK.hs\_clk\_hi = \frac{pclk\_count}{3} - 1$$

Equation 12-9: Hs-Mode SCL Low Time Calculation

$$I2Cn\_HS\_CLK.hs\_clk\_lo = \frac{(pclk\_count \times 2)}{3} - 1$$

Compensate for rounding errors in the calculation by rounding up the *I2Cn_HS_CLK.hs_clk_hi* result if the least significant bit is set.

12.14 TX FIFO Preloading

There may be situations where, when operating as a slave, firmware wants to preload the TX FIFO prior to a transmission, such as when clock stretching is disabled. Firmware may also want to respond to an external master requesting data by sending a NACK until the requested data is ready to transmit, rather than sending an ACK and then holding the bus low while the data is prepared. By default, however, Address Match and General Call Match clear the TX FIFO preventing firmware from preloading data into the TX FIFO. Firmware can change this behavior by enabling TX FIFO Preloading.

When TX FIFO Preloading is enabled, the application firmware controls ACKs to the external master using the TX Ready (*I2Cn_TXCTRL1.txrdy*) bit. When *I2Cn_TXCTRL1.txrdy* is set to 0, hardware automatically NACKs all read transactions from the Master. Setting *I2Cn_TXCTRL1.txrdy* to 1 sends an ACK to the Master on the next read transaction and transmits the data in the TX FIFO. Preloading the TX FIFO must be complete prior to setting the *I2Cn_TXCTRL1.txrdy* field to 1.

The required steps for implementing TX FIFO Preloading in an application are as follow:

1. Set *I2Cn_TXCTRL1.txrdy* to 0
2. Enable TX FIFO Preloading by setting *I2Cn_TXCTRL1.txpreld* to 1.
3. If the TX FIFO Lockout Flag (*I2Cn_INTFL0.txloi*) is set to 1, write 1 to clear the flag and enable writes to the TX FIFO.
4. Enable DMA or Interrupts if required.
5. Load the TX FIFO with the data to send when the Master sends the next read request.
6. Set *I2Cn_TXCTRL1.txrdy* to 1 to automatically let the hardware send the preloaded FIFO on the next read from a Master.
7. *I2Cn_TXCTRL1.txrdy* is cleared by hardware when a read occurs, and data is transmitted from the TX FIFO.
   a. Once cleared, the application firmware may repeat the Preloading process or disable TX FIFO Preloading.

Note: The TX FIFO Lockout flag is set if an error condition occurs while TX FIFO Preloading is enabled.

12.15 Master Mode Receiver Operation

When in Master Mode, initiating a Master Receiver operation begins with the following sequence:

1. Write the number of data bytes to receive to the I²C Receive Count field (*I2Cn_RXCTRL1.rxcnt).*
2. Write the Slave Address to the TX FIFO with the R/W bit set to 1
3. Send a START condition by setting *I2Cn_MSTR_MODE.start* = 1
4. The slave address is transmitted by the controller from the TX FIFO.
5. The I²C controller receives an ACK from the slave and the controller sets address ACK field (*I2Cn_INTFL0.adracki* = 1).
6. The I²C controller receives data from the slave and automatically ACKs each byte.
7. Once *I2Cn_RXCTRL1.rxcnt* data bytes have been received, the I²C controller sends a NACK to the slave and sets the Transfer Done Interrupt Status Flag (*I2Cn_INTFL0.donei* = 1).
8. If *I2Cn_MSTR_MODE.restart* or *I2Cn_MSTR_MODE.stop* is set, then the I²C controller sends a repeated START or STOP, respectively.

12.16 I²C Registers

The I2C0 base peripheral address is 0x4001 D000. The I2C1 base peripheral address is 0x4001 E000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 12-3: I²C Registers

Offset	Register Name	Access	Description
[0x0000]	*I2Cn_CTRL0*	R/W	I²C Control 0 Register
[0x0004]	*I2Cn_STATUS*	RO	I²C Status Register
[0x0008]	*I2Cn_INTFL0*	R/W1C	I²C Interrupt Flags 0 Register
[0x000C]	*I2Cn_INTEN0*	R/W	I²C Interrupt Enable 0 Register
[0x0010]	*I2Cn_INTFL1*	R/W1C	I²C Interrupts Flags 1 Register
[0x0014]	*I2Cn_INTEN1*	R/W	I²C Interrupts Enable 1 Register
[0x0018]	*I2Cn_FIFOLEN*	RO	I²C FIFO Length Register
[0x001C]	*I2Cn_RXCTRL0*	R/W	I²C Receive Control 0 Register
[0x0020]	*I2Cn_RXCTRL1*	R/W	I²C Receive Control 1 Register 1
[0x0024]	*I2Cn_TXCTRL0*	R/W	I²C Transmit Control 0 Register 0
[0x0028]	*I2Cn_TXCTRL1*	R/W	I²C Transmit Control 1 Register 1
[0x002C]	*I2Cn_FIFO*	R/W	I²C Transmit and Receive FIFO Register
[0x0030]	*I2Cn_MSTR_MODE*	R/W	I²C Master Mode Register

Offset	Register Name	Access	Description
[0x0034]	*I2Cn_CLKLO*	R/W	I²C Clock Low Time Register
[0x0038]	*I2Cn_CLKHI*	R/W	I²C Clock High Time Register
[0x003C]	*I2Cn_HS_CLK*	R/W	I²C Hs-Mode Clock Control Register
[0x0040]	*I2Cn_TIMEOUT*	R/W	I²C Timeout Register
[0x0044]	*I2Cn_SLADDR*	R/W	I²C Slave Address Register
[0x0048]	*I2Cn_DMA*	R/W	I²C DMA Enable Register

1.1.2 I²C Register Details

Table 12-4: I²C Control Registers 0

I²C Control 0 Register				I2Cn_CTRL0	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**	
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.	
15	hsmode	R/W	0	**High Speed Mode** Set this field to 1 to enable High-speed mode (Hs-mode) operation. This field must be set to 0 for Standard, Fast or Fast-Plus operation. 0: Hs-mode not enabled. 1: Hs-mode enabled.	
14	-	R/W	0	**Reserved for Future Use** Do not modify this field.	
13	scl_ppm	R/W	0	**SCL Push-Pull Mode Enable** Setting this field enables push-pull mode for the SCL hardware pin. This field should not be set unless any external slave device will never actively drive SCL low. 0: SCL operates in standard I²C open-drain mode 1: SCL operates in push-pull mode without the need for a pull-up resistor. Only recommended when in Master mode and external slaves will not drive SCL low.	
12	scl_strd	R/W	0	**SCL Clock Stretch Control** 0: Enable Slave clock stretching 1: Disable Slave clock stretching	
11	read	R	0	**Read/Write Bit Status** Returns the logic level of the R/W bit on a received address match (*I2Cn_INTFL0.ami* = 1) or general call match (*I2Cn_INTFL0.gci* = 1). This bit is valid for three SCL clock cycles after the address match status flag is set.	
10	swoe	R/W	0	**Software output Enabled** When set, pins SDA and SCL are directly controlled by the fields *I2Cn_CTRL0.sdao* and *I2Cn_CTRL0.sclo*, rather than the I²C controller. Setting this field to 1 enables software bit bang control of I²C. 0: The I²C controller manages the SDA and SCL pins in hardware. 1: SDA and SCL are controller by firmware using the *I2Cn_CTRL0.sdao* and *I2Cn_CTRL0.sclo* fields.	
9	sda	RO	-	**SDA Status** Returns the current logic level of the SDA pin. 0: SDA pin is logic low. 1: SDA pin is logic high.	

I²C Control 0 Register			I2Cn_CTRL0	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
8	scl	RO	-	**SCL Status** Returns the current logic level of the SCL hardware pin. 0: SCL pin is logic low. 1: SCL pin is logic high.
7	sdao	R/W	0	**SDA Pin Control** Set the state of the SDA hardware pin (actively pull low or float). 0: Pull SDA Low 1: Release SDA *Note: Only valid when I2Cn_CTRL0.swoe = 1*
6	sclo	R/W	0	**SCL Pin Control** Set the state of the SCL hardware pin (actively pull low or float). 0: Pull SCL low 1: Release SCL *Note: Only valid when I2Cn_CTRL0.swoe=1*
5	-	R/W	0	**Reserved for Future Use** Do not modify this field.
4	ack	R/W	0	**Interactive Receive Mode (IRXM) Acknowledge** If IRM is enabled (*I2Cn_CTRL0.irxm* = 1), this field determines if the hardware sends an ACK or a NACK to an IRM transaction. 0: Respond to IRM with ACK 1: Respond to IRM with NACK
3	irxm	R/W	0	**Interactive Receive Mode (IRXM)** When receiving data, allows for an Interactive Receive Mode (IRM) interrupt event after each received byte of data. The I²C peripheral hardware can be enabled to send either an ACK or NACK for IRM. See *Interactive Receive Mode* section for detailed information. 0: Disable Interactive Receive Mode 1: Enable Interactive Receive Mode *Note: Only set this field when the I²C bus is inactive.*
2	gcen	R/W	0	**General Call Address Enable** Set this field to 1 to enable General Call Address Acknowledgement. 0: Ignore General Call Address 1: Acknowledge General Call Address
1	mst	R/W	0	**Master Mode Enable** Setting this field to 1 enables Master mode operation for the I²C peripheral. Setting this field to 0 enables the I²C peripheral for Slave mode operation. 0: Slave mode enabled 1: Master mode enabled
0	i2cen	R/W	0	**I²C Enable** Set this field to 1 to enable the I²C peripheral. 0: I²C peripheral disabled 1: I²C peripheral enabled

Table 12-5: I²C Status Registers

I²C Status Register			I2Cn_STATUS	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:12	-	R/W	-	**Reserved for Future Use** Do not modify this field.
11:8	stat	RO	0	**I²C Controller Status** This field indicates the status of the I²C controller. *stat* / Controller Status: 0: Idle 1: Master Transmit address 2: Master Receive address ACK 3: Master Transmit extended address 4: Master Receive extended address ACK 5: Slave Receive address 6: Slave Transmit address ACK 7: Slave Receive extended address 8: Slave transit extended address ACK 9: Transmit data (Master or Slave) 10: Receive data ACK (Master or Slave) 11: Receive data (Master or Slave) 12: Transmit data ACK (Master or Slave) 13: NACK stage (Master or Slave) 14: Reserved for Future Use 15: Another master is addressing another slave. The transaction is ongoing but the I²C controller is not actively part of the transaction.
5	ckmd	RO	0	**SCL Drive Status** This field indicates if an external device is behaving as a master by actively driving the SCL line. 0: External device not driving SCL 1: External device is a Master actively driving the SCL pin
4	txf	RO	0	**TX FIFO Full** When set, the TX FIFO is full. 0: TX FIFO is not full 1: TX FIFO full
3	txe	RO	1	**TX FIFO Empty** If set, the TX FIFO is empty. 0: TX FIFO is not empty 1: TX FIFO is empty
2	rxf	RO	0	**RX FIFO Full** If set, the RX FIFO is full. 0: RX FIFO not full 1: RX FIFO Full
1	rxe	RO	1	**RX FIFO Empty** If set, the RX FIFO is empty. 0: RX FIFO is not empty 1: RX FIFO is empty

I²C Status Register				I2Cn_STATUS	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**	
0	busy	RO	0	**Bus Busy** If set, the I²C bus is active. 0: Bus is idle 1: Bus is busy	

Table 12-6: I²C Interrupt Status Flags Registers 0

I²C Interrupt Status Flags 0 Register				I2Cn_INTFL0	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**	
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.	
15	txloi	R/W1C	0	**TX FIFO Locked Interrupt Flag** If set, the TX FIFO is locked and writes to the TX FIFO are ignored. This field is set to 1 by hardware to prevent stale data from being transmitted from the TX FIFO. When set, the TX FIFO is automatically flushed. Writes to the TX FIFO are ignored until this flag is cleared. Write 1 to clear. 0: TX FIFO not locked. 1: TX FIFO is locked and all writes to the TX FIFO are ignored.	
14	stoperi	R/W1C	0	**Out of Sequence STOP Interrupt Flag** This flag is set if a STOP condition occurs on the I²C Bus out of expected sequence. Write 1 to clear this field. Writing 0 has no effect. 0: Error condition has not occurred. 1: Out of sequence STOP condition occurred.	
13	strteri	R/W1C	0	**Out of Sequence START Interrupt Flag** This flag is set if a START condition occurs on the I²C Bus out of expected sequence. Write 1 to clear this field. Writing 0 has no effect. 0: Error condition has not occurred. 1: Out of sequence START condition occurred.	
12	dnreri	R/W1C	0	**Slave Mode Do Not Respond Interrupt Flag** This occurs if an address match is made, but the TX FIFO or RX FIFO is not ready. Write 1 to clear this field. Writing 0 has no effect. 0: Error condition has not occurred. 1: I²C address match has occurred and either the TX or RX FIFO is not configured.	
11	dateri	R/W1C	0	**Master Mode: Data NACK from External Slave Interrupt Flag** This flag is set by hardware if a NACK is received from a slave on the I²C bus. This flag is only valid if the I2Cn peripheral is configured for Master Mode operation. Write 1 to clear. Write 0 has no effect. 0: Error condition has not occurred. 1: Data NACK received from a slave.	
10	adreri	R/W1C	0	**Master Mode: Address NACK from Slave Error Flag** This flag is set by hardware if an Address NACK is received from a slave on the I²C bus. This flag is only valid if the I2Cn peripheral is configured for Master Mode operation. Write 1 to clear. Write 0 has no effect. 0: Error condition has not occurred. 1: Address NACK received from a slave.	
9	toeri	R/ W1C	0	**Timeout Error Interrupt Flag** This occurs when this device holds SCL low longer than the programmed timeout value. Applies to both Master and Slave Mode. Write 1 to clear. Write 0 has no effect. 0: Timeout error has not occurred. 1: Timeout error occurred.	

I²C Interrupt Status Flags 0 Register		I2Cn_INTFL0		[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
8	arberi	R/ W1C	0	**Master Mode Arbitration Lost Interrupt Flag** Write 1 to clear. Write 0 has no effect. 0: Condition has not occurred. 1: Condition occurred.
7	adracki	R/ W1C	0	**Master Mode: Address ACK from External Slave Interrupt Flag** This field is set when a slave address ACK is received. Write 1 to clear. Write 0 has no effect. 0: Condition has not occurred. 1: The slave device ACK for the address was received.
6	stopi	R/ W1C	0	**Slave Mode: STOP Condition Interrupt Flag** This flag is set by hardware when a STOP condition is detected on the I²C bus. Write 1 to clear. Write 0 has no effect. 0: Stop condition has not occurred 1: Stop condition occurred
5	txthi	RO	1	**TX FIFO Threshold Level Interrupt Flag** This field is set by hardware if the number of bytes in the Transmit FIFO is less than or equal to the Transmit FIFO threshold level. Write 1 to clear. This field is automatically cleared by hardware when the TX FIFO contains fewer bytes than the TX threshold level. 0: TX FIFO contains more bytes than the TX threshold level. 1: TX FIFO contains TX threshold level or fewer bytes (Default).
4	rxthi	RO	1	**RX FIFO Threshold Level Interrupt Flag** This field is set by hardware if the number of bytes in the Receive FIFO is greater than or equal top the Receive FIFO threshold level. This field is automatically cleared when the RX FIFO contains fewer bytes than the RX threshold setting. 0: RX FIFO contains fewer bytes than the RX threshold level. 1: RX FIFO contains at least RX threshold level of bytes (Default).
3	ami	R/W1C	0	**Slave Mode: Address Match Status Interrupt Flag** In Slave Mode operation, a slave mode address match condition has occurred. Write 1 to clear. Writing 0 has no effect. 0: Slave address match has not occurred. 1: Slave address match occurred.
2	gci	R/W1C	0	**Slave Mode: General Call Address Match Received Interrupt Flag** In Slave Mode operation, a general call address match condition has occurred. Write 1 to clear. Writing 0 has no effect. 0: General call address match has not occurred. 1: General call address match occurred.
1	irxmi	R/W1C	0	**Interactive Receive Mode Interrupt Flag** Write 1 to clear. Writing 0 is ignored. 0: Interrupt condition has not occurred. 1: Interrupt condition occurred.
0	donei	R/W1C	0	**Transfer Complete Interrupt Flag** This flag is set for both Master and Slave mode for both transmit and receive operations on the SCL falling edge after an ACK is received or sent. Write 1 to clear. Writing 0 has no effect. 0: Transfer is not complete. 1: Transfer complete.

