MAX32620

User's Guide

UG6242; Rev 0; 01/16

maxim integrated™

Contents

1 Disclaimer and Revision History

Disclaimer

LIFE SUPPORT POLICY

MAXIM'S PRODUCTS ARE NOT DESIGNED, INTENDED OR AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT, LIFE SUSTAINING, DEVICES OR SYSTEMS OR APPLICATIONS, INCLUDING BUT NOT LIMITED TO, NUCLEAR, TRANSPORTATION OPERATING SYSTEMS, IN WHICH THE FAILURE OF SUCH GOODS COULD REASONABLY BE EXPECTED TO RESULT IN PERSONAL INJURY, LOSS OF LIFE OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF MAXIM INTEGRATED PRODUCTS, INC.

As used herein

Life support, life sustaining devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

Revision History

Version	Changes	Date
1.00	Initial release for rev A4.	01-11-2016

2 Overview

Introduction

The **MAX32620** User Guide is targeted to hardware, embedded firmware and application developers. This guide provides information on how to use and configure the **MAX32620** memory, peripherals and registers. For ordering information, complete feature sets, package information, and electrical specifications, refer to the **MAX32620/MAX32621** data sheet.

Related Documents

- ARM® SM Cortex®-M4 Technical Reference Manual available from www.arm.com
- **MAX32620/MAX32621** data sheet

Devices Covered by This Guide

This User Guide covers functionality common to both the **MAX32620** and **MAX32621** devices. For simplicity's sake, when the device being used is referred to in the text, it is referred to as **MAX32620**, even though the text applies equally well to the **MAX32621**.

The following functionality is specific to the **MAX32621** only and is not covered in this guide:

- MAA modular arithmetic accelerator
- PRNG pseudo-random number generator

For details on the above **MAX32621** specific topics, refer to the User Guide Supplement document for the **MAX32621**.

Key Device Features

The **MAX32620** is a low-power, mixed signal microcontroller based on the ARM Cortex-M4 32-bit core with a maximum operating frequency of 96MHz.

Application code on the **MAX32620** runs from an onboard 2MB program flash memory, with a 256KB SRAM available for general application use. An 8KB instruction cache improves execution throughput, and a transparent code scrambling scheme is used to protect customer intellectual property residing in the program flash memory. Additionally, a SPI Execute In Place (XIP) external memory interface allows application code and data (up to 16MB) to be accessed from an external SPI memory device.

The **MAX32620** includes a 10-bit sigma-delta ADC with a multiplexer front end for four external input channels (two of which are 5.5V tolerant) and six internal channels. Dedicated divided supply input channels allow direct monitoring of onboard power supplies such as V_{DD12}, V_{DD18}, V_{DDB}, and V_{RTC} by the ADC. Built in limit monitors allow converted input samples to be compared against user-configurable high and low limits, with an option to trigger an interrupt and wake the CPU

from a low power mode if attention is required.

The **MAX32620** includes a wide variety of communications and interface peripherals. It is designed to interface to an off-chip front end module (using a dedicated SPI master interface) to communicate over a Bluetooth® Low Energy wireless band. Other communications peripherals include a USB 2.0 slave interface, three master SPI interfaces, four UART interfaces with hardware flow control and multi-drop support, three master I²C interfaces , and a slave I²C interface.

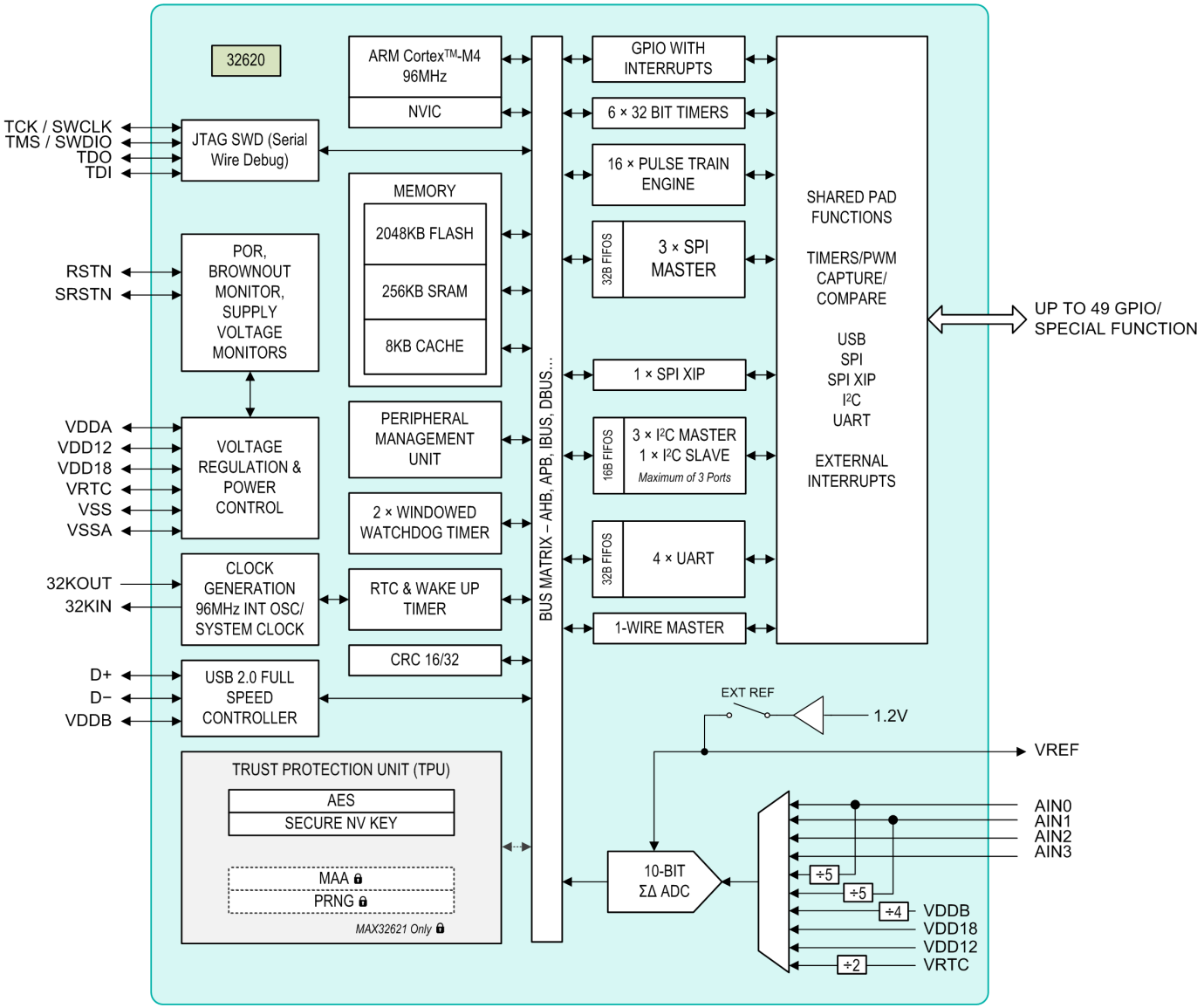

Figure 2.1: MAX32620 Block Diagram

2.1 Core and Architecture

ARM® Cortex®-M4 Core

The **MAX32620** is based on the ARM Cortex-M4 32-bit RISC CPU, which implements the ARMv7-M architectural profile. The implementation of the Cortex-M4 core used in the **MAX32620** is targeted for a maximum operating frequency of 96MHz and provides the following features.

- 32-bit data path with mixed 16-bit and 32-bit instructions (Thumb-2 instruction set)
- Single cycle multiply-accumulate (MAC) with 16/32-bit multiply operations and 32/64-bit accumulate operations
- DSP extensions include hardware division operation (2-12 cycles)
- Extended data processing instructions for SIMD (single instruction multiple data) operations, with dual 16-bit and quad 8-bit operations
- Single precision Floating Point Unit (FPU) extension for floating point arithmetic (IEEE^{SM} 754 compliant)
- Memory Protection Unit (MPU) for RTOS support
- Nested vectored interrupt controller (NVIC) with multiple interrupt priority levels and nested interrupt support
- 4GB total memory space, shared by code memory, data memory, and peripheral registers
- Low power, highly energy efficient core reduces power consumption
- Built-in debug functionality and tracing with JTAG port and Serial-Wire (SW) Debug interface (connects to internal Debug Access Port)
- Power saving sleep mode

2.1.1 Core Parameters

When the Cortex-M4 core is instantiated in a design, values must be selected for configurable parameters in the core. For the **MAX32620** design, key Cortex-M4 core parameters are shown below.

Parameter	Value	Description
NUM_IRQ	64	Number of Interrupts supported by the Cortex-M4
LVL_WIDTH	3	Specifies the number of bits of interrupt priority levels supported. At a width of three, there are eight levels supported. At a width of eight (the maximum allowed), 256 levels are supported.
MPU_PRESENT	1	The MPU (memory protection unit) is included on this device, with 8 protection regions defined.
BB_PRESENT	1	Bit-banding (memory mapped bit) operations are supported on this device.
DEBUG_LVL	3	Full debug with data matching. All debug functionality is present including data matching for watchpoint generation.
TRACE_LVL	0	Standard trace. ITM, TPIU, and DWT triggers and counters are present. ETM and HTM port are not present.
RESET_ALL_REGS	1	Registers are set to a known reset state.
JTAG_PRESENT	1	JTAG Debug Access Port is included in this design.
CLKGATE_PRESENT	1	Architectural gates are included to minimize dynamic power dissipation.
FPU_PRESENT	1	Floating Point Unit functionality is included in this design.

2.1.2 Generic Memory Map

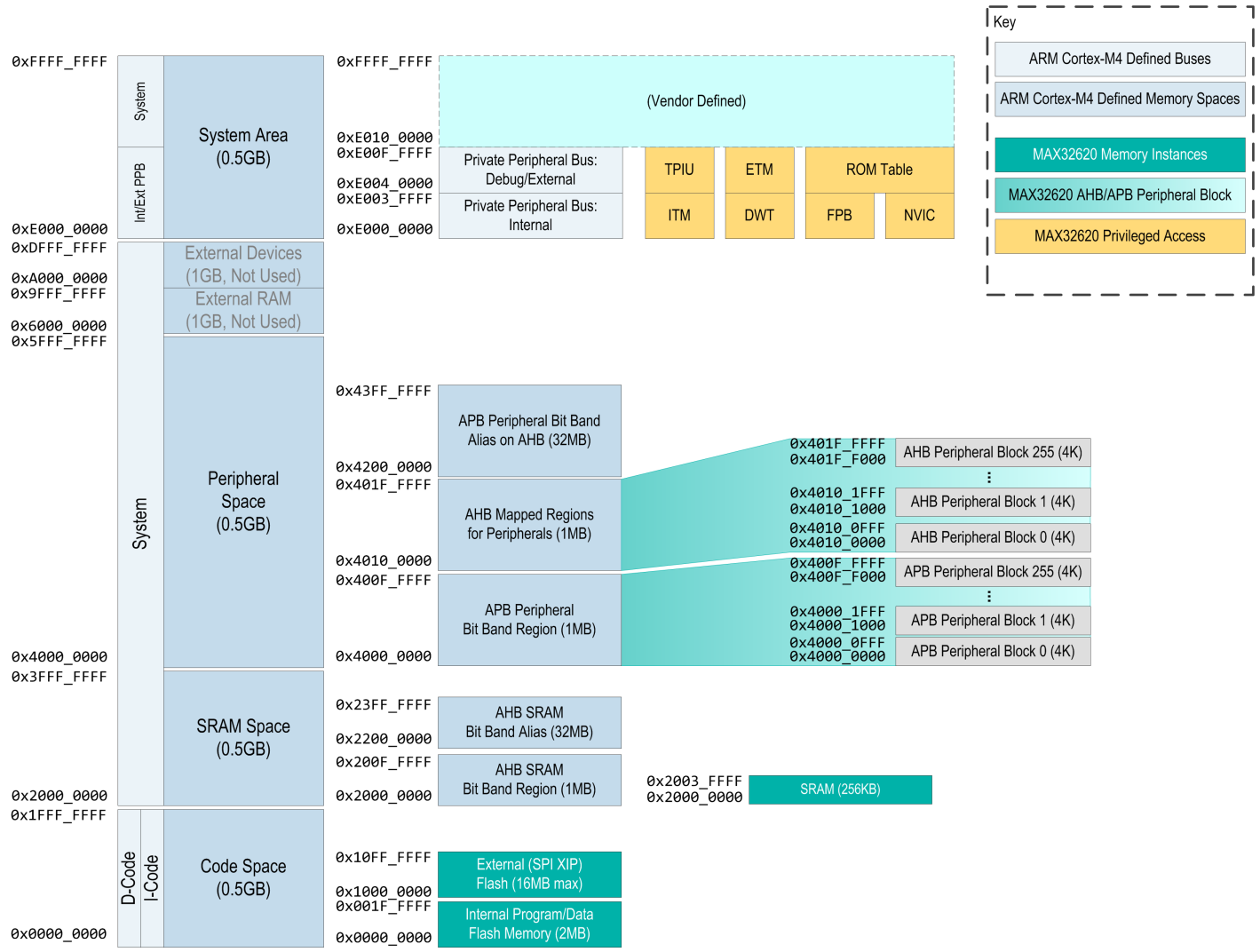

Figure 2.2: Memory Map

2.1.3 AHB Buses

The standard ARM Advanced High Performance Bus (AHB-Lite version) is used for several different system bus masters on the **MAX32620**. All buses are 32 bits in width.

- **I-Code**: Performs instruction fetches from internal code memory regions. On the **MAX32620**, accesses to internal program flash memory and external SPI XIP memory (for instruction decoding purposes) are cached to improve execution throughput.
- **D-Code**: Performs data fetches from program flash memory; this includes literal local constant fetches. These data fetches are not cached, unlike instruction code fetches.
- **System**: Performs data read/write and bit band operations on internal SRAM, and data read/write operations on peripherals (including bit band operations), and vendor defined expansion devices in the system area.

Note Bit band operations are translated internally by the ARM core into a read-modify-write sequence, and so only the core itself (when performing instruction execution) can read or write to locations using the bit banding function. The bit banding alias areas, although they are shown on the memory map, do not exist as separate logical mapped areas, and so they cannot be accessed by other AHB masters (such as the PMU) since they do not exist at this layer.

2.1.4 APB Buses

The External PPB (private peripheral bus) is a 32-bit bus based on the APB (Advanced Peripheral Bus) standard. It is intended for adding components to the private peripheral bus area which are not intended for general application use, since privileged operating mode is required to access this area. The **MAX32620** does not currently map any non-core components to this bus area.

The majority of the digital and analog peripherals on the **MAX32620** are controlled by registers that are memory mapped into the Peripheral region, from address 0x4000_0000 to 0x400F_FFFF (in the bit banding enabled region). These peripherals are connected to the CPU core using a lower-speed APB peripheral bus (connected to the System AHB-Lite bus through an AHB-to-APB bridge).

Peripherals which require higher speed access for large data transfers have control/buffer regions mapped to the AHB bus from address 0x4010_0000 to 0x401F_FFFF. These regions are designed to allow more rapid data transfer directly through the AHB bus, without having to go through the AHB-to-APB bridge. Peripherals using this type of interface include SPI, I²C, UART, ADC, AES, CRC, and USB.

2.1.5 Nested Vectored Interrupt Controller (NVIC)

The **MAX32620** includes the standard Nested Vectored Interrupt Controller (NVIC) as implemented for the Cortex-M4 ARM core. The NVIC supports high speed, deterministic interrupt response, interrupt masking and multiple interrupt sources. External interrupts support rising or falling edge trigger mode, as well as level triggered mode.

With the core instantiation parameters given above, the NVIC will support up to 64 distinct interrupt sources (including internal and external interrupts). Programmable interrupt priority is supported, allowing up to eight priority levels to be used (3-bit width for priority field).

2.1.6 ARM Debug

The **MAX32620** includes the standard JTAG debug engine as implemented for the Cortex-M4 ARM core. The JTAG TAP interface is supported, along with the reduced-pin-count Serial-Wire Debug Interface.

Since this is a standard ARM core component, it is instantiated as-is along with the rest of the Cortex-M4 core and cannot be modified to support special requirements for the **MAX32620** design. Accordingly, there are two separate JTAG TAP engines on the **MAX32620**: the ARM Debug JTAG and the Maxim Test JTAG. These are chained to allow physical access via a single JTAG TAP pin set.

There are two JTAG TAP device addresses for the **MAX32620**. The address for the ARM Debug, first in the JTAG chain, is 0x4BA00477; the address for the Maxim Test JTAG, second in the JTAG chain, is 0x07F67197.

Standard features supported by the ARM debug engine include the ability to set up to six breakpoints and two watchpoints, access main system memory and peripheral registers even when the CPU is running, and pause, trace, or reset the CPU.

> **Note** The debug engine is coupled to the CPU only for clocking and reset purposes. If the debug engine pauses the CPU, other peripherals and functions on the **MAX32620** are not paused and continue to operate normally.

2.2 Power Supplies and Modes

2.2.1 Digital Supply Voltages

The following digital supply voltages are required by the **MAX32620**. These are typically provided by an external power management IC (PMIC). The **MAX32620** does not include internal regulators to derive lower-voltage supplies from higher-voltage ones; this functionality (along with battery supply management and battery charging functions) is intended to be provided by the companion PMIC device.

2.2.1.1 VDD18 (Nominal 1.8V) Digital Power Supply

This digital power supply is used to power the I/O pad circuitry (all I/O pads on the **MAX32620** operate from a nominal 1.8V rail) as well as flash, comparators used for power supply monitoring, the two relaxation oscillators on the **MAX32620**, and certain power management and monitoring functions.

2.2.1.2 VDD12 (Nominal 1.2V) Digital Power Supply

This digital power supply is used to power most of the digital logic on the **MAX32620**, including the CM4 core, flash memory, SRAM, and digital peripherals.

2.2.1.3 VRTC (Nominal 1.8V) RTC Power Supply

This power supply is used to maintain functions on the **MAX32620** that must continue operating under all power management modes. It is used to power the 32kHz RTC oscillator, the 8kHz nanoring oscillator, and 'always-on' functions including the Real Time Clock, the power sequencer, and power management functions including the retention controller.

2.2.2 Analog Supply Voltages

The following analog supply voltages are required by the **MAX32620**. These are typically provided by an external power management IC (PMIC). The **MAX32620** does not include internal regulators to derive lower-voltage supplies from higher-voltage ones; this functionality (along with battery supply management and battery charging functions) is intended to be provided by the companion PMIC device.

2.2.2.1 VDDA (Nominal 1.8V) Analog Power Supply

This power supply is used to power analog functionality required on the **MAX32620** for the ADC and analog front end. These analog functions include the ADC itself, analog input multiplexing, internal reference generation, and internal/external reference selection.

2.2.2.2 VDDB (Nominal 3.3V) USB Power Supply

This power supply (used by the USB PHY and related functions) must be provided directly at the V_{DDB} pin since the **MAX32620** does not include an internal USB supply regulator.

2.2.3 Power Management Modes

The **MAX32620** supports 4 major power states which are user configurable to application specific needs. State-specific configuration details are stored in data retention power domain registers that are continuously powered across all modes of operation. Firmware-controlled power gating and hardware-controlled clock gating of peripherals allow designers to optimize system power consumption. The 4 power states supported are LP0:STOP (with or without RTC), LP1:STANDBY (with data retention), LP2:PMU (ARM asleep) and LP3:RUN (ARM active).

2.2.4 Power Supply Monitoring

Several programmable supply voltage monitors are provided, which allow a power fail warning interrupt and/or a system reset to be triggered when the supply voltage drops below a preset level (depending on the type of power supply and firmware settings). Supply voltage monitors are available to monitor all digital and analog externally provided power supplies on the device.

2.3 Clock Sources

2.3.1 Internal 96MHz Relaxation Oscillator

The **MAX32620** includes a high-frequency internal relaxation oscillator designed to operate at a nominal frequency of 96MHz. The output of this relaxation oscillator is used as the primary clock source for most digital logic on the device.

The **MAX32620** does not support operation from an external high-frequency crystal or clock source.

2.3.2 Internal 44MHz Relaxation Oscillator

A secondary high-frequency internal relaxation oscillator is also available on the **MAX32620**, this one targeted to run at a nominal frequency of 44MHz. The output of this relaxation oscillator is used as a secondary timing source for cryptographic functions in the TPU including the random number generator, the AES engine, and the MAA (on the **MAX32621** only). This increases security and makes certain types of timing-based attacks and power-analysis-based attacks against the TPU more difficult, due to the fact that certain internal cryptographic operations are driven by this secondary, non-observable clock.

2.3.3 RTC 32kHZ Crystal Oscillator

A 32kHz crystal oscillator (with the 32kHz crystal connected between the 32KIN and 32KOUT pins) is used to generate the 32kHz clock that is used by the Real Time Clock timers.

An external clock source may also be used by the **MAX32620** in place of a 32kHz crystal. For this configuration, the external clock source (which must meet the electrical/timing requirements given in the datasheet) is connected to the device on the 32KIN pin.

The 32KHz crystal oscillator is required for USB compliance as this circuit is used for internal frequency calibration of the 96MHz relaxation oscillator.

2.4 Memory

2.4.1 Internal Flash Memory

The **MAX32620** includes 2MB (or 512K x 32 bits) of internal flash memory. This internal flash memory may be used to store application program code as well as constant static data used by application code. Locations in the internal flash memory are programmed one doubleword (32 bits) at a time. The internal flash is logically divided into 256 logical pages of 8KB (or 2048 x 32 bits) each. When erasing data in internal flash memory, it is not possible to erase a single doubleword (32 bit) location independently. Instead, the flash contents may be erased either a by erasing a single logical page at a time (page erase), or by erasing the entire internal flash memory at once (mass erase). Erased (blank) locations in the internal flash memory will always read as all ones (0xFFFF_FFFF).

For read access, the internal flash memory is mapped into the standard code/data space region beginning at address 0x0000_0000. Modifications to the flash memory array (either program or erase operations) are handled by the Flash Controller module.

The beginning of program flash memory (starting at address 0x0000_0000) is also the default location for the ARM exception/interrupt vector table. Since this table contains initialization information such as the reset vector address which is required for the system to properly initialize, this table must be loaded into the flash in order for the **MAX32620** to execute any application code.

2.4.2 8KB Instruction Cache

The 8KB (2048 x 32 bits) instruction cache is used to cache instruction codes that have been recently fetched from locations in the internal flash memory. By avoiding repeated refetches of frequently accessed code, the instruction cache helps to improve execution throughput. Instruction fetches from the ARM CPU are handled by the instruction cache, which either returns the previously cached instructions (cache hit), or fetches the requested data from the internal flash (cache miss) and stores it for future access. Fetches between the instruction cache and the internal flash memory go through the code descrambler, so that the content stored in the instruction cache has already been descrambled. If the external SPI flash memory interface (SPI XIP) is being used, instruction fetches from the external SPI flash memory device are also cached in the same manner as instructions fetched from the internal flash memory. However, content stored in the external SPI flash memory is not scrambled.

Code access to the internal data SRAM is not cached; instruction fetches from SRAM are always performed directly. Data fetches (D-Code, as opposed to I-Code fetches for the purposes of decoding instructions) are also not cached and are performed directly. This includes fetches of local constant literals that are used by certain ARM instruction op codes.

Firmware has the option to flush the instruction cache manually at any point using a control register. When code mapped into a cached instruction space is updated (for example, if an in-application programming modification is made to an executable area of program flash), the cache should be flushed to ensure that the latest version of the code will be accessible and that stale cache contents will not be used instead of the new flash programmed values.

2.4.3 Internal SRAM

The internal SRAM on the **MAX32620** is 256KB in size and has a 32-bit internal width. It is mapped into the SRAM bit-banding access region beginning at address 0x2000_0000, and so it can be read/written either a full 32-bit word at a time, or a single bit at a time using the bit-band alias region (beginning at 0x2200_0000).

The bit-banding function can only be used when the SRAM is being accessed in data space by the ARM core itself, since the ARM core handles the remapping from the bit-banding alias area to a read-modify-write sequence (or single read/mask/shift for a bit read function) of the standard memory area.

The SRAM can be read or written freely in data space by the application, and can be used for either code or data access. The contents of the SRAM are not battery-backed. The SRAM is also used to hold the ARM CPU stack.

2.4.4 Peripheral Management Unit (PMU)

The Peripheral Management Unit (PMU) on the **MAX32620** is a programmable state machine that allows servicing of multiple onboard peripherals without using the CPU, thereby significantly improving overall system power consumption. It provides a generalized, flexible mechanism to perform automatic read and/or write

sequences to peripherals and areas of internal SRAM and is capable of operation during ARM sleep mode (LP2:PMU).

Peripherals which can be read from or written to using PMU channels (accessing AHB mapped memory areas) include:

- The ADC
- Any of the UART instances
- Any of the SPI master instances
- Any of the I²C master instances
- The I²C slave instance
- Any of the USB endpoint buffers (since they are stored in the main SRAM area)
- The CRC engine
- The AES engine
- The MAA (on the **MAX32621** only)

The PMU controller can also be used to read from the internal flash memory, the internal SRAM, the external SPI XIP flash memory (if enabled), or any peripheral register area which is normally accessible on the System bus.