Table 12-7: I²C Interrupt Enable 0 Registers

I²C Interrupt Enable 0 Register			I2Cn_INTEN0	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.
15	txloie	R/W	0	**TX FIFO Locked Out Interrupt Enable** Set this field to enable events for TX FIFO lock events. 0: Interrupt disabled. 1: Interrupt enabled.
14	stoperie	R/W	0	**Out of Sequence STOP condition detected Interrupt Enable** Set this field to enable events for an out of sequence STOP condition event. 0: Interrupt disabled. 1: Interrupt enabled.
13	strterie	R/W	0	**Out of Sequence START condition detected Interrupt Enable** Set this field to enable events for an out of sequence START condition event. 0: Interrupt disabled. 1: Interrupt enabled.
12	dnrerie	R/W	0	**Slave Mode Do Not Respond Interrupt Enable** Set this field to enable events in Slave Mode when the Do Not Respond condition occurs. 0: Interrupt disabled. 1: Interrupt enabled.
11	daterie	R/W	0	**Master Mode Received Data NACK from Slave Interrupt Enable** Set this field to enable events for Master Mode external device data NACK events. 0: Interrupt disabled. 1: Interrupt enabled.
10	adrerie	R/W	0	**Master Mode Received Address NACK from Slave Interrupt Enable** Set this field to enable events for Master Mode slave device address NACK events. 0: Interrupt disabled. 1: Interrupt enabled.
9	toerie	R/W	0	**Timeout Error Interrupt Enable** Set this field to enable events for a timeout error interrupt event. 0: Interrupt disabled. 1: Interrupt enabled.
8	arberie	R/W	0	**Master Mode Arbitration Lost Interrupt Enable** Set this field to enable events in Master Mode for arbitration lost events. 0: Interrupt disabled. 1: Interrupt enabled.
7	adrackie	R/W	0	**Received Address ACK from Slave Interrupt Enable** Set this field to enable events for Master Mode slave device address ACK events. 0: Interrupt disabled. 1: Interrupt enabled.
6	stopie	R/W	0	**STOP Condition Detected Interrupt Enable** Set this field to enable interrupt events for STOP conditions. 0: Interrupt disabled. 1: Interrupt enabled.
5	txthie	R/W	0	**TX FIFO Threshold Level Interrupt Enable** Set this field to enable interrupt events when a TX FIFO threshold event occurs. 0: Interrupt disabled. 1: Interrupt enabled.

I²C Interrupt Enable 0 Register			I2Cn_INTEN0	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
4	rxthie	R/W	0	**RX FIFO Threshold Level Interrupt Enable** Set this field to enable interrupt events when an RX FIFO threshold event occurs. 0: Interrupt disabled. 1: Interrupt enabled.
3	amie	R/W	0	**Slave Mode Incoming Address Match Interrupt Enable** Set this field to enable the slave mode address match interrupt event. 0: Interrupt disabled. 1: Interrupt enabled.
2	gcie	R/W	0	**Slave Mode General Call Address Match Received Interrupt Enable** Set this field to enable the slave mode general call address match received interrupt event. 0: Interrupt disabled. 1: Interrupt enabled.
1	irxmie	R/W	0	**Interactive Receive Interrupt Enable** Set this field to enable the interactive receive interrupt event. 0: Interrupt disabled. 1: Interrupt enabled.
0	doneie	R/W	0	**Transfer Complete Interrupt Enable** Set this field to enable the transfer complete interrupt event. 0: Interrupt disabled. 1: Interrupt enabled.

Table 12-8: I²C Interrupt Status Flags 1 Registers

I²C Interrupt Status Flags 1 Register			I2Cn_INTFL1	[0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
31:2	-	R/W	-	**Reserved for Future Use** Do not modify this field.
1	txufi	R/W1C	0	**Slave Mode: TX FIFO Underflow Status Flag** In Slave Mode operation, the hardware sets this flag automatically if the TX FIFO is empty and the master requests more data by sending an ACK after the previous byte transferred. 0: Slave mode TX FIFO underflow condition has not occurred. 1: Slave mode TX FIFO underflow condition occurred.
0	rxofi	R/W1C	0	**Slave Mode: RX FIFO Overflow Status Flag** In Slave Mode operation, the hardware sets this flag automatically when an RX FIFO overflow occurs. Write 1 to clear. Writing 0 has no effect. 0: Slave mode RX FIFO overflow event has not occurred. 1: Slave mode RX FIFO overflow condition occurred (data lost).

Table 12-9: I²C Interrupt Enable Registers 1

I²C Interrupt Enable 1 Register			I2Cn_INTEN1	[0x0014]
Bits	**Name**	**Access**	**Reset**	**Description**
31:2	-	R/W	0	**Reserved for Future Use** Do not modify this field.
1	txufie	R/W	0	**Slave Mode TX FIFO Underflow Interrupt Enable** In slave mode operation, set this field to enable the TX FIFO underflow interrupt. 0: Interrupt disabled. 1: Interrupt enabled.

I²C Interrupt Enable 1 Register			I2Cn_INTEN1	[0x0014]
Bits	**Name**	**Access**	**Reset**	**Description**
0	rxofie	R/W	0	**Slave Mode RX FIFO Overflow Interrupt Enable** In slave mode operation, set this field to enable the RX FIFO overflow interrupt. 0: Interrupt disabled. 1: Interrupt enabled.

Table 12-10: I²C FIFO Length Registers

I²C FIFO Length Register			I2Cn_FIFOLEN	[0x0018]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.
15:8	txlen	RO	8	**TX FIFO Length** Returns the length of the TX FIFO. 8: 8-byte TX FIFO.
7:0	rxlen	RO	8	**RX FIFO Length** Returns the length of the RX FIFO. 8: 8-byte RX FIFO.

Table 12-11: I²C Receive Control Registers 0

I²C Receive Control Register 0			I2Cn_RXCTRL0	[0x001C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:12	-	R/W	0	**Reserved for Future Use** Do not modify this field.
11:8	rxth	R/W	0	**RX FIFO Threshold Level** Set this field to the required number of bytes to trigger a RX FIFO threshold event. When the number of bytes in the RX FIFO is equal to or greater than this field, the hardware sets the *I2Cn_INTFL0.rxthi* bit indicating an RX FIFO threshold level event. 0: 0 bytes or more in the RX FIFO causes a threshold event. 1: 1+ bytes in the RX FIFO triggers a receive threshold event (recommended minimum value). … 8: RX FIFO threshold event only occurs when the RX FIFO is full.
7	rxfsh	R/W1O	0	**Flush RX FIFO** Write 1 to this field to initiate a RX FIFO flush, clearing all data in the RX FIFO. This field is automatically cleared by hardware when the RX FIFO flush completes. Writing 0 has no effect. 0: RX FIFO flush complete or not active. 1: Flush the RX FIFO
6:1	-	R/W	0	**Reserved for Future Use** Do not modify this field.
0	dnr	R/W	0	**Do Not Respond** Slave mode operation only. 0: If the RX FIFO contains data and an external master requests a WRITE transaction, respond to an address match with an ACK but NACK the subsequent data byte(s). (No additional data is written into the RX FIFO.) 1: If the RX FIFO contains data and a master requests a WRITE transaction, do not respond to an address match and send a NACK instead.

Table 12-12: I²C Receive Control 1 Registers

I²C Receive Control 1 Register			I2Cn_RXCTRL1	[0x0020]
Bits	**Name**	**Access**	**Reset**	**Description**
31:12	-	R/W	0	**Reserved for Future Use** Do not modify this field.
11:8	rxfifo	R	0	**RX FIFO Byte Count** Returns the number of bytes currently in the RX FIFO. 0: No data in the RX FIFO. ... 8: 8 bytes in the RX FIFO (max value).
7:0	rxcnt	R/W	1	**RX FIFO Transaction Byte Count** When in Master Mode, write the number of bytes to be received in a transaction from 1 to 256. 0 represents 256. 0: 256 byte receive transaction. 1: 1 byte receive transaction. 2: 2 byte receive transaction. … 255: 255 byte receive transaction.

Table 12-13: I²C Transmit Control Registers 0

I²C Transmit Control Register 0			I2Cn_TXCTRL0	[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
31:12	-	R/W	-	**Reserved for Future Use** Do not modify this field.
11:8	txth	R/W	0	**TX FIFO Threshold Level** Sets the level for a Transmit FIFO threshold event interrupt. If the number of bytes remaining in the TX FIFO falls to this level or lower the interrupt flag *I2Cn_INTFL0.txthi* is set indicating a TX FIFO Threshold Event occurred. 0: 0 bytes remaining in the TX FIFO triggers a TX FIFO threshold event. 1: 1 byte or less remaining in the TX FIFO triggers a TX FIFO threshold event (recommended minimum value). … 7: 7 or fewer bytes remaining in the TX FIFO triggers a TX FIFO threshold event
7	txfsh	R/W1O	0	**TX FIFO Flush** Write this field to 1 to initiate a TX FIFO flush, clearing all remaining data from the transmit FIFO. 0: TX FIFO flush is complete or not active. 1: Flush the TX FIFO *Note: Hardware automatically clears this bit to 0 after it is written to 1 when the flush is completed.* *Note: If I2Cn_INTFL0.txloi = 1, then I2Cn_TXCTRL0.txfsh = 1.*
6:1	-	R/W	0	**Reserved for Future Use** Do not modify this field.

I²C Transmit Control Register 0			I2Cn_TXCTRL0	[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
0	txpreld	R/W	0	**TX FIFO Preload Mode Enable** 0: Normal operation. An address match in Slave Mode, or a General Call address match, will flush and lock the TX FIFO so it cannot be written and set *I2Cn_INTFL0.txloi.* 1: TX FIFO Preload Mode. An address match in Slave Mode, or a General Call address match, will not lock the TX FIFO and will not set *I2Cn_INTFL0.txloi.* This allows firmware to preload data into the TX FIFO. The status of the I²C is controllable at *I2Cn_TXCTRL1.txrdy.*

Table 12-14: I²C Transmit Control Registers 1

I²C Transmit Control Register 1			I2Cn_TXCTRL1	[0x0028]
Bits	**Name**	**Access**	**Reset**	**Description**
31:12	-	R/W	0	**Reserved for Future Use** Do not modify this field.
11:8	txfifo	RO	0	**Transmit FIFO Byte Count Status** Contains the number of bytes remaining in the TX FIFO
7:6	-	R/W	0	**Reserved for Future Use** Do not modify this field.
5	flsh_nack_dis	R/W	0	**TX FIFO Auto Flush Disable for NACK** Setting this field to 1 disables the TX FIFO Automatic Flush when a NACK is received at the end of a slave transaction. 0: The TX FIFO is automatically flushed if a NACK is received at the end of a slave transaction. 1: The TX FIFO is ***not flushed*** when a NACK is received at the end of a slave transaction. *Note: This field is valid for slave mode operation only and is ignored if the I²C mode is set to master.*
4	flsh_sladdr_dis	R/W	0	**TX FIFO Auto Flush Disable for Slave Address Match** Setting this field to 1 disables the TX FIFO Automatic Flush when a Slave Address Match occurs. 0: The TX FIFO is automatically flushed on a Slave Address Match. 1: The TX FIFO is ***not flushed*** on a Slave Address Match. *Note: This field is valid for slave mode operation only and is ignored if the I²C mode is set to master.*
3	-	R/W	0	**Reserved for Future Use** Do not modify this field.
2	flsh_gcaddr_dis	R/W	0	**TX FIFO Auto Flush Disable on General Call Address Match** Setting this field to 1 disables the TX FIFO Automatic Flush when a General Call Address Match occurs. 0: The TX FIFO is automatically flushed on a General Call Address Match. 1: The TX FIFO is ***not flushed*** on a General Call Address Match. *Note: This field is valid for slave mode operation only and is ignored if the I²C mode is set to master.*

I²C Transmit Control Register 1			I2Cn_TXCTRL1	[0x0028]
Bits	**Name**	**Access**	**Reset**	**Description**
1	txlast	R/W1O	0	**Slave Mode Transmit Last** This bit decides what to do if the I²C is in Slave Mode, is transmitting data to a Master, and the TX FIFO is empty. 0: Hold SCL low. This pauses transmission until data is written to the TX FIFO. 1: End transaction by releasing SCL. Cleared on a STOP/RESTART condition, or if *I2Cn_INTFL0.txloi*=1 (transmit FIFO locked for writing).
0	txrdy	R/W1O	1	**Transmit FIFO Preload Ready Status** When TX FIFO Preload Mode is enabled, *I2Cn_TXCTRL0.txpreld* = 1, this bit is automatically cleared to 0. While this bit is 0, if the I²C hardware receives a slave address match a NACK is sent. Once the I²C hardware is ready (firmware has preloaded the TX FIFO, configured the DMA, etc.) application firmware must set this bit to 1 so the I²C hardware will send an ACK on a slave address match. When TX FIFO Preload Mode is disabled, *I2Cn_TXCTRL0.txpreld* = 1, this bit is forced to 1 and the I²C hardware behaves normally.

Table 12-15: I²C Data Registers

I²C Data Register			I2Cn_FIFO	[0x002C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	R/W	0	**Reserved for Future Use** Do not modify this field.
7:0	data	R/W	0xFF	**I²C FIFO Data Register** Reading from this register pops data from the RX FIFO and writes to this register pushes data onto the TX FIFO. If the RX FIFO is empty, reading this field returns 0xFF. *Note: If the RX FIFO is empty, reads from this field return 0xFF.* *Note: If the TX FIFO is full, writes to this field are ignored.*

Table 12-16: I²C Master Mode Control Registers

I²C Master Mode Control Register			I2Cn_MSTR_MODE	[0x0030]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	R/W	0	**Reserved for Future Use** Do not modify this field.
7	sea	R/W	0	**Slave Extended Addressing** 0: Send a 7-bit address to the slave 1: Send a 10-bit address to the slave
6:3	-	R/W	0	**Reserved for Future Use** Do not modify this field.
2	stop	R/W1O	0	**Send STOP Condition** 0: Stop condition completed or inactive. 1: Send a STOP Condition *Note: This bit is automatically cleared by hardware when the STOP condition begins.*

I²C Master Mode Control Register			I2Cn_MSTR_MODE	[0x0030]
Bits	**Name**	**Access**	**Reset**	**Description**
1	restart	R/W1O	0	**Send Repeated START Condition** After sending data to a slave, instead of sending a STOP condition the master may send another START to retain control of the bus. 0: Repeated start condition complete or inactive. 1: Send a Repeated START *Note: This bit is automatically cleared by hardware when the repeated START condition begins.*
0	start	R/W1O	0	**Start Master Mode Transfer** 0: Master mode transfer inactive. 1: Start Master Mode Transfer *Note: This bit is automatically cleared by hardware when the transfer is completed or aborted.*

Table 12-17: I²C SCL Low Control Register

I²C Clock Low Control			I2Cn_CLKLO	[0x0034]
Bits	**Name**	**Access**	**Reset**	**Description**
31:9	-	R/W	0	**Reserved for Future Use** Do not modify this field.
8:0	scl_lo	R/W	1	**Clock Low Time** In Master Mode, this configures the SCL low time. $t_{SCL\_LOW} = f_{I2C\_CLK} \times (scl\_lo + 1)$ *Note: 0 is not a valid setting for this field.*

Table 12-18: I²C SCL High Control Register

I²C Clock High Control Register			I2Cn_CLKHI	[0x0038]
Bits	**Name**	**Access**	**Reset**	**Description**
31:9	-	R/W	0	**Reserved for Future Use** Do not modify this field.
8:0	scl_hi	R/W	1	**Clock High Time** In Master Mode, this configures the SCL high time. $t_{SCL\_HIGH} = {}^{1}/_{f_{I2C\_CLK}} \times (scl\_hi + 1)$ In both Master and Slave Mode, this also configures the time SCL is held low after new data is loaded from the TX FIFO or after firmware clears irxmi during Interactive Receive Mode. *Note: 0 is not a valid setting for this field.*

Table 12-19: I²C Timeout Registers

I²C Timeout Register			I2Cn_HS_CLK	[0x003C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.

I²C Timeout Register			I2Cn_HS_CLK	[0x003C]
Bits	**Name**	**Access**	**Reset**	**Description**
15:8	hs_clk_hi	R/W	0	**Hs-Mode Clock High Time** This field sets the Hs-Mode clock low count. In Slave mode, this is the time SCL is held low after data is output on SDA. The following equation defined the clock low time. $t_{HS\_CLK\_HI} = {}^{1}/_{f_{I2C\_CLK}} \times (hs\_clk\_hi + 1)$ *Note: Refer to section JL:KIJ:KLJ:LKJ:KLJ:L for details on the requirements for the Hs-mode clock high and low times.*
7:0	hs_clk_lo	R/W	0	**Hs-Mode Clock Low Time** This field sets the Hs-Mode clock low count. In Slave mode, this is the time SCL is held low after data is output on SDA. The following equation defined the clock low time. $t_{HS\_CLK\_LO} = {}^{1}/_{f_{I2C\_CLK}} \times (hs\_clk\_lo + 1)$ *Note: Refer to section 12.13 SCL Clock Generation for Hs-Mode for details on the requirements for the Hs-mode clock high and low times.*

Table 12-20: I²C Timeout Registers

I²C Timeout Register			I2Cn_TIMEOUT	[0x0040]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.
15:0	to	R/W	0	**Bus Error SCL Timeout Period** Set this value to the number of I²C clock cycles desired to cause a bus timeout error. The I2Cn peripheral timeout timer starts when it pulls SCL low. After the I2Cn peripheral releases the line, if the line is not pulled high prior to the timeout number of I²C clock cycles, a bus error condition is set (*I2Cn_INTFL0.toeri* = 1) and the I2Cn peripheral releases the SCL and SDA lines 0: Timeout disabled. All other values result in a timeout calculation of: $t_{BUS\_TIMEOUT} = {}^{1}/_{f_{I2C\_CLK}} \times to$ *Note: The timeout counter monitors the I2Cn controller's driving of the SCL pin, not an external I²C master driving the SCL pin.*

Table 12-21: I²C Slave Address Register

I²C Slave Address Register			I2Cn_SLADDR	[0x0044]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.
15	ea	R/W	0	**Slave Mode Extended Address Select** When this I²C is operating in Slave Mode, this bit selects whether sla contains a 7-bit or 10-bit address. 0: 7-bit addressing 1: 10-bit addressing
14:10	-	R/W	0	**Reserved for Future Use** Do not modify this field.

I²C Slave Address Register		I2Cn_SLADDR		[0x0044]
Bits	**Name**	**Access**	**Reset**	**Description**
9:0	sla	R/W	0	**Slave Mode Slave Address** When I²C peripheral is operating in Slave Mode, *I2Cn_CTRL0.mst* = 0, this field must be set as the desired slave address for the peripheral.

Table 12-22: I²C DMA Register

I²C DMA Register		I2Cn_DMA		[0x0048]
Bits	**Name**	**Access**	**Reset**	**Description**
31:2	-	R/W	0	**Reserved for Future Use** Do not modify this field.
1	rxen	R/W	0	**RX DMA Channel Enable** 0: Disable RX DMA channel 1: Enable RX DMA channel
0	txen	R/W	0	**TX DMA Channel Enable** 0: Disable TX DMA channel 1: Enable TX DMA channel

13 Serial Peripheral Interface 0 (SPI0)

The Serial Peripheral Interface 0 (SPI0) is a highly configurable, synchronous communications peripheral that interfaces to SPI devices and supports both Master and Slave modes.

13.1 SPI Port 0

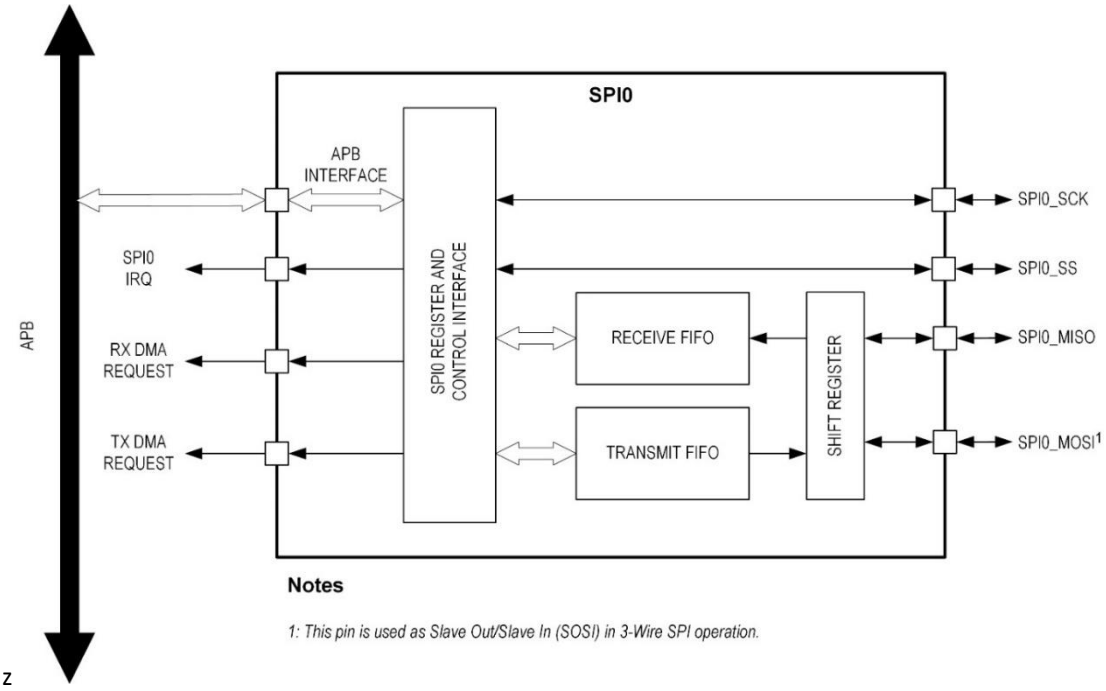
Features:

Four-Wire, full-duplex communication

- Three-Wire, half-duplex communication supported
- Selectable number of bits per character
  - 1-bit to 16-bits
- Master and Slave mode support
- Wakeup from *SLEEP* based on configurable Transmit and Receive FIFO Levels
- Four SPI modes (mode 0, 1, 2, and 3)
- Programmable Serial Clock (SCK) frequency and duty cycle
- Master mode operation up to 60MHz
- Slave mode operation up to 48MHz
- Independent clock phase and clock polarity control enabling support for SPI Mode 0, 1, 2, and 3
- 32-byte Transmit FIFO, 32-byte Receive FIFO

The SPI0 Peripheral are independently configurable as for master or slave operation. A SPI network uses a single master with one or more slave devices for a given transaction. A high level block diagram of the SPI0 peripheral is shown in *Figure 13-1*, *below*.

Figure 13-1: SPI0 Block Diagram



z

13.2 Overview

13.2.1 Four-Wire SPI Signals

SPI devices operate as either a Master or Slave device. In four-wire SPI, four signals are required for communication as shown in *Table 13-1*, *below*.

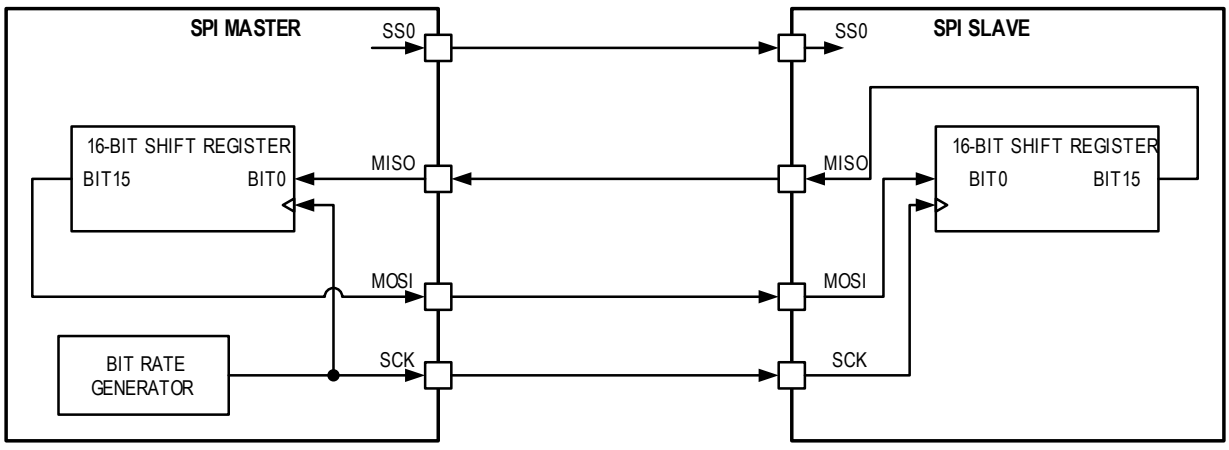
Table 13-1: Four-Wire SPI Signals

Signal	Description	Direction
SCK	Serial Clock	The master generates the Serial Clock signal, which is an output from the master and an input to the slave.
MOSI	Master Output Slave Input	In master mode, this signal is used as an output for sending data to the Slave. In slave mode this is the input data from the Master.
MISO	Master Input Slave Output	In master mode, this signal is used as an input for receiving data from the Slave. In slave mode, this signal is an output for transmitting data to the Master.
SS	Slave Select	In master mode, this signal is an output used to select a slave device prior to communication. In slave mode, this signal is an input used to indicate the Master is going to start communication.

The MAX32660 supports a single slave select pin, SPI0_SS0, for SPI0.

A typical four-wire SPI network is shown in *Figure 13-2*, *below*. In a typical SPI network, the master device selects the slave device using the slave select output. The master starts the communication by selecting the slave device by asserting the slave select output. The master then starts the SPI clock via the SCK output pin. When a slave device's slave select pin is deasserted, the device is required to put the SPI pins in tri-state mode.

Figure 13-2: 4-Wire SPI Connection Diagram



13.2.2 Three-Wire SPI Signals

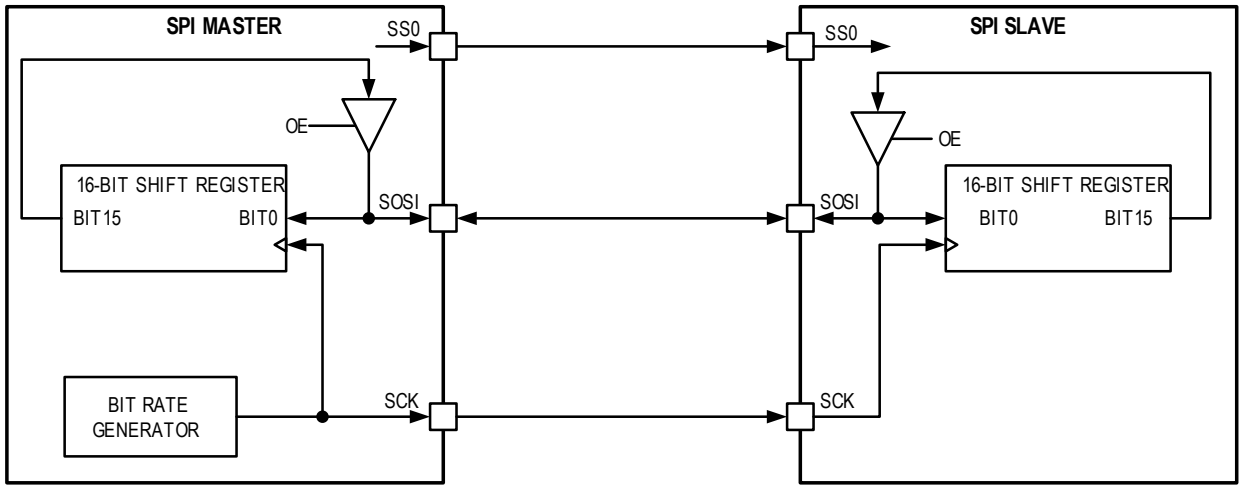
*Three-wire SPI is supported by the **MAX32660** family of microcontrollers. In this variant the MOSI and MISO lines are combined and used as a bi-directional half-duplex communication pin. Three-wire also uses a serial clock generated by the master and a slave select pin controlled by the master. Table 13-1 describes each of the signals used in three-wire SPI communication.*

Table 13-2: Three-Wire SPI Signals

Signal	Description	Direction
SCK	Serial Clock	The master generates the Serial Clock signal, which is an output from the master and an input to the slave.
SISO	Slave Input Slave Output	This is a half-duplex, bidirectional I/O pin used for communication between the SPI master and slave in a three-wire SPI communication network
SS	Slave Select	In master mode, this signal is an output used to select a slave device prior to communication. In slave mode, this signal is an input used to indicate an external SPI master is starting communication.

A three-wire SPI network is shown in *Figure 13-3*, *below*. The master device selects the slave device using the slave select output. The communication starts with the master asserting the slave select line and then starting the clock (SCK). In a 3-Wire SPI communication, the master and slave must both know the intended direction of the data. For a write, the master drives the data out the SISO pin. For a read, the master must release the SISO line and let the slave drive the SISO line. In the MAX32660 the direction of transmission is controlled using the FIFO enables. See *Three-Wire SPI Read and Write* for detailed information.

Figure 13-3: 3-Wire SPI Connection Diagram



13.3 SPI Configuration

Before configuring the SPI peripheral, first disable the SPI port by setting *SPI0_CTRL0.spi_en* to 0.

13.3.1 Pin Configuration

Pin selection and configuration is required to use the SPI port. *Table 13-3* shows the pin selection options for each of the SPI ports for each package available. The Alternate Function Name column maps the typical SPI Signal name to the Alternate Function name on the MAX32660 family of parts. The required pins for SPI0 are all mapped to Alternate Function 1 on GPIO Port 0 and are available on the same pins in both the 16-WLP and the 20-TQFN as shown in *Table 13-3*.

Table 13-3: SPI0 Pins

SPI Port	SPI Signal	Alternate Function Name	Alternate Function Number	GPIO	
				16-WLP	20-TQFN
SPI0	SCK	SPI0_SCK	AF1	P0.6	P0.6
	MOSI (SISO)	SPI0_MOSI	AF1	P0.5	P0.5
	MISO	SPI0_MISO	AF1	P0.4	P0.4
	SS	SPI0_SS0	AF1	P0.7	P0.7

Four-wire SPI uses SCK, MISO, MOSI, and the SS pin, whereas three-wire SPI uses SCK, MOSI(SISO) and the SS pin. The following steps outline setting up the GPIO0 pins for the SPI alternate function usage for SPI0.

1. Set the GPIO pin for alternate function operation.
   a. SPI0_SCK: Set GPIO0_EN[6] to 0.
   b. SPI0_MOSI (SISO): Set GPIO0_EN[5] to 0.
   c. SPI0_MISO: Set GPIO0_EN[4] to 0.
      i. If using three-wire SPI this step is not required.
   d. SPI0_SS0: Set GPIO0_EN[7] to 0.
2. Select AF1 using the GPIO0_AF_SEL register.
   a. SPI0_SCK: Set GPIO0_AF_SEL[6] to 0
   b. SPI0_MOSI: Set GPIO0_AF_SEL[5] to 0.
   c. SPI0_MISO: Set GPIO0_AF_SEL[4] to 0.
      i. If using three-wire SPI this step is not required.
   d. SPI0_SS0: Set GPIO0_AF_SEL[7] to 0.

13.3.2 Master Configuration

For SPI Master operation, set the master enable bit to 1 (*SPI0_CTRL0.mm_en* = 1).

In a single master network the slave select pin operates as an output only. Configure the slave select pin as an output by setting *SPI0_CTRL0.ss_io* to 1. Additionally, prior to starting a SPI Master mode transaction, set the Slave Select Enable bit to 1 (*SPI0_CTRL0.ss_sel* = 1) to use the SPI0_SS0 pin as the slave select output.

13.3.3 Slave Configuration

For slave operation, set the master enable bit to 0 (*SPI0_CTRL0.mm_en* = 0). In slave mode, the slave select pin is used for to indicate that the SPI Master is starting a communication with the SPI Slave. Set the slave select pin as an input by setting *SPI0_CTRL0.ss_io* to 0.

13.3.4 Three and Four Wire SPI Configuration

Select three-wire SPI or four-wire SPI communication using the *SPI0_CTRL2*.*three_wire* bit. Set *SPI0_CTRL2*.*three_wire* = 0 for four-wire mode or set this bit to 1 to select three-wire operation.

13.3.5 SPI Peripheral Clock

The System Peripheral Clock, PCLK, drives the SPI0 peripheral clock. The SPI0 provides an internal clock, SPI0_CLK, that is used within the SPI peripheral for the base clock to control the module and generate the SCK clock when in master mode. Set the SPI0 internal clock using the field *SPI0_CLK_CFG*.*scale* as shown in *Equation 13-1*. Valid settings for *SPI0_CLK_CFG*.*scale* are 0 to 8, allowing a divisor of 1 to 256.

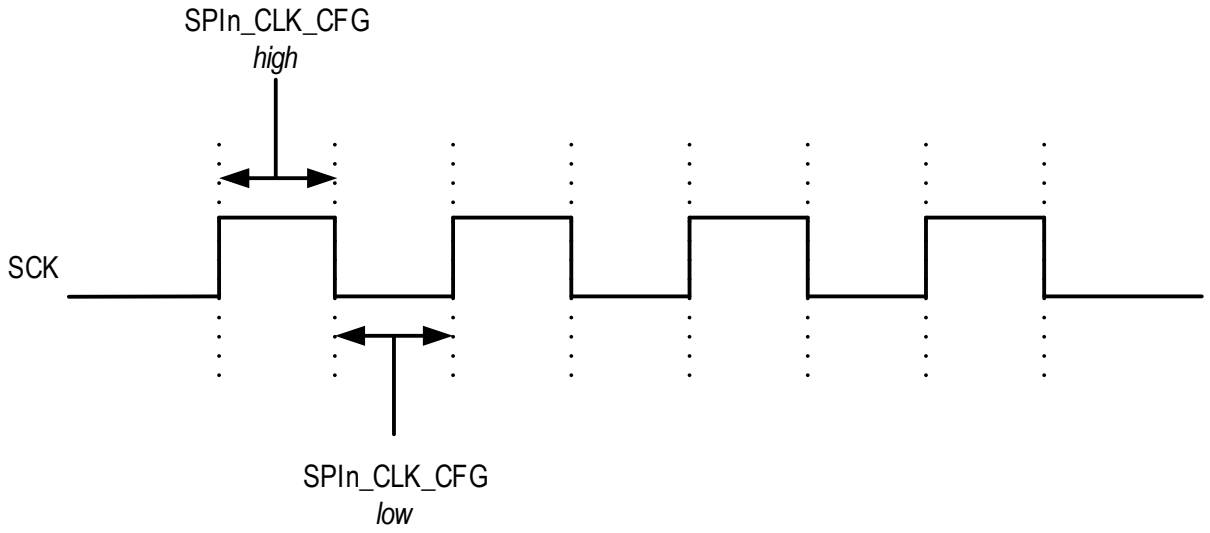
Equation 13-1: SPI Peripheral Clock

$$f_{SPI\_CLK} = \frac{f_{PCLK}}{2^{scale}}$$

13.3.6 Master Mode Serial Clock Generation

In master and multi-master mode the SCK clock is generated by the master. The SPI0 provides control for both the high time and low time of the SCK clock. This control allows setting the high and low times for the SCK to duty cycles other than 50% if required. The SCK clock uses the SPI peripheral clock as a base value and the high and low values are a count of the number of f_{SPI_CLK} clocks. *Figure 13-4*, *below*, visually represents the use of the *SPI0_CLK_CFG*.*hi* and *SPI0_CLK_CFG*.*low* fields. Refer to *Equation 13-2* and *Equation 13-3* for calculating the SCK high and low time from the *hi* and *low* field values.