2.4.5 Flash Information Block

The flash information block on the **MAX32620** allows production trim values and other nonvolatile information that will be written during the production process (e.g., device configuration and test details / logging data) to be stored in a separate dedicated area of internal flash memory.

The flash information block can be mapped to the AHB bus in the code space area, beginning at byte address location 0x0020_0000 (which is immediately following the highest internal flash memory address). When the flash information block is mapped to this memory region, it can be accessed in the same manner as the main flash; that is, it can be read from in code space (although accesses to it are not cached under any circumstances, and its contents are not subject to the scrambling/descrambling function used by the contents of the main program flash memory). It is possible to write to locations in the flash information block by writing the appropriate sequence directly to the flash controller registers.

However, this mapping of the flash information block (and direct read/write access to its contents) is only intended for testing and trimming purposes during the factory production test sequence. Once production test of the **MAX32620** has completed (or at least the last stage where trim operations are performed has been completed), a lock option setting is set in the information block to prevent future modifications to the trim and option settings (except for those which are explicitly allowed to be set later by the user, such as the DSB access key and the auto-lock security option). Setting the info block lockout setting also removes the information block from the memory map, which means that the only way to view its contents after that point is by reading the copies of the info block settings that have been copied by the hardware into the trim shadow registers. These registers are mapped to a different area in the APB peripheral region.

The flash controller automatically loads trim and other configuration values from the appropriate locations in the flash information block following a power on reset (POR). To prevent misconfiguration of trim and other option settings in the case of a random (i.e., unprogrammed), blank, or corrupted information block, an "info block valid" field is checked by the flash controller before the remaining locations in the information block memory are scanned. If the info block valid field (which is the first addressed location in the information block array) does not match the predetermined valid key value, the information block is presumed to be invalid, and instead of the trim registers being set to copies of the data contained in the information block, they are loaded with predetermined "default" values to prevent undesired or erratic system performance. These default values are designed to allow maximum access (with a minimum of security settings applied) for ease of testing and configuration.

2.4.6 Flash Memory Controller

The flash memory controller on the **MAX32620** handles control and timing signals for programming and erase operations on both the internal flash memory and the flash information block. The flash information block is normally written during production test only, and is not generally intended to be modified by the user application. Two exceptions to this are the Destructive Security Bypass access key value and the Auto-Lock option value. Each of these two fields may be programmed (one time only) by application firmware using a special procedure. This special programming procedure is allowed (for these two fields only) even after access to the rest of the information block has been locked out.

Functions provided by the flash controller for general use by application firmware include:

- Mass erase of the entire internal flash memory
- Page erase of a single page 8KB in the internal flash memory
- Write to one location (programmed 32 bits at a time) in the internal flash memory
- Special one-time writes by the user to the DSB Access Key and/or the Auto-Lock option values in the flash information block

2.5 Analog Peripherals

2.5.1 10-Bit ADC

The **MAX32620** includes a 10-bit sigma-delta analog-to-digital converter (ADC) with four external analog inputs and additional analog input channels to measure four internal power supply levels. The ADC also includes an internal reference voltage generator and a high impedance input buffer.

The ADC voltage reference can be generated from either an internal or external reference voltage. Two of the external analog inputs can be configured to divide the input voltage by five before samples are converted by the ADC.

2.5.2 ADC Sample Limit Monitoring

An optional feature allows samples captured by the ADC to be automatically compared against user-programmable high and low limits. Up to four channel limit pairs can be configured in this way. The comparison allows the ADC to trigger an interrupt (and potentially wake the CPU from a low power sleep mode) when a captured sample goes outside the preprogrammed limit range. Since this comparison is performed directly by the sample limit monitors, it can be performed even while the main CPU is suspended in low power mode LP2:PMU.

2.6 Digital Peripherals

2.6.1 GPIO Pins w/Interrupt and Wakeup Capability

The **MAX32620** includes seven GPIO ports with eight pins per port (for P0 through P5; P6 has only one GPIO pin) for a total of 49 GPIO pins.

Pins may be multiplexed with one or more digital peripheral functions. GPIO pins may be individually switched between GPIO and alternate peripheral input/output functions using the appropriate control registers.

All GPIO pins can be configured individually by firmware to act as external interrupt sources. All GPIO pins also have the option (which is separate from configuring a GPIO pin to act as an external interrupt source) to be configured to act as wakeup sources.

All GPIO pins support standard I/O operating modes including buffered logic inputs, high impedance, weak pullup and pulldown modes, open drain, and standard drive high/low outputs.

2.6.2 32-Bit Timer/Counters

The **MAX32620** includes six 32-bit timer/counter modules with the following features:

- 32-bit up/down count with auto reload mode
- One-shot or continuous operation mode
- Programmable 16-bit prescaler
- PWM output generation mode
- Capture/compare modes
- External input pin for timer input, clock gating or capture, limited to an input frequency of 1/4 of the peripheral clock
- Timer output pin
- Timer interrupt

Each 32-bit timer/counter module also has the option to be split into two separate 16-bit timers (dual 16-bit timer mode) for a possible total of twelve 16-bit timers. When a 32-bit timer is split into a pair of 16-bit timers, each 16-bit timer in this pair has the following features:

- 16-bit up/down count with auto reload mode
- One-shot or continuous operation mode
- Programmable 16-bit prescaler (setting is shared by both 16-bit timers in the pair)
- Timer interrupt (separate interrupt for each 16-bit timer in the pair)

2.6.3 Windowed Watchdog Timers

The **MAX32620** includes two independent watchdog timers (WDT) with window support. The watchdog timers run independently from each other and the CPU and have multiple clock source options for ensuring system stability. The watchdog uses a 32-bit timer with prescaler to generate the watchdog reset. When enabled, the watchdog timers must be fed/reset prior to timeout or within a specified window of time if window mode is enabled. Failure to do so before the watchdog times out will result in a watchdog reset event.

The first watchdog instance (WDT0) can be configured to trigger a system reset (reset of digital core) when it generates a watchdog reset. The second watchdog instance (WDT1) can be configured to generate a system reboot (equivalent to digital POR event) when it generates a watchdog reset.

Multiple clock sources are available for each watchdog timer and can be independently configured using system manager settings.

2.6.4 32-Bit Real Time Clock with Time of Day Alarm

A binary real-time clock (RTC) keeps the time of day in absolute seconds with configurable resolution determined by a programmable prescaler. Two time-of-day (counter comparison) alarms and independent sub-second (interval) alarm can cause an interrupt or wake the device from stop mode.

The independent sub-second alarm runs from the same RTC and allows the application to support interrupts with a minimum interval of approximately 244 microseconds. This creates an additional timer that can be used to measure long periods of time without performance degradation.

2.6.5 SPI Masters

The **MAX32620** includes three SPI master interface modules. Each SPI master module provides an independent master-mode-only serial communication channel that communicates synchronously with peripheral devices in a single or multiple slave system. Depending on the other peripherals and GPIO pins that are in use by the application, up to two separate SPI master modules are available for general use. A third SPI master module is reserved for Bluetooth module communication (when applicable).

The SPI master modules support half- or full-duplex communications with single, dual, or quad data transmission modes, and can be operated in master mode only. Each SPI master also supports configuration of active SS state (active low or active high) through the slave active select. The PMU can be used to access both the transmit (transaction) and receive FIFO buffers.

2.6.6 I²C

The **MAX32620** includes three I²C bus master modules which can be used to communicate with a wide variety of other I²C-enabled slave devices. The I²C bus is a two-wire, bidirectional bus which includes a ground line and two bus lines: the serial data line (SDA) and the serial clock line (SCL). Both the SDA and SCL lines must be driven as open-collector/drain outputs. External resistors are required to pull the lines to a logic-high state.

In the master mode, the I²C bus master has ownership of the I²C bus, drives the clock, and generates the START and STOP conditions. This allows the I²C bus master to send data to a slave or receive data from a slave as required.

The **MAX32620** also includes a single I²C slave interface module which acts as an I²C bus slave device. Unlike the general-purpose FIFO-based I²C master interface, the I2CS handles not only the low-level timing and protocol required to communicate with the I²C bus, but it also implements a set of predefined commands which allow an external I²C bus master to read and write from a set of bidirectional "mailbox" byte registers. This protocol handling is performed automatically by the I2CS module, without requiring any intervention on the part of the **MAX32620** CPU. The I²C slave interface module relies on an externally generated clock to drive SCL and responds to data and commands only when requested by the I²C bus master.

2.6.7 UART

The **MAX32620** includes four UART interface modules. The UART interface supports full-duplex asynchronous data transfers, and also provides optional hardware flow control using the RTS and CTS signal lines.

Options supported by each UART interface module include:

- Hardware flow control using RTS (Ready to Send), CTS (Clear to Send), or both
- 32-byte transmit and receive FIFOs
- Programmable interrupts for receive and transmit
- Independent baud rate generator
- Character sizes of 5, 6, 7, or 8 bits
- Configurable optional parity bit for even or odd parity checking
- Optional multidrop (single master, multiple slave) mode using parity bit to mark slave address characters

2.6.8 USB 2.0 Device Slave with Integrated Transceiver

The **MAX32620** includes a USB device controller module which is compliant with the USB 2.0 specification, providing full-speed operation as a USB peripheral device. The USB module includes an integrated PHY interface, which allows the external USB pins to be connected directly to the USB bus. This reduces required board space and overall system cost.

The USB module includes an integrated AHB bus master which is used to write to and read from the buffers for each supported/active endpoint. These buffers are located in the standard system SRAM (in user-configurable locations), so they can be accessed by firmware directly as well as by the USB DMA engine. A total of seven endpoint buffers are supported with configurable selection of IN or OUT (endpoints 1 through 7). Endpoint 0 is read-only.

2.6.9 CRC Hardware Block with CRC16 and CRC32

A CRC hardware module is included to provide fast calculations and integrity checking of application software and data. The CRC module supports both CRC-16-CCITT and CRC-32 polynomial modes. Both the CRC-16 and CRC-32 operations (per doubleword of data loaded) complete in a single system clock cycle.

Additional features of the CRC module include:

- Programmable start seed
- Programmable start address
- Programmable length
- Direct load or PMU-based memory load support

2.7 Security Features

2.7.1 Trust Protection Unit (TPU)

The **MAX32620** supports several types cryptographic and security peripherals which are grouped together to form the Trust Protection Unit, or TPU. Depending on the device configuration, the TPU may include cryptographic peripherals (such as the AES engine or the MAA) as well as other security features (such as the random number generator). These peripherals and features can be used by applications to protect critical user information within the device.

2.7.2 AES Cryptographic Engine

One component found in the TPU is an Advanced Encryption Standard (AES) cryptographic engine. This cryptographic engine allows 256-bit AES encryption and decryption operations to be performed in hardware without CPU intervention.

The AES control register selects encryption or decryption mode, controls interrupt notification of the processor upon an operation completion, and selects whether or not the key expansion (generation of first/last round key) is performed before the encryption or decryption operation begins. For multiple-block encryption or decryption operations using a single key, the key expansion is performed on the first block only. This allows subsequent operations using the same key to complete more quickly by reusing the previously generated round key information.

The working AES key, cryptographic working space, and input and output parameters (plaintext to ciphertext or vice versa) are stored in a dedicated internal AES memory. This memory is automatically cleared (rapid zeroization) in the event of a tamper response.

2.7.3 Battery-Backed AES Secure Key Storage

The battery-backed (by the V_{RTC} supply voltage) RTC module contains the Secure Key Storage (SKS) area, which is a dedicated set of battery backed registers. The Secure Key Storage can be used store a master AES key (up to 128 bits in length) or other critical data. Data in the SKS area will be automatically cleared (rapid zeroization) in the event of a tamper response.

The master AES key can be generated by the device using a pseudo-random number algorithm. This key is intended for use in encrypting other sensitive information that might be stored in other locations on the device (such as the main system SRAM or the program flash memory) that will not be automatically wiped in the event of a tamper response. However, if this sensitive information has been encrypted before storage using an AES master key, then once a tamper response occurs and this master key has been deleted, the sensitive information will not be recoverable by an attacker at that point. Even if the ciphertext version of the information can be recovered in some way, the key that was used to encrypt that data no longer exists.

Another potential use of this key would be to encrypt larger keys that might be stored in long-term, nonvolatile storage on the device (such as a public key, private key, or certificate, which might be stored in the main program flash). By encrypting the larger keys with the AES master key, they may safely be stored in a nonsecure location such as the main SRAM or program flash memory. If a tamper response occurs, the AES master key will be wiped, effectively destroying the other encrypted keys or data as well since the encrypted information is now useless.

The Secure Key Storage area (which has a capacity of four 32-bit registers or 128 bits total) does not necessarily have to be used for an AES master key. The SKS can be used to hold any type of secure data that must be immediately erased in the event of a tamper response.

2.7.4 Code Scrambling

All application code and data loaded into the internal flash memory is scrambled in both content and location by hardware before it is stored in the flash. When data is retrieved from the flash, it is descrambled before arriving at the program cache (for instruction fetches) or the main data bus (for data fetches). Both the scrambling and descrambling operations are transparent to the end user.

3 Memory, Register Mapping, and Access

3.1 Memory, Register Mapping, and Access Overview

The ARM Cortex-M4 architecture 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.

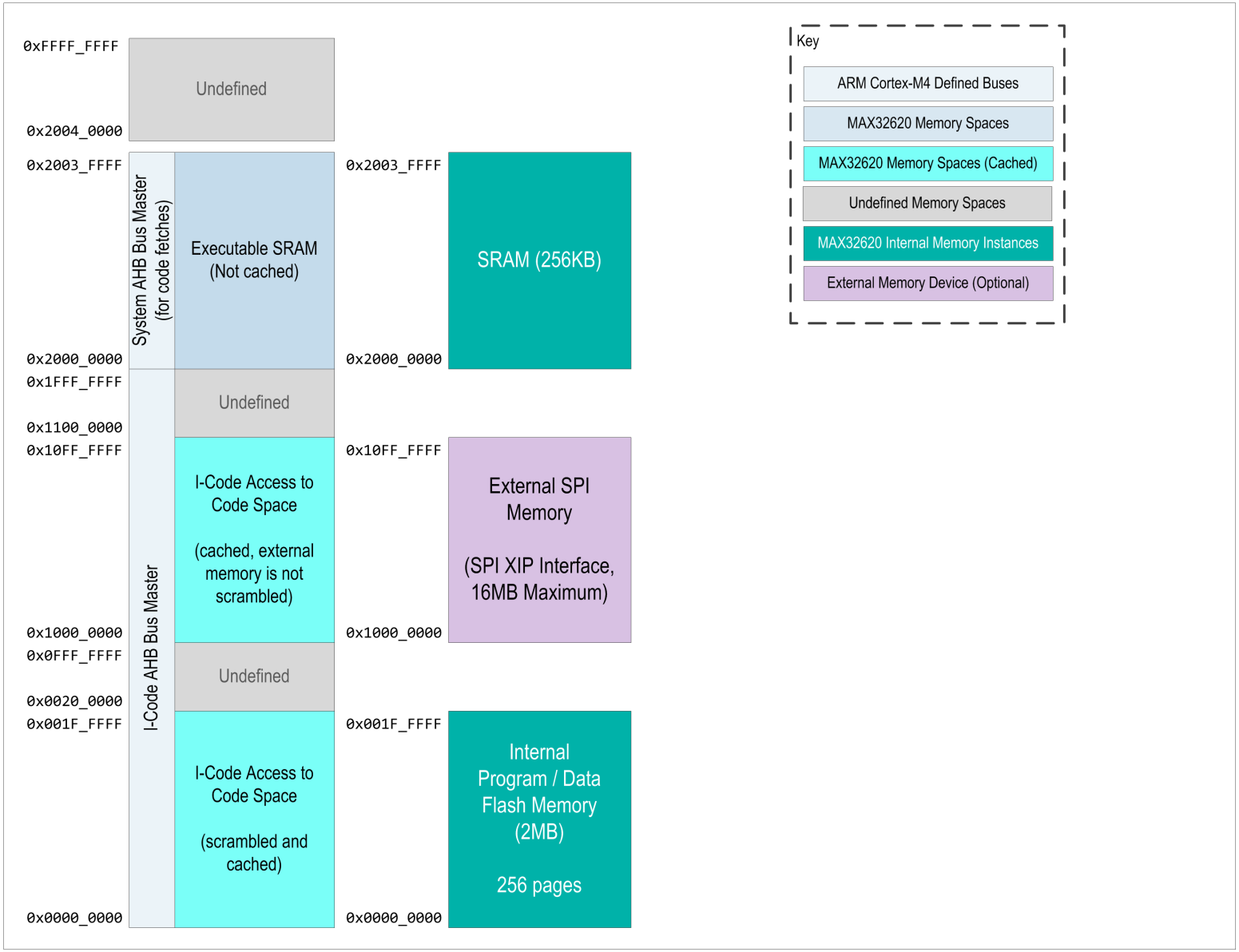

Figure 3.1: Code Memory Mapping

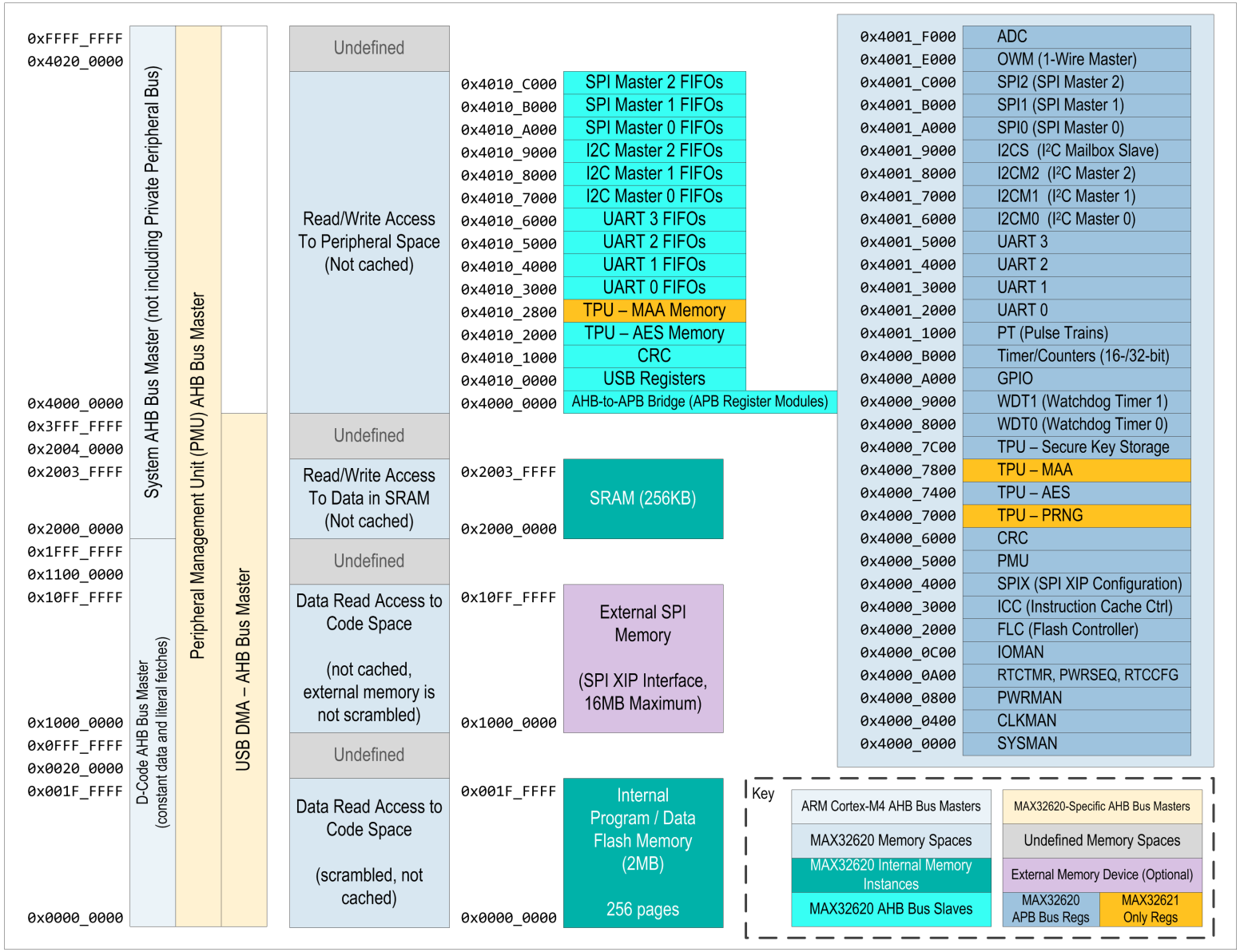

Figure 3.2: Data Memory Mapping

3.2 Standard Memory Regions

A number of 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 **MAX32620**, a Cortex-M4-based device, must contain some code and data memory for application code and variable/stack use, as well as certain components which are part of the instantiated core.

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 Cortex-M4 core and 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 **MAX32620**, the code space memory area contains the main internal flash memory, which holds the majority of 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 0x001F_FFFF. This program memory area must also contain the default system vector table (located initially at address 0x0000_0000), which contains the reset vector for the device and the initial settings for all system exception handlers and interrupt handlers.

The code space memory on the **MAX32620** also contains the mapping for the flash information block, from 0x0020_0000 to 0x0020_1FFF. However, this mapping is generally only present during production test; it is disabled once the information block has been loaded with valid data and the info block lockout option has been set. This memory is accessible for data reads only and cannot be used for code execution.

Optionally, the SPIX (SPI XIP, or Execute In Place) module can be used to expand the available code and data memory space for the **MAX32620**. This expansion consists of mapping the contents of an external SPI flash memory device (up to 16MB) into a read-only area of the code memory map. If enabled, the external memory is mapped starting at byte address 0x1000_0000 up to a maximum of 0x10FF_FFFF (for a 16MB device). This external memory can be used for code execution as well as static data storage.

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 **MAX32620**, this memory area contains the main system SRAM 256KB, which is mapped from 0x2000_0000 to 0x2003_FFFF. The entirety of the SRAM memory space on the **MAX32620** is contained within the dedicated ARM Cortex-M4 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 core 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 PMU AHB bus master will not trigger a bit-banding operation and will instead result in an AHB bus error.

The SRAM area on the **MAX32620** can be used to contain executable code. Code stored in the SRAM is accessed directly for execution (using the system bus) and is not cached or code scrambled.

The SRAM is also 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 by the system exception table (which is, by default, stored at the beginning of the internal flash memory).

The general purpose PMU engine and the function specific AHB bus master included in the USB peripheral block can both access the SRAM to use as general storage or working space. Specifically in the case of the USB interface, SRAM memory area can be used to store the descriptor table for the endpoint buffers as well as the endpoint buffers themselves.

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 **MAX32620**, 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 core. 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 core. If another memory bus master (such as the PMU 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 **MAX32620**, 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).

A secondary region within the peripheral memory space (0x04010_0000 to 0x401F_FFFF) allows peripherals that require more rapid data transfer to handle this data transfer using their own local AHB slave instances (instead of going indirectly through the AHB-to-APB bridge). This allows peripherals which have FIFOs or other functions requiring large amounts of data to be transferred quickly (such as the ADC or communications peripherals like SPI) to benefit from the more rapid data transfer rate of the AHB bus.

Peripheral	Register Module(s)	Instance #	APB Start Address (Note 1)	AHB Start Address
SysMan	CLKMAN		0x4000_0000	
	PWRMAN		0x4000_0000	
	IOMAN		0x4000_0000	
Flash Controller	FLC		0x4000_2000	
I-Cache Controller	ICC		0x4000_3000	
SPI XIP	SPIX		0x4000_4000	
PMU	PMU		0x4000_5000	
USB	USB			0x4010_0000
CRC	CRC		0x4000_6000	0x4010_1000
TPU	TPU, AES, MAA		0x4000_7000	0x4010_2000
Watchdog Timer 0	WDT	0	0x4000_8000	
Watchdog Timer 1	WDT	1	0x4000_9000	
GPIO	GPIO		0x4000_A000	
32b/2x16b Timer 0	TMR	0	0x4000_B000	
32b/2x16b Timer 1	TMR	1	0x4000_C000	
32b/2x16b Timer 2	TMR	2	0x4000_D000	
32b/2x16b Timer 3	TMR	3	0x4000_E000	
32b/2x16b Timer 4	TMR	4	0x4000_F000	
32b/2x16b Timer 5	TMR	5	0x4001_0000	
Pulse Trains 0-15	PTG, PT(0-15)		0x4001_1000	
UART Interface 0	UART	0	0x4001_2000	0x4010_3000
UART Interface 1	UART	1	0x4001_3000	0x4010_4000

Peripheral	Register Module(s)	Instance #	APB Start Address (Note 1)	AHB Start Address
UART Interface 2	UART	2	0x4001_4000	0x4010_5000

\ UART Interface 3 | UART | 3 | 0x4001_5000 | 0x4010_6000 I²C Master 0 | I2CM | 0 | 0x4001_6000 | 0x4010_7000 I²C Master 1 | I2CM | 1 | 0x4001_7000 | 0x4010_8000 \ I²C Master 2 | I2CM | 2 | 0x4001_8000 | 0x4010_9000 I²C Slave | I2CS | | 0x4001_9000 | SPI Master 0 | SPI | 0 | 0x4001_A000 | 0x4010_A000 SPI Master 1 | SPI | 1 | 0x4001_B000 | 0x4010_B000 SPI Master 2 | SPI | 2 | 0x4001_C000 | 0x4010_C000 One-Wire Master | OWM | | 0x4001_E000 | ADC | ADC | | 0x4001_F000 |

Note (1) Only 32-bit width access is permitted when reading or writing registers in the APB mapped area. Accesses of 8-bit width and 16-bit width are not supported and will result in an AHB bus fault.