Figure 13-4: SCK Clock Rate Control



Equation 13-2: SCK High Time

$$t_{SCK\_HI} = t_{SPI\_CLK} \times SPIn\_CLK\_CFG.high$$

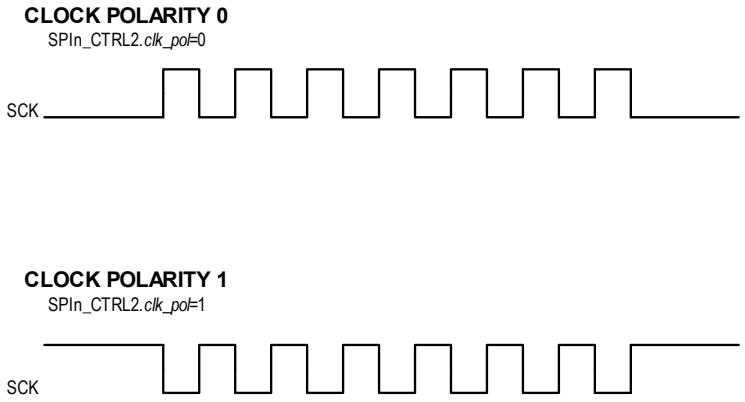
Equation 13-3: SCK Low Time

$$t_{SCK\_LOW} = t_{SPI\_CLK} \times SPIn\_CLK\_CFG.low$$

13.3.7 Clock Phase and Polarity Control

SPIn supports four combinations of clock and phase polarity as shown in *Table 13-4*, *below*. Clock polarity is controlled using the bit *SPI0_CTRL2.clk_pol* and determines if the clock is active high or active low as shown in *Figure 13-5*. Clock polarity does not affect the transfer format for SPI. Clock phase determines when the data must be stable for sampling. Setting the clock phase to 0, *SPI0_CTRL2.clk_pha* = 0, dictates the SPI data is sampled on the initial SPI clock edge regardless of clock polarity. Phase 1, *SPI0_CTRL2.clk_pha* = 1, results in data sample occurring on the second edge of the clock regardless of clock polarity.

Figure 13-5: SPI Clock Polarity



For proper data transmission, the clock phase and polarity must be identical for the SPI master and slave. The master always places data on the MOSI line a half-cycle before the SCLK edge for the slave to latch the data.

Table 13-4. Clock Phase and Polarity Operation

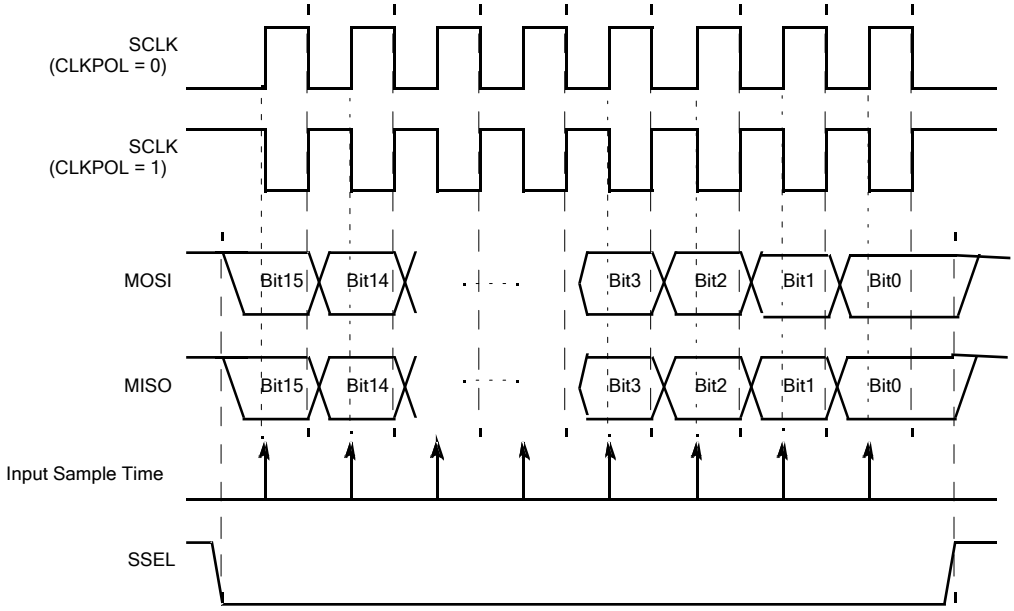
SPI0_CTRL2 clk_pha	SPI0_CTRL2 clk_pol	SCK Transmit Edge	SCK Receive Edge	SCK Idle State
0	0	Falling	Rising	Low
0	1	Rising	Falling	High
1	0	Rising	Falling	Low
1	1	Falling	Rising	High

13.3.8 Transfer Format Phase 0

Figure 13-7 is the timing diagram for an SPI 16-bit transfer in which the clock phase is cleared (*SPI0_CTRL2.clk_pha* = 0). The two SCK waveforms show active low (*SPI0_CTRL2.clk_pol* = 0) and active high (*SPI0_CTRL2.clk_pol* = 1). The diagram may be interpreted as either a master or slave timing diagram since the SCLK, MISO and MOSI pins are directly connected between the master and the slave.

In the case of multi-character transfers with SS0 remaining asserted between characters, the output data will change at the end of the Bit0 (final clock edge) to reflect the output value for Bit15 of the next character.

Figure 13-6. SPI Timing (SPI0_CTRL2.clk_pha = 0)

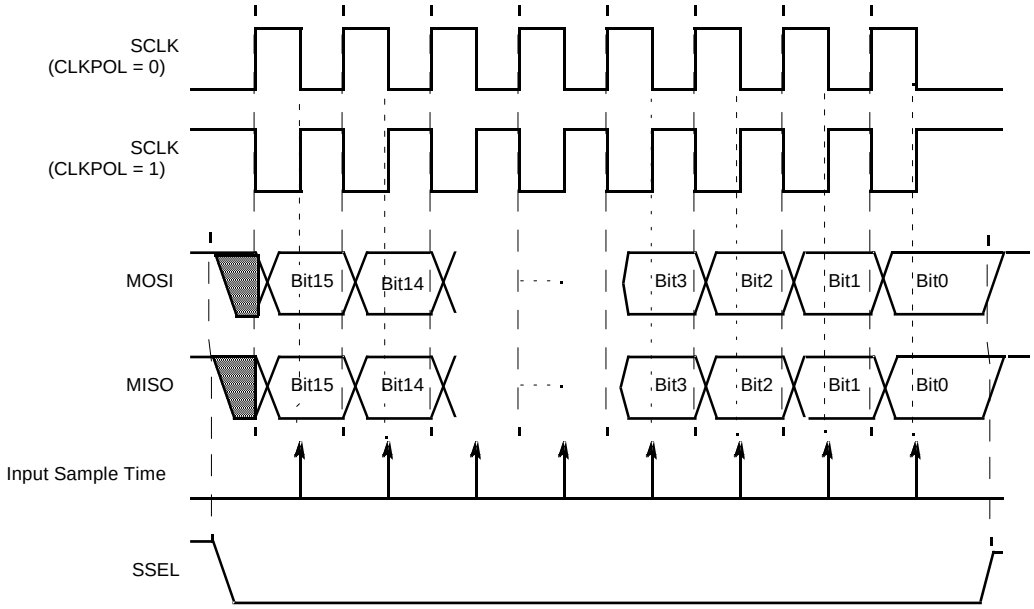


13.3.9 Transfer Format Phase 1

Figure 13-7 is the timing diagram for an SPI transfer in which the clock phase is set (*SPI0_CTRL2.clk_pha* = 1). The two SCLK waveforms show active low (*SPI0_CTRL2.clk_pol* = 0) and active high (*SPI0_CTRL2.clk_pol* = 1). The diagram may be interpreted as either a master or slave timing diagram since the SCLK, MISO and MOSI pins are directly connected between the master and the slave.

In the case of multi-character transfers with SS0 remaining asserted between characters, the bit 0 output data will remain stable until the clock edge which starts bit 15 of the next character or until the SS0 deasserts at the end of the transfer.

Figure 13-7. SPI Timing (SPI0_CTRL2.clk_pha = 1)



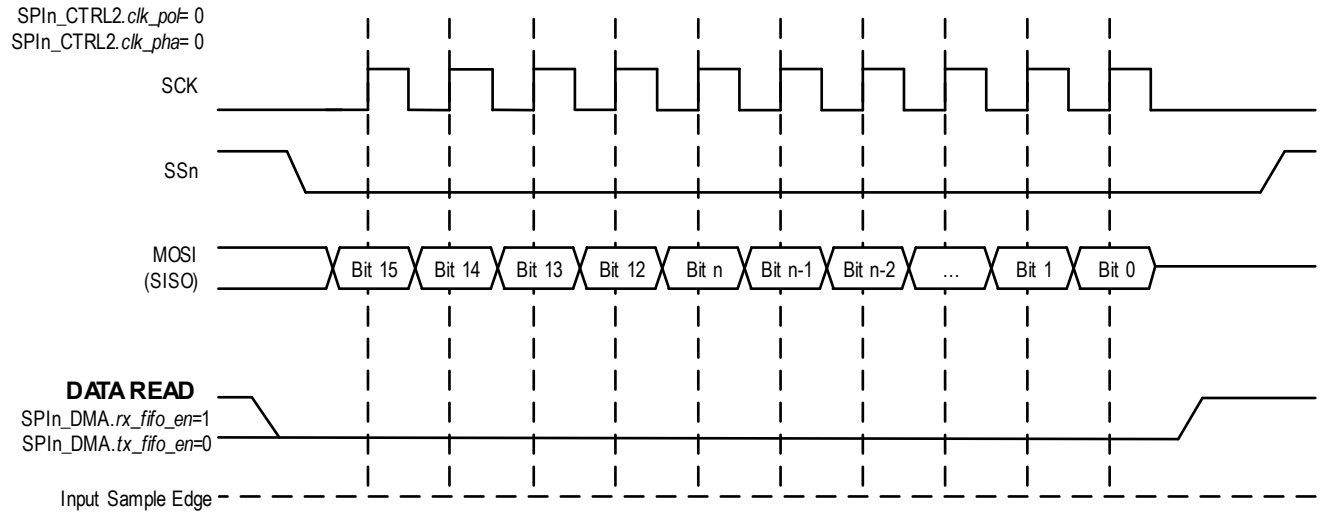
13.3.10 Three-Wire SPI Read and Write

In three-wire SPI, read and write transactions are controlled using the SPI FIFO enable bits. For a read transaction, enable the Receive FIFO and disable the Transmit FIFO.

13.3.10.1 Read Transaction

Figure 13-8 shows a three-wire SPI read transaction. The direction is set to a read by enabling the receive FIFO (*SPI0_DMA.rx_fifo_en* = 1) and disabling the transmit FIFO (*SPI0_DMA.tx_fifo_en* = 0). The SPI0_MOSI(SISO) pin is automatically set as an input by hardware based on the FIFO enable bits.

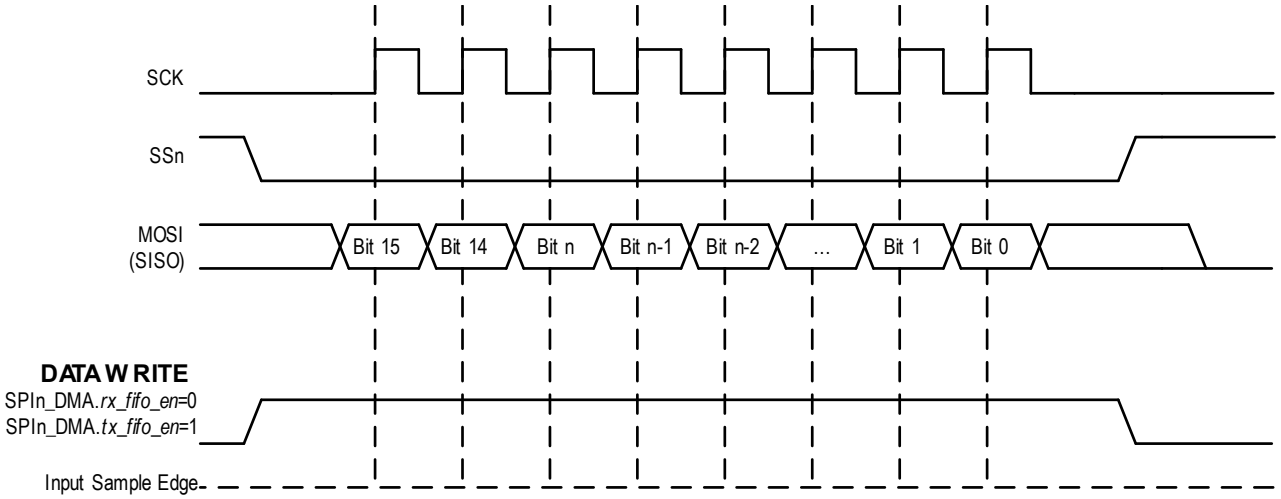
Figure 13-8: Three-Wire SPI Read



13.3.10.2 Write Transaction

Figure 13-9 shows a three-wire SPI write transaction. The direction is set to write by disabling the receive FIFO (*SPI0_DMA.rx_fifo_en* = 0) and enabling the transmit FIFO (*SPI0_DMA.tx_fifo_en* = 1). The SPI0_MOSI(SISO) pin is automatically set as to an output by hardware based on the FIFO enable bits. Data should be loaded to the transmit FIFO for the write.

Figure 13-9: Three-Wire SPI Write



13.3.11 Additional Configuration

Interrupt events are configured using the *SPI0_INT_EN* register.

Wakeup events are configured using the *SPI0_WAKE_EN* register.

The DMA is configured using *SPI0_DMA*.

Set *SPI0_CTRL0*.*start* = 1 to begin a Master Mode transmission.

Do not modify the SPI timing registers while a SPI transaction is in progress. Modifying any SPI timing register while a SPI transfer is in progress will result in an invalid SPI communication transaction.

To prevent a stall condition when in Master Mode, ensure that the transmit FIFO does not empty until the entire transmission is complete.

13.3.12 SPI FIFOs

The Transmit FIFO hardware is 32 bytes deep. The write data width can be 8-, 16- or 32-bits wide. A 16-bit write queues a 16-bit word to the FIFO hardware. A 32-bit write queues two 16-bit words to the FIFO hardware with the least significant word dequeued first. Bytes must be written to two consecutive byte addresses, with the odd byte as the most significant byte, and the even byte as the least significant byte. The FIFO logic waits for both the odd and even bytes to be written to this register space before dequeuing the 16-bit result to the FIFO.

The Receive FIFO hardware is 32 bytes deep. Read data width can be 8-, 16- or 32-bits. A byte read from this register dequeues one byte from the FIFO. A 16-bit read from this register dequeues two bytes from the FIFO, least significant byte first. A 32-bit read from this register dequeues four bytes from the FIFO, least significant byte first.

13.3.13 SPI Interrupts and Wakeups

The SPI supports multiple interrupt sources. Interrupt source events can come from the FIFOs, the SS and SR signals, and SPI status. Status flags for each interrupt are set regardless of the state of the interrupt enable bit for that event. Each interrupt flag field is set once when the condition is satisfied and remains set until cleared by the application. Write 1 to clear a specific interrupt flag field.

The following FIFO interrupts are supported:

- Transmit FIFO Empty
- Transmit FIFO Level below threshold, where the level is set by firmware.
- Receive FIFO Full
- Receive FIFO Level above threshold, where the level is set by firmware.
- Transmit FIFO Underrun (Slave mode only, Master mode stalls the clock)
- Transmit FIFO Overrun
- Receive FIFO Underrun
- Receive FIFO Overrun (Slave Mode only, Master Mode will stall the clock)

Note: On the MAX32660 SPI0 use of the Transmit FIFO level interrupt is recommended to avoid the Transmit FIFO empty and the TX FIFO underrun conditions from occurring in both Master or Slave mode operation.