3.2.4 External RAM Space

The external RAM space area of memory is intended for use in mapping off-chip external memory and is defined from byte address range 0x6000_0000 to 0x9FFF_FFFF (1GB maximum). The **MAX32620** does not implement this memory area. However, as previously noted the SPIX module can be used to map external SPI flash memory device contents into a read-only code/data memory area.

3.2.5 External Device Space

The external device space area of memory is intended for use in mapping off-chip device control functions onto the AHB bus. This memory space is defined from byte address range 0xA000_0000 to 0xDFFF_FFFF (1GB maximum). The **MAX32620** does not implement this memory area.

3.2.6 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 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 PMU or the USB.

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.

3.2.7 System Area (Vendor Defined)

The system area (vendor defined) memory space is reserved for vendor (system integrator) specific functions that are not handled by another memory area. It is defined from byte address range 0xE010_0000 to 0xFFFF_FFFF. The **MAX32620** does not implement this memory region.

3.3 Device Memory Instances

This section details physical memory instances on the **MAX32620** (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 particular peripheral, are not covered here.

3.3.1 Main Program Flash Memory

The main program flash memory is 2MB in size and consists of 256 logical pages of 8KB each.

3.3.2 Instruction Cache Memory

The instruction cache is 8KB in size and is used to cache instructions fetched using the I-Code bus. This includes instructions fetched from the internal flash memory as well as instructions fetched from an external SPI memory device (if SPIX is enabled). Note that the cache is used for instruction fetches only. Data fetches (including code literal values) from the internal flash memory or external SPIX memory do not use the instruction cache.

When code is fetched from the internal flash memory, it is descrambled before being cached. However, contents of the external SPI flash memory are never scrambled/descrambled.

3.3.3 Information Block Flash Memory

The information block is a separate flash instance with a single 8KB page. It is used to store trim settings (option configuration and analog trim) as well as other nonvolatile device-specific information intended for use by firmware.

3.3.4 System SRAM

The system SRAM is 256KB in size and can be used for general purpose data storage, the ARM system stack, USB data transfers (endpoints), and code execution if desired.

3.3.5 AES Key and Working Space Memory

The AES key memory and working space for AES operations (including input and output parameters) are located in a dedicated register file memory tied to the AES engine block. This AES memory is mapped into AHB space for rapid firmware access. In the event of a tamper detection, the AES memory will be automatically erased by hardware.

3.3.6 MAA Key and Working Space Memory

The MAA contains a dedicated memory for key storage, input and output parameters for operations, and working space. It is mapped into the AHB memory space for ease of loading and unloading.

Like the MAA itself, the MAA memory is only usable on the **MAX32621** version of the device.

3.3.7 TPU Memory Secure Key Storage Area

The **MAX32620** contains a specialized 128-bit memory that is designed to preserve critical data (such as a 128-bit AES key) even when the device is in the lowest power-saving state. As long as the RTC power supply is still available, the contents of this memory will be retained, even if the AES block and the main SRAM are shut down completely. In the event of a tamper response, the contents of this memory area will be automatically erased by hardware.

The Secure Key Storage Area consists of four V_{RTC}-backed 32-bit registers: TPU_TSR_SKS0, TPU_TSR_SKS1, TPU_TSR_SKS2, and TPU_TSR_SKS3.

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.

3.4.1 Core AHB Interface - 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 and the external SPI flash memory (if SPI XIP is enabled). 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 or the external SPI flash memory when a cache miss occurs.

3.4.2 Core AHB Interface - D-Code

This AHB master is used by the ARM core for data fetches from memory instances located in code space from byte addresses 0x0000_0000 to 0x1FFF_FFFF. This bus master has access to the internal flash memory, the external SPI flash memory (if SPI XIP is enabled), and the information block (if it has not been locked).

3.4.3 Core AHB Interface - 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.

3.4.4 AHB Master - Peripheral Management Unit (PMU)

The PMU bus master has access to all off-core memory areas (equivalent to areas accessible by the System bus plus areas accessible by the D-Code bus). It does not have access to the ARM Private Peripheral Bus area.

3.4.5 AHB Master - USB Endpoint Buffer Manager

The USB AHB bus master is used to manage endpoint buffers in the SRAM. It has access to the SRAM (read/write, for storage and retrieval of endpoint buffer data), as well as the internal and/or external flash data contents (which can be used to contain static data for transmission by the USB).

3.5 Flash Controller and Instruction Cache

3.5.1 Overview

The flash memory controller on the **MAX32620** handles control and timing signals for programming and erase operations on the internal flash memory. The flash controller is used during the application development and release cycle to erase the internal flash memory and then load the application code and data. This is typically done using an external debug adapter (using the provided JTAG or Serial Wire Debug interfaces), but it is also possible to implement a custom bootloader to use other interfaces to load or update application code under firmware control.

Once an application has been loaded, user application firmware can also use the flash controller to perform in-application programming operations. In-application programming allows contents of internal flash memory (application code or data) to be erased and reprogrammed as needed under firmware control. Internal flash memory locations can be programmed one 32-bit doubleword at a time, but in order to erase previously programmed data, a page erase operation must be used to clear all the contents of a logical page (8KB) of internal flash memory.

The Instruction Cache (ICC) is related to the flash controller and is used to cache up to 8KB of application code in order to reduce access time and improve overall application speed, particularly in tight loops where the same portion of code may be accessed multiple times. This operation is transparent to the user, and generally the operation of the instruction cache does not need to be of concern to the user application except when instruction code located in internal flash memory or external (SPIX) flash memory is altered. In this instance, the cache must be manually flushed by the application to remove stale contents.

The Instruction Cache is disabled following any system reset and must be explicitly enabled at the start of application code in order for it to be used.

3.5.2 Flash Controller Operations

This section details the procedures used to program and erase the contents of internal flash memory using the flash controller. Note that the flash controller is only used when programming or erasing the **internal** flash memory. Any external SPI flash memory mapped into memory space using the SPIX interface must be programmed using a standard SPI master interface; the flash controller is not involved.

Flash Write Operation

Writes to the internal flash memory are performed one 32-bit doubleword at a time. In order to write to a location in internal flash memory, the location must be in the erased state (all ones, with the value FFFF_FFFFh). If the location has previously been programmed to a different value, it cannot be reprogrammed without first erasing the entire flash page containing that location.

If the flash location has the contents FFFF_FFFFh, it can be programmed to a new value using the following procedure.

1. Verify that the FLC_CTRL.pending bit reads 0. If this bit reads 1, the flash controller is still initializing or completing a previously issued command. Once the bit returns to 0, the flash controller is ready to perform the write operation.
2. Write FLC_CTRL.flsh_unlock to 0010b (2 decimal) to unlock the 'safety' preventing inadvertent write and erase operations from being triggered. This field will only need to be written to this value once following each reset, but rewriting it before each operation won't cause any problems.
3. Write FLC_FADDR to the byte address of the 32-bit doubleword location to be written in flash. To ensure that the address is 32-bit aligned, bits 1 and 0 of this address value should both be zero. Values other than 00b for the low two address bits will be ignored during a write, and the hardware will force these two bits to 0 instead.
4. Write FLC_FDATA to the new 32-bit value that will be written to the location in flash memory.
5. Write FLC_CTRL.write to 1 to start the flash write operation.
6. Monitor the state of the FLC_CTRL.pending bit. This bit should read 1 while the flash operation is in progress. After the flash operation completes, this bit will return to a 0 state.
7. The FLC_INTR register can be checked to determine the success or failure of the flash write operation. If the flash operation halted due to a failure condition, the FLC_INTR.failed_if flag will be set to 1 by hardware. This flag is not auto-cleared and must be cleared by application firmware after the value has been checked.

Flash Page Erase Operation

Typically, when the internal flash contents are modified under application software control, the flash page erase operation is used to erase the contents of the internal flash (one 8KB page at a time) for any areas of the flash that need to be rewritten to new values. There is also a 'mass erase' operation that can be used to clear

the entire contents of internal flash in one operation, but this is generally only used when the application software is initially loaded (using an external JTAG or Serial Wire debug controller).

To erase a page of internal flash memory under application control, the following procedure can be used.

1. Verify that the FLC_CTRL.pending bit reads 0. If this bit reads 1, the flash controller is still initializing or completing a previously issued command. Once the bit returns to 0, the flash controller is ready to perform the write operation.
2. Write FLC_CTRL.flsh_unlock to 0010b (2 decimal) to unlock the 'safety' preventing inadvertent write and erase operations from being triggered. This field will only need to be written to this value once following each reset, but rewriting it before each operation won't cause any problems.
3. An additional key field must be written before each page erase operation because the page erase operation clears a large section of flash memory. This is intended to prevent accidental page erases by runaway or malfunctioning application code. To enable the page erase operation, write FLC_CTRL.erase_code to 55h.
4. Write FLC_FADDR to the byte address of any 32-bit doubleword location inside the flash page to be erased. Effectively, the low 13 bits of this byte address will be ignored, so any location within the desired page area will have the same effect. For example, any address from 00_0000h...00_1FFFh will trigger an erase of page 0, addresses from 00_2000h...00_3FFFh will trigger an erase of page 1, and so on. There are 256 pages in total in the internal flash, which makes the address range of the highest page (page 255) 1F_E000h..1F_FFFFh.
5. Write FLC_CTRL.page_erase to 1 to start the page erase operation.
6. Monitor the state of the FLC_CTRL.pending bit. This bit should read 1 while the flash operation is in progress. After the flash operation completes, this bit will return to a 0 state.
7. The FLC_INTR register can be checked to determine the success or failure of the flash write operation. If the flash operation halted due to a failure condition, the FLC_INTR.failed_if flag will be set to 1 by hardware. This flag is not auto-cleared and must be cleared by application firmware after the value has been checked.

3.5.3 Instruction Cache Controller Operations

Enabling the Cache Controller

Following system reset, application software must enable the instruction cache by writing ICC_CTRL_STAT.enable to 1. After this bit has been written, the ICC_CTRL_STAT.ready bit can be monitored to determine when the cache is enabled and ready for use. When the cache is ready, the ICC_CTRL_STAT.ready bit will read 1; otherwise, the bit will read 0 indicating that the cache is not yet ready.

Flushing the Cache Contents

The cache stores the contents of previous instruction fetches from the internal flash memory (and or external SPI flash, if SPIX is enabled) so that when the same locations are accessed again, the data can be available more quickly from the cache without having to go through the delay of another flash read access cycle.

As long as the contents of instruction code in internal/external flash memory remain static, the cache will operate as intended. However, in the event that application software performs one or more write or erase operations on the internal flash as detailed in the previous section, then the new data contained in the internal flash memory will not be reflected in the contents held in the cache. If the CPU attempts to fetch program code from locations in the internal flash memory that have been modified, then the cache will return the 'stale' or previously fetched contents from the cache and not the new contents from the flash memory itself.

To force the instruction cache to flush all cached values and reload contents from the internal/external flash memory for any subsequent program code fetches, write any value to the ICC_INVDT_ALL register. This will cause all currently cached data to be cleared. While the cache is clearing, the ICC_CTRL_STAT.ready bit will read 0; this bit will change to a read value of 1 when the cache is ready for use again.

Since the instruction cache is also used to store fetches from the external SPI flash memory (if SPI XIP is enabled), then if application code stored in external flash is modified (or if the SPIX module is reconfigured to switch slave select lines and access a different device), the instruction cache will need to be flushed in this case as well.

3.5.4 Registers (FLC)

Address	Register	Access	Description	Reset By
0x40002000	FLC_FADDR	R/W	Flash Operation Address	Sys
0x40002004	FLC_FCKDIV	R/W	Flash Clock Pulse Divisor	Sys
0x40002008	FLC_CTRL	***	Flash Control Register	Sys
0x40002024	FLC_INTR	R/W	Flash Controller Interrupt Flags and Enable/Disable 0	Sys
0x40002030	FLC_FDATA	R/W	Flash Operation Data Register	Sys
0x40002050	FLC_PERFORM	R/W	Flash Performance Settings	Sys
0x40002080	FLC_STATUS	R/O	Security Status Flags	
0x40002088	FLC_SECURITY	R/W	Flash Controller Security Settings	POR
0x4000209C	FLC_BYPASS	R/O	Status Flags for DSB Operations	Sys
0x40002100	FLC_USER_OPTION	W/O	Used to set DSB Access code and Auto-Lock in info block	Sys
0x40002140	FLC_CTRL2	R/W	Flash Control Register 2	Sys
0x40002144	FLC_INTFL1	W1C	Interrupt Flags Register 1	Sys
0x40002148	FLC_INTEN1	R/W	Interrupt Enable/Disable Register 1	Sys
0x40002170	FLC_BL_CTRL	R/W	Bootloader Control Register	Sys
0x40002174	FLC_TWK	R/W	FLC TWK Cycle Count	Sys
0x4000217C	FLC_SLM	R/W	Sleep Mode Register	Sys

3.5.4.1 FLC_FADDR

FLC_FADDR.faddr

Field	Bits	Sys Reset	Access	Description
faddr	21:0	00_0000h	R/W	Flash Operation Address

Byte address for flash write and erase operations.

3.5.4.2 FLC_FCKDIV

FLC_FCKDIV.fckdiv

Field	Bits	Sys Reset	Access	Description
fckdiv	6:0	50	R/W	Flash Clock Pulse Divisor

The value of this field is used to generate a 1 microsecond width pulse from the system clock source. This pulse is used when performing write or erase operations on the flash memory. This field should be set to the system clock source frequency in MHz (or as close as possible); for example, if the system clock is running at 96MHz, this field should be set to 96.

3.5.4.3 FLC_CTRL

FLC_CTRL.write

Field	Bits	Sys Reset	Access	Description
write	0	0	R/W	Start Flash Write Operation

Writing this bit to 1 attempts to start a flash write operation using the contents of the two registers (which must be written before this bit is set):

- FLC_FADDR = Byte address of location to write.
- FLC_FDATA = Value to write to flash location.

FLC_CTRL.mass_erase

Field	Bits	Sys Reset	Access	Description
mass_erase	1	0	R/W	Start Flash Mass Erase Operation

Writing this bit to 1 attempts to start a flash mass erase operation (of the main program flash). The Flash Erase Code must first be written to AAh for this operation to be accepted by the FLC.

FLC_CTRL.page_erase

Field	Bits	Sys Reset	Access	Description
page_erase	2	0	R/W	Start Flash Page Erase Operation

Writing this bit to 1 attempts to start a flash page erase operation (in the main program flash for addresses from 00_0000h-1F_FFFFh). The Flash Erase Code must first be written to 55h for this operation to be accepted by the FLC.

FLC_CTRL.erase_code

Field	Bits	Sys Reset	Access	Description
erase_code	15:8	00h	R/W	Flash Erase Code

This is a key field that must be written in order to trigger a flash mass erase or page erase operation; the intention is to make it more difficult to trigger these operations by mistake.

- AAh: Enable a mass erase operation
- 55h: Enable a page erase operation

This field is auto-cleared by the hardware to zero following any mass erase or page erase operation.

FLC_CTRL.info_block_unlock

Field	Bits	Sys Reset	Access	Description
info_block_unlock	16	0	R/O	Flash Info Block Locked

- 1: Flash info block is accessible (not locked)
- 0: Flash info block is locked - it may not be targeted by write operations, and it is not mapped into AHB memory space.

Note Even if the info block is unlocked, this field will only read 1 if the flsh_unlock field has also been set to 0010b.

FLC_CTRL.write_enable

Field	Bits	Sys Reset	Access	Description
write_enable	17	0	R/O	Flash Writes Enabled

- 0: Flash writes are not currently enabled.
- 1: Flash writes are currently enabled.

FLC_CTRL.pending

Field	Bits	Sys Reset	Access	Description
pending	24	0	R/O	Flash Controller Status

This bit goes to an active high state (1) whenever the flash controller is performing a read, write, or erase operation. Once the operation has completed, the bit will return to 0.

FLC_CTRL.info_block_valid

Field	Bits	Sys Reset	Access	Description
info_block_valid	25	0	R/O	Info Block Valid Status

- 0: Flash info block does not contain valid contents. Security, test and trim settings have been set to default values by the FLC.
- 1: Flash info block is valid, and its contents have been loaded into the trim shadow registers.

FLC_CTRL.auto_incre_mode

Field	Bits	Sys Reset	Access	Description
auto_incre_mode	27	0	R/W	Address Auto-Increment Mode

- 0: Auto-increment mode disabled.
- 1: Auto-increment mode enabled; flash address will be automatically incremented by 4 following a write operation.

FLC_CTRL.flsh_unlock

Field	Bits	Sys Reset	Access	Description
flsh_unlock	31:28	0000b	R/W	Flash Write/Erase Enable

0010b/2d: Flash write and erase operations are enabled.

3.5.4.4 FLC_INTR

FLC_INTR.started_if

Field	Bits	Sys Reset	Access	Description
started_if	0	0	R/W	Flash Operation Started

Write to zero to clear bit to 0.

Set to 1 by hardware when a flash operation (write or erase) starts.

FLC_INTR.failed_if

Field	Bits	Sys Reset	Access	Description
failed_if	1	0	R/W	Flash Operation Failed

Write to zero to clear bit to 0.

Set to 1 by hardware when an error occurs during a flash operation

FLC_INTR.started_ie

Field	Bits	Sys Reset	Access	Description
started_ie	8	0	R/W	Flash Operation Started Interrupt Enable

- 0: Interrupt disabled.
- 1: Setting the Flash Operation Started bit to 1 will cause the FLC to generate an interrupt.

FLC_INTR.failed_ie

Field	Bits	Sys Reset	Access	Description
failed_ie	9	0	R/W	Flash Operation Failed Interrupt Enable

- 0: Interrupt disabled.
- 1: Setting the Flash Operation Failed bit to 1 will cause the FLC to generate an interrupt.

3.5.4.5 FLC_FDATA

FLC_FDATA

Sys Reset	Access	Description
00000000h	R/W	Flash Operation Data Register

This register is used as an input parameter for various flash controller operations.

3.5.4.6 FLC_PERFORM

FLC_PERFORM.delay_se_en

Field	Bits	Sys Reset	Access	Description
delay_se_en	0	1	R/W	Delay SE Enable

- 0: Original flash access design; xaddr and yaddr are activated the same clock edge as SE
- 1: Inserts an additional clock cycle before SE is driven active (safety margin, slows down read accesses) (default)

Note This bit is intended for internal testing purposes only. Application firmware should not modify the default value of this bit.

FLC_PERFORM.fast_read_mode_en

Field	Bits	Sys Reset	Access	Description
fast_read_mode_en	8	0	R/W	Fast Read Mode Enable

- 0: Original flash access mode, 8-9 cycles for read access (default)
- 1: Fast read mode - the flash controller will perform back-to-back flash reads for an access time of 2-3 cycles per read cycle (after the 2nd)

Note This bit is intended for internal testing purposes only. Application firmware should not modify the default value of this bit.

3.5.4.7 FLC_STATUS

FLC_STATUS.jtag_lock_window

Field	Bits	Sys Reset	Access	Description
jtag_lock_window	0	n/a	R/O	Debug Locked - Hardware Window

- 0: No lockout from this source
- 1: The debug lockout is being asserted because the debug hardware window lockout feature is active, and the window of 1024 cycles following reset has not yet elapsed.

FLC_STATUS.jtag_lock_static

Field	Bits	Sys Reset	Access	Description
jtag_lock_static	1	n/a	R/O	Debug Locked - Firmware Lockout

- 0: No lockout from this source
- 1: The debug lockout is being asserted because the Disable Debug (bit 0) in the FLC_SECURITY register has been set to 1. This is typically done by firmware as part of a user-defined security scheme.

FLC_STATUS.auto_lock

Field	Bits	Sys Reset	Access	Description
auto_lock	3	n/a	R/O	Debug Locked - Auto Lock

- 0: No lockout from this source
- 1: The debug lockout is being asserted unconditionally because the Auto Lock option has been enabled in the flash info block. The only way to remove this setting is to erase the info block (if the info block has not been locked) or to use the FLC_BYPASS register to perform a Factory Global Erase operation (Super-Wipe).

FLC_STATUS.trim_update_done

Field	Bits	Sys Reset	Access	Description
trim_update_done	29	n/a	R/O	Trim Update Done

Set to 1 by hardware once the info block trim has been loaded.

FLC_STATUS.info_block_valid

Field	Bits	Sys Reset	Access	Description
info_block_valid	30	n/a	R/O	Info Block Valid

Once the Trim Update Done field is set to 1, the value of this bit can be checked to see the status of the info block. If this bit returns 1, the contents of the info block are valid. If this bit returns 0, the contents of the info block are invalid and the trim has been loaded with default values. The info block should ALWAYS be valid on production tested devices.

3.5.4.8 FLC_SECURITY

FLC_SECURITY.debug_disable

Field	Bits	Sys Reset	Alt Reset	Access	Description
debug_disable	7:0	no effect	POR:00h	R/W	Debug Lockout

This field can be set to two values as follows:

- 00h: Debug access is unlocked; the ARM debugger can be used normally.
- 01h: Debug access is locked out. All access to the ARM debug engine is blocked.

To set this field to 01h (to set debug lockout), write the value A5h to this field. This field cannot be altered if the security_lock field has been set to 1000b.

FLC_SECURITY.mass_erase_lock

Field	Bits	Sys Reset	Alt Reset	Access	Description
mass_erase_lock	11:8	no effect	POR:0000b	R/W	Mass Erase Lockout

This field can be set to two values as follows:

- 0000b: Mass erase of the main internal flash memory can be performed in the usual manner.
- 0001b: The mass erase operation is locked out.

To set this field to 0001b (lock out mass erase), write the value Ch to this field. This field cannot be altered if the security_lock field has been set to 1000b.

FLC_SECURITY.security_lock

Field	Bits	Sys Reset	Alt Reset	Access	Description
security_lock	31:28	no effect	POR:0000b	R/W	Security Lock

Once this field is set to 1000b, the SECURITY register and the flash page protect registers may no longer be modified.

To set this field to 1000b, write the value 5 to the field. This field can only be set by firmware; it cannot be cleared.

3.5.4.9 FLC_BYPASS

FLC_BYPASS.destruct_bypass_erase

Field	Bits	Sys Reset	Access	Description
destruct_bypass_erase	0	0	R/O	Destructive Security Bypass In Progress

1:Operation is in progress.

FLC_BYPASS.superwipe_erase

Field	Bits	Sys Reset	Access	Description
superwipe_erase	1	0	R/O	Superwipe Erase In Progress

1:Operation is in progress.

FLC_BYPASS.destruct_bypass_complete

Field	Bits	Sys Reset	Access	Description
destruct_bypass_complete	2	0	R/O	Destructive Security Bypass Erase Complete

1:Operation has completed.

FLC_BYPASS.superwipe_complete

Field	Bits	Sys Reset	Access	Description
superwipe_complete	3	0	R/O	Superwipe Erase Complete

1:Operation has completed.

3.5.4.10 FLC_USER_OPTION

FLC_USER_OPTION

Sys Reset	Access	Description
00000000h	W/O	Used to set DSB Access code and Auto-Lock in info block

To set DESTRUCTIVE_BYPASS_CODE:

- Write your 32 bit DESTRUCTIVE_BYPASS_CODE to FLC_FDATA0
- Write 0x7502 to this FLC_USER_OPTION[15:0] register.
- Wait for write to flash to complete.
- The code will be saved into the INFO block array of the flash.
- Do this prior to setting AUTO_LOCK.
- This does not set AUTO_LOCK or any other security settings.

To set AUTO_LOCK:

- Write 0xAE01_10EA to FLC_FDATA0
- Write 0x7501 to this FLC_USER_OPTION[15:0] register.
- Wait for write to flash to complete.
- The AUTO_LOCK bit will be set in the INFO block array.
- Upon reset, the device will automatically lock (no debug).

3.5.4.11 FLC_CTRL2

FLC_CTRL2.flash_lve

Field	Bits	Sys Reset	Access	Description
flash_lve	0	0	R/W	Flash LVE Enable

Reserved for test purposes only. This bit should not be used by application firmware.

FLC_CTRL2.frc_fclk1_on

Field	Bits	Sys Reset	Access	Description
frc_fclk1_on	1	0	R/W	Force FCLK1 On

Reserved for test purposes only. This bit should not be used by application firmware.