The SPI supports interrupts for the internal state of the SPI as well as external signals. The following transmission interrupts are supported:

- SS Asserted or Deasserted
- Transmission Complete
- Slave Mode Transaction Aborted
- Multi-Master Fault

SPI0 has four Wakeup (WAKE) sources that can wake the core from *SLEEP* mode when the WAKE event occurs. The following WAKE events are supported:

- Wake on RX FIFO Full
- Wake on TX FIFO Empty
- Wake on RX FIFO Level crossed
- Wake on TX FIFO Level crossed

13.4 SPI0 Registers

The SPI0 base peripheral address is 0x4004 6000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 13-5: SPI0 Master Register Addresses and Descriptions

Offset	Register Name	Access	Description
[0x0000]	*SPI0_DATA*	R/W	SPI FIFO Data Register
[0x0004]	*SPI0_CTRL0*	R/W	SPI Master Signals Control Register
[0x0008]	*SPI0_CTRL1*	R/W	SPI Transmit Packet Size Register
[0x000C]	*SPI0_CTRL2*	R/W	SPI Static Configuration Register
[0x0010]	*SPI0_SS_TIME*	R/W	SPI Slave Select Timing Register
[0x0014]	*SPI0_CLK_CFG*	R/W	SPI Master Clock Configuration Register
[0x001C]	*SPI0_DMA*	R/W	SPI DMA Control Register
[0x0020]	*SPI0_INT_FL*	R/W1O	SPI Interrupt Status Flags Register
[0x0024]	*SPI0_INT_EN*	R/W	SPI Interrupt Enable Register
[0x0028]	*SPI0_WAKE_FL*	R/W1O	SPI Wakeup Status Flags Register
[0x002C]	*SPI0_WAKE_EN*	R/W	SPI Wakeup Enable Register
[0x0030]	*SPI0_STAT*	RO	SPI Active Status Register

13.4.1 SPI0 Register Details

Table 13-6: SPI FIFO Data Registers

SPIn FIFO Data Register			SPI0_DATA	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	-	R/W	0	**SPI FIFO Data Register** Reading from this register dequeues data from the receive FIFO. Writes to this register queues data to the transmit FIFO. Reads and writes with this register are in 1-byte, 2-byte, or 4-byte widths only.

Table 13-7: SPI Master Signals Control Registers

SPI Master Signals Control Register			SPI0_CTRL0	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:20	-	R/W	0	**Reserved for Future Use** Do not modify this field.

SPI Master Signals Control Register		SPI0_CTRL0		[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
19:16	ss_sel	R/W	0	**Slave Select Enable** Set this field to 1 to enable the slave select pin for SPI Master mode. When this field is set to 1, the SPI0_SS0 pin is used for the slave select output when the next SPI transaction is started (*SPI0_CTRL0.start* = 1). 0: SPI0_SS0 is disabled 1: SPI0_SS0 is enabled for output in SPI Master mode operation. 2 15: Reserved for Future Use.
15:9	-	R/W	0	**Reserved for Future Use** Do not modify this field.
8	ss_ctrl	R/W	0	**Master Mode Slave Select Control** In Master Mode operation, this bit controls the state of the slave select line at the end of a transmission. 0: Slave Select is deasserted at the end of a transmission 1: Slave Select stays asserted at the end of a transmission
7:6	-	R/W	0	**Reserved for Future Use** Do not modify this field.
5	start	R/WAC	0	**Master Mode Start Data Transmission** This bit is cleared by hardware. Writing a 0 is ignored. 0: Hardware automatically sets this field to 0 when the transaction has been initiated. 1: Master initiates a data transmission. Ensure that all pending transactions are complete before writing a 1. *Note: At least 1 byte must be loaded in the TX FIFO prior to setting this bit to 1.* *Note: This field is only used when the SPI is configured for Master Mode (SPI0_CTRL0.mm_en = 1).*
4	ss_io		0	**Master Mode Slave Select Output** This field must be set to 0 for the MAX32660. 0: Slave Select is an output 1: Reserved for Future Use *Note: This field is only used when the SPI is configured for Master Mode (SPI0_CTRL0.mm_en = 1).*
3:2	-	R/W	0	**Reserved for Future Use** Do not modify this field.
1	mm_en	R/W	0	**SPI Master Mode Enable** This field selects between slave mode and master mode operation for the SPI port. Write this field to 0 to operate as an SPI slave. Setting this field to 1 sets the port as an SPI master. 0: SPI port is in Slave Mode. 1: SPI is in Master Mode
0	spi_en	R/W	0	**SPI Enable/Disable** This field enables the SPI port instance. Setting this field disables the SPI port, but does not change the contents of the receive or transmit FIFOs or other SPI registers. 0: SPI port is disabled 1: SPI port is enabled

Table 13-8: SPI Transmit Packet Size Register

SPI Transmit Packet Size Register			SPI0_CTRL1	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	rx_num_char	R/W	0	**Number of Receive Characters** Number of characters to receive in RX FIFO. *Note: If the SPI port is set to operate in 4-wire mode, this field is ignored and the tx_num_chars field is used for both the number of characters to receive or transmit.*
15:0	tx_num_char	R/W	0	**Number of Transmit Characters** Number of characters to transmit from TX FIFO. *Note: In 4-wire mode, this also applies to the RX FIFO.*

Table 13-9: SPI Static Configuration Registers

SPI Static Configuration Register			SPI0_CTRL2	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:17	-	R/W	0	**Reserved for Future Use** Do not modify this field.
16	ss_pol	R/W	0	**Slave Select Polarity** Controls the polarity of the SPI0 SS signal 0: SS is active low 1: SS is active high
15	three_wire	R/W	0	**Three-Wire Mode Enable** 0: Four-wire mode enabled. 1: Three-wire mode enabled (Single IO Mode only).
14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:12	data_width	R/W	0	**SPI Data Width** Set this field to the required number of SDIO pins required. For Three-wire mode this field must be set to 0. Four-wire SPI mode supports Single or Dual IO Mode. 0: 1-data pin (Single IO Mode - MOSI) 1: 2-data pins (Dual IO Mode - MOSI/MISO) 2: Reserved 3: Reserved
11:8	num_bits	R/W	0x0	**Number of Bits per Character** 1-bit and 9-bit character lengths are not supported in Slave Mode
7:2	-	R/W	0	**Reserved for Future Use** Do not modify this field.
1	clk_pol	R/W	0	**Clock Polarity** Selects the SPI clock polarity. 0: Normal clock. Use when in SPI Mode 0 and Mode 1 1: Inverted clock. Use when in SPI Mode 2 and Mode 3
0	clk_pha	R/W	0	**Clock Phase** 0: Data sampled on clock rising edge. Use when in SPI Mode 0 and Mode 2 1: Data sampled on clock falling edge. Use when in SPI Mode 1 and Mode 3

Table 13-10: SPI Slave Select Timing Register

SPI Slave Select Timing			SPI0_SS_TIME	[0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
31:24	-	R/W	0	**Reserved for Future Use** Do not modify this field.
23:16	ssinact	R/W	0	**SS Inactive Clock Delay** This is the time SS is inactive, and the bus is inactive between character transmission. It is the number of system clock cycles from the time a character is transmitted, and SS is inactive to the time SS is active and a new character is transmitted. 0: 256 1: 1 2: 2 3:3 … … 254: 254 255: 255
15:8	ssact2	R/W	0	**Slave Select Active After Last SCLK** Number of system clock cycles that SS is active from the last SCLK edge to when SS is inactive. 0: 256 1: 1 2: 2 3:3 … … 254: 254 255: 255
7:0	ssact1	R/W	0	**Slave Select Active to First SCLK** Number of system clock cycles between the time SS is asserted until the first SCLK edge. 0: 256 1: 1 2: 2 3:3 … … 254: 254 255: 255

Table 13-11: SPI Master Clock Configuration Registers

SPI Master Clock Configuration Register			SPI0_CLK_CFG	[0x0014]
Bits	**Name**	**Access**	**Reset**	**Description**
31:20	-	R/W	0	**Reserved for Future Use** Do not modify this field.

SPI Master Clock Configuration Register			SPI0_CLK_CFG	[0x0014]
Bits	**Name**	**Access**	**Reset**	**Description**
19:16	scale	R/W	0	**System Clock to SPI Clock Scale Factor** Scales the Peripheral Clock (PCLK) by 2^{scale} to generate the SPI module clock. $$f_{SPI\_CLK} = \frac{f_{PCLK}}{2^{scale}}$$ 0x0 - 0x8: Scales the system clock by the set value to generate the internal SPI clock 0x9 - 0xF: Invalid *Note: The microcontroller System Clock is scaled by scale to generate the internal SPI clock. The external SPI clock, SCLK, is generated by setting the low cycle time, low, and the high cycle time, hi.* *Note: If scale=0, hi=0, and lo=0, character sizes of 2 and 10 bits are not supported.*
15:8	hi	R/W	0x00	**SCLK Hi Clock Cycles Control** 0x0: Hi duty cycle control disabled. Only valid if scale = 0. 0x1 – 0xF: Number of internal SPI clocks that SCLK is high. *Note: If scale=0, hi=0, and lo=0, character sizes of 2 and 10 bits are not supported.*
7:0	lo	R/W	0x00	**SCLK Low Clock Cycles Control** 0x0: Low duty cycle control disabled. Only valid if *SPI0_CLK_CFG.scale* = 0. 0x1 – 0xF: Number of internal SPI clocks that SCLK is low *Note: If SPI0_CLK_CFG.scale=0, SPI0_CLK_CFG.hi=0, and SPI0_CLK_CFG.low=0, character sizes of 2 and 10 bits are not supported.*

Table 13-12: SPI DMA Control Registers

SPI DMA Control Register			SPI0_DMA	[0x001C]
Bits	**Name**	**Access**	**Reset**	**Description**
31	rx_dma_en	R/W	0	**RX DMA Enable** 0: RX DMA is disabled. Any pending DMA requests are cleared 1: RX DMA is enabled
30	-	R/W	0	**Reserved for Future Use** Do not modify this field.
29:24	rx_fifo_cnt	R	0	**Number of Bytes in the RX FIFO** Read returns the number of bytes currently in the RX FIFO
23	rx_fifo_clear	W	-	**Clear the RX FIFO** 1: Clear the RX FIFO and any pending RX FIFO flags in *SPI0_INT_FL*. This should be done when the RX FIFO is inactive. Writing a 0 has no effect.
22	rx_fifo_en	R/W	0	**RX FIFO Enabled** 0: RX FIFO disabled 1: RX FIFO enabled
21	-	R/W	0	**Reserved for Future Use** Do not modify this field.

SPI DMA Control Register			SPI0_DMA	[0x001C]
Bits	**Name**	**Access**	**Reset**	**Description**
20:16	rx_fifo_level	R/W	0	**RX FIFO Threshold Level** When the RX FIFO contains more bytes than the value set in this field, a DMA request is triggered, and the *SPI0_INT_FL.rx_level* interrupt flag is set. Valid levels for this field are from 0x00 to 0x1E. 0x00: 1 byte in the RX FIFO generates a *SPI0_INT_FL.rx_level* interrupt flag. 0x01: 2 bytes in the RX FIFO generates an interrupt. … n: n+1 bytes in the RX FIFO sets the *SPI0_INT_FL.rx_level* interrupt flag. … 0x1E: Maximum allowed value for this field. 0x1F bytes in the RX FIFO set the *SPI0_INT_FL.rx_level* interrupt flag. 0x1F is not a valid value.
15	tx_dma_en	R/W	0	**TX DMA Enable** 0: TX DMA is disabled. Any pending DMA requests are cleared 1: TX DMA is enabled
14	-	R/W	0	**Reserved for Future Use** Do not modify this field.
13:8	tx_fifo_cnt	R0	0	**Number of Bytes in the TX FIFO** Read returns the number of bytes currently in the TX FIFO
7	tx_fifo_clear	W1O	-	**Clear the TX FIFO** Set this field to flush the TX FIFO. Write 1 only. Write 0 is ignored. 0: TX FIFO flush not active. 1: Clear the TX FIFO and any pending TX FIFO flags in *SPI0_INT_FL*. This should be done when the TX FIFO is inactive. *Note: Writing 0 has no effect.*
6	tx_fifo_en	R/W	0	**TX FIFO Enabled** Enable the TX FIFO by setting this field to 1. 0: TX FIFO disabled 1: TX FIFO enabled
5	-	R/W	0	**Reserved for Future Use** Do not modify this field.
4:0	tx_fifo_level	R/W	0x10	**TX FIFO Threshold Level** When the TX FIFO has fewer than the value set in this field, a DMA request is triggered, and the *SPI0_INT_FL.tx_level* interrupt flag is set.

Table 13-13: SPI Interrupt Flag Registers

SPI Interrupt Flag Register			SPI0_INT_FL	[0x0020]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W1C	0	**Reserved for Future Use** Do not modify this field.
15	rx_und	R/W1C	0	**RX FIFO Underrun Flag** Set when a read is attempted from an empty RX FIFO.
14	rx_ovr	R/W1C	0	**RX FIFO Overrun Flag** Set if SPI is in Slave Mode, and a write to a full RX FIFO is attempted. If the SPI is in Master Mode, this bit is not set as the SPI stalls the clock until data is read from the RX FIFO.

SPI Interrupt Flag Register			SPI0_INT_FL	[0x0020]
Bits	**Name**	**Access**	**Reset**	**Description**
13	tx_und	R/W1C	0	**TX FIFO Underrun Flag** Set if SPI is in Slave Mode, and a read from empty TX FIFO is attempted. If SPI0 is in Master Mode, this bit is not set as the SPI stalls the clock until data is written to the empty TX FIFO. *Note: This condition should be avoided by using the SPI0_INT_FL.tx_level interrupt flag to ensure that the underrun condition does not occur.*
12	tx_ovr	R/W1C	0	**TX FIFO Overrun Flag** Set when a write is attempted to a full TX FIFO. 0: Condition has not occurred. 1: Condition occurred. Write 1 to clear.
11	m_done	R/W1C	0	**Master Data Transmission Complete Flag** Set if SPI is in Master Mode and all data transmission is complete.
10	-	R/W	0	**Reserved for Future Use** Do not modify this field.
9	abort	R/W1C	0	**Slave Mode Transaction Abort Detected Flag** Set if the SPI is in Slave Mode, and SS is deasserted before a complete character is received.
8	-	R/W1C	0	**Reserved for Future Use** Do not modify this field.
7:6	-	R/W	0	**Reserved for Future Use** Do not modify this field.
5	ssd	R/W1C	0	**Slave Select Deasserted Flag** This flag is set when the slave select pin is deasserted and the SPI is operating in slave mode.
4	ssa	R/W1C	0	**Slave Select Asserted Flag** This flag is set when the slave select pin is asserted and the SPI is operating in slave mode. 1: Slave Select pin is asserted from a deasserted state.
3	rx_full	R/W1C	0	**RX FIFO Full Flag** This flag is set when the RX FIFO is full. Write 1 to clear. Clearing this flag and not reading data from the RX FIFO will result in this flag being set by hardware automatically when another byte is received on the SPI port. 1: RX FIFO is full.
2	rx_level	R/W1C	0	**RX FIFO Threshold Level Crossed Flag** Set when the RX FIFO exceeds the level set in the *SPI0_DMA.rx_fifo_level*.
1	tx_empty	R/W1C	1	**TX FIFO Empty Flag** This field is set when the Transmit FIFO is empty. Write 1 to clear. This flag must be cleared by the application directly. Writing data to the TX FIFO does not clear this flag automatically. *Note: On the MAX32660 SPI0 port, the SPI0_DMA.tx_fifo_level should be used to avoid an empty TX FIFO condition.*
0	tx_level	R/W1C	0	**TX FIFO Threshold Level Crossed Flag** Set when the TX FIFO is less than the value in *SPI0_DMA.tx_fifo_level.* Write 1 to clear. This flag must be cleared by the application explicitly. Correcting the condition by writing data to the TX FIFO does not clear this flag.

Table 13-14: SPI Interrupt Enable Registers

SPI Interrupt Enable Register			SPI0_INT_EN	[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.
15	rx_und	R/W	0	**RX FIFO Underrun Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
14	rx_ovr	R/W	0	**RX FIFO Overrun Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
13	tx_und	R/W	0	**TX FIFO Underrun Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
12	tx_ovr	R/W	0	**TX FIFO Overrun Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
11	m_done	R/W	0	**Master Data Transmission Done Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
10	-	R/W	0	**Reserved for Future Use** Do not modify this field.
9	abort	R/W	0	**Slave Mode Abort Detected Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
8	fault	R/W	0	**Multi-Master Fault Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
7:6	-	R/W	0	**Reserved for Future Use** Do not modify this field.
5	ssd	R/W	0	**Slave Select Deasserted Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
4	ssa	R/W	0	**Slave Select Asserted Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
3	rx_full	R/W	0	**RX FIFO Full Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
2	rx_level	R/W		**RX FIFO Threshold Level Crossed Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
1	tx_empty	R/W	0	**TX FIFO Empty Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled
0	tx_level	R/W	0	**TX FIFO Threshold Level Crossed Interrupt Enable** 0: Interrupt is disabled 1: Interrupt is enabled

Table 13-15: SPI Wakeup Status Flags Registers

SPI Wakeup Status Flags			SPI0_WAKE_FL	[0x0028]
Bits	**Name**	**Access**	**Reset**	**Description**
31:4	-	R/W	0	**Reserved for Future Use** Do not modify this field.
3	rx_full	R/W1C	0	**Wake on RX FIFO Full Flag** 0: Wake condition has not occurred. 1: Wake condition occurred.
2	rx_level	R/W1C	0	**Wake on RX FIFO Threshold Level Crossed Flag** 0: Wake condition has not occurred. 1: Wake condition occurred.
1	tx_empty	R/W1C	0	**Wake on TX FIFO Empty Flag** 0: Wake condition has not occurred. 1: Wake condition occurred.
0	tx_level	R/W1C	0	**Wake on TX FIFO Threshold Level Crossed Flag** 0: Wake condition has not occurred. 1: Wake condition occurred.