FLC_CTRL2.bypass_ahb_fail

Field	Bits	Sys Reset	Access	Description
bypass_ahb_fail	15:8	0	R/W	AHB Fail Bypass

When using the AHB to write to flash, the fail flag will get set if a read access to the flash occurs before the write to the flash completes. Both AHB writes and AHB reads are arbitrated by the flash controller and the fail flag does not indicate a failed write or read. Thus, this fail flag can be ignored on subsequent writes or reads. If this field is 0 and the fail flag gets set, subsequent flash accesses will be ignored until the fail flag is cleared. Setting this field to 1 removes the fail flag from being evaluated and all subsequent flash accesses will be processed as normal.

Write 0x45 to this field to set the field to 1. Write 0x54 to this field to clear the field to 0.

This field is also used when modifying the values of other bits in this register; these operations do not change the value of this field.

3.5.4.12 FLC_INTFL1

FLC_INTFL1.sram_addr_wrapped

Field	Bits	Sys Reset	Access	Description
sram_addr_wrapped	0	0	W1C	SRAM Address Wrapped Interrupt Flag

This flag is set to 1 by hardware when an AHB read or write access occurs to an out-of-range SRAM location (due to trim configuration settings).

Write 1 to clear.

FLC_INTFL1.invalid_flash_addr

Field	Bits	Sys Reset	Access	Description
invalid_flash_addr	1	0	W1C	Invalid Flash Address Interrupt Flag

This flag is set to 1 by hardware when an AHB read or write access occurs to an out-of-range internal flash memory location (due to trim configuration settings).

Write 1 to clear.

FLC_INTFL1.flash_read_locked

Field	Bits	Sys Reset	Access	Description
flash_read_locked	2	0	W1C	Flash Read from Locked Area Interrupt Flag

This flag is set to 1 by hardware when an attempt is made to read from a page of internal flash memory that has been locked by setting a bit in the appropriate DISABLE_XR register.

Write 1 to clear.

FLC_INTFL1.trim_update_done

Field	Bits	Sys Reset	Access	Description
trim_update_done	3	0	W1C	Trim Update Complete Interrupt Flag

This flag is set to 1 by hardware when the FLC has completed its scan of the flash info block following power-up.

Write 1 to clear.

3.5.4.13 FLC_INTEN1

FLC_INTEN1.sram_addr_wrapped

Field	Bits	Sys Reset	Access	Description
sram_addr_wrapped	0	0	R/W	SRAM Address Wrapped Interrupt Enable/Disable

- 0: Interrupt disabled (default).
- 1: Enables FLC_INTFL1.sram_addr_wrapped as an FLC interrupt source.

FLC_INTEN1.invalid_flash_addr

Field	Bits	Sys Reset	Access	Description
invalid_flash_addr	1	0	R/W	Invalid Flash Address Interrupt Enable/Disable

- 0: Interrupt disabled (default).
- 1: Enables FLC_INTFL1.invalid_flash_addr as an FLC interrupt source.

FLC_INTEN1.flash_read_locked

Field	Bits	Sys Reset	Access	Description
flash_read_locked	2	0	R/W	Flash Read from Locked Area Interrupt Enable/Disable

- 0: Interrupt disabled (default).
- 1: Enables FLC_INTFL1.flash_read_locked as an FLC interrupt source.

FLC_INTEN1.trim_update_done

Field	Bits	Sys Reset	Access	Description
trim_update_done	3	0	R/W	Trim Update Complete Interrupt Enable/Disable

- 0: Interrupt disabled (default).
- 1: Enables FLC_INTFL1.trim_update_done as an FLC interrupt source.

3.5.4.14 FLC_BL_CTRL

FLC_BL_CTRL

Sys Reset	Access	Description
00000000h	R/W	Bootloader Control Register

Internal use only. Contents of this register should not be modified by application firmware.

3.5.4.15 FLC_TWK

FLC_TWK

Sys Reset	Access	Description
00000000h	R/W	FLC TWK Cycle Count

Internal use only. Contents of this register should not be modified by application firmware.

3.5.4.16 FLC_SLM

FLC_SLM

Sys Reset	Access	Description
00000000h	R/W	Sleep Mode Register

Internal use only. Contents of this register should not be modified by application firmware.

3.5.5 Registers (ICC)

Address	Register	Access	Description	Reset By
0x40003000	ICC_ID	R/O	Cache ID Register (INTERNAL USE ONLY)	Sys
0x40003004	ICC_MEM_CFG	R/O	Memory Configuration Register	Sys
0x40003100	ICC_CTRL_STAT	***	Control and Status	Sys
0x40003700	ICC_INVDT_ALL	W/O	Invalidate (Clear) Cache Control	Sys

3.5.5.1 ICC_ID

ICC_ID.rtl_version

Field	Bits	Sys Reset	Access	Description
rtl_version	5:0	xxxxxx	R/O	Version

Internal/test use only; not for use by application software.

ICC_ID.part_num

Field	Bits	Sys Reset	Access	Description
part_num	9:6	xxxx	R/O	Number ID

Internal/test use only; not for use by application software.

ICC_ID.cache_id

Field	Bits	Sys Reset	Access	Description
cache_id	15:10	xxxxxx	R/O	Cache ID

Internal/test use only; not for use by application software.

3.5.5.2 ICC_MEM_CFG

ICC_MEM_CFG.cache_size

Field	Bits	Sys Reset	Access	Description
cache_size	15:0	0008h	R/O	Instruction Cache Size

Size of the instruction cache memory (in KB).

For the **MAX32620**, this value is 8KB.

ICC_MEM_CFG.main_memory_size

Field	Bits	Sys Reset	Access	Description
main_memory_size	31:16	0040h	R/O	Internal Flash Memory Size

Size of the main internal flash memory (in KB, divided by 32).

For the **MAX32620**, this value is 2048KB (2MB).

3.5.5.3 ICC_CTRL_STAT

ICC_CTRL_STAT.enable

Field	Bits	Sys Reset	Access	Description
enable	0	0	R/W	Cache Enable

- 0: Instruction cache is disabled (bypass mode)
- 1: Instruction cache is enabled

ICC_CTRL_STAT.ready

Field	Bits	Sys Reset	Access	Description
ready	16	0	R/O	Cache Ready Status

- 0: Cache is invalidated/in a reset state
- 1: Cache is active and ready for use

3.5.5.4 ICC_INVDT_ALL

ICC_INVDT_ALL

Sys Reset	Access	Description
00000000h	W/O	Invalidate (Clear) Cache Control

Writing any value to this register triggers a cache invalidate operation. The Cache Ready Status bit will indicate when this has completed.

All reads return 0.

3.5.6 Registers (TRIM)

Address	Register	Access	Description	Reset By
0x4000102C	TRIM_REG11_ADC_TRIM0	R/W	Shadow Trim for ADC R0	Sys
0x40001030	TRIM_REG12_ADC_TRIM1	R/W	Shadow Trim for ADC R1	Sys
0x40001034	TRIM_FOR_PWR_REG5	R/O	Shadow Trim for PWRSEQ Register REG5	Sys
0x40001038	TRIM_FOR_PWR_REG6	R/O	Shadow Trim for PWRSEQ Register REG6	Sys

3.5.6.1 TRIM_REG11_ADC_TRIM0

TRIM_REG11_ADC_TRIM0.adctrim_x0r0

Field	Bits	Sys Reset	Access	Description
adctrim_x0r0	9:0	special	R/W	Gain Error Trim X0 R0

This field is automatically initalized from the flash info block by hardware.

This field should not be modified by the user except in the case when an external ADC reference is being used. In this case, the value of this field should be changed to 0x200.

TRIM_REG11_ADC_TRIM0.adctrim_x1r0

Field	Bits	Sys Reset	Access	Description
adctrim_x1r0	25:16	special	R/W	Gain Error Trim X1 R0

This field is automatically initalized from the flash info block by hardware.

This field should not be modified by the user except in the case when an external ADC reference is being used. In this case, the value of this field should be changed to 0x200.

3.5.6.2 TRIM_REG12_ADC_TRIM1

TRIM_REG12_ADC_TRIM1.adctrim_x0r1

Field	Bits	Sys Reset	Access	Description
adctrim_x0r1	9:0	special	R/W	Gain Error Trim X0 R1

This field is automatically initalized from the flash info block by hardware.

This field should not be modified by the user except in the case when an external ADC reference is being used. In this case, the value of this field should be changed to 0x200.

TRIM_REG12_ADC_TRIM1.adctrim_x1r1

Field	Bits	Sys Reset	Access	Description
adctrim_x1r1	25:16	special	R/W	Gain Error Trim X1 R1

This field is automatically initalized from the flash info block by hardware.

This field should not be modified by the user except in the case when an external ADC reference is being used. In this case, the value of this field should be changed to 0x200.

TRIM_REG12_ADC_TRIM1.adctrim_dc

Field	Bits	Sys Reset	Access	Description
adctrim_dc	31:28	special	R/W	Offset Error Trim

This field is automatically initalized from the flash info block by hardware.

This field should not be modified by the user. For proper ADC operation, this field should remain at its default value.

3.5.6.3 TRIM_FOR_PWR_REG5

TRIM_FOR_PWR_REG5

Sys Reset	Access	Description
special	R/O	Shadow Trim for PWRSEQ Register REG5

This register is automatically initialized from the flash info block by hardware.

Following boot, the contents of this register must be copied by application firmware to the power sequencer register PWRSEQ_REG5.

3.5.6.4 TRIM_FOR_PWR_REG6

TRIM_FOR_PWR_REG6

Sys Reset	Access	Description
special	R/O	Shadow Trim for PWRSEQ Register REG6

This register is automatically initialized from the flash info block by hardware.

Following boot, the contents of this register must be copied by application firmware to the power sequencer register PWRSEQ_REG6.

4 System Configuration and Management

4.1 Power Ecosystem and Operating Modes

4.1.1 Power Ecosystem

The **MAX32620** has multiple operating modes with many user configurable options, offering significant flexibility in total power consumption. These options are stored in configuration registers and are continuously powered across all modes of operation. These registers dictate which analog and digital peripherals are enabled during low power modes. The V_{RTC} power supply provides power to the configuration registers during loss-of-power events on the main digital and analog supplies.

The **MAX32620** supports four power modes: LP0:STOP, LP1:STANDBY, LP2:PMU, LP3:RUN. The Power State Diagram shows a state diagram of these power modes.

In the low power modes LP0:STOP and LP1:STANDBY, the **MAX32620** is controlled by the Power Sequencer. In the LP2:PMU mode, the **MAX32620** is controlled by the PMU. In the LP3:RUN mode, it is controlled by the CPU.

When a wakeup event is detected, the **MAX32620** exits the low power mode (LP0:STOP or LP1:STANDBY) and enters the LP3:RUN mode.

Note Some power modes can only be reached via transitions from specific modes. Further transitioning information is found in the power mode sections below. The following is a typical power state transition sequence: (LP0:STOP or LP1:STANDBY) → LP3:RUN → LP2:PMU → LP3:RUN → (LP0:STOP or LP1:STANDBY).

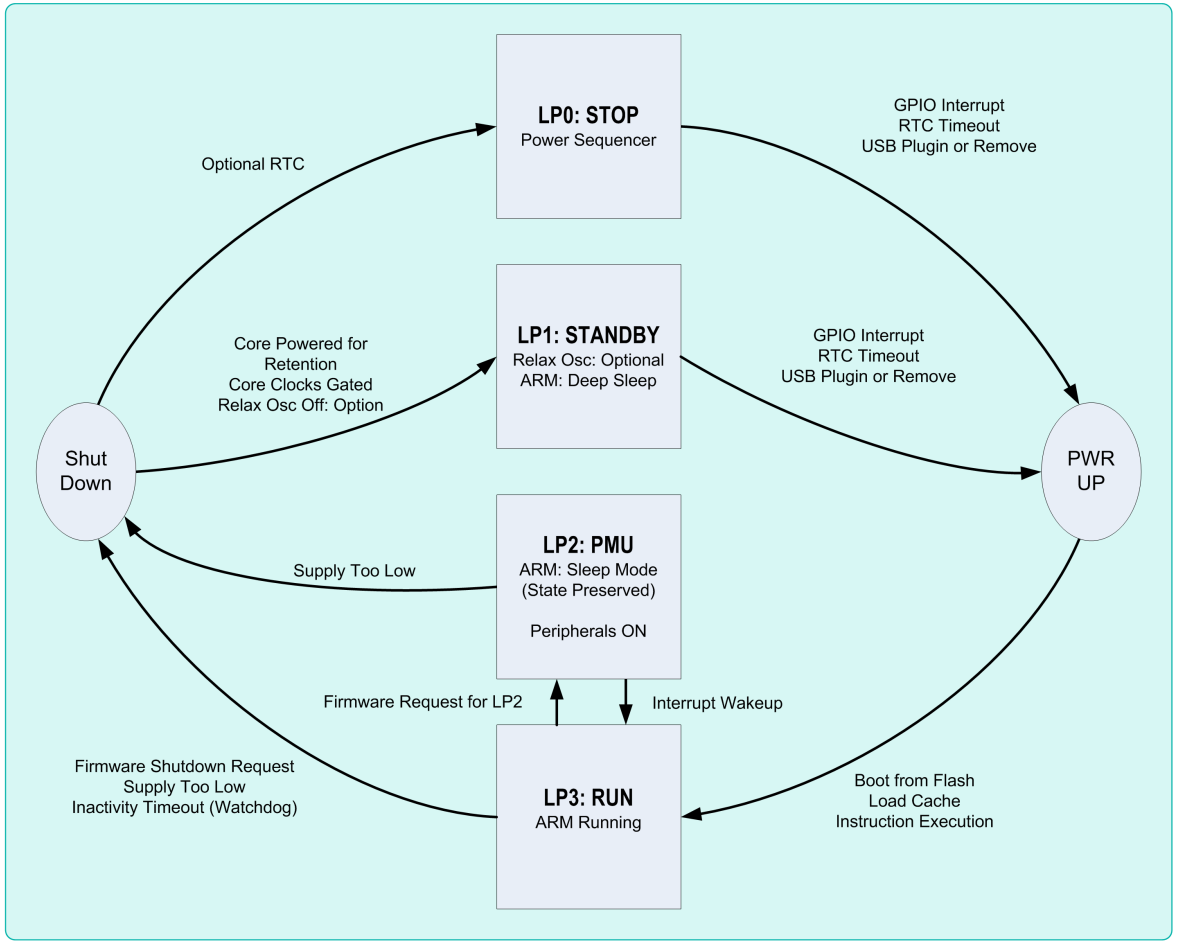

Figure 4.1: Power State Diagram

4.1.2 Power Modes

Note Low Power Modes LP0:STOP and LP1:STANDBY can only transition to/from the LP3:RUN mode. In order to enter LP2:PMU from one of these modes, the device must first enter LP3:RUN.

4.1.2.1 Low Power Mode 0 (LP0:STOP)

LP0:STOP is the lowest power mode supported by the **MAX32620**.

This state is useful when the **MAX32620** is not expected to perform CPU-level processing tasks for an extended period of time and a slightly slower wakeup period is tolerable. During this state, all power to most digital and analog circuitry on the device is shut off, including power to the ARM core, the internal SRAM, and all digital and analog peripherals except for those discussed below.

The only current draw sources in this state are the power sequencer, the RTC clock (if enabled), key data retention registers, and POR/failsafe circuitry.

4.1.2.2 Low Power Mode 1 (LP1:STANDBY)

LP1:STANDBY is a core data retention mode which supports fast wakeup time while maintaining ultra-low power. In this mode, all clocks are gated off and the core logic is in a static, low-power state. The CPU is in deep sleep in this mode, and all data is preserved.

In order to achieve the lowest possible power consumption during LP1:STANDBY, it is recommended to turn off all unused analog circuitry.

4.1.2.3 Low Power Mode 2 (LP2:Peripheral Management Unit)

The PMU is in control of the system when using LP2:PMU. During this mode, the CPU of the **MAX32620** is in sleep mode. This mode enables the lowest noise floor for analog measurements and reduced operating power for peripherals.

The PMU can direct data movement via programmable op codes for:

- Peripheral to Memory
- Memory to Peripheral
- Analog to Memory
- Synchronization of analog measurements
- Control and synchronization of pulse train signals and events

4.1.2.4 Low Power Mode 3 (LP3:RUN)

In the LP3:RUN mode, the CPU is running application code. During this state, the CPU and all digital and analog peripherals are fully powered and awake. The clocks to each peripheral are gated dynamically, which results in power savings for those peripherals which are not in use.

4.1.2.5 Wakeup Events from LP0:STOP and LP1:STANDBY

The following events can wake up the **MAX32620** from the LP0:STOP or LP1:STANDBY modes.

- RTC timer interrupt
- GPIO level detection
- USB connection/disconnection

Each of these events is configurable and must be enabled by the firmware.

Note Certain wakeup events can be masked out by writing to the PWRSEQ_MSK_FLAGS register.

4.1.3 Power State Matrix Control Options

The Power State Diagram shows the four major power states and how control is handled in the **MAX32620**.

The figure illustrates:

- Hardware-controlled powering of circuit blocks
- Firmware-controllable power options
- Firmware-controllable clock gating

Note The power manager will power the minimal amount of circuitry necessary to achieve functionality of each mode. To achieve the lowest optimized power solution, the user must fully analyze the use case and choose the appropriate power and clock gating options.

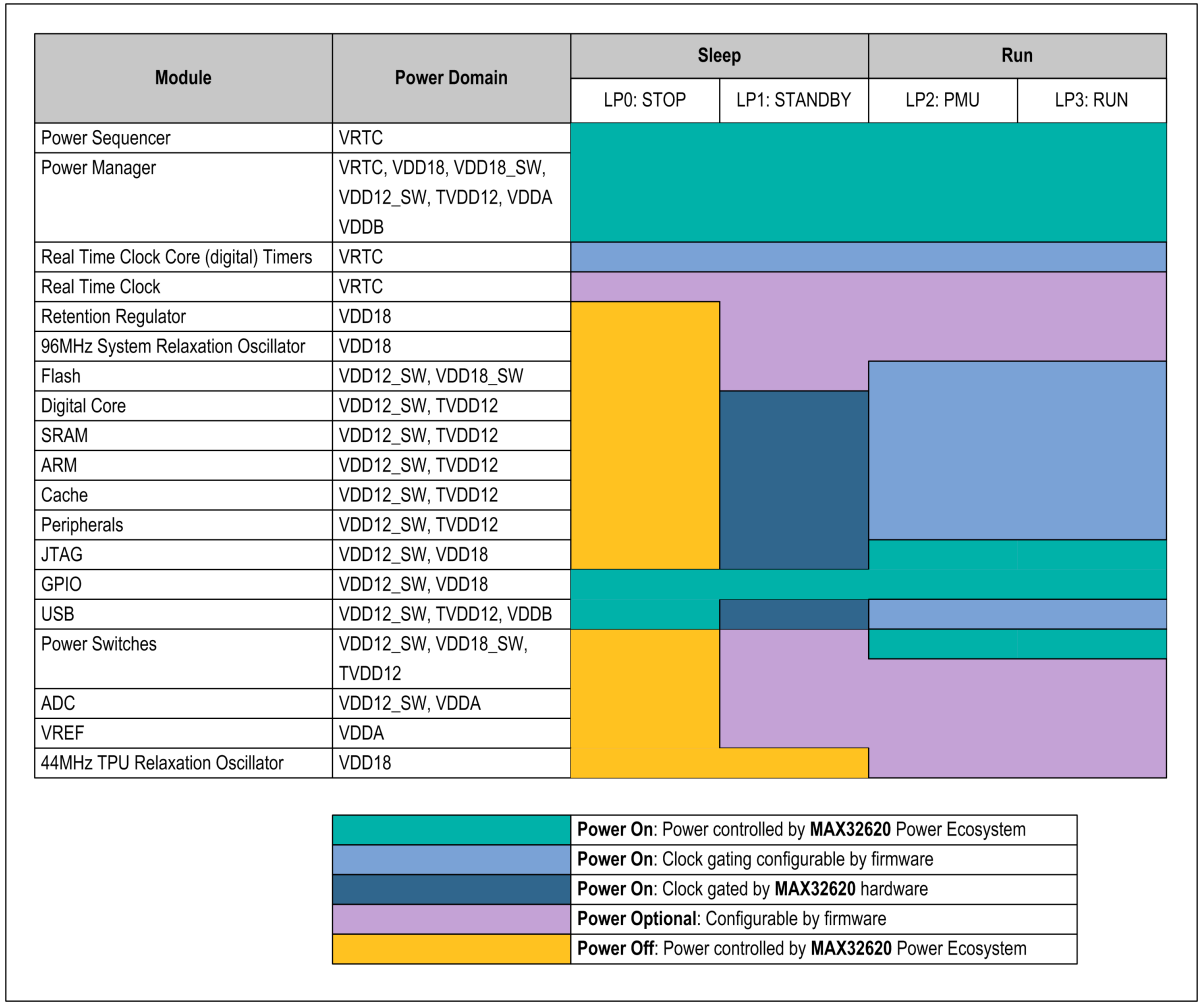

Figure 4.2: Power and Clock Gating Options

4.1.4 Power Ecosystem

The **MAX32620** has multiple power domains that are controlled by the power management block. This includes the Power Sequencer, Power Manager, Real Time Clock (RTC), SVM, ADC, and GPIO. The configuration registers for the Power Manager are within the battery backed V_{RTC} domain. The registers are configurable by firmware and dictate what analog and digital peripherals are enabled while in each of the operating modes.

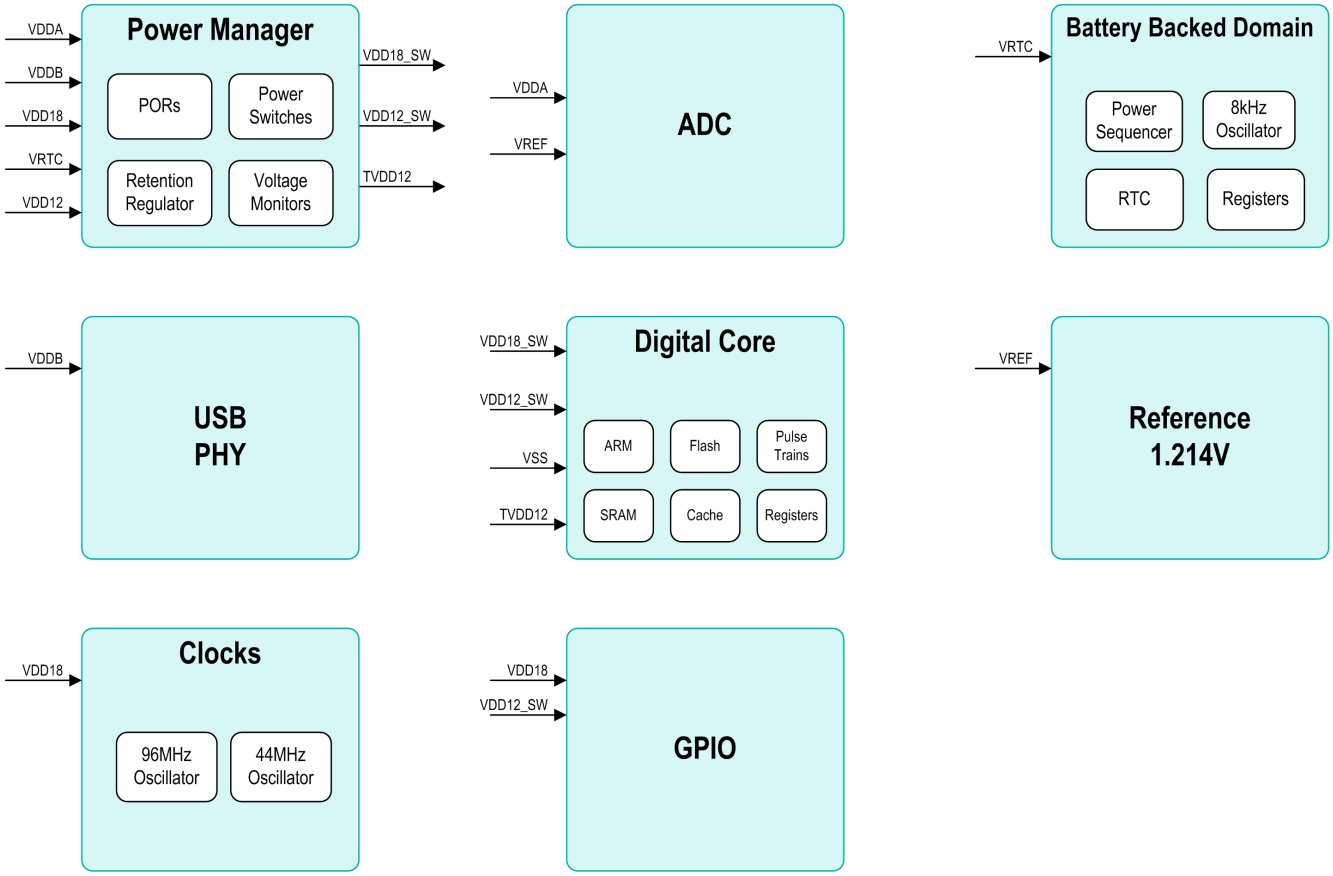

Figure 4.3: Power Ecosystem

4.1.5 Power Manager

In addition to power distribution and management, the Power Manager monitors power levels using Supply Voltage Monitor (SVM) blocks. These are used to trigger warning and/or reset events, depending on the supply and the voltage level detected.