Table 13-16: SPI Wakeup Enable Registers

SPI Wakeup Enable			SPI0_WAKE_EN	[0x002C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:4	-	R/W	0	**Reserved for Future Use** Do not modify this field.
3	rx_full	R/W	0	**Wake on RX FIFO Full Enable** 0: Wake event is disabled 1: Wake event is enabled.
2	rx_level	R/W	0	**Wake on RX FIFO Threshold Level Crossed Enable** 0: Wake event is disabled 1: Wake event is enabled.
1	tx_empty	R/W	0	**Wake on TX FIFO Empty Enable** 0: Wake event is disabled 1: Wake event is enabled.
0	tx_level	R/W	0	**Wake on TX FIFO Threshold Level Crossed Enable** 0: Wake event is disabled 1: Wake event is enabled.

Table 13-17: SPI Status Registers

SPI Status Register			SPI0_STAT	[0x0030]
Bits	**Name**	**Access**	**Reset**	**Description**
31:1	-	R/W	0	**Reserved for Future Use** Do not modify this field.
0	busy	R	0	**SPI Active Status** 0: SPI is not active. In Master Mode, cleared when the last character is sent. In Slave Mode, cleared when SS is deasserted. 1: SPI is active. In Master Mode, set when transmit starts. In Slave Mode, set when SS is asserted.

14 SPIMSS (SPI1/I²S)

14.1 Overview

The SPIMSS peripheral provides either an SPI interface or an I²S interface. When set to SPI mode, the peripheral supports SPI with a four-wire full-duplex serial bus operating as either a master or slave. When set to I²S mode, the peripheral support full-duplex bi-directional I/O, a signal for Left/Right clock and a bit rate clock signal for master or slave communications.

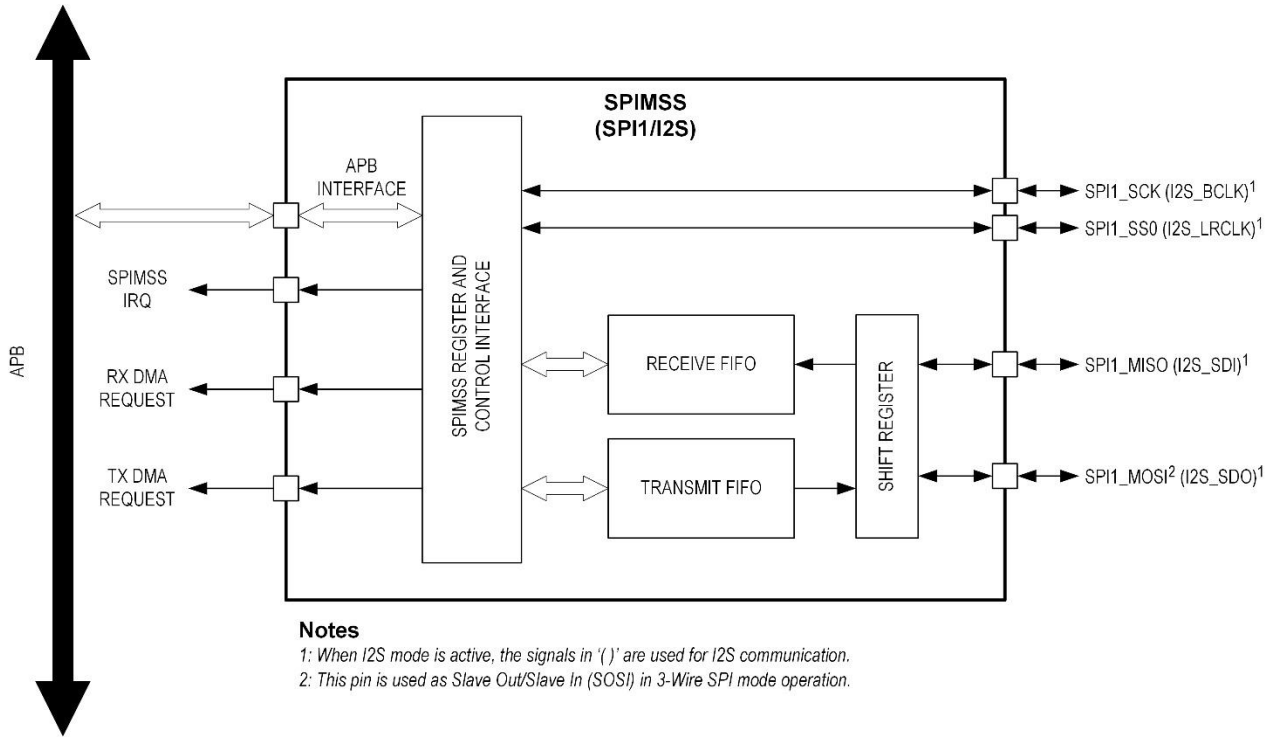
The SPIMSS (SPI1/I²S) peripheral supports Inter-IC Sound (I²S) protocol for 16-bit mono or stereo audio transfer to or from an external I²S audio codec.

14.1.1 Features

- Dedicated Bit Rate Generator
- Eight entry, 16-bit wide transmit and receive FIFOs
- DMA support for both transmit and receive operations
- SPI Features
  - Full-duplex, synchronous communication of 1 to 16-bit characters
  - Four-wire interface
  - Data transfers rates up to one-fourth the peripheral clock frequency ($f_{PCLK}/4$)
  - Master and Slave mode SPI operation
  - 1 Slave Select Pin
- I²S Features
  - 16-bit audio transfer
  - I²S master or slave mode
  - Bidirectional, full-duplex communication supported
  - Left-Right Clock and Bit Clock generation in I²S master mode

Figure 14-1 shows a high-level block diagram of the SPIMSS peripheral including the external interface signals, control unit, receive and transmit FIFOs, and single shift register common to the transmit and receive data path for SPI or I²S operation. The SPIMSS peripheral is configurable to operate as a SPI port or an I²S port. The same physical pins are used for SPI or I²S operation. The reset default sets the SPIMSS to SPI operation. Enabling I²S, setting *SPIMSS_I2S_CTRL*.*i2s_en* to 1, results in hardware using the pins as defined for I²S.

Figure 14-1. SPIMSS Block Diagram



An SPI system has a single master and one or more slaves for any given transaction. The SPIMSS supports single master mode networks only when operating in SPI master mode.

14.1.2 Four-Wire SPI Signals

SPI devices operate as either a master or slave device. In four-wire SPI, four signals are required for communication as shown in *Table 14-3, below*.

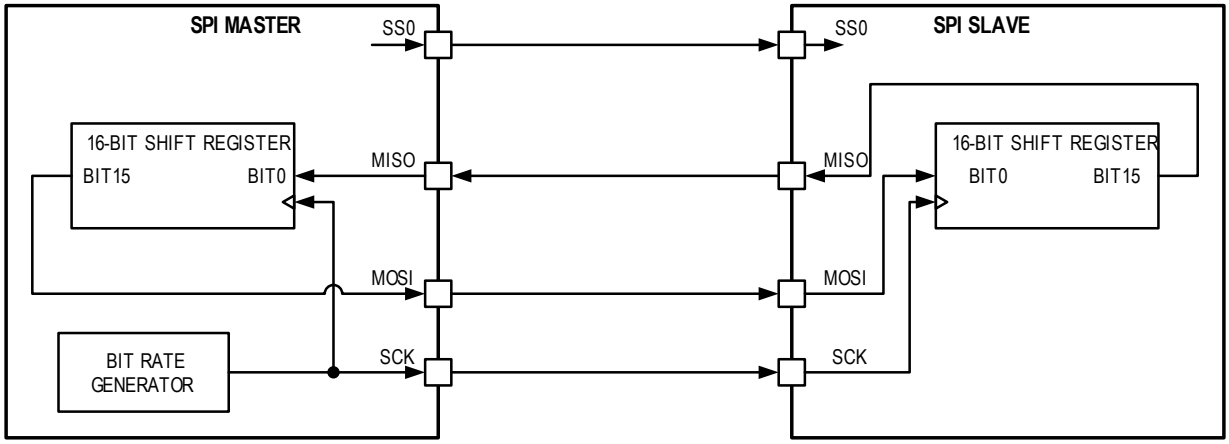
Table 14-1: Four-Wire SPI Signals

Signal	MAX32660 Alternate Function Name	Description	Direction
SCK	SPI1_SCK	Serial Clock	The master generates the Serial Clock signal, which is an output from the master and an input to the slave.
MOSI	SPI1_MOSI	Master Output Slave Input	In master mode, this signal is used as an output for sending data to the slave. In slave mode this is the input data from the master.
MISO	SPI1_MISO	Master Input Slave Output	In master mode, this signal is used as an input for receiving data from the slave. In slave mode, this signal is an output for transmitting data to the master.
SS	SPI1_SS0	Slave Select	In master mode, this signal is an output used to select a slave device prior to communication. In slave mode, this signal is an input used to indicate the master is going to start communication.

Note: The MAX32660 supports a single slave select pin in the SPIMSS peripheral for SPI1. This pin's alternate function name is SPI1_SS0.

A typical four-wire SPI network is shown in *Figure 14-2*, *below*. In a typical SPI network, the master device selects the slave device using the slave select output pin. The master starts the communication by selecting the slave device by asserting the slave select output pin. The master then starts the SPI clock via the SCK output pin. When a slave device's slave select pin is deasserted, the device is required to put the SPI pins in tri-state mode.

Figure 14-2: 4-Wire SPI Connection Diagram



14.1.3 I²S Signals

I²S devices operate as either a master or slave device. Three signals are required for communication as shown in *Table 14-3*, *below*. For a slave I²S device, the SDI signal is used for audio input and the SDO pin is not required. For a master I²S device, the SDO pin is used for audio output and the SDI pin is not required.

Table 14-2: I²S Signals

Signal	MAX32660 Alternate Function Name	Name	Description
WS	I2S_LRCLK	Word Select	The word select clock indicates which channel is currently being sent. I²S supports two channels, channel 1 and channel 2, left and right respectively. The word select signal is often referred to as left-right clock (LRCLK). Channel 1, the left audio, is transmitted when the LRCLK is low. Channel 2, the right audio, is transmitted when the LRCLK is high.
SCK	I2S_BCLK	Continuous Serial Clock	SCK is referred to as the Bit Clock, BCLK. The BCLK pulses once for each discrete bit of data on the data lines.
SDI	I2S_SDI	Serial Data In	Audio data input line. Data is signed, encoded as two's complement with the MSB first.
SDO	I2S_SDO	Serial Data Out	Audio data output line.

14.2 SPIMSS Configuration

Before configuring the SPIMSS peripheral, first disable any SPIMSS activity by setting the *SPIMSS_CTRL.enable* field to 0.

14.2.1 SPIMSS Pin Configuration for SPI and I²S Operation

Pin selection and configuration is required to use the SPIMSS for either SPI or I²S operation. The following information applies to both I²S or SPI operation. *Table 14-3* shows the pin selection options for the SPIMSS (SPI1/I²S) for each package available. The Alternate Function Name column maps the typical SPI signal name to the Alternate Function name on the MAX32660 family of parts. The required pins for SPIMSS are all mapped to Alternate Function 2 on GPIO Port 0 and are available on the same pins in both the 16-WLP and the 20-TQFN as shown in *Table 14-3*.

For SPI operation, an external pull-up resistor should be used to prevent floating input signals when operating the SPI signals in open drain mode (refer to the *SPIMSS_CTRL.wor* bit) or high impedance mode.

When the SPI or I²S are not enabled, *SPIMSS_CTRL.enable* = 0, all the SPI1 pins are put into high-impedance mode.

Table 14-3: SPIMSS Pins for SPI1 and I²S

SPI Signal	I²S Signal	Alternate Function Name	Alternate Function Number	GPIO	
				16-WLP	20-TQFN
SCK	WS (BCLK)	SPI1_SCK (I2S_BCLK)	AF2	P0.2	P0.2
MOSI (SISO)	SDO	SPI1_MOSI (I2S_SDO)	AF2	P0.1	P0.1
MISO	SDI	SPI1_MISO (I2S_SDI)	AF2	P0.0	P0.0
SS	BCLK (LRCLK)	SPI1_SS0 (I2S_LRCLK)	AF2	P0.3	P0.3

Four-wire SPI uses SCK, MISO, MOSI, and the SS pin. I²S requires BCLK, LRCLK and either SDO or SDI. The following steps outline setting up the GPIO pins for the SPI1/I²S alternate function usage.

1. Set the GPIO pin for alternate function operation.
   a. SPI1_SCK (I2S_BCLK): Set GPIO0_EN[2] to 0.
   b. SPI1_MOSI (I2S_SDO): Set GPIO0_EN[1] to 0.
   c. SPI1_MISO (I2S_SDI): Set GPIO0_EN[0] to 0.
   d. SPI1_SS0 (I2S_LRCLK): Set GPIO0_EN[3] to 0.
2. Select AF2 using the GPIO0_AF_SEL register.
   a. SPI1_SCK (I2S_BCLK):: Set GPIO0_AF_SEL[2] to 1
   b. SPI1_MOSI (I2S_SDO): Set GPIO0_AF_SEL[1] to 1.
   c. SPI1_MISO (I2S_SDI):: Set GPIO0_AF_SEL[0] to 1.
   d. SPI1_SS0 (I2S_LRCLK): Set GPIO0_AF_SEL[3] to 1.

14.3 SPI Operation

SPI is a full-duplex, synchronous, character-oriented serial communication channel that supports a four-wire interface consisting of a serial clock, SCK, a slave select line, SS, and two data lines, MOSI and MISO. The SPIMSS consists of a transmit and receive shift register, a transmit FIFO, a receive FIFO, a bit rate generator and a control unit as shown in *Figure 14-1*.

During an SPI transfer, data is sent and received simultaneously by both the master and slave device. When an SPI transfer occurs, a multi-bit character, selectable from 1-bit to 16 bits, is shifted out on the data output line and a multi-bit character is simultaneously shifted in on the data input pin. A 16-bit shift register in the master and another 16-bit shift register in the slave are connected as a circular buffer with the most significant bit (bit15) sent first. The SPIMSS contains two 8 entry, 16-bit FIFOs to support transmit and receive data. Any data in the transmit FIFO is moved into the shift register at the start of every new SPI transfer if there is data in the transmit FIFO. At the end of each SPI character transmitted, data is moved from the input shift register into the receive FIFO.

14.3.1 Serial Clock

The Serial Clock (SCK) synchronizes data movement in and out of the device through the SPI1_MOSI and SPI1_MISO pins. The master drives the serial clock out its SCK pin to the slave's SCK pin. When SPI1 is set to master mode, the SPIMSS bit rate generator creates the serial clock and outputs it on the SPI1_SCK pin. When SPI1 is configured for slave operation the SPI1_SCK pin is an input from the external master. Slave devices ignore the SCK signal unless their slave select pin is asserted.

When SPI1 is configured for slave operation, the maximum SCK input frequency supported is $f_{SCK} = {}^{f_{PCLK}}/_{8}$. For example, if $f_{PCLK} = 48MHz$, the maximum SPI clock frequency supported in slave mode for SPI1 is 6MHz.

In both master and slave devices, data is shifted on one edge of the SCK and is sampled on the opposite edge where data is stable. Data availability and sampling time is controlled using the SPI phase control field, *SPIMSS_CTRL*.*phase*. The SCK clock polarity field, *SPIMSS_CTRL*.*clkpol*, controls if the SCK signal is active high or active low.

SPIMSS supports four combinations of SCK phase and polarity. Clock Polarity (*SPIMSS_CTRL*.*clkpol*) selects an active low/high clock and has no effect on the transfer format. Clock Phase (*SPIMSS_CTRL*.*phase*) selects one of two different transfer formats.

For proper data transmission, the clock phase and polarity must be identical for the SPI master and slave. The master always places data on the MOSI line a half-cycle before the SCLK edge for the slave to latch the data.

Table 14-4. Clock Phase and Polarity Operation

SPIMSS_CTRL ***phase***	*SPIMSS_CTRL* ***clkpol***	**SCLK Transmit Edge**	**SCLK Receive Edge**	**SCLK Idle State**
0	0	Falling	Rising	Low
0	1	Rising	Falling	High
1	0	Rising	Falling	Low
1	1	Falling	Rising	High

14.3.2 SPI Slave Select Configuration

The Slave Select (SS) signal is used to select a specific slave device during SPI transfers or to distinguish left and right channel audio data in I²S mode. In an SPI system with multiple slaves, the master must provide separate slave select signals to each slave. The slave select is set to active prior to any communication with a slave and must remain in the active state for the full duration of each character transferred at a minimum. The slave select signal may stay low during the transfer of multiple characters or may deassert between each character. The *SPIMSS_MODE*.*ssv* bit is not used in SPI slave mode.

14.3.2.1 SPI Master Mode

If using SPI1 as a SPI master, set the slave select pin, SPI1_SS0, as an output (*SPIMSS_CTRL*.*ss_io* = 1). The polarity of the slave select signal is selected via the *SPIMSS_MODE*.*ssv* bit and defaults to active low (*SPIMSS_MODE*.*ssv*).