4.1.6 Power Sequencer

The Power Sequencer controls the **MAX32620** during Power Modes. During exit from LP0:STOP or LP1:STANDBY, the Power Sequencer transfers control to the system manager. The **MAX32620** then enters LP3:RUN. One of the primary functions of the Power Sequencer is to ensure that power, clock, and reset signals are stable prior to entry into LP3:RUN mode.

During LP0:STOP and LP1:STANDBY, wakeup interrupts are continuously monitored. Once an interrupt event occurs, the Power Sequencer automatically enables SVMs, firmware-selected peripherals, and the main system high-frequency relaxation oscillator.

When the Power Sequencer determines that all power, clock, and reset signals are valid, the clock gating for the CPU and digital core is released, and the **MAX32620** is allowed to enter LP3:RUN mode.

4.1.6.1 Power Mode Transitioning to Low Power Modes

To take full advantage of the low power modes of operation in the **MAX32620**, application firmware will need to spend as much time as possible in either the LP0:STOP or LP1:STANDBY modes.

Prior to entering those modes, it is extremely important to set up any wakeup interrupt(s) that will be used to exit LP0:STOP or LP1:STANDBY and return to active mode.

4.1.7 Registers (PWRMAN)

Address	Register	Access	Description	Reset By
0x40000800	PWRMAN_PWR_RST_CTRL	***	Power Reset Control and Status	Sys
0x40000804	PWRMAN_INTFL	W1C	Interrupt Flags	Sys
0x40000808	PWRMAN_INTEN	R/W	Interrupt Enable/Disable Controls	Sys
0x4000080C	PWRMAN_SVM_EVENTS	R/O	SVM Event Status Flags (read-only)	
0x40000810	PWRMAN_WUD_CTRL	R/W	Wake-Up Detect Control	Sys
0x40000814	PWRMAN_WUD_PULSE0	W/O	WUD Pulse To Mode Bit 0	Sys
0x40000818	PWRMAN_WUD_PULSE1	W/O	WUD Pulse To Mode Bit 1	Sys
0x4000081C	PWRMAN_WUD_SEEN0	R/O	Wake-up Detect Status for P0/P1/P2/P3	Sys
0x40000820	PWRMAN_WUD_SEEN1	R/O	Wake-up Detect Status for P4/P5/P6/P7	Sys
0x40000838	PWRMAN_DIE_TYPE	R/O	Die Type ID Register	Sys
0x4000083C	PWRMAN_BASE_PART_NUM	R/O	Base Part Number	
0x40000840	PWRMAN_MASK_ID0	R/O	Mask ID Register 0	
0x40000844	PWRMAN_MASK_ID1	***	Mask ID Register 1	Sys
0x40000848	PWRMAN_PERIPHERAL_RESET	R/W	Peripheral Reset Control Register	Sys

4.1.7.1 PWRMAN_PWR_RST_CTRL

PWRMAN_PWR_RST_CTRL.afe_powered

Field	Bits	Sys Reset	Access	Description
afe_powered	2	0	R/W	AFE Powered

- 0:Entire AFE is powered off
- 1:AFE is powered on globally; individual AFE controls may be used to power sub-features of AFE on and off as needed.

PWRMAN_PWR_RST_CTRL.io_active

Field	Bits	Sys Reset	Access	Description
io_active	3	1	R/W	I/O Active

- 0: Puts all I/O (GPIO-only) pins in the lowest power state.
- 1: All I/O pins are powered on normally.

PWRMAN_PWR_RST_CTRL.usb_powered

Field	Bits	Sys Reset	Access	Description
usb_powered	4	0	R/W	USB Powered

1: Powers on the USB block.

PWRMAN_PWR_RST_CTRL.pullups_enabled

Field	Bits	Sys Reset	Access	Description
pullups_enabled	5	1	R/W	Static Pullups Enabled

1: Enables static pullups on dedicated I/O pins.

PWRMAN_PWR_RST_CTRL.firmware_reset

Field	Bits	Sys Reset	Access	Description
firmware_reset	8	0	R/W	Firmware Initiated Reset

Initiates a system reset when set to 1. This bit is self-clearing.

PWRMAN_PWR_RST_CTRL.arm_lockup_reset

Field	Bits	Sys Reset	Access	Description
arm_lockup_reset	9	0	R/W	ARM Lockup Reset

If this bit is set to 1, a system reset will be automatically triggered when the ARM core asserts its lockup state output signal.

PWRMAN_PWR_RST_CTRL.tamper_detect

Field	Bits	Sys Reset	Access	Description
tamper_detect	16	special	R/O	Reset Caused By - Tamper Detect

Note Tamper detection does not cause assertion of reset on the **MAX32620**.

PWRMAN_PWR_RST_CTRL.arm_lockup

Field	Bits	Sys Reset	Access	Description
arm_lockup	17	special	R/O	Reset Caused By - ARM Lockup

PWRMAN_PWR_RST_CTRL.fw_command_arm

Field	Bits	Sys Reset	Access	Description
fw_command_arm	18	special	R/O	Reset Caused By - Firmware Commanded Reset (ARM Core)

PWRMAN_PWR_RST_CTRL.watchdog_timeout

Field	Bits	Sys Reset	Access	Description
watchdog_timeout	19	special	R/O	Reset Caused By - Watchdog Timeout

PWRMAN_PWR_RST_CTRL.fw_command_sysman

Field	Bits	Sys Reset	Access	Description
fw_command_sysman	20	special	R/O	Reset Caused By - Firmware Commanded Reset (SysMan)

PWRMAN_PWR_RST_CTRL.srstn_assertion

Field	Bits	Sys Reset	Access	Description
srstn_assertion	21	special	R/O	Reset Caused By - External System Reset

PWRMAN_PWR_RST_CTRL.por

Field	Bits	Sys Reset	Access	Description
por	22	special	R/O	Reset Caused By - Power On Reset (POR)

PWRMAN_PWR_RST_CTRL.low_power_mode

Field	Bits	Sys Reset	Access	Description
low_power_mode	31	0	R/W	Power Manager Dynamic Clock Gating Enable

1: Enables dynamic clock gating for pwrman functions.

4.1.7.2 PWRMAN_INTFL

PWRMAN_INTFL.v1_2_warning

Field	Bits	Sys Reset	Access	Description
v1_2_warning	0	0	W1C	1.2V Warning Monitor Int Flag

Write 1 to clear.

Set to 1 by hardware when the associated SVM event monitor detects the monitored condition.

PWRMAN_INTFL.v1_8_warning

Field	Bits	Sys Reset	Access	Description
v1_8_warning	1	0	W1C	1.8V Warning Monitor Int Flag

Write 1 to clear.

Set to 1 by hardware when the associated SVM event monitor detects the monitored condition.

PWRMAN_INTFL.rtc_warning

Field	Bits	Sys Reset	Access	Description
rtc_warning	2	0	W1C	RTC Warning Monitor Int Flag

Write 1 to clear.

Set to 1 by hardware when the associated SVM event monitor detects the monitored condition.

PWRMAN_INTFL.vdda_warning

Field	Bits	Sys Reset	Access	Description
vdda_warning	3	0	W1C	VDDA Warning Monitor Int Flag

Write 1 to clear.

Set to 1 by hardware when the associated SVM event monitor detects the monitored condition.

PWRMAN_INTFL.vddb_warning

Field	Bits	Sys Reset	Access	Description
vddb_warning	4	0	W1C	VDDB Warning Monitor Int Flag

Write 1 to clear.

Set to 1 by hardware when the associated SVM event monitor detects the monitored condition.

4.1.7.3 PWRMAN_INTEN

PWRMAN_INTEN.v1_2_warning

Field	Bits	Sys Reset	Access	Description
v1_2_warning	0	0	R/W	1.2V Warning Monitor Int Enable

0:Int disabled; 1:Interrupt enabled for the associated SVM monitor event.

PWRMAN_INTEN.v1_8_warning

Field	Bits	Sys Reset	Access	Description
v1_8_warning	1	0	R/W	1.8V Warning Monitor Int Enable

0:Int disabled; 1:Interrupt enabled for the associated SVM monitor event.

PWRMAN_INTEN.rtc_warning

Field	Bits	Sys Reset	Access	Description
rtc_warning	2	0	R/W	RTC Warning Monitor Int Enable

0:Int disabled; 1:Interrupt enabled for the associated SVM monitor event.

PWRMAN_INTEN.vdda_warning

Field	Bits	Sys Reset	Access	Description
vdda_warning	3	0	R/W	VDDA Warning Monitor Int Enable

0:Int disabled; 1:Interrupt enabled for the associated SVM monitor event.

PWRMAN_INTEN.vddb_warning

Field	Bits	Sys Reset	Access	Description
vddb_warning	4	0	R/W	VDDB Warning Monitor Int Enable

0:Int disabled; 1:Interrupt enabled for the associated SVM monitor event.

4.1.7.4 PWRMAN_SVM_EVENTS

PWRMAN_SVM_EVENTS.v1_2_warning

Field	Bits	Sys Reset	Access	Description
v1_2_warning	0	n/a	R/O	1.2V Warning Monitor Event Input

Current state of the associated SVM event input.

PWRMAN_SVM_EVENTS.v1_8_warning

Field	Bits	Sys Reset	Access	Description
v1_8_warning	1	n/a	R/O	1.8V Warning Monitor Event Input

Current state of the associated SVM event input.

PWRMAN_SVM_EVENTS.rtc_warning

Field	Bits	Sys Reset	Access	Description
rtc_warning	2	n/a	R/O	RTC Warning Monitor Event Input

Current state of the associated SVM event input.

PWRMAN_SVM_EVENTS.vdda_warning

Field	Bits	Sys Reset	Access	Description
vdda_warning	3	n/a	R/O	VDDA Warning Monitor Event Input

Current state of the associated SVM event input.

PWRMAN_SVM_EVENTS.vddb_warning

Field	Bits	Sys Reset	Access	Description
vddb_warning	4	n/a	R/O	VDDB Warning Monitor Event Input

Current state of the associated SVM event input.

4.1.7.5 PWRMAN_WUD_CTRL

PWRMAN_WUD_CTRL.pad_select

Field	Bits	Sys Reset	Access	Description
pad_select	5:0	0	R/W	Wake-Up Pad Select

Selects which pad to modify WUD/Weak latch states.

Pads are numbered from 0-48, where 0-7 corresponds to P0.0-P0.7, 8-15 corresponds to P1.0-P1.7, and so on through 48=P6.0.

PWRMAN_WUD_CTRL.pad_mode

Field	Bits	Sys Reset	Access	Description
pad_mode	9:8	00b	R/W	Wake-Up Pad Signal Mode

Defines WUD signal to be sent to selected pad.

- 0 = Clear/Activate WUD
- 1 = Set WUD Act Hi/Set WUD Act Lo
- 2 = Set Weak Hi/Set Weak Lo
- 3 = No pad state change

PWRMAN_WUD_CTRL.clear_all

Field	Bits	Sys Reset	Access	Description
clear_all	12	0	R/W	Clear All WUD Pad States

When set forces all pads into Clr WUD/Weak state until cleared.

PWRMAN_WUD_CTRL.ctrl_enable

Field	Bits	Sys Reset	Access	Description
ctrl_enable	16	0	R/W	Enable WUD Control Modification

Set to 1 to enable control of WUD pad logic. Should be set to 0 (to disable modifications) when not actively configuring WUD pad modes.

4.1.7.6 PWRMAN_WUD_PULSE0

PWRMAN_WUD_PULSE0

Sys Reset	Access	Description
n/a	W/O	WUD Pulse To Mode Bit 0

Writing to this register issues a pulse to the selected WUD pad mode[0] for one clock.

The effect on the pad behavior depends on the Wake-Up Pad Signal Mode as set in WUD_CTRL.

- 0 = Clr WUD/Weak
- 1 = Set WUD Act Hi
- 2 = Set Weak Hi
- 3 = No pad state change

4.1.7.7 PWRMAN_WUD_PULSE1

PWRMAN_WUD_PULSE1

Sys Reset	Access	Description
n/a	W/O	WUD Pulse To Mode Bit 1

Writing to this register issues a pulse to the selected WUD pad mode[1] for one clock.

The effect on the pad behavior depends on the Wake-Up Pad Signal Mode as set in WUD_CTRL.

- 0 = WUD Activate
- 1 = Set WUD Act Lo
- 2 = Set Weak Lo
- 3 = No pad state change;

4.1.7.8 PWRMAN_WUD_SEEN0

PWRMAN_WUD_SEEN0.[gpio0, gpio1, gpio2, gpio3, gpio4, gpio5, gpio6, gpio7]

Field	Bits	Sys Reset	Access	Description
gpio0	0	0	R/O	Wake-Up Detect Status for P0.0
gpio1	1	0	R/O	Wake-Up Detect Status for P0.1
gpio2	2	0	R/O	Wake-Up Detect Status for P0.2
gpio3	3	0	R/O	Wake-Up Detect Status for P0.3
gpio4	4	0	R/O	Wake-Up Detect Status for P0.4
gpio5	5	0	R/O	Wake-Up Detect Status for P0.5
gpio6	6	0	R/O	Wake-Up Detect Status for P0.6
gpio7	7	0	R/O	Wake-Up Detect Status for P0.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN0.[gpio8, gpio9, gpio10, gpio11, gpio12, gpio13, gpio14, gpio15]

Field	Bits	Sys Reset	Access	Description
gpio8	8	0	R/O	Wake-Up Detect Status for P1.0
gpio9	9	0	R/O	Wake-Up Detect Status for P1.1
gpio10	10	0	R/O	Wake-Up Detect Status for P1.2
gpio11	11	0	R/O	Wake-Up Detect Status for P1.3
gpio12	12	0	R/O	Wake-Up Detect Status for P1.4
gpio13	13	0	R/O	Wake-Up Detect Status for P1.5
gpio14	14	0	R/O	Wake-Up Detect Status for P1.6
gpio15	15	0	R/O	Wake-Up Detect Status for P1.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN0.[gpio16, gpio17, gpio18, gpio19, gpio20, gpio21, gpio22, gpio23]

Field	Bits	Sys Reset	Access	Description
gpio16	16	0	R/O	Wake-Up Detect Status for P2.0
gpio17	17	0	R/O	Wake-Up Detect Status for P2.1
gpio18	18	0	R/O	Wake-Up Detect Status for P2.2
gpio19	19	0	R/O	Wake-Up Detect Status for P2.3
gpio20	20	0	R/O	Wake-Up Detect Status for P2.4
gpio21	21	0	R/O	Wake-Up Detect Status for P2.5
gpio22	22	0	R/O	Wake-Up Detect Status for P2.6
gpio23	23	0	R/O	Wake-Up Detect Status for P2.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN0.[gpio24, gpio25, gpio26, gpio27, gpio28, gpio29, gpio30, gpio31]

Field	Bits	Sys Reset	Access	Description
gpio24	24	0	R/O	Wake-Up Detect Status for P3.0
gpio25	25	0	R/O	Wake-Up Detect Status for P3.1
gpio26	26	0	R/O	Wake-Up Detect Status for P3.2
gpio27	27	0	R/O	Wake-Up Detect Status for P3.3
gpio28	28	0	R/O	Wake-Up Detect Status for P3.4
gpio29	29	0	R/O	Wake-Up Detect Status for P3.5
gpio30	30	0	R/O	Wake-Up Detect Status for P3.6
gpio31	31	0	R/O	Wake-Up Detect Status for P3.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

4.1.7.9 PWRMAN_WUD_SEEN1

PWRMAN_WUD_SEEN1.[gpio32, gpio33, gpio34, gpio35, gpio36, gpio37, gpio38, gpio39]

Field	Bits	Sys Reset	Access	Description
gpio32	0	0	R/O	Wake-Up Detect Status for P4.0
gpio33	1	0	R/O	Wake-Up Detect Status for P4.1
gpio34	2	0	R/O	Wake-Up Detect Status for P4.2
gpio35	3	0	R/O	Wake-Up Detect Status for P4.3
gpio36	4	0	R/O	Wake-Up Detect Status for P4.4
gpio37	5	0	R/O	Wake-Up Detect Status for P4.5
gpio38	6	0	R/O	Wake-Up Detect Status for P4.6
gpio39	7	0	R/O	Wake-Up Detect Status for P4.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN1.[gpio40, gpio41, gpio42, gpio43, gpio44, gpio45, gpio46, gpio47]

Field	Bits	Sys Reset	Access	Description
gpio40	8	0	R/O	Wake-Up Detect Status for P5.0
gpio41	9	0	R/O	Wake-Up Detect Status for P5.1
gpio42	10	0	R/O	Wake-Up Detect Status for P5.2
gpio43	11	0	R/O	Wake-Up Detect Status for P5.3
gpio44	12	0	R/O	Wake-Up Detect Status for P5.4
gpio45	13	0	R/O	Wake-Up Detect Status for P5.5
gpio46	14	0	R/O	Wake-Up Detect Status for P5.6
gpio47	15	0	R/O	Wake-Up Detect Status for P5.7

Displays wakeup detection status of the 8 listed GPIO pads,

- bit 0: Px.0
- bit 1: Px.1
- bit 2: Px.2
- bit 3: Px.3
- bit 4: Px.4
- bit 5: Px.5
- bit 6: Px.6
- bit 7: Px.7 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

PWRMAN_WUD_SEEN1.gpio48

Field	Bits	Sys Reset	Access	Description
gpio48	16	0	R/O	Wake-Up Detect Status for P6.0

Displays wakeup detection status of the GPIO pad,

- bit 0: P6.0 where a '1' bit represents a wakeup condition detected.

Bits for any I/O pads that are not in Wakeup Detect Mode will always read 0.

4.1.7.10 PWRMAN_DIE_TYPE

PWRMAN_DIE_TYPE

Sys Reset	Access	Description
n/a	R/O	Die Type ID Register

Always reads 4D453032h (ASCII 'ME02').

4.1.7.11 PWRMAN_BASE_PART_NUM

PWRMAN_BASE_PART_NUM.base_part_number

Field	Bits	Sys Reset	Access	Description
base_part_number	15:0	n/a	R/O	Base Part Number

Always reads 7F67h (32615 decimal).

4.1.7.12 PWRMAN_MASK_ID0

PWRMAN_MASK_ID0.revision_id

Field	Bits	Sys Reset	Access	Description
revision_id	3:0	n/a	R/O	Revision ID

Device revision information.

This field is intended for internal test purposes only.

PWRMAN_MASK_ID0.mask_id

Field	Bits	Sys Reset	Access	Description
mask_id	31:4	n/a	R/O	Mask ID[27:0]

Mask identification information - low 28 bits.

This field is intended for internal test purposes only.

4.1.7.13 PWRMAN_MASK_ID1

PWRMAN_MASK_ID1.mask_id

Field	Bits	Sys Reset	Access	Description
mask_id	30:0	n/a	R/O	Mask ID[58:28]

Mask identification information - high 31 bits.

This field is intended for internal test purposes only.

PWRMAN_MASK_ID1.mask_id_enable

Field	Bits	Sys Reset	Access	Description
mask_id_enable	31	0	R/W	Enable Mask ID

Must set to 1 in order for the Mask ID fields to be readable.

This field is intended for internal test purposes only.

4.1.7.14 PWRMAN_PERIPHERAL_RESET

PWRMAN_PERIPHERAL_RESET.ssb

Field	Bits	Sys Reset	Access	Description
ssb	0	0	R/W	Reset SSB

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.spix

Field	Bits	Sys Reset	Access	Description
spix	1	0	R/W	Reset SPI XIP

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.pmu

Field	Bits	Sys Reset	Access	Description
pmu	2	0	R/W	Reset PMU

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.usb

Field	Bits	Sys Reset	Access	Description
usb	3	0	R/W	Reset USB

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.crc

Field	Bits	Sys Reset	Access	Description
crc	4	0	R/W	Reset CRC

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.tpu

Field	Bits	Sys Reset	Access	Description
tpu	5	0	R/W	Reset TPU

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.watchdog0

Field	Bits	Sys Reset	Access	Description
watchdog0	6	0	R/W	Reset Watchdog Timer 0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.gpio

Field	Bits	Sys Reset	Access	Description
gpio	7	0	R/W	Reset GPIO

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.timer0

Field	Bits	Sys Reset	Access	Description
timer0	8	0	R/W	Reset Timer/Counter Module 0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.timer1

Field	Bits	Sys Reset	Access	Description
timer1	9	0	R/W	Reset Timer/Counter Module 1

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.timer2

Field	Bits	Sys Reset	Access	Description
timer2	10	0	R/W	Reset Timer/Counter Module 2

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.timer3

Field	Bits	Sys Reset	Access	Description
timer3	11	0	R/W	Reset Timer/Counter Module 3

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.timer4

Field	Bits	Sys Reset	Access	Description
timer4	12	0	R/W	Reset Timer/Counter Module 4

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.timer5

Field	Bits	Sys Reset	Access	Description
timer5	13	0	R/W	Reset Timer/Counter Module 5

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.pulse_train

Field	Bits	Sys Reset	Access	Description
pulse_train	14	0	R/W	Reset All Pulse Trains

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.uart0

Field	Bits	Sys Reset	Access	Description
uart0	15	0	R/W	Reset UART 0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.uart1

Field	Bits	Sys Reset	Access	Description
uart1	16	0	R/W	Reset UART 1

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.uart2

Field	Bits	Sys Reset	Access	Description
uart2	17	0	R/W	Reset UART 2

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.uart3

Field	Bits	Sys Reset	Access	Description
uart3	18	0	R/W	Reset UART 3

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.i2cm0

Field	Bits	Sys Reset	Access	Description
i2cm0	19	0	R/W	Reset I2C Master 0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.i2cm1

Field	Bits	Sys Reset	Access	Description
i2cm1	20	0	R/W	Reset I2C Master 1

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.i2cm2

Field	Bits	Sys Reset	Access	Description
i2cm2	21	0	R/W	Reset I2C Master 2

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.i2cs

Field	Bits	Sys Reset	Access	Description
i2cs	22	0	R/W	Reset I2C Slave

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.spim0

Field	Bits	Sys Reset	Access	Description
spim0	23	0	R/W	Reset SPI Master 0

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.spim1

Field	Bits	Sys Reset	Access	Description
spim1	24	0	R/W	Reset SPI Master 1

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.spim2

Field	Bits	Sys Reset	Access	Description
spim2	25	0	R/W	Reset SPI Master 2

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.spib

Field	Bits	Sys Reset	Access	Description
spib	26	0	R/W	Reset SPI Bridge

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.owm

Field	Bits	Sys Reset	Access	Description
owm	27	0	R/W	Reset 1-Wire Master

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

PWRMAN_PERIPHERAL_RESET.adc

Field	Bits	Sys Reset	Access	Description
adc	28	0	R/W	Reset ADC

- 0: Peripheral is released to run normally.
- 1: Peripheral is held in a reset state.

4.1.8 Registers (PWRSEQ)

Address	Register	Access	Description	Reset By
0x40000A30	PWRSEQ_REG0	***	Power Sequencer Control Register 0	POR PWRSEQ
0x40000A34	PWRSEQ_REG1	R/W	Power Sequencer Control Register 1	PWRSEQ
0x40000A38	PWRSEQ_REG2	R/W	Power Sequencer Control Register 2	PWRSEQ
0x40000A3C	PWRSEQ_REG3	R/W	Power Sequencer Control Register 3	PWRSEQ
0x40000A40	PWRSEQ_REG4	R/W	Power Sequencer Control Register 4 (Internal Test Only)	PWRSEQ
0x40000A44	PWRSEQ_REG5	R/W	Power Sequencer Control Register 5 (Trim 0)	PWRSEQ
0x40000A48	PWRSEQ_REG6	R/W	Power Sequencer Control Register 6 (Trim 1)	PWRSEQ
0x40000A4C	PWRSEQ_REG7	R/O	Power Sequencer Control Register 7 (Trim 2)	Sys
0x40000A50	PWRSEQ_FLAGS	***	Power Sequencer Flags	Sys
0x40000A54	PWRSEQ_MSK_FLAGS	R/W	Power Sequencer Flags Mask Register	PWRSEQ
0x40000A60	PWRSEQ_RETN_CTRL0	R/W	Retention Control Register 0	PWRSEQ
0x40000A64	PWRSEQ_RETN_CTRL1	R/W	Retention Control Register 1	PWRSEQ

4.1.8.1 PWRSEQ_REG0

PWRSEQ_REG0.pwr_lp1

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_lp1	0	X	PWRSEQ:0	R/W	Shutdown Power Mode Select

- 0: Shutdown to LP0:STOP (default)
- 1: Shutdown to LP1:STANDBY

Note This bit is also reset to zero by any of the following conditions/events:
- System Reboot event
- Whenever the Power Fail Event Detected flag is set

PWRSEQ_REG0.pwr_first_boot

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_first_boot	1	X	PWRSEQ:1	R/W	Wake on First Boot

Wakeup on first power automatically.

PWRSEQ_REG0.pwr_sys_reboot

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_sys_reboot	2	X	PWRSEQ:0	W/O	Firmware System Reboot Request

Writing a 1 to this bit triggers a system reboot.