14.3.2.2 SPI Slave Mode

When using SPI slave mode, *SPIMSS_CTRL.mode = 0*, configure the slave select pin, SPI1_SS0, as an input by clearing *SPIMSS_CTRL.ss_io*. The *SPIMSS_MODE.ssv* bit is not used in SPI slave mode.

14.3.3 SPI Character Size

SPI transmits and receives characters simultaneously. The transmit and receive character size is the same. Set the number of bits per character using the *SPIMSS_MODE.numbits* field.

14.3.4 SPI Data Movement

Data movement in SPI mode is controlled using one of the following methods:

- Synchronous operation
  - Application polling of the *SPIMSS_INT_FL.txst* bit to transfer single words.
  - Application polling of the *SPIMSS_DMA.tx_fifo_level* or *SPIMSS_DMA.rx_fifo_level* fields enables transfers of up to eight characters at a time.
- Asynchronous Operation
  - The *SPIMSS_CTRL.irqe* bit can be set to enable data and error interrupts. The *SPIMSS_CTRL*.str bit may be used if desired to force a "startup" data interrupt. A data interrupt will be generated on completion of each character transfer.
- DMA Operation
  - Control of data transferred is enabled via the *SPIMSS_DMA.rx_dma_en* and/or *SPIMSS_DMA.tx_dma_en* bits. The *SPIMSS_DMA.tx_fifo_level* and *SPIMSS_DMA.rx_fifo_level* control when DMA requests are asserted. When DMA is enabled, the SPIMSS data interrupt is disabled by hardware, error interrupts remain available. DMA operation is beneficial for block transfers as the CPU only needs to service one DMA interrupt per block of data versus one interrupt for each character transferred if data interrupt-based transfer is used.

The SPIMSS Data Register, *SPIMSS_DATA*, is used for transferring data for both transmit and receive operations.

For incoming data, the receive data is shifted into an internal shift register. Once a full character is received, the character is automatically moved into the receive FIFO. Read received data using the *SPIMSS_DATA* register.

For outgoing data, data written to the *SPIMSS_DATA* register is automatically moved to the transmit FIFO. The data is then transmitted via the shift register. When the shift register is empty, any data in the transmit FIFO is moved to the shift register.

Note: When the SPIMSS is not actively transmitting or receiving data (SPIMSS_CTRL.enable = 0), data written to the SPIMSS Data Register is stored in the transmit FIFO, if the transmit FIFO is not full. Any data in the transmit FIFO when the SPIMSS enable bit is set to 1 is transmitted immediately by the hardware. Flush the transmit FIFO at any time by setting the SPIMSS_DMA.tx_fifo_clr bit to 1.

With the SPIMSS configured as a SPI master, writing data to the *SPIMSS_DATA* register initiates the data transmission. With the SPIMSS configured as a SPI slave, writing data *SPIMSS_DATA* register loads the shift register in preparation for the next data transfer with the external master. In either SPI master or slave mode, when the transmit FIFO is full, writes to the *SPIMSS_DATA* register are ignored and result in a transmit overrun error interrupt (*SPIMSS_INT_FL.tovr* = 1).

Data is shifted out starting with the most significant bit first (bit 15). The last bit received will reside in the least significant bit, (bit 0). In SPI mode, when the character length is less than 16 bits, set by the *SPIMSS_MODE.numbits* field, the transmit character must be left justified in the *SPIMSS_DATA* register. A received character of less than 16 bits is always right justified, with the last bit received loaded into the least significant bit (bit 0). For example, if SPI1 is configured for 4-bit characters, write transmit data to *SPIMSS_DATA[15:12]* and received characters are read from *SPIMSS_DATA[3:0]*.

The application overhead required to left justify the transmit data can be eliminated by setting the *tx_lj* bit in the *SPIMSS_MODE* register. When *SPIMSS_MODE.tx_lj* = 1, transmit data is always written by software or DMA to *SPIMSS_DATA* in right justified form and hardware performs the left justify according to *SPIMSS_MODE.numbits* when the shift register is loaded. For the 4-bit character example, when *SPIMSS_MODE.tx_lj* = 1, transmit data is written to *SPIMSS_DATA* [3:0] and hardware shifts these to bits *SPIMSS_DATA*[15:12] when the shift register is loaded. The *SPIMSS_MODE.tx_lj* bit has no effect on receive data which is always right justified.

14.3.5 SPI Master Configuration

Perform the following steps to configure SPI1 for master mode:

- Set *SPIMSS_CTRL.enable* to 0 to disable the SPIMSS peripheral.
- Enable SPI master mode by setting *SPIMSS_CTRL.mode* to 1.
- Set the SPIMSS SPI1 pins to wired OR/open drain, if required, by setting *SPIMSS_CTRL.wor* to 1.
- Set the slave select pin to an output by setting *SPIMSS_CTRL.ss_io* = 1
- Configure the clock phase and polarity, if required, using the *SPIMSS_CTRL.phase* and *SPIMSS_CTRL.clkpol* bits.
- Set the number of bits per character using the *SPIMSS_MODE.numbits* field.
- Configure the Bit Rate Generator for the desired bit transmission rate. Refer to section *SPIMSS Bit Rate Generator*.
- Load data in the Transmit FIFO to send to the slave or configure the Transmit and Receive DMA.
- Enable SPI1 by setting the SPIMSS enable bit, *SPIMSS_CTRL.enable,* to 1.
- Optionally force a SPIMSS data interrupt by setting *SPIMSS_CTRL.str* to 1.

The *SPIMSS_CTRL.phase, SPIMSS_CTRL.clkpol* and the *SPIMSS_MODE.numbits* fields must be consistent with the slave SPI device for successful operation. The *SPIMSS_MODE.ssv* bit controls the asserted level for the slave select output pin, SPI1_SS0. Set *SPIMSS_MODE.ssv* to 1 to set the slave select to active high. The typical SPI device uses an active low slave select signal and the *SPIMSS_MODE.ssv* defaults to 0 for an active low slave select output.

14.3.6 SPI Slave Configuration

Perform the following steps to configure SPI1 for slave mode:

3. Disable the SPI1 by clearing the enable bit (*SPIMSS_CTRL.enable* = 0)
4. Set SPI1 to slave mode, by setting the *SPIMSS_CTRL.mode* bit to 0.
5. Set *SPIMSS_CTRL.wor* to 0 for wired OR/open drain for all the SPIMSS (SPI1/I^{2}S) I/O pins.
6. Set *SPIMSS_CTRL.ss_io* to 0 to set the slave select pin (SPI1_SS0) as an input.
7. Configure the clock phase and polarity, if required, using the *SPIMSS_CTRL.phase* and *SPIMSS_CTRL.clkpol* bits.
8. Set the number of bits per character using the *SPIMSS_MODE.numbits* field.
9. Load data in the Transmit FIFO to send to the master for the next communication or configure the Transmit and Receive DMA.
10. Enable SPI1 by setting the enable bit, *SPIMSS_CTRL.enable,* to 1 to begin transmitting and receiving data.

The *SPIMSS_CTRL.phase* and *SPIMSS_CTRL.clkpol* bits and the *SPIMSS_MODE.numbits* field must be set to be consistent with the SPI master. The *SPIMSS_CTRL.str* bit may be used, if desired, to force a start interrupt when the master SPI device asserts the slave select pin, SPI1_SS0. The baud rate generator timer interrupt flag, *SPIMSS_CTRL.birq,* and the slave select value bit, *SPIMSS_CTRL.ssv,* are not used in SPI slave mode. The SPI bit rate generator is not used in slave mode, so the Mode Register, *SPIMSS_MODE*, does not need initialization.

If the slave has data to send to the master, the data should be written before the transaction starts (first edge of SCK after the slave select pin, SPI1_SS0, is asserted). If the *SPIMSS_DATA* register is not written prior to the slave transaction, the SPI1_MISO pin will output whatever value was written last into the *SPIMSS_DATA* Register. Avoid this issue by loading the transmit FIFO or configure transmit DMA prior to enabling SPI1.

14.4 I²S Mode

Configure the SPIMSS peripheral for I²S mode as follows:

- *SPIMSS_CTRL.enable* = 0, disable the SPIMSS for SPI and I²S.
- *SPIMSS_I2S_CTRL.i2s_en* = 1, enable I²S mode
- *SPIMSS_CTRL.phase* = 0, set the phase
- *SPIMSS_CTRL.clkpol* = 0
- *SPIMSS_MODE.numbits* = 0 (to select 16-bit characters)
- *SPIMSS_CTRL.enable* = 1

The *SPIMSS_CTRL.mmen* and *SPIMSS_CTRL.ss_io* bits are set in accordance with either master or slave mode of operation. The *SPIMSS_MODE.ssv* bit is ignored by hardware in I²S mode. In I²S, the master hardware sources the word select pin, I2S_BCLK, and the left-right clock pin, I2S_LRCLK. In I²S mode, the word select signal toggles between consecutive audio words. I2S_LRCLK is low for the left channel data and is high for the right channel data.

The receive and/or transmit DMA channels must be enabled when operating in I²S mode. Typically, audio data will only flow in one direction as defined by the *SPIMSS_DMA.rx_dma_en* or *SPIMSS_DMA.tx_dma_en* bits, however audio data may be transferred in both directions simultaneously if desired. Data in the transmit buffer should be initialized with the first 16-bit character containing a left channel audio sample, then alternating right and left channel 16-bit audio samples. When audio data is being received, the first sample written into the receive buffer will be a left channel audio sample.

14.4.1 Mute

The *SPIMSS_I2S_CTRL.i2s_mute* bit in the I²S Control Register can be set by software asynchronously to the DMA transfers to silence the transmit output. At the beginning of the next left channel audio sample after *SPIMSS_I2S_CTRL.i2s_mute* is asserted, DMA and FIFO accesses will continue, however, the data read from the transmit FIFO will be discarded and replaced with zeroes. When *SPIMSS_I2S_CTRL.i2s_mute* is deasserted, the transmit output will resume at the beginning of the next left channel audio sample.

14.4.2 Pause

The *SPIMSS_I2S_CTRL.i2s_pause* bit can be set by software asynchronously to the DMA transfers to halt DMA and FIFO accesses. At the beginning of the next left channel audio sample after *SPIMSS_I2S_CTRL.i2s_pause* is set to 1, both transmit and receive DMA and FIFO accesses will halt and the transmit data will be forced to zero. At the beginning of the next left channel audio sample after *SPIMSS_I2S_CTRL.i2s_pause* is set to 0, the DMA accesses will resume from the position at which the pause occurred. Pause takes precedence over mute.

14.4.3 Mono

The *SPIMSS_I2S_CTRL.i2s_mono* bit in the I²S Control Register is set to select single channel audio data vs. stereo format. In mono mode each transmit data word read from the transmit FIFO is duplicated for both left and right channel output words. The receive channel will read the data from the left channel (I2S_LRCLK pin low) and ignore data in the right channel (I2S_LRCLK pin high). This allows DMA buffers for mono mode to be one-half the size of DMA buffers for stereo mode.

14.4.4 Left Justify

The *SPIMSS_I2S_CTRL.i2s_lj* bit selects the phase of the I2S_LRCLK signal versus the data. When *SPIMSS_I2S_CTRL.i2s_lj* = 0 (normal I²S mode), the audio data lags the transition of the I2S_LRCLK signal by one I2S_BCLK period as shown in *Figure 14-3*, *below*. When *SPIMSS_I2S_CTRL.i2s_lj* = 1, the audio data is “left justified” so that the most significant bit of the channel audio data is available when the I2S_LRCLK signal transitions.

Figure 14-3: I²S Audio Data in Standard I²S Operation

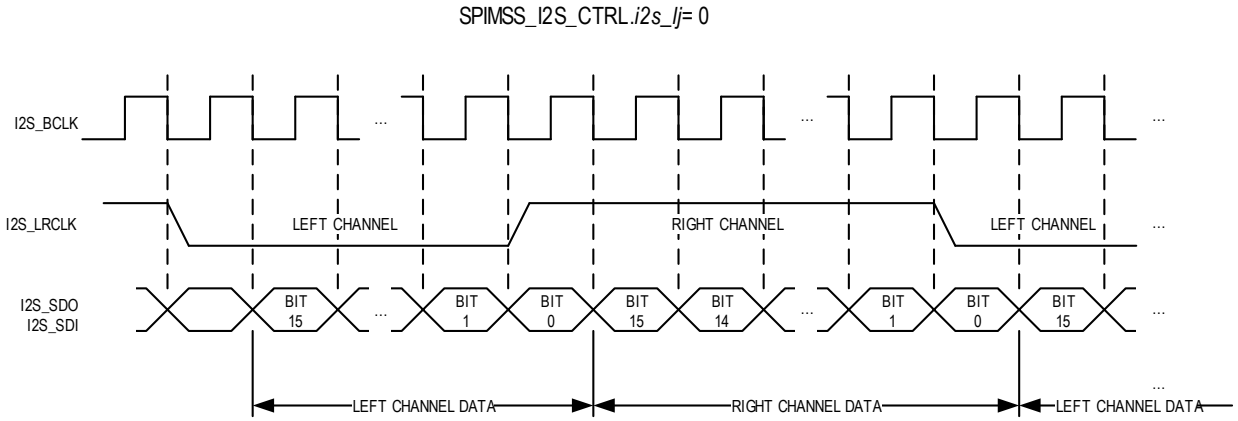
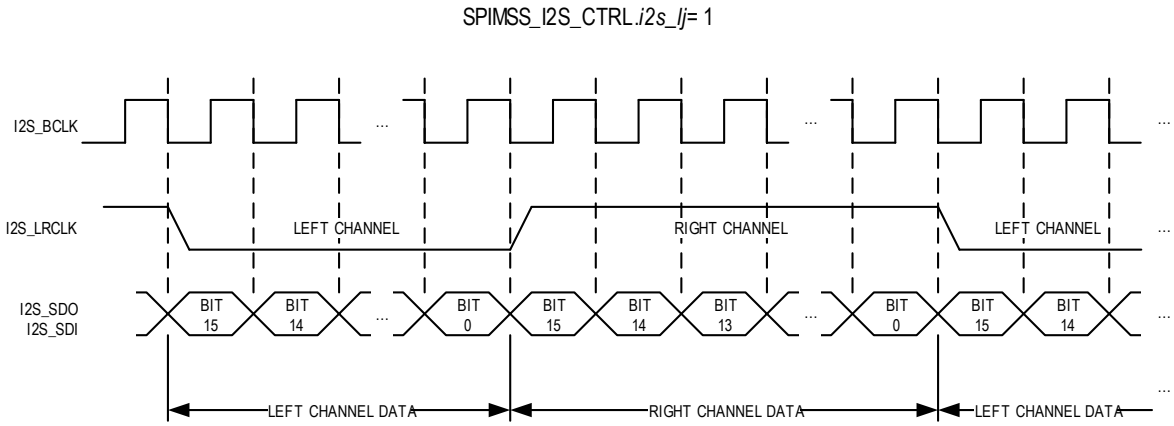


Figure 14-4: I²S Mode (i2s_en=1, i2s_lj=1)



14.5 SPI and I²S Error Detection

The SPIMSS peripheral includes error detection logic to recognize when communication errors occur for either SPI or I²S communications. If the *SPIMSS_CTRL.irqe* bit is set to 1, error conditions generate a SPIMSS IRQ. The SPIMSS Interrupt Flag Register, *SPIMSS_INT_FL*, includes the error flags described below.

14.5.1 Transmit Overrun

A transmit overrun error indicates a write to the transmit FIFO was attempted when the internal transmit FIFO was full in either SPI or I²S modes. An overrun condition sets the *SPIMSS_INT_FL.tovr* bit to 1. Writing a 1 to *SPIMSS_INT_FL.tovr* clears this error flag.

Note: A transmit FIFO overrun in I²S mode may result in mixing left and right channel data. Software should reinitialize the DMA channel, the Transmit FIFO and restart the I²S transfer.

14.5.2 SPI Slave Mode Abort

A SPI slave mode abort error indicates that the SPI1_SS0 pin deasserted before all bits in a character were transferred. The next time SPI1_SS0 asserts, the SPI1_MISO pin outputs *SPIMSS_DATA[15]*, regardless of where the previous transaction aborted. A slave mode abort sets the *SPIMSS_INT_FL.abt* error flag to 1. Writing a 1 to *SPIMSS_INT_FL.abt* clears this error flag.

Note: This error interrupt does not occur in SPI Master or I²S mode.

14.5.3 Receive Overrun

This error occurs if data is received and the Receive FIFO is full and applies to SPI master, SPI slave and I²S modes. The *SPIMSS_INT_FL.rovr* error flag is set to 1 if this error occurs. Writing a 1 to *SPIMSS_INT_FL.rovr* bit clears this error flag.

Note: A Receive Overrun error in I²S mode results in left/right channel data corruption. If a Receive Overrun error occurs, the application must flush the Receive FIFO, reinitialize the receive DMA channel, restart the I²S transfer.

14.6 SPI1 and I²S Interrupts

The SPIMSS provides interrupt support for a variety of conditions for SPI and I²S modes. Setting *SPIMSS_CTRL.irqe* to 1 enables the SPIMSS IRQ. The SPIMSS generates an IRQ when one of the following interrupt conditions occur.

14.6.1 Data Interrupt

A data interrupt occurs when a the Transmit FIFO is empty and the transmit shift register shifts out the last bit of the active transmit word The Data Interrupt flag applies to both transmit and receive data for SPI and I²S because the transmit and receive paths are interlocked.