PWRSEQ_REG0.pwr_flashen_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_flashen_run	3	X	PWRSEQ:1	R/W	Enable Flash Operation during Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_flashen_slp

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_flashen_slp	4	X	PWRSEQ:0	R/W	Enable Flash Operation during Sleep Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_retregen_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_retregen_run	5	X	PWRSEQ:0	R/W	Enable Retention Regulator Operation during Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_retregen_slp

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_retregen_slp	6	X	PWRSEQ:0	R/W	Enable Retention Regulator Operation during Sleep Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_roen_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_roen_run	7	X	PWRSEQ:1	R/W	Enable 96MHz System Relaxation Oscillator in Run Mode

- 0: Disabled
- 1: Enabled

Note This bit should never be set to zero; if the system enters Run mode with the 96MHz system oscillator disabled, the device will enter a hard lockup state requiring a full power cycle of ALL supplies (including VRTC) to recover.

PWRSEQ_REG0.pwr_roen_slp

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_roen_slp	8	X	PWRSEQ:0	R/W	Enable 96MHz System Relaxation Oscillator in Sleep Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_nren_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_nren_run	9	X	VRTC_POR:0	R/W	Enable Nano Oscillator in Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_nren_slp

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_nren_slp	10	X	VRTC_POR:0	R/W	Enable Nano Oscillator in Sleep Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_rtcen_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_rtcen_run	11	X	VRTC_POR:0	R/W	Enable Real Time Clock Operation during Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_rtcen_slp

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_rtcen_slp	12	X	VRTC_POR:0	R/W	Enable Real Time Clock Operation during Sleep Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_svm12en_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_svm12en_run	13	X	PWRSEQ:1	R/W	Enable VDD12_SW SVM operation during Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_svm18en_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_svm18en_run	15	X	PWRSEQ:1	R/W	Enable VDD18_SW SVM operation during Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_svmrtcen_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_svmrtcen_run	17	X	PWRSEQ:1	R/W	Enable VRTC SVM operation during Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_svm_vddb_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_svm_vddb_run	19	X	PWRSEQ:1	R/W	Enable VDDB SVM operation during Run Mode

- 0: Disabled
- 1: Enabled

Internal VDDB POR must be released before actual enable.

PWRSEQ_REG0.pwr_svmtvdd12en_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_svmtvdd12en_run	21	X	PWRSEQ:1	R/W	Enable TVDD12 SVM operation during Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_vdd12_swen_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vdd12_swen_run	23	X	PWRSEQ:1	R/W	Enable VDD12 power switch during Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_vdd12_swen_slp

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vdd12_swen_slp	24	X	PWRSEQ:0	R/W	Enable VDD12 power switch during Sleep Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_vdd18_swen_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vdd18_swen_run	25	X	PWRSEQ:1	R/W	Enable VDD18 power switch during Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_vdd18_swen_slp

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vdd18_swen_slp	26	X	PWRSEQ:0	R/W	Enable VDD18 power switch during Sleep Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_tvdd12_swen_run

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_tvdd12_swen_run	27	X	PWRSEQ:1	R/W	Enable TVDD12 power switch during Run Mode

- 0: Disabled
- 1: Enabled

PWRSEQ_REG0.pwr_tvdd12_swen_slp

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_tvdd12_swen_slp	28	X	PWRSEQ:0	R/W	Enable TVDD12 power switch during Sleep Mode

- 0: Disabled
- 1: Enabled

4.1.8.2 PWRSEQ_REG1

PWRSEQ_REG1.pwr_clr_io_event_latch

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_clr_io_event_latch	0	X	PWRSEQ:0	R/W	Clear all GPIO Event Seen Latches

Write to 1 to clear latches.

After writing this bit to 1 to trigger the operation, firmware must manually clear this bit back to 0 for proper operation. This may be done immediately after writing the bit to 1; no delay is needed before clearing the bit.

PWRSEQ_REG1.pwr_clr_io_cfg_latch

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_clr_io_cfg_latch	1	X	PWRSEQ:0	R/W	Clear all GPIO Configuration Latches

Write to 1 to clear all GPIO configuration latches (Event Seen, Event Seen Active High/Low, Weak Pullup/down).

After writing this bit to 1 to trigger the operation, firmware must manually clear this bit back to 0 for proper operation. This may be done immediately after writing the bit to 1; no delay is needed before clearing the bit.

PWRSEQ_REG1.pwr_mbus_gate

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_mbus_gate	2	X	PWRSEQ:0	R/W	Freeze GPIO MBus State

- 0: Normal GPIO MBus operation
- 1: Freeze GPIO MBus state

PWRSEQ_REG1.pwr_discharge_en

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_discharge_en	3	X	PWRSEQ:0	R/W	Enable Flash Discharge During Powerfail Event

- 0: Disabled
- 1: Enabled

PWRSEQ_REG1.pwr_tvdd12_well

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_tvdd12_well	4	X	PWRSEQ:0	R/W	TVDD12 Well Switch

- 0: VDD12
- 1: TVDD12

4.1.8.3 PWRSEQ_REG2

PWRSEQ_REG2.pwr_vdd12_hyst

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vdd12_hyst	1:0	X	PWRSEQ:1	R/W	VDD12_SW Comparator Hysteresis Setting

- 0: 0 mV
- 1: 12.5 mV
- 2: 25 mV
- 3: 50 mV

PWRSEQ_REG2.pwr_vdd18_hyst

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vdd18_hyst	3:2	X	PWRSEQ:1	R/W	VDD18_SW Comparator Hysteresis Setting

- 0: 0 mV
- 1: 12.5 mV
- 2: 25 mV
- 3: 50 mV

PWRSEQ_REG2.pwr_vrtc_hyst

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vrtc_hyst	5:4	X	PWRSEQ:1	R/W	VRTC Comparator Hysteresis Setting

- 0: 0 mV

- 1: 12.5 mV
- 2: 25 mV
- 3: 50 mV

PWRSEQ_REG2.pwr_vddb_hyst

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vddb_hyst	7:6	X	PWRSEQ:1	R/W	VDDB Comparator Hysteresis Setting

- 0: 0 mV
- 1: 12.5 mV
- 2: 25 mV
- 3: 50 mV

PWRSEQ_REG2.pwr_tvdd12_hyst

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_tvdd12_hyst	9:8	X	PWRSEQ:1	R/W	TVDD12 Comparator Hysteresis Setting

- 0: 0 mV
- 1: 12.5 mV
- 2: 25 mV
- 3: 50 mV

4.1.8.4 PWRSEQ_REG3

PWRSEQ_REG3.pwr_rosel

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_rosel	2:0	X	PWRSEQ:5	R/W	Relaxation Oscillator Stable Timeout for Wakeup

This timeout delay only applies when the device is transitioning back to Run mode from either LP0 or LP1. The timeout setting ensures that the 96MHz relaxation oscillator output has stabilized before code execution begins.

- 0: No delay (bypass)
- 1: 140ns
- 2: 300ns

- 3: 620ns
- 4: 1.26μs
- 5: 2.54μs (default)
- 6: 5.10μs
- 7: 10.22μs

PWRSEQ_REG3.pwr_fltrrosel

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_fltrrosel	5:3	X	PWRSEQ:3	R/W	Power Supply Filter Timeout for Wakeup

This timeout delay only applies when the device is transitioning back to Run mode from either LP0 or LP1. The timeout setting ensures that the power supply has time to stabilize before code execution begins.

- 0: No delay (bypass)
- 1: 60ns
- 2: 140ns
- 3: 300ns (default)
- 4: 620ns
- 5: 1.26μs
- 6: 2.54μs
- 7: 5.10μs

PWRSEQ_REG3.pwr_svm_clk_mux

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_svm_clk_mux	7:6	X	PWRSEQ:0	R/W	SVM Clock Mux

- 0: Nanoring (default)
- 1: RTC clock (for test purposes only)
- 2: Relaxation oscillator (for test purposes only)
- 3: External clock (for test purposes only)

This field is intended for internal test use only and should not be modified by user application firmware.

PWRSEQ_REG3.pwr_ro_clk_mux

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_ro_clk_mux	9:8	X	PWRSEQ:0	R/W	Relaxation Clock Mux

- 0: Relaxation oscillator (default)
- 1: External clock (for test purposes only)
- 2: Nanoring (for test purposes only)
- 3: RTC clock (for test purposes only)

This field is intended for internal test use only and should not be modified by user application firmware.

PWRSEQ_REG3.pwr_failsel

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_failsel	12:10	X	PWRSEQ:0	R/W	Timeout before rebooting during PowerFail/BootFail events.

These timeout values are based on the nanoring clock.

- 0: No delay (bypass, default)
- 1: 8ms
- 2: 16ms
- 3: 32ms
- 4: 64ms
- 5: 128ms
- 6: 256ms
- 7: 512ms

4.1.8.5 PWRSEQ_REG4

PWRSEQ_REG4.pwr_tm_ps_2_gpio

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_tm_ps_2_gpio	0	X	PWRSEQ:0	R/W	Internal Use Only

This bit is intended for internal test use only and should not be modified by user application firmware.

PWRSEQ_REG4.pwr_tm_fast_timers

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_tm_fast_timers	1	X	PWRSEQ:0	R/W	Internal Use Only

This bit is intended for internal test use only and should not be modified by user application firmware.

PWRSEQ_REG4.pwr_usb_dis_comp

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_usb_dis_comp	3	X	PWRSEQ:0	R/W	Internal Use Only

This bit is intended for internal test use only and should not be modified by user application firmware.

PWRSEQ_REG4.pwr_ro_tstclk_en

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_ro_tstclk_en	4	X	PWRSEQ:0	R/W	Internal Use Only

This bit is intended for internal test use only and should not be modified by user application firmware.

PWRSEQ_REG4.pwr_nr_clk_gate_en

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_nr_clk_gate_en	5	X	PWRSEQ:0	R/W	Internal Use Only

This bit is intended for internal test use only and should not be modified by user application firmware.

PWRSEQ_REG4.pwr_ext_clk_in_en

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_ext_clk_in_en	6	X	PWRSEQ:0	R/W	Internal Use Only

This bit is intended for internal test use only and should not be modified by user application firmware.

PWRSEQ_REG4.pwr_pseq_32k_en

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_pseq_32k_en	7	X	PWRSEQ:0	R/W	Internal Use Only

This bit is intended for internal test use only and should not be modified by user application firmware.

4.1.8.6 PWRSEQ_REG5

PWRSEQ_REG5.pwr_trim_svm_bg

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_trim_svm_bg	8:0	X	PWRSEQ:0	R/W	Power Manager Bandgap Trim Setting

This field must be initialized by firmware following initial powerup by copying from the TRIM_FOR_PWR_REG5 register to this register.

PWRSEQ_REG5.pwr_trim_bias

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_trim_bias	14:9	X	PWRSEQ:0	R/W	Power Manager Bias Current Trim Setting

This field must be initialized by firmware following initial powerup by copying from the TRIM_FOR_PWR_REG5 register to this register.

PWRSEQ_REG5.pwr_trim_retreg_5_0

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_trim_retreg_5_0	20:15	X	PWRSEQ:0	R/W	Retention Regulator Trim Setting (Bits 5:0)

This field must be initialized by firmware following initial powerup by copying from the TRIM_FOR_PWR_REG5 register to this register.

PWRSEQ_REG5.pwr_rtc_trim

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_rtc_trim	24:21	X	PWRSEQ:0	R/W	Real Time Clock Trim Setting

This field must be initialized by firmware following initial powerup by copying from the TRIM_FOR_PWR_REG5 register to this register.

4.1.8.7 PWRSEQ_REG6

PWRSEQ_REG6.pwr_trim_usb_bias

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_trim_usb_bias	2:0	X	PWRSEQ:0	R/W	USB Bias Current Trim Setting

This field must be initialized by firmware following initial powerup by copying from the TRIM_FOR_PWR_REG6 register to this register.

PWRSEQ_REG6.pwr_trim_usb_pm_res

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_trim_usb_pm_res	6:3	X	PWRSEQ:0	R/W	USB Data Plus Slew Rate Trim Setting

This field must be initialized by firmware following initial powerup by copying from the TRIM_FOR_PWR_REG6 register to this register.

PWRSEQ_REG6.pwr_trim_usb_dm_res

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_trim_usb_dm_res	10:7	X	PWRSEQ:0	R/W	USB Data Minus Slew Rate Trim Setting

This field must be initialized by firmware following initial powerup by copying from the TRIM_FOR_PWR_REG6 register to this register.

PWRSEQ_REG6.pwr_trim_osc_vref

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_trim_osc_vref	19:11	X	PWRSEQ:17Ch	R/W	Relaxation Oscillator Trim Setting

This field must be initialized by firmware following initial powerup by copying from the TRIM_FOR_PWR_REG6 register to this register.

PWRSEQ_REG6.pwr_trim_crypto_osc

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_trim_crypto_osc	28:20	X	PWRSEQ:0	R/W	Crypto Oscillator Trim Setting

This field must be initialized by firmware following initial powerup by copying from the TRIM_FOR_PWR_REG6 register to this register.

4.1.8.8 PWRSEQ_REG7

PWRSEQ_REG7.pwr_flash_pd_lookahead

Field	Bits	Sys Reset	Access	Description
pwr_flash_pd_lookahead	0	0	R/O	Flash Powerdown Lookahead Flag

When this flag returns 1, the flash is about to be powered down in preparation for entry to LP0 or LP1 mode.

4.1.8.9 PWRSEQ_FLAGS

PWRSEQ_FLAGS.pwr_first_boot

Field	Bits	Sys Reset	Access	Description
pwr_first_boot	0	s	R/O	Initial Boot Event Detected Flag

PWRSEQ_FLAGS.pwr_sys_reboot

Field	Bits	Sys Reset	Access	Description
pwr_sys_reboot	1	s	R/O	Firmware Reset Event Detected Flag

PWRSEQ_FLAGS.pwr_power_fail

Field	Bits	Sys Reset	Access	Description
pwr_power_fail	2	s	W1C	Power Fail Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_boot_fail

Field	Bits	Sys Reset	Access	Description
pwr_boot_fail	3	s	W1C	Boot Fail Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_flash_discharge

Field	Bits	Sys Reset	Access	Description
pwr_flash_discharge	4	s	W1C	Flash Discharged During Powerfail Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_iowakeup

Field	Bits	Sys Reset	Access	Description
pwr_iowakeup	5	s	W1C	GPIO Wakeup Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_vdd12_rst_bad

Field	Bits	Sys Reset	Access	Description
pwr_vdd12_rst_bad	6	s	W1C	VDD12_SW Comparator Tripped Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_vdd18_rst_bad

Field	Bits	Sys Reset	Access	Description
pwr_vdd18_rst_bad	7	s	W1C	VDD18_SW Comparator Tripped Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_vrtc_rst_bad

Field	Bits	Sys Reset	Access	Description
pwr_vrtc_rst_bad	8	s	W1C	VRTC Comparator Tripped Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_vddb_rst_bad

Field	Bits	Sys Reset	Access	Description
pwr_vddb_rst_bad	9	s	W1C	VDDB Comparator Tripped Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_tvdd12_rst_bad

Field	Bits	Sys Reset	Access	Description
pwr_tvdd12_rst_bad	10	s	W1C	TVDD12 Comparator Tripped Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_por18z_fail_latch

Field	Bits	Sys Reset	Access	Description
pwr_por18z_fail_latch	11	s	W1C	POR18 and POR18_bg have been tripped

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.rtc_cmpr0

Field	Bits	Sys Reset	Access	Description
rtc_cmpr0	12	s	R/O	RTC Comparator 0 Match Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.rtc_cmpr1

Field	Bits	Sys Reset	Access	Description
rtc_cmpr1	13	s	R/O	RTC Comparator 1 Match Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.rtc_prescale_cmp

Field	Bits	Sys Reset	Access	Description
rtc_prescale_cmp	14	s	R/O	RTC Prescale Comparator Match Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.rtc_rollover

Field	Bits	Sys Reset	Access	Description
rtc_rollover	15	s	R/O	RTC Rollover Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_usb_plug_wakeup

Field	Bits	Sys Reset	Access	Description
pwr_usb_plug_wakeup	16	s	W1C	USB Power Connect Wakeup Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_usb_remove_wakeup

Field	Bits	Sys Reset	Access	Description
pwr_usb_remove_wakeup	17	s	W1C	USB Power Remove Wakeup Event Detected Flag

Write 1 to clear event detected flag.

PWRSEQ_FLAGS.pwr_tvdd12_bad

Field	Bits	Sys Reset	Access	Description
pwr_tvdd12_bad	18	s	W1C	Retention Regulator POR Tripped Event Detected Flag

Write 1 to clear event detected flag.

4.1.8.10 PWRSEQ_MSK_FLAGS

PWRSEQ_MSK_FLAGS.pwr_sys_reboot

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_sys_reboot	1	X	PWRSEQ:1	R/W	Mask for system reboot detect

- 0: No effect.
- 1: If a watchdog is asserting a system reboot output, this will cause the device to wake up from LP0.

PWRSEQ_MSK_FLAGS.pwr_power_fail

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_power_fail	2	X	PWRSEQ:1	R/W	Mask for previous power fail detect

- 0: Normal operation.
- 1: The power sequencer will *not* shut down due to a power fail event.

PWRSEQ_MSK_FLAGS.pwr_boot_fail

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_boot_fail	3	X	PWRSEQ:1	R/W	Mask for previous boot fail detect

- 0: Normal operation.
- 1: The power sequencer will *not* restart the boot/resume sequence due to a boot failure event.

PWRSEQ_MSK_FLAGS.pwr_flash_discharge

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_flash_discharge	4	X	PWRSEQ:1	R/W	Mask for flash discharge event

- 0: Normal operation.
- 1: The power sequencer will *not* safely discharge flash in the event of a power failure.

PWRSEQ_MSK_FLAGS.pwr_iowakeup

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_iowakeup	5	X	PWRSEQ:1	R/W	Mask for GPIO wakeup event detect

- 0: GPIO pins cannot be used for wakeup detection. Also, in this mode, an active low external reset on SRSTn will not wake the device up from LP1 or LP0.
- 1: GPIO pins can be used for wakeup detection in LP1/LP0 (note that wakeup detection must be enabled individually for each GPIO pin that will be used in this manner). Also, wakeup detection will be enabled on the SRSTn pin for LP1/LP0.

PWRSEQ_MSK_FLAGS.pwr_vdd12_rst_bad

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vdd12_rst_bad	6	X	PWRSEQ:1	R/W	Mask for VDD12 rst event

- 0: Normal operation.
- 1: The power sequencer will *not* shut down due to a VDD12 rst event.

PWRSEQ_MSK_FLAGS.pwr_vdd18_rst_bad

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vdd18_rst_bad	7	X	PWRSEQ:1	R/W	Mask for VDD18 rst event

- 0: Normal operation.
- 1: The power sequencer will *not* shut down due to a VDD18 rst event.

PWRSEQ_MSK_FLAGS.pwr_vrtc_rst_bad

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vrtc_rst_bad	8	X	PWRSEQ:1	R/W	Mask for VRTC rst event

- 0: Normal operation.
- 1: The power sequencer will *not* shut down due to a VRTC rst event.

PWRSEQ_MSK_FLAGS.pwr_vddb_rst_bad

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_vddb_rst_bad	9	X	PWRSEQ:1	R/W	Mask for VDDB rst event

- 0: Normal operation.
- 1: The power sequencer will *not* shut down due to a VDDB rst event.

PWRSEQ_MSK_FLAGS.pwr_tvdd12_rst_bad

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_tvdd12_rst_bad	10	X	PWRSEQ:1	R/W	Mask for TVDD12 rst event

- 0: Normal operation.
- 1: The power sequencer will *not* shut down due to a TVDD12 rst event.

PWRSEQ_MSK_FLAGS.pwr_por18z_fail_latch

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_por18z_fail_latch	11	X	PWRSEQ:1	R/W	Mask for POR18 powerfail event

- 0: Normal operation.
- 1: The power sequencer will not latch any POR18 powerfail event.

PWRSEQ_MSK_FLAGS.rtc_cmpr0

Field	Bits	Sys Reset	Alt Reset	Access	Description
rtc_cmpr0	12	X	PWRSEQ:1	R/W	Mask for RTC compare 0 event

This mask bit is inverted in sense from the other non-RTC mask bits in this register.

- 0: Event can be detected and used as a wakeup source
- 1: Event is masked

PWRSEQ_MSK_FLAGS.rtc_cmpr1

Field	Bits	Sys Reset	Alt Reset	Access	Description
rtc_cmpr1	13	X	PWRSEQ:1	R/W	Mask for RTC compare 1 event

This mask bit is inverted in sense from the other non-RTC mask bits in this register.

- 0: Event can be detected and used as a wakeup source
- 1: Event is masked

PWRSEQ_MSK_FLAGS.rtc_prescale_cmp

Field	Bits	Sys Reset	Alt Reset	Access	Description
rtc_prescale_cmp	14	X	PWRSEQ:1	R/W	Mask for RTC prescale compare event

This mask bit is inverted in sense from the other non-RTC mask bits in this register.

- 0: Event can be detected and used as a wakeup source
- 1: Event is masked

PWRSEQ_MSK_FLAGS.rtc_rollover

Field	Bits	Sys Reset	Alt Reset	Access	Description
rtc_rollover	15	X	PWRSEQ:1	R/W	Mask for RTC rollover event

This mask bit is inverted in sense from the other non-RTC mask bits in this register.

- 0: Event can be detected and used as a wakeup source
- 1: Event is masked

PWRSEQ_MSK_FLAGS.pwr_usb_plug_wakeup

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_usb_plug_wakeup	16	X	PWRSEQ:1	R/W	Mask for USB plug connect event

- 0: Event is masked
- 1: Event can be detected and used as a wakeup source

PWRSEQ_MSK_FLAGS.pwr_usb_remove_wakeup

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_usb_remove_wakeup	17	X	PWRSEQ:1	R/W	Mask for USB plug disconnect event

- 0: Event is masked
- 1: Event can be detected and used as a wakeup source

PWRSEQ_MSK_FLAGS.pwr_tvdd12_bad

Field	Bits	Sys Reset	Alt Reset	Access	Description
pwr_tvdd12_bad	18	X	PWRSEQ:1	R/W	Mask for TVDD12 power fail event

- 0: Normal operation.
- 1: The power sequencer will *not* shut down due to a TVDD12 power fail event (applies during LP1 only).

4.1.8.11 PWRSEQ_RETN_CTRL0

PWRSEQ_RETN_CTRL0.retn_ctrl_en

Field	Bits	Sys Reset	Alt Reset	Access	Description
retn_ctrl_en	0	X	PWRSEQ:0	R/W	Retention Controller Enable

This bit controls the top-level enable/disable mode for the retention controller.

- 0: Disabled.
- 1: Enabled.

PWRSEQ_RETN_CTRL0.rc_rel_ccg_early

Field	Bits	Sys Reset	Alt Reset	Access	Description
rc_rel_ccg_early	1	X	PWRSEQ:0	R/W	Early Core Clock Gate Release

- 0: Normal behavior.
- 1: Allows the retention controller to release the core clock gate early, before the TWake period has expired.

PWRSEQ_RETN_CTRL0.rc_use_flc_twk

Field	Bits	Sys Reset	Alt Reset	Access	Description
rc_use_flc_twk	2	X	PWRSEQ:1	R/W	Enable Flash Controller TWake Timer

- 0: Flash controller TWake is ignored.
- 1: Allow the flash controller to implement its own TWake timeout.

PWRSEQ_RETN_CTRL0.rc_poll_flash

Field	Bits	Sys Reset	Alt Reset	Access	Description
rc_poll_flash	3	X	PWRSEQ:0	R/W	Enable Automatic Flash Polling for Wakeup

- 0: Normal behavior.
- 1: Allow automatic polling of the flash memory to determine when the flash is awake and ready for use by the application.

PWRSEQ_RETN_CTRL0.restore_override

Field	Bits	Sys Reset	Alt Reset	Access	Description
restore_override	4	X	PWRSEQ:0	R/W	Restore Override (Reserved).

Reserved. For proper retention controller operation, this bit should always be set to 0.

4.1.8.12 PWRSEQ_RETN_CTRL1

PWRSEQ_RETN_CTRL1.rc_twk

Field	Bits	Sys Reset	Alt Reset	Access	Description
rc_twk	3:0	X	PWRSEQ:0xB	R/W	Retention Controller TWake Cycle Count

The number of micro seconds before the twk (flash timer wakeup period) stage of retention controller is complete, thus enabling the flash controller to take over (which may or may not implement its own twk circuit).

PWRSEQ_RETN_CTRL1.sram_fms

Field	Bits	Sys Reset	Alt Reset	Access	Description
sram_fms	7:4	X	PWRSEQ:0xC	R/W	SRAM Margin Setting (Reserved).

Reserved. For proper operation, this field should be left at its default value.

4.2 Interrupt Vector Table

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
16	0	Crypto Oscillator Stable	CLKMAN_INTEN.crypto_stable	CLKMAN_INTFL.crypto_stable
17	1	VDD12 Power Supply Warning	PWRMAN_INTEN.v1_2_warning	PWRMAN_INTFL.v1_2_warning
		VDD18 Power Supply Warning	PWRMAN_INTEN.v1_8_warning	PWRMAN_INTFL.v1_8_warning
		VRTC Power Supply Warning	PWRMAN_INTEN.rtc_warning	PWRMAN_INTFL.rtc_warning
		VDDA Power Supply Warning	PWRMAN_INTEN.vdda_warning	PWRMAN_INTFL.vdda_warning
		VDDB Power Supply Warning	PWRMAN_INTEN.vddb_warning	PWRMAN_INTFL.vddb_warning
18	2	Flash Controller Operation Started	FLC_INTR.started_ie	FLC_INTR.started_if
		Flash Controller Operation Failed	FLC_INTR.failed_ie	FLC_INTR.failed_if
19	3	RTC Time-of-Day Alarm 0	RTCTMR_INTEN.comp0	RTCTMR_FLAGS.comp0
20	4	RTC Time-of-Day Alarm 1	RTCTMR_INTEN.comp1	RTCTMR_FLAGS.comp1
21	5	RTC Interval Alarm	RTCTMR_INTEN.prescale_comp	RTCTMR_FLAGS.prescale_comp
22	6	RTC Counter Overflow	RTCTMR_INTEN.overflow	RTCTMR_FLAGS.overflow
		RTC Trim Adjustment	RTCTMR_INTEN.trim	RTCTMR_FLAGS.trim
23	7	PMU Channel n Interrupt	PMUn_CFG.int_en	PMUn_CFG.interrupt
24	8	USB DPLUS Activity	USB_DEV_INTEN.dpact	USB_DEV_INTFL.dpact
		USB Remote Wakeup Done	USB_DEV_INTEN.rwu_dn	USB_DEV_INTFL.rwu_dn
		USB Bus Activity	USB_DEV_INTEN.bact	USB_DEV_INTFL.bact
		USB Bus Reset In Progress	USB_DEV_INTEN.brst	USB_DEV_INTFL.brst
		USB Suspend	USB_DEV_INTEN.susp	USB_DEV_INTFL.susp
		USB No VBUS Present	USB_DEV_INTEN.no_vbus	USB_DEV_INTFL.no_vbus
		USB VBUS Connect	USB_DEV_INTEN.vbus	USB_DEV_INTFL.vbus

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
24	8	USB Bus Reset Completed	USB_DEV_INTEN.brst_dn	USB_DEV_INTFL.brst_dn
		USB Setup Packet	USB_DEV_INTEN.setup	USB_DEV_INTFL.setup
		USB Endpoint IN	USB_DEV_INTEN.ep_in	USB_DEV_INTFL.ep_in
		USB Endpoint OUT	USB_DEV_INTEN.ep_out	USB_DEV_INTFL.ep_out
		USB Endpoint NAK	USB_DEV_INTEN.ep_nak	USB_DEV_INTFL.ep_nak
		USB DMA Error	USB_DEV_INTEN.dma_err	USB_DEV_INTFL.dma_err
		USB Buffer Overflow	USB_DEV_INTEN.buf_ovr	USB_DEV_INTFL.buf_ovr
25	9	AES Interrupt	AES_CTRL.inten	AES_CTRL.intfl
26	10	MAA Done	MAA_CTRL.inten	MAA_CTRL.if_done
		MAA Error	MAA_CTRL.inten	MAA_CTRL.if_error
27	11	Watchdog 0 Timeout Interrupt	WDT0_ENABLE.timeout	WDT0_FLAGS.timeout
28	12	Watchdog 0 Pre-window Interrupt	WDT0_ENABLE.pre_win	WDT0_FLAGS.pre_win
29	13	Watchdog 1 Timeout Interrupt	WDT1_ENABLE.timeout	WDT1_FLAGS.timeout
30	14	Watchdog 1 Pre-window Interrupt	WDT1_ENABLE.pre_win	WDT1_FLAGS.pre_win
31	15	GPIO External Interrupt on P0.0	GPIO_INTEN_P0.pin0	GPIO_INTFL_P0.pin0
		GPIO External Interrupt on P0.1	GPIO_INTEN_P0.pin1	GPIO_INTFL_P0.pin1
		GPIO External Interrupt on P0.2	GPIO_INTEN_P0.pin2	GPIO_INTFL_P0.pin2
		GPIO External Interrupt on P0.3	GPIO_INTEN_P0.pin3	GPIO_INTFL_P0.pin3
		GPIO External Interrupt on P0.4	GPIO_INTEN_P0.pin4	GPIO_INTFL_P0.pin4
		GPIO External Interrupt on P0.5	GPIO_INTEN_P0.pin5	GPIO_INTFL_P0.pin5
		GPIO External Interrupt on P0.6	GPIO_INTEN_P0.pin6	GPIO_INTFL_P0.pin6
		GPIO External Interrupt on P0.7	GPIO_INTEN_P0.pin7	GPIO_INTFL_P0.pin7

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
32	16	GPIO External Interrupt on P1.0	GPIO_INTEN_P1.pin0	GPIO_INTFL_P1.pin0
		GPIO External Interrupt on P1.1	GPIO_INTEN_P1.pin1	GPIO_INTFL_P1.pin1
		GPIO External Interrupt on P1.2	GPIO_INTEN_P1.pin2	GPIO_INTFL_P1.pin2
		GPIO External Interrupt on P1.3	GPIO_INTEN_P1.pin3	GPIO_INTFL_P1.pin3
		GPIO External Interrupt on P1.4	GPIO_INTEN_P1.pin4	GPIO_INTFL_P1.pin4
		GPIO External Interrupt on P1.5	GPIO_INTEN_P1.pin5	GPIO_INTFL_P1.pin5
		GPIO External Interrupt on P1.6	GPIO_INTEN_P1.pin6	GPIO_INTFL_P1.pin6
		GPIO External Interrupt on P1.7	GPIO_INTEN_P1.pin7	GPIO_INTFL_P1.pin7
33	17	GPIO External Interrupt on P2.0	GPIO_INTEN_P2.pin0	GPIO_INTFL_P2.pin0
		GPIO External Interrupt on P2.1	GPIO_INTEN_P2.pin1	GPIO_INTFL_P2.pin1
		GPIO External Interrupt on P2.2	GPIO_INTEN_P2.pin2	GPIO_INTFL_P2.pin2
		GPIO External Interrupt on P2.3	GPIO_INTEN_P2.pin3	GPIO_INTFL_P2.pin3
		GPIO External Interrupt on P2.4	GPIO_INTEN_P2.pin4	GPIO_INTFL_P2.pin4
		GPIO External Interrupt on P2.5	GPIO_INTEN_P2.pin5	GPIO_INTFL_P2.pin5
		GPIO External Interrupt on P2.6	GPIO_INTEN_P2.pin6	GPIO_INTFL_P2.pin6
		GPIO External Interrupt on P2.7	GPIO_INTEN_P2.pin7	GPIO_INTFL_P2.pin7
34	18	GPIO External Interrupt on P3.0	GPIO_INTEN_P3.pin0	GPIO_INTFL_P3.pin0
		GPIO External Interrupt on P3.1	GPIO_INTEN_P3.pin1	GPIO_INTFL_P3.pin1
		GPIO External Interrupt on P3.2	GPIO_INTEN_P3.pin2	GPIO_INTFL_P3.pin2
		GPIO External Interrupt on P3.3	GPIO_INTEN_P3.pin3	GPIO_INTFL_P3.pin3
		GPIO External Interrupt on P3.4	GPIO_INTEN_P3.pin4	GPIO_INTFL_P3.pin4
		GPIO External Interrupt on P3.5	GPIO_INTEN_P3.pin5	GPIO_INTFL_P3.pin5
		GPIO External Interrupt on P3.6	GPIO_INTEN_P3.pin6	GPIO_INTFL_P3.pin6
		GPIO External Interrupt on P3.7	GPIO_INTEN_P3.pin7	GPIO_INTFL_P3.pin7

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
35	19	GPIO External Interrupt on P4.0	GPIO_INTEN_P4.pin0	GPIO_INTFL_P4.pin0
		GPIO External Interrupt on P4.1	GPIO_INTEN_P4.pin1	GPIO_INTFL_P4.pin1
		GPIO External Interrupt on P4.2	GPIO_INTEN_P4.pin2	GPIO_INTFL_P4.pin2
		GPIO External Interrupt on P4.3	GPIO_INTEN_P4.pin3	GPIO_INTFL_P4.pin3
		GPIO External Interrupt on P4.4	GPIO_INTEN_P4.pin4	GPIO_INTFL_P4.pin4
		GPIO External Interrupt on P4.5	GPIO_INTEN_P4.pin5	GPIO_INTFL_P4.pin5
		GPIO External Interrupt on P4.6	GPIO_INTEN_P4.pin6	GPIO_INTFL_P4.pin6
		GPIO External Interrupt on P4.7	GPIO_INTEN_P4.pin7	GPIO_INTFL_P4.pin7
36	20	GPIO External Interrupt on P5.0	GPIO_INTEN_P5.pin0	GPIO_INTFL_P5.pin0
		GPIO External Interrupt on P5.1	GPIO_INTEN_P5.pin1	GPIO_INTFL_P5.pin1
		GPIO External Interrupt on P5.2	GPIO_INTEN_P5.pin2	GPIO_INTFL_P5.pin2
		GPIO External Interrupt on P5.3	GPIO_INTEN_P5.pin3	GPIO_INTFL_P5.pin3
		GPIO External Interrupt on P5.4	GPIO_INTEN_P5.pin4	GPIO_INTFL_P5.pin4
		GPIO External Interrupt on P5.5	GPIO_INTEN_P5.pin5	GPIO_INTFL_P5.pin5
		GPIO External Interrupt on P5.6	GPIO_INTEN_P5.pin6	GPIO_INTFL_P5.pin6
		GPIO External Interrupt on P5.7	GPIO_INTEN_P5.pin7	GPIO_INTFL_P5.pin7
37	21	GPIO External Interrupt on P6.x	GPIO_INTEN_P6.pin0	GPIO_INTFL_P6.pin0
38	22	Timer 0 Interrupt 0 (x32 or x16/0)	TMR0_INTEN.timer0	TMR0_INTFL.timer0
39	23	Timer 0 Interrupt 1 (x16/1)	TMR0_INTEN.timer1	TMR0_INTFL.timer1
40	24	Timer 1 Interrupt 0 (x32 or x16/0)	TMR1_INTEN.timer0	TMR1_INTFL.timer0
41	25	Timer 1 Interrupt 1 (x16/1)	TMR1_INTEN.timer1	TMR1_INTFL.timer1
42	26	Timer 2 Interrupt 0 (x32 or x16/0)	TMR2_INTEN.timer0	TMR2_INTFL.timer0
43	27	Timer 2 Interrupt 1 (x16/1)	TMR2_INTEN.timer1	TMR2_INTFL.timer1

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
44	28	Timer 3 Interrupt 0 (x32 or x16/0)	TMR3_INTEN.timer0	TMR3_INTFL.timer0
45	29	Timer 3 Interrupt 1 (x16/1)	TMR3_INTEN.timer1	TMR3_INTFL.timer1
46	30	Timer 4 Interrupt 0 (x32 or x16/0)	TMR4_INTEN.timer0	TMR4_INTFL.timer0
47	31	Timer 4 Interrupt 1 (x16/1)	TMR4_INTEN.timer1	TMR4_INTFL.timer1
48	32	Timer 5 Interrupt 0 (x32 or x16/0)	TMR5_INTEN.timer0	TMR5_INTFL.timer0
49	33	Timer 5 Interrupt 1 (x16/1)	TMR5_INTEN.timer1	TMR5_INTFL.timer1
50	34	UART 0 TX Done	UART0_INTEN.tx_done	UART0_INTFL.tx_done
		UART 0 TX Unstalled	UART0_INTEN.tx_unstalled	UART0_INTFL.tx_unstalled
		UART 0 TX FIFO Almost Empty	UART0_INTEN.tx_fifo_ae	UART0_INTFL.tx_fifo_ae
		UART 0 RX FIFO Not Empty	UART0_INTEN.rx_fifo_not_empty	UART0_INTFL.rx_fifo_not_empty
		UART 0 RX Stalled	UART0_INTEN.rx_stalled	UART0_INTFL.rx_stalled
		UART 0 RX FIFO Almost Full	UART0_INTEN.rx_fifo_af	UART0_INTFL.rx_fifo_af
		UART 0 RX FIFO Overflow	UART0_INTEN.rx_fifo_overflow	UART0_INTFL.rx_fifo_overflow
		UART 0 RX Framing Error	UART0_INTEN.rx_framing_err	UART0_INTFL.rx_framing_err
		UART 0 RX Parity Error	UART0_INTEN.rx_parity_err	UART0_INTFL.rx_parity_err
51	35	UART 1 TX Done	UART1_INTEN.tx_done	UART1_INTFL.tx_done
		UART 1 TX Unstalled	UART1_INTEN.tx_unstalled	UART1_INTFL.tx_unstalled
		UART 1 TX FIFO Almost Empty	UART1_INTEN.tx_fifo_ae	UART1_INTFL.tx_fifo_ae
		UART 1 RX FIFO Not Empty	UART1_INTEN.rx_fifo_not_empty	UART1_INTFL.rx_fifo_not_empty
		UART 1 RX Stalled	UART1_INTEN.rx_stalled	UART1_INTFL.rx_stalled
		UART 1 RX FIFO Almost Full	UART1_INTEN.rx_fifo_af	UART1_INTFL.rx_fifo_af
		UART 1 RX FIFO Overflow	UART1_INTEN.rx_fifo_overflow	UART1_INTFL.rx_fifo_overflow
		UART 1 RX Framing Error	UART1_INTEN.rx_framing_err	UART1_INTFL.rx_framing_err
		UART 1 RX Parity Error	UART1_INTEN.rx_parity_err	UART1_INTFL.rx_parity_err

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
52	36	UART 2 TX Done	UART2_INTEN.tx_done	UART2_INTFL.tx_done
		UART 2 TX Unstalled	UART2_INTEN.tx_unstalled	UART2_INTFL.tx_unstalled
		UART 2 TX FIFO Almost Empty	UART2_INTEN.tx_fifo_ae	UART2_INTFL.tx_fifo_ae
		UART 2 RX FIFO Not Empty	UART2_INTEN.rx_fifo_not_empty	UART2_INTFL.rx_fifo_not_empty
		UART 2 RX Stalled	UART2_INTEN.rx_stalled	UART2_INTFL.rx_stalled
		UART 2 RX FIFO Almost Full	UART2_INTEN.rx_fifo_af	UART2_INTFL.rx_fifo_af
		UART 2 RX FIFO Overflow	UART2_INTEN.rx_fifo_overflow	UART2_INTFL.rx_fifo_overflow
		UART 2 RX Framing Error	UART2_INTEN.rx_framing_err	UART2_INTFL.rx_framing_err
		UART 2 RX Parity Error	UART2_INTEN.rx_parity_err	UART2_INTFL.rx_parity_err
53	37	UART 3 TX Done	UART3_INTEN.tx_done	UART3_INTFL.tx_done
		UART 3 TX Unstalled	UART3_INTEN.tx_unstalled	UART3_INTFL.tx_unstalled
		UART 3 TX FIFO Almost Empty	UART3_INTEN.tx_fifo_ae	UART3_INTFL.tx_fifo_ae
		UART 3 RX FIFO Not Empty	UART3_INTEN.rx_fifo_not_empty	UART3_INTFL.rx_fifo_not_empty
		UART 3 RX Stalled	UART3_INTEN.rx_stalled	UART3_INTFL.rx_stalled
		UART 3 RX FIFO Almost Full	UART3_INTEN.rx_fifo_af	UART3_INTFL.rx_fifo_af
		UART 3 RX FIFO Overflow	UART3_INTEN.rx_fifo_overflow	UART3_INTFL.rx_fifo_overflow
		UART 3 RX Framing Error	UART3_INTEN.rx_framing_err	UART3_INTFL.rx_framing_err
		UART 3 RX Parity Error	UART3_INTEN.rx_parity_err	UART3_INTFL.rx_parity_err
54	38	Pulse Train 0	PTG_INTEN.pt0	PTG_INTFL.pt0
		Pulse Train 1	PTG_INTEN.pt1	PTG_INTFL.pt1
		Pulse Train 2	PTG_INTEN.pt2	PTG_INTFL.pt2
		Pulse Train 3	PTG_INTEN.pt3	PTG_INTFL.pt3

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
54	38	Pulse Train 4	PTG_INTEN.pt4	PTG_INTFL.pt4
		Pulse Train 5	PTG_INTEN.pt5	PTG_INTFL.pt5
		Pulse Train 6	PTG_INTEN.pt6	PTG_INTFL.pt6
		Pulse Train 7	PTG_INTEN.pt7	PTG_INTFL.pt7
		Pulse Train 8	PTG_INTEN.pt8	PTG_INTFL.pt8
		Pulse Train 9	PTG_INTEN.pt9	PTG_INTFL.pt9
		Pulse Train 10	PTG_INTEN.pt10	PTG_INTFL.pt10
		Pulse Train 11	PTG_INTEN.pt11	PTG_INTFL.pt11
		Pulse Train 12	PTG_INTEN.pt12	PTG_INTFL.pt12
		Pulse Train 13	PTG_INTEN.pt13	PTG_INTFL.pt13
		Pulse Train 14	PTG_INTEN.pt14	PTG_INTFL.pt14
		Pulse Train 15	PTG_INTEN.pt15	PTG_INTFL.pt15
55	39	I2CM0 TX Done	I2CM0_INTEN.tx_done	I2CM0_INTFL.tx_done
		I2CM0 TX Nacked	I2CM0_INTEN.tx_nacked	I2CM0_INTFL.tx_nacked
		I2CM0 TX Lost Arbitration	I2CM0_INTEN.tx_lost_arbitr	I2CM0_INTFL.tx_lost_arbitr
		I2CM0 TX Timeout	I2CM0_INTEN.tx_timeout	I2CM0_INTFL.tx_timeout
		I2CM0 TX FIFO Empty	I2CM0_INTEN.tx_fifo_empty	I2CM0_INTFL.tx_fifo_empty
		I2CM0 TX FIFO $\frac{3}{4}$ Empty	I2CM0_INTEN.tx_fifo_3q_empty	I2CM0_INTFL.tx_fifo_3q_empty
		I2CM0 RX FIFO Not Empty	I2CM0_INTEN.rx_fifo_not_empty	I2CM0_INTFL.rx_fifo_not_empty
		I2CM0 RX FIFO $\frac{1}{2}$ Full	I2CM0_INTEN.rx_fifo_2q_full	I2CM0_INTFL.rx_fifo_2q_full
		I2CM0 RX FIFO $\frac{3}{4}$ Full	I2CM0_INTEN.rx_fifo_3q_full	I2CM0_INTFL.rx_fifo_3q_full
		I2CM0 RX FIFO Full	I2CM0_INTEN.rx_fifo_full	I2CM0_INTFL.rx_fifo_full

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
56	40	I2CM1 TX Done	I2CM1_INTEN.tx_done	I2CM1_INTFL.tx_done
		I2CM1 TX Nacked	I2CM1_INTEN.tx_nacked	I2CM1_INTFL.tx_nacked
		I2CM1 TX Lost Arbitration	I2CM1_INTEN.tx_lost_arbitr	I2CM1_INTFL.tx_lost_arbitr
		I2CM1 TX Timeout	I2CM1_INTEN.tx_timeout	I2CM1_INTFL.tx_timeout
		I2CM1 TX FIFO Empty	I2CM1_INTEN.tx_fifo_empty	I2CM1_INTFL.tx_fifo_empty
		I2CM1 TX FIFO $\frac{3}{4}$ Empty	I2CM1_INTEN.tx_fifo_3q_empty	I2CM1_INTFL.tx_fifo_3q_empty
		I2CM1 RX FIFO Not Empty	I2CM1_INTEN.rx_fifo_not_empty	I2CM1_INTFL.rx_fifo_not_empty
		I2CM1 RX FIFO $\frac{1}{2}$ Full	I2CM1_INTEN.rx_fifo_2q_full	I2CM1_INTFL.rx_fifo_2q_full
		I2CM1 RX FIFO $\frac{3}{4}$ Full	I2CM1_INTEN.rx_fifo_3q_full	I2CM1_INTFL.rx_fifo_3q_full
		I2CM1 RX FIFO Full	I2CM1_INTEN.rx_fifo_full	I2CM1_INTFL.rx_fifo_full
57	41	I2CM2 TX Done	I2CM2_INTEN.tx_done	I2CM2_INTFL.tx_done
		I2CM2 TX Nacked	I2CM2_INTEN.tx_nacked	I2CM2_INTFL.tx_nacked
		I2CM2 TX Lost Arbitration	I2CM2_INTEN.tx_lost_arbitr	I2CM2_INTFL.tx_lost_arbitr
		I2CM2 TX Timeout	I2CM2_INTEN.tx_timeout	I2CM2_INTFL.tx_timeout
		I2CM2 TX FIFO Empty	I2CM2_INTEN.tx_fifo_empty	I2CM2_INTFL.tx_fifo_empty
		I2CM2 TX FIFO $\frac{3}{4}$ Empty	I2CM2_INTEN.tx_fifo_3q_empty	I2CM2_INTFL.tx_fifo_3q_empty
		I2CM2 RX FIFO Not Empty	I2CM2_INTEN.rx_fifo_not_empty	I2CM2_INTFL.rx_fifo_not_empty
		I2CM2 RX FIFO $\frac{1}{2}$ Full	I2CM2_INTEN.rx_fifo_2q_full	I2CM2_INTFL.rx_fifo_2q_full
		I2CM2 RX FIFO $\frac{3}{4}$ Full	I2CM2_INTEN.rx_fifo_3q_full	I2CM2_INTFL.rx_fifo_3q_full
		I2CM2 RX FIFO Full	I2CM2_INTEN.rx_fifo_full	I2CM2_INTFL.rx_fifo_full

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
58	42	I2CS Update to Mailbox Byte 0	I2CS_INTEN.byte0	I2CS_INTFL.byte0
		I2CS Update to Mailbox Byte 1	I2CS_INTEN.byte1	I2CS_INTFL.byte1
		I2CS Update to Mailbox Byte 2	I2CS_INTEN.byte2	I2CS_INTFL.byte2
		I2CS Update to Mailbox Byte 3	I2CS_INTEN.byte3	I2CS_INTFL.byte3
		I2CS Update to Mailbox Byte 4	I2CS_INTEN.byte4	I2CS_INTFL.byte4
		I2CS Update to Mailbox Byte 5	I2CS_INTEN.byte5	I2CS_INTFL.byte5
		I2CS Update to Mailbox Byte 6	I2CS_INTEN.byte6	I2CS_INTFL.byte6
		I2CS Update to Mailbox Byte 7	I2CS_INTEN.byte7	I2CS_INTFL.byte7
		I2CS Update to Mailbox Byte 8	I2CS_INTEN.byte8	I2CS_INTFL.byte8
		I2CS Update to Mailbox Byte 9	I2CS_INTEN.byte9	I2CS_INTFL.byte9
		I2CS Update to Mailbox Byte 10	I2CS_INTEN.byte10	I2CS_INTFL.byte10
		I2CS Update to Mailbox Byte 11	I2CS_INTEN.byte11	I2CS_INTFL.byte11
		I2CS Update to Mailbox Byte 12	I2CS_INTEN.byte12	I2CS_INTFL.byte12
		I2CS Update to Mailbox Byte 13	I2CS_INTEN.byte13	I2CS_INTFL.byte13
		I2CS Update to Mailbox Byte 14	I2CS_INTEN.byte14	I2CS_INTFL.byte14
		I2CS Update to Mailbox Byte 15	I2CS_INTEN.byte15	I2CS_INTFL.byte15

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
58	42	I2CS Update to Mailbox Byte 16	I2CS_INTEN.byte0	I2CS_INTFL.byte0
		I2CS Update to Mailbox Byte 17	I2CS_INTEN.byte17	I2CS_INTFL.byte17
		I2CS Update to Mailbox Byte 18	I2CS_INTEN.byte18	I2CS_INTFL.byte18
		I2CS Update to Mailbox Byte 19	I2CS_INTEN.byte19	I2CS_INTFL.byte19
		I2CS Update to Mailbox Byte 20	I2CS_INTEN.byte20	I2CS_INTFL.byte20
		I2CS Update to Mailbox Byte 21	I2CS_INTEN.byte21	I2CS_INTFL.byte21
		I2CS Update to Mailbox Byte 22	I2CS_INTEN.byte22	I2CS_INTFL.byte22
		I2CS Update to Mailbox Byte 23	I2CS_INTEN.byte23	I2CS_INTFL.byte23
		I2CS Update to Mailbox Byte 24	I2CS_INTEN.byte24	I2CS_INTFL.byte24
		I2CS Update to Mailbox Byte 25	I2CS_INTEN.byte25	I2CS_INTFL.byte25
		I2CS Update to Mailbox Byte 26	I2CS_INTEN.byte26	I2CS_INTFL.byte26
		I2CS Update to Mailbox Byte 27	I2CS_INTEN.byte27	I2CS_INTFL.byte27
		I2CS Update to Mailbox Byte 28	I2CS_INTEN.byte28	I2CS_INTFL.byte28
		I2CS Update to Mailbox Byte 29	I2CS_INTEN.byte29	I2CS_INTFL.byte29
		I2CS Update to Mailbox Byte 30	I2CS_INTEN.byte30	I2CS_INTFL.byte30
		I2CS Update to Mailbox Byte 31	I2CS_INTEN.byte31	I2CS_INTFL.byte31