The data interrupt does not occur if either Transmit DMA or Receive DMA is enabled because the DMA interface handles the data interrupt condition directly in hardware.

A data interrupt is indicated as follows:

- The *SPIMSS_INT_FL.irq* flag is set to 1.
- No error condition flags are set in the *SPIMSS_INT_FL* register.
  - *SPIMSS_INT_FL[6:2]* = 0

Clear the data interrupt condition by writing 1 to *SPIMSS_INT_FL.irq*.

14.6.2 Forced Interrupt

To start the data transfer process, an SPI interrupt may be forced by software by writing a 1 to the *SPIMSS_CTRL.str* bit in the SPI Control Register.

14.6.3 Error Condition Interrupt

If any of the SPI error conditions occurs as described in section *SPI and I2S Error Detection*, *above*, the corresponding error bit in the *SPIMSS_INT_FL* register and the *SPIMSS_INT_FL.irq* bit are set to 1. The error flags and the irq bit should be cleared simultaneously by writing both to 1 simultaneously. For example, a transmit overrun error is indicated by the *SPIMSS_INT_FL.tovr* bit set to 1 and the *SPIMSS_INT_FL.irq* bit set to 1. Clear this condition by writing 0x06 to *SPIMSS_INT_FL*.

14.6.4 Bit Rate Generator Time-out Interrupt

If the SPI is disabled, an SPI interrupt can be generated by a Bit Rate Generator time-out. This timer function must be enabled by setting the *SPIMSS_CTRL.birq* bit to 1.

14.7 SPIMSS Bit Rate Generator

14.7.1 SPI and I²S Slave Mode

The Bit Rate Generator is not used in SPI or I²S slave mode. When configured as a SPI or I²S slave, the maximum SCK or BCLK frequency is $f_{PCLK}/_{8}$.

14.7.2 SPI and I²S Master Mode Bit Rate Generator

For SPI or I²S master operation, the Bit Rate Generator (BRG) creates a lower frequency clock for SCK or BCLK for data transmission synchronization between the master and the external slave. The input to the Bit Rate Generator is the System Peripheral clock, PCLK. The Bit Rate Generator register is a 16-bit reload value, *SPIMSS_BRG.div*, for the Bit Rate Generator. The reload value, *SPIMSS_BRG.div*, must be greater than or equal to 0x02 for SPI and I²S master operation with a maximum bit rate frequency of $f_{PCLK}/_{4}$. *Equation 14-1* shows the equation for calculating the SPI master and I²S master bit rate frequency.

Equation 14-1: SPI Master and I²S Master Bit Rate Calculation

$$Bit\ Rate\ \left(bits/_{sec}\right) = \left(\frac{f_{PCLK}}{2 \times SPIMSS\_BRG.div}\right)$$

$$Note: For\ SPIMSS\_BRG.div = 0, use\ 2^{16}$$

14.7.3 Timer Mode

When SPI1 or I²S is not active, the Bit Rate Generator can function as a continuous mode 16-bit timer with interrupt on time-out. To configure the Bit Rate Generator as a timer with interrupt on time-out, complete the following procedure:

1. Set *SPIMSS_CTRL.enable* = 0 to stop any SPI or I²S activity.
2. Disable the transmit and receive FIFOs
   a. *SPIMSS_DMA.tx_fifo_en* = 0
   b. *SPIMSS_DMA.rx_fifo_*en = 0
3. Disable DMA mode
   a. *SPIMSS_DMA.tx_dma_en* = 0
   b. *SPIMSS_DMA.rx_dma_en* = 0
4. Load the desired 16-bit divisor into the SPIMSS Bit Rate Generator Register, *SPIMSS_BRG.div.*
5. Set *SPIMSS_CTRL.birq* = 1 to enable the Bit Rate Generator
6. Enable the SPIMSS peripheral by setting *SPIMSS_CTRL.enable* = 1

14.8 SPIMSS (SPI1/I²S) Registers

The SPIMSS base peripheral address is 0x4001 9000. Refer to *Table 3-1: APB Peripheral Base Address Map* for the addresses of all APB mapped peripherals.

Table 14-5: SPIMSS Register Offsets, Access and Descriptions

Offset	Register Name	Access	Description
[0x0000]	*SPIMSS_DATA*	R/W	SPIMSS Data Register
[0x0004]	*SPIMSS_CTRL*	R/W	SPIMSS Control Register
[0x0008]	*SPIMSS_INT_FL*	R/W	SPIMSS Interrupt Flag Register
[0x000C]	*SPIMSS_MODE*	R/W	SPIMSS Mode Register
[0x0014]	*SPIMSS_BRG*	R/W	SPIMSS Bit Rate Register
[0x0018]	*SPIMSS_DMA*	R/W	SPIMSS DMA Register
[0x001C]	*SPIMSS_I2S_CTRL*	R/W	SPIMSS I²S Control Register

14.8.1 SPIMSS Register Details

Table 14-6. SPIMSS Data Register

SPIMSS Data Register			SPIMSS_DATA	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.
15:0	data	R/W	0	**SPIMSS Data** Refer to the *SPI Data Movement* section for details.

Table 14-7: SPIMSS Control Register

SPIMSS Control Register			SPIMSS_CTRL	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	R/W	0	**Reserved for Future Use** Do not modify this field.
7	irqe	R/W	0	**Interrupt Request Enable** Set to enable interrupts for the SPIMSS peripheral. 0: SPIMSS (SPI1/I²S) interrupts are disabled. 1: SPI MSS (SPI1/I²S) interrupts are enabled. Interrupt requests are sent to the Interrupt Controller *Note that if transmit or receive DMA is enabled, the transmit data complete interrupt is disabled, but other interrupt sources are available.*
6	str	R/W	0	**Start SPI Interrupt** Setting this bit starts a SPIMSS interrupt request. Setting this bit also sets *SPIMSS_INT_FL.irq* to 1. Setting this bit forces the SPIMSS to send an interrupt request to the Interrupt Controller if *SPIMSS_CTRL.irqe* = 1. This bit is cleared by writing a 0 to this bit or by writing a 1 to *SPIMSS_INT_FL.irq*.
5	birq	R/W	0	**Bit Rate Generator Timer Interrupt Request** Enable or disable the Bit Rate Generator if the SPIMSS is enabled (*SPIMSS_CTRL.enable* = 1). 0: Clearing this bit disables the Bit Rate Generation timer function. 1: Setting this bit to 1 enables the Bit Rate Generation timer function and enables the time-out interrupt. *Note: If SPIMSS_CTRL.enable = 0, this bit has no effect.*

SPIMSS Control Register			SPIMSS_CTRL	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
4	phase	R/W	0	**Phase Select** Refer to the section *13.3.7 Clock Phase and Polarity Control* for details. 0: Data is valid prior to first serial clock edge. 1: Data transition occurs after the first serial clock edge.
3	clkpol	R/W	0	**Clock Polarity** Sets the idle state for the SCK clock pin after a character transaction. 0: SCK idles low (0) after character transmission or reception. 1: SCK idles high (1) after character transmission or reception.
2	wor	R/W	0	**Wired OR (Open Drain) Enable** Set to enable wired OR for the SPIMSS signal pins (SPI1_SCK, SPI1_SS0, SPI1_MOSI, SPI1_MISO). 0: Wired OR configuration disabled. 1: Wired OR configuration enabled.
1	mmen	R/W	0	**SPI Master Mode Enable** Set this field to enable master mode for SPI1. 0: SPI set to slave mode operation 1: SPI set to master mode operation
0	enable	R/W	0	**SPI1/I²S Enable** Set this field to enable operation of the SPIMSS as configured, either for SPI1 or I²S. If the Transmit FIFO contains data, the data is considered valid and is transmitted. If data is in the Receive FIFO, the data is considered valid and is used as received data. 0: Disable SPI1 and I²S. 1: Enable SPI1 or I²S, if configured. *Note: This bit should be set to 1 only after the SPIMSS is configured for operation as either SPI or I²S. Setting this bit to 0 does not reset or change any configuration of the SPI or I²S and does not affect any data in the Transmit or Receive FIFOs.* *Note: Clear the Transmit and Receive FIFOs, if desired, prior to setting this field to 1 by writing 1 to SPIMSS_DMA.tx_fifo_clr or SPIMSS_DMA.rx_fifo_clr respectively.*

Table 14-8: SPIMSS Interrupt Flag Register

SPIMSS Interrupt Flag Register			SPIMSS_INT_FL	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	R/W	0	**Reserved for Future Use** Do not modify this field.
7	irq	R/W1C	0	**SPIMSS Interrupt Request Flag** This bit is set by hardware when an SPIMSS interrupt request is pending. Write 1 to clear. 0: No SPIMSS interrupt request is pending 1: An SPIMSS interrupt request is pending *Note: This field cannot be cleared unless all interrupt flags in this register are cleared.*
6	tovr	R/W1C	0	**Transmit Overrun Flag** This bit is set by hardware when a transmit FIFO overrun has occurred. Write 1 to clear. 0: No SPI interrupt request is pending 1: An SPI interrupt request is pending

SPIMSS Interrupt Flag Register			SPIMSS_INT_FL	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
5	col	R/W1C	0	**Collision Flag** This bit is set by hardware when a multi-master collision (mode fault) occurs. Write 1 to clear. 0: No multi-master collision has occurred 1: A multi-master collision has occurred
4	abt	R/W1C	0	**Slave Mode Transaction Abort Flag** This bit is set by hardware when a slave mode transaction abort occurs. Write 1 to clear. 0: No slave mode transaction abort has occurred 1: A slave mode transaction abort has occurred
3	rovr	R/W1C	0	**Receive Overrun Flag** This bit is set by hardware when a receive FIFO overrun occurs. Write 1 to clear. 0: No FIFO overrun has occurred 1: A FIFO overrun has occurred.
2	tund	R/W1C	0	**Transmit Underrun Flag** This bit is set by hardware to indicate a transmit FIFO underrun has occurred. Write 1 to clear. 0: No FIFO underrun has occurred 1: A FIFO underrun has occurred
1	txst	RO	0	**Transmit Status** This field reads 1 if a SPIMSS data transmission is currently in progress. 0: No data transmission currently in progress. 1: Data transmission currently in progress
0	slas	R/W	0	**Slave Select** If the SPI is in slave mode, this bit indicates if the SPI is selected. If the SPI is in master mode, this bit has no meaning. 0: Slave SPI is selected 1: Slave SPI is not selected

Table 14-9: SPIMSS Mode Register

SPIMSS Mode Register			SPIMSS_MODE	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	R/W	0	**Reserved for Future Use** Do not modify this field.
7	tx_lj	R/W	0	**Transmit Data Alignment** Selects left or right alignment when data is loaded into the *SPIMSS_DATA.data* field for transmission if the character size is less than 16-bits. 0: Data is LSB aligned with the unused bits set to 0 up to the MSB (right aligned) 1: Data is MSB aligned with the unused bits set to 0 down to the LSB (left aligned)
6	-	R/W	0	**Reserved for Future Use** Do not modify this field.

SPIMSS Mode Register			SPIMSS_MODE	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
5:2	numbits	R/W	0	**Number of Data Bits per Character to Transfer** This field contains the number of bits to shift for each character transfer. Refer to the data movement chapter for information on valid bit positions when the character length is less than 16-bits. 0b0000: 16-bits 0b0001: 1-bits 0b0010: 2-bits … 0b1110: 14-bits 0b1111: 15-bits *Note: Setting this field to 0 (default) sets the number of bits per character to 16.*
1	ss_io	R/W	0	**Slave Select Input/Output Mode** Setting this field to 1 sets the slave select pin, SPI1_SS0, as an output. Clearing this field sets the slave select pin, SPI1_SS0, to an input. 0: The SPI1_SS0 pin is configured as an input. 1: The SPI1_SS0 pin is configured as an output *Note: This field is only used if the SPIMSS is in SPI Master mode (SPIMSS_CTRL.mode = 1).*
0	ssv	R/W	0	**Slave Select Value** This indicates the value of the SPI1_SSO (I2S_LRCLK) pin if the SPIMSS slave select pin is configured as an output (*SPIMSS_MODE.ss_io* = 1), writing this field drives the pin to the value written. If the slave select pin is set to an input (*SPIMSS_MODE.ss_io* = 0), reading this field returns the level of the slave select pin.

Table 14-10: SPIMSS Bit Rate Generator Register

SPIMSS Bit Rate Generator Register			SPIMSS_BRG	[0x0014]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved for Future Use** Do not modify this field.
15:0	div	R/W	0	**Bit Rate Reload Value** The SPI Bit Rate register is a 16-bit reload value for the SPI Bit Rate Generator. The reload value, div, must be greater than or equal to 2 for proper SPI operation. Refer to section *14.7.2 SPI and I2S Master Mode Bit Rate Generator*.

Table 14-11: SPIMSS DMA Register

SPIMSS DMA Register			SPIMSS_DMA	[0x0018]
Bits	**Name**	**Access**	**Reset**	**Description**
31	rx_dma_en	R/W	0	**Receive DMA Enable** Disabling clears any active request to the DMA controller. 0: Disable RX DMA requests 1: Enable RX DMA requests
30:28	-	R/W	0	**Reserved for Future Use** Do not modify this field.

SPIMSS DMA Register			SPIMSS_DMA	[0x0018]
Bits	**Name**	**Access**	**Reset**	**Description**
27:24	rx_fifo_cnt	R/W	0	**Receive FIFO Count** 0b0000: RX FIFO empty (0 entries) 0b0001: RX FIFO contains 1 entry 0b0010: RX FIFO contains 2 entries 0b0011: RX FIFO contains 3 entries … 0b1000: RX FIFO contains 15 entries
23:21	-	R/W	0	**Reserved for Future Use** Do not modify this field.
20	rx_fifo_clr	R/W	0	**Receive FIFO Clear** Write 1 to reset the Receive FIFO. Writing 0 has no effect. 0: Ignored 1: Reset Receive FIFO
19	-	R/W	0	**Reserved for Future Use** Do not modify this field.
18:16	rx_fifo_lvl	R/W	0	**Receive FIFO Level** Sets the RX FIFO DMA request threshold. This configures the number of filled RX FIFO entries before activating an RX DMA request. 000: Request Receive DMA when RX FIFO contains 1 entry 001: Request Receive DMA when RX FIFO contains 2 entries 010: Request Receive DMA when RX FIFO contains 3 entries … 111: Request Receive DMA when RX FIFO contains 8 entries
15	tx_dma_en	R/W	0	**Transmit DMA Enable** Disabling clears any active request to the DMA controller. 0: Disable TX DMA requests 1: Enable TX DMA requests
14:12	-	R/W	0	**Reserved for Future Use** Do not modify this field.
11:8	tx_fifo_cnt	R/W	0	**Transmit FIFO Count** 0b0000: TX FIFO empty (0 entries) 0b0001: TX FIFO contains 1 entry 0b0010: TX FIFO contains 2 entries 0b0011: TX FIFO contains 3 entries … 0b1000: TX FIFO contains 15 entries
7:5	-	R/W	0	**Reserved for Future Use** Do not modify this field.
4	tx_fifo_clr	R/W	0	**Transmit FIFO Clear** Write 1 to reset the Receive FIFO. Writing 0 has no effect. 0: Ignored 1: Reset Receive FIFO
3	-	R/W	0	**Reserved for Future Use** Do not modify this field.

SPIMSS DMA Register			SPIMSS_DMA	[0x0018]
Bits	**Name**	**Access**	**Reset**	**Description**
2:0	tx_fifo_lvl	R/W	0	**Transmit FIFO Level** Sets the TX FIFO DMA request threshold. This configures the number of empty TX FIFO entries before activating a Transmit DMA request. 0b000: Request Transmit DMA when TX FIFO has 1 free entry. 0b001: Request Transmit DMA when TX FIFO has 2 free entries. 0b010: Request Transmit DMA when TX FIFO has 3 free entries. … 0b111: Request Transmit DMA when TX FIFO has 8 free entries.

Table 14-12: SPIMSS I²S Control Register

SPIMSS I2S Control Register			SPIMSS_I2S_CTRL	[0x001C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:5	-	R/W	0	**Reserved for Future Use** Do not modify this field.
4	i2s_lj	R/W	0	**I²S Left Justify** 0: Normal I²S audio protocol - audio data lags left/right channel signal by one SCLK period. 1: Audio data is synchronized with SSEL (left/right channel signal).
3	i2s_mono	R/W	0	**I²S Monophonic Audio Mode** Set this field to enable monophonic audio mode. In this mode, each transmit data word is replicated on both left and right channels. Receive data is taken from left channel, right channel receive data is ignored. 0: Stereophonic audio. 1: Monophonic audio format
2	i2s_pause	R/W	0	**I²S Pause Transmit/Receive** 0: Normal transmission/reception. 1: Halt transmit and receive FIFO and DMA accesses, transmit 0.
1	i2s_mute	R/W	0	**I²S Mute Transmit** 0: Normal transmit. 1: Transmit data is replaced with 0
0	i2s_en	R/W	0	**I²S Mode Enable** Set to enable I²S mode. 0: I²S mode is disabled. 1: I²S mode enabled.

15 Trademarks

Arm and Cortex are registered trademarks of ARM Limited.

16 Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION
0	7/18	Initial Release.