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
59	43	SPI Master 0 TX Stalled	SPI0_INTEN.tx_stalled	SPI0_INTFL.tx_stalled
		SPI Master 0 RX Stalled	SPI0_INTEN.rx_stalled	SPI0_INTFL.rx_stalled
		SPI Master 0 TX Ready	SPI0_INTEN.tx_ready	SPI0_INTFL.tx_ready
		SPI Master 0 RX Done	SPI0_INTEN.rx_done	SPI0_INTFL.rx_done
		SPI Master 0 TX FIFO Almost Empty	SPI0_INTEN.tx_fifo_ae	SPI0_INTFL.tx_fifo_ae
		SPI Master 0 RX FIFO Almost Full	SPI0_INTEN.rx_fifo_af	SPI0_INTFL.rx_fifo_af
60	44	SPI Master 1 TX Stalled	SPI1_INTEN.tx_stalled	SPI1_INTFL.tx_stalled
		SPI Master 1 RX Stalled	SPI1_INTEN.rx_stalled	SPI1_INTFL.rx_stalled
		SPI Master 1 TX Ready	SPI1_INTEN.tx_ready	SPI1_INTFL.tx_ready
		SPI Master 1 RX Done	SPI1_INTEN.rx_done	SPI1_INTFL.rx_done
		SPI Master 1 TX FIFO Almost Empty	SPI1_INTEN.tx_fifo_ae	SPI1_INTFL.tx_fifo_ae
		SPI Master 1 RX FIFO Almost Full	SPI1_INTEN.rx_fifo_af	SPI1_INTFL.rx_fifo_af
61	42	SPI Master 2 TX Stalled	SPI2_INTEN.tx_stalled	SPI2_INTFL.tx_stalled
		SPI Master 2 RX Stalled	SPI2_INTEN.rx_stalled	SPI2_INTFL.rx_stalled
		SPI Master 2 TX Ready	SPI2_INTEN.tx_ready	SPI2_INTFL.tx_ready
		SPI Master 2 RX Done	SPI2_INTEN.rx_done	SPI2_INTFL.rx_done
		SPI Master 2 TX FIFO Almost Empty	SPI2_INTEN.tx_fifo_ae	SPI2_INTFL.tx_fifo_ae
		SPI Master 2 RX FIFO Almost Full	SPI2_INTEN.rx_fifo_af	SPI2_INTFL.rx_fifo_af

Exception	IRQ Num	Interrupt Condition	Interrupt Enable	Interrupt Flag
63	44	1-Wire Master Reset Done	OWM_INTEN.ow_reset_done	OWM_INTFL.ow_reset_done
		1-Wire Master TX Data Empty	OWM_INTEN.tx_data_empty	OWM_INTFL.tx_data_empty
		1-Wire Master RX Data Ready	OWM_INTEN.rx_data_ready	OWM_INTFL.rx_data_ready
		1-Wire Master Data Line Short	OWM_INTEN.line_short	OWM_INTFL.line_short
		1-Wire Master Data Line Low	OWM_INTEN.line_low	OWM_INTFL.line_low
64	45	ADC Done	ADC_INTR.adc_done_ie	ADC_INTR.adc_done_if
		ADC Reference Ready	ADC_INTR.adc_ref_ready_ie	ADC_INTR.adc_ref_ready_if
		ADC Sample Above Hi Limit	ADC_INTR.adc_hi_limit_ie	ADC_INTR.adc_hi_limit_if
		ADC Sample Below Lo Limit	ADC_INTR.adc_lo_limit_ie	ADC_INTR.adc_lo_limit_if
		ADC Overflow	ADC_INTR.adc_overflow_ie	ADC_INTR.adc_overflow_if
		ADC RO Calibration Done	ADC_INTR.ro_cal_done_ie	ADC_INTR.ro_cal_done_if

4.3 Resets and Reset Sources

This section describes resets and the reset sources for the **MAX32620**.

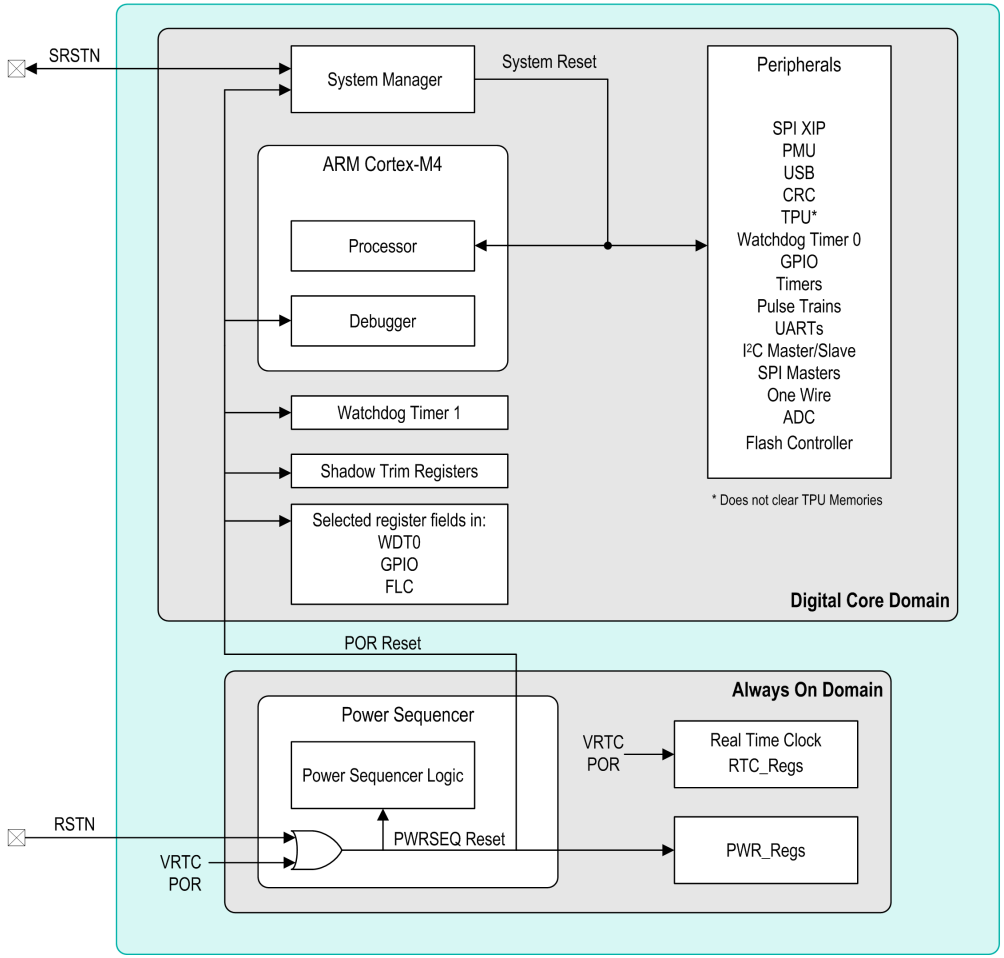

Figure 4.4: Reset Block Diagram

4.3.1 System Reset

The system reset triggers a reset of the operational state of most digital and analog blocks, including the ARM Cortex-M4 CPU.

Sources that can trigger a system reset include:

- Active low signal on the SRSTN external reset pin.
- Firmware requested reset.
- Reset generated by ARM core due to fault.
- Reset generated by watchdog timer WDT0.

4.3.1.1 System Reset Pin

SRSTN: System Reset, Active-Low Input/Output

The **MAX32620** remains in system reset while an external reset signal (active low) is asserted on this pin. Once the external reset is released, the SRSTN pin logic state will return to high due to the internal pullup on the pin.

When system reset is asserted, the device resets the ARM core, most digital registers, and most peripherals (resetting most of the core logic on the V_{DD12} supply). This reset does not affect the POR-only registers, RTC, or ARM JTAG debug interface.

After the device senses assertion of SRSTN, the pin will automatically reconfigure as an output and will be driven active low by the **MAX32620**. The device continues to output for six system clock cycles and then repeats the input sensing/output driving until SRSTN is sensed as de-asserted.

This pin is internally connected with an internal 25kohm pullup to the V_{RTC} supply, and may be left unconnected. SRSTN is normally connected to a JTAG header pin for In-circuit System Programming (ICSP).

4.3.2 Power-On Reset (POR)

The Power-On Reset condition causes the **MAX32620** to reset all functions and blocks that are reset by a System Reset. Additionally, the Power-On Reset also resets additional blocks and registers including certain system configuration registers and the ARM JTAG debugging interface.

A POR Reset is typically triggered when a power failure occurs on one or more digital supplies as configured by the user.

Other sources that can trigger a POR Reset include:

- Shutdown to LP0:STOP (digital core state is lost).
- Reset generated by watchdog timer WDT1.
- Firmware requested system reboot.

4.3.3 Power Sequencer Reset (PWRSEQ_Reset)

Power sequencer logic (which operates in the 'always on' logic domain powered by V_{RTC} generates the Power-On Reset (POR) output.

The power sequencer itself (along with its associated registers) is reset when a Power Sequencer Reset occurs. This also causes the Power-On Reset and System Reset signals to be propagated downstream to the rest of the system.

A Power Sequencer Reset occurs under the following conditions:

- Active low signal on the RSTN pin.
- Failure of the backup power supply V_{RTC} (VRTC_POR).

4.3.3.1 RSTN (PWRSEQ Reset) Pin

RSTN: Power-on Reset, Active-Low Input

The **MAX32620** remains in reset while this pin is in its active state. When the pin transitions to its inactive state, the device performs a Power Sequencer Reset and begins execution. This pin is internally connected with an internal 25kohm pullup to the V_{RTC} supply. This pin can be left unconnected.

4.3.4 VRTC Power On Reset (VRTC_POR)

A VRTC Power On Reset is the deepest reset level in the system. This type of reset occurs only when a failure occurs on the VRTC backup power supply.

When a VRTC_POR reset occurs, it has the following effects:

- All other reset signals are asserted: PWRSEQ Reset, POR Reset, and System Reset.
- All registers, blocks, and internal device states are reset, with the exception of information retained in non-volatile internal flash memory.

Generally, registers in the RTC module are only reset by VRTC_POR.

4.4 Device Clock Sources and Configuration

4.4.1 Clock Sources

There are a number of global and local clocks used by different peripherals and subsystems on the **MAX32620**. All of these are generated (either directly or indirectly) from one of the clock sources described below. These clock sources are not synchronized to one another; in the cases where a peripheral or block makes use of more than one clock source, clock synchronization between the domains is performed locally at that point.

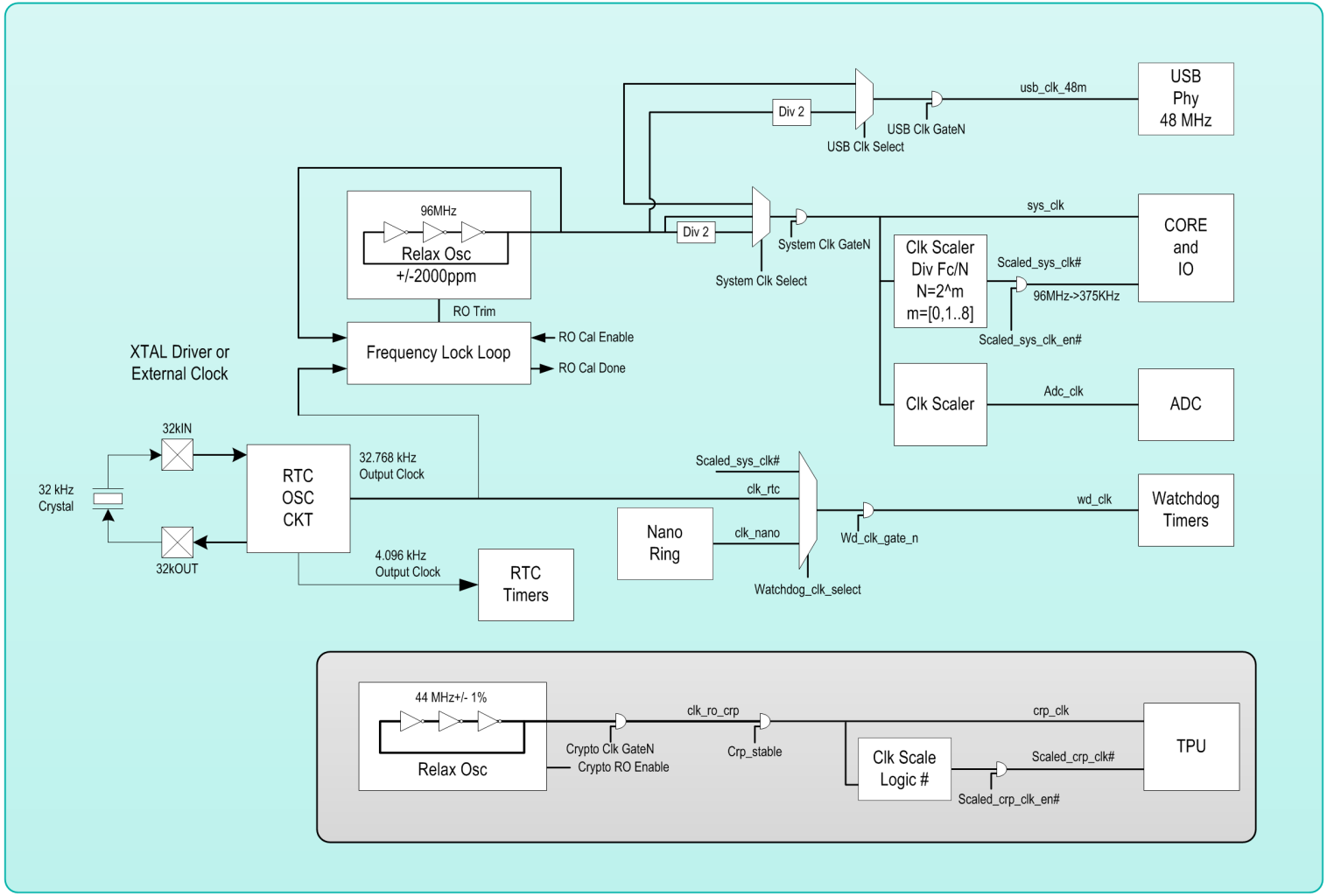

Figure 4.5: Clock Subsystem Diagram

4.4.1.1 System Relaxation Oscillator (96MHz)

The primary clock source on the **MAX32620** is a high-speed relaxation oscillator targeted to run at 96MHz. The output of this oscillator is (by default) the primary system oscillator, which is used as the timing basis for most blocks and peripherals on the **MAX32620**, including the ARM CPU and the AMBA bus matrix.

The system relaxation oscillator is factory trimmed during producton to run at a frequency of 96MHz with an accuracy of $\pm$ 1.6%. This level of accuracy is sufficient for the core digital logic on the device and all peripherals except for the USB interface and the Real Time Clock (RTC). The RTC uses its own dedicated 32kHz external crystal oscillator (or resonator) as a timing basis.

The USB runs from the 96MHz oscillator output, but it requires a higher frequency accuracy than can be obtained using the one-time factory trim. In order to improve the accuracy of the 96MHz oscillator to the level required by the USB interface, the 96MHz oscillator can be calibrated against the output from the 32kHz RTC oscillator.

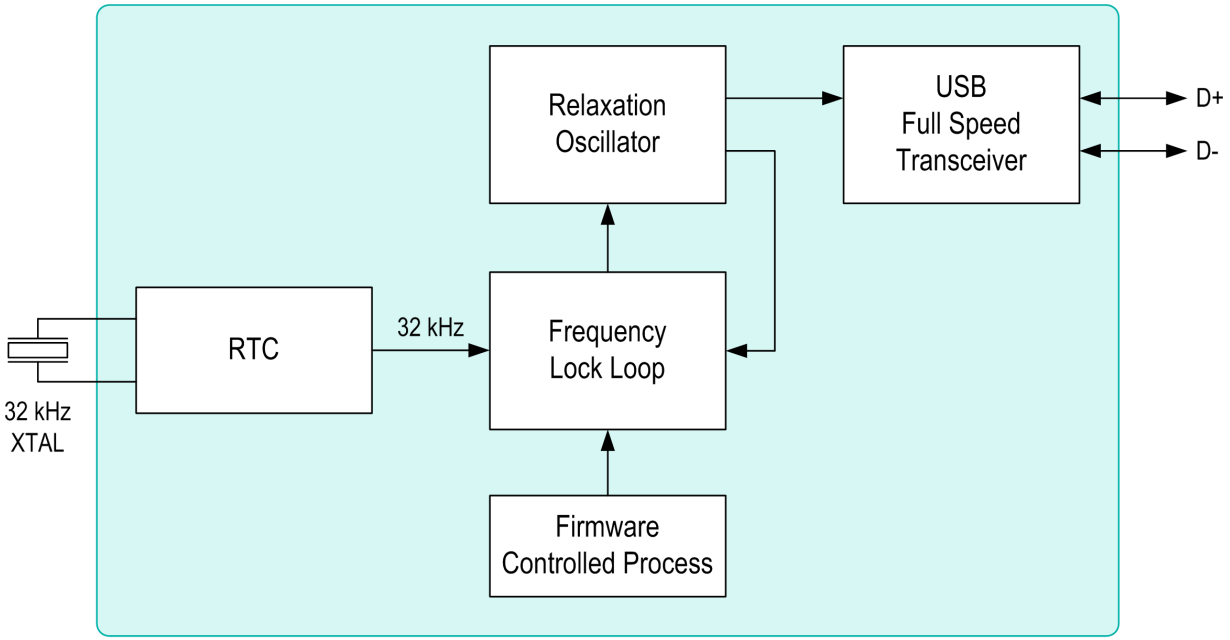

Figure 4.6: Calibrating the System Relaxation Oscillator

Using this method, the accuracy of the 96MHz oscillator output can be improved to $\pm$ 0.25%. This feature eliminates the need for an external high frequency crystal,

PLL and driver circuits. The calibration process is controlled by application firmware; calibration should be performed periodically to compensate for any frequency shifts induced by temperature and supply voltage changes.

4.4.1.1.1 Operation and Power Mode Options

The system relaxation oscillator is always shut off in the LP0:STOP mode to conserve power.

By default, the system relaxation oscillator is turned off in the LP1:STANDBY mode. When the **MAX32620** exits the LP1:STANDBY mode to return to active LP3:RUN state, the power sequencer re-enables the system relaxation oscillator automatically. However, this requires a delay of 5μs. It is possible to configure the system relaxation oscillator to continue running during LP1:STANDBY mode by setting the power sequencer control bit PWRSEQ_REG0.pwr_roen_slp to 1. This will remove the 5μs startup delay at a cost of additional power consumption during LP1:STANDBY mode.

The system relaxation oscillator must always be running during the active operation modes LP2:PMU and LP3:RUN.

4.4.1.1.2 Operation and Power Mode Options

In order for the USB interface to be used, the frequency calibration process must be used to calibrate the system relaxation oscillator against the output of the 32kHz RTC oscillator. The steps for the frequency calibration are outlined below.

Step	Action	Register I/O
1	Enable the 32kHz RTC (if not already enabled)	PWRSEQ_REG0.pwr_rtcen_run = 1
2	Read Relaxation Oscillator Flash Trim shadow register	InitTrim[8:0] = PWRSEQ_REG6.pwr_trim_osc_vref
3	Write InitTrim to freq. cal. initial condition register	ADC_RO_CAL1.trm_init = InitTrim[8:0]
4	Load Initial Trim to active frequency trim register	Write ADC_RO_CAL0.ro_cal_load to 1 (self-clearing)
5	Enable frequency loop to control RO trim	ADC_RO_CAL0.ro_cal_en = 1
6	Run Frequency Calibration	ADC_RO_CAL0.ro_cal_run = 1
8	After 50mS, stop Frequency Calibration	ADC_RO_CAL0.ro_cal_run = 0
9	Read the final frequency trim value	FinalTrim[8:0] = ADC_RO_CAL0.ro_trm
10	Write Final trim to Sys RO Flash Trim shadow register	PWRSEQ_REG6.pwr_trim_osc_vref = FinalTrim[8:0]
11	Disable RTC oscillator (if not needed for RTC timer)	PWRSEQ_REG0.pwr_rtcen_run = 0

The **MAX32620** also has the ability to self-time the frequency calibration and generate an interrupt when complete, as outlined in the table below.

Step	Action	Register I/O
1	Enable the 32kHz RTC (if not already enabled)	PWRSEQ_REG0.pwr_rtcen_run = 1
2	Enable the RO Calibration complete interrupt	ADC_INTR.ro_cal_done_ie = 1
3	Clear RO Calibration complete interrupt	ADC_INTR.ro_cal_done_if = 1 (write 1 to clear)
4	Read Relaxation Oscillator Flash Trim shadow register	InitTrim[8:0] = PWRSEQ_REG6.pwr_trim_osc_vref
5	Write InitTrim to freq. cal. initial condition register	ADC_RO_CAL1.trm_init = InitTrim[8:0]
6	Load Initial Trim to active frequency trim register	Write ADC_RO_CAL0.ro_cal_load to 1 (self-clearing)
7	Enable frequency loop to control RO trim	ADC_RO_CAL0.ro_cal_en = 1
8	Run Frequency Calibration in Atomic mode	ADC_RO_CAL0.ro_cal_atomic = 1
9	Detect Interrupt Service request	...
10	Stop Frequency Calibration	ADC_RO_CAL0.ro_cal_run = 0
11	Disable the RO Calibration complete interrupt	ADC_INTR.ro_cal_done_ie = 0
12	Read the final frequency trim value	FinalTrim[8:0] = ADC_RO_CAL0.ro_trm
13	Write Final trim to Sys RO Flash Trim shadow register	PWRSEQ_REG6.pwr_trim_osc_vref = FinalTrim[8:0]
14	Disable RTC oscillator (if not needed for RTC timer)	PWRSEQ_REG0.pwr_rtcen_run = 0

4.4.1.2 TPU Relaxation Oscillator (44MHz)

The **MAX32620** provides a dedicated on-chip oscillator used to supply a separate isolated clock to reduce opportunities of clock interference attacks. Specifically, timing-based analysis and fault injection attacks on sensitive functions of the device are mitigated with this feature. Vulnerabilities protected by the isolated cryptographic internal oscillator include timing-based and security functions such as the AES and the MAA among others.

Effectively decoupling these sensitive functions from the main system clock allows encryption and decryption operations to take a consistent amount of time regardless of the current system clock rate. Also, this provides a more variable (and not externally observable) clock to reduce the opportunity for an attacker to perform power or timing analysis attacks against the cryptographic and security functions.

The TPU relaxation oscillator is targeted to run at approximately 44MHz. This oscillator is always powered on in active power modes (LP3:RUN and LP2:PMU) and is disabled in LP0:STOP and LP1:STANDBY modes. In order for the output from the TPU oscillator to be used by other modules in the **MAX32620**, the cryptographic clock source must be enabled by setting CLKMAN_CLK_CONFIG.crypto_enable to 1.

4.4.1.3 RTC External Oscillator (32kHz)

The external oscillator on the **MAX32620** is designed to be used with a standard 32.768kHz watch crystal. These crystals are very high impedance and do not require a high current oscillator circuit. The 32kHz oscillator has been designed specifically to handle these crystals and is compatible with their high impedance and limited power handling capability. The oscillator power dissipation is very low to minimize power draw on the V_{RTC} power supply.

The oscillator is designed to be self-biasing. An external resistor should NOT be connected across the crystal. A resonator can also be used with the 32kHz oscillator if needed. The oscillator frequency can be adjusted for 25C accuracy, as well as for temperature effects. See the Real Time Clock (RTC) section for details.

Firmware Control of the External 32kHz Clock

There are two bits that control whether the RTC timer and 32kHz external oscillator are enabled. In order to enable the RTC timer and 32kHz external oscillator during LP3:RUN and LP2:PMU, the PWRSEQ_REG0.pwr_rtcen_run bit must be set to 1. In order to enable the RTC timer and 32kHz external oscillator during LP1:STANDBY and LP0:STOP, the PWRSEQ_REG0.pwr_rtcen_slp bit must be set to 1.

4.4.2 Clock Configuration

The **MAX32620** clock subsystem provides a high degree of flexibility to optimize an embedded system implementation. The user has options to trade-off external components depending on the circuitry required by the desired application(s). A block diagram of the **MAX32620** clock subsystem is provided in the figure above.

4.4.2.1 System Clock Configuration

The main system clock on the **MAX32620** is used as a timing base by many of the underlying architectural blocks on the device, including the AHB and APB buses, the SRAM, internal flash memory, and various system management functions (such as clock management, power management, and I/O function management).

This primary system clock is also known as the "system clock source", or the undivided system clock. This is the highest-frequency clock present on the device. The selection for this clock is made by setting the CLKMAN_CLK_CTRL.system_source_select register field as detailed below.

Field Setting	System Clock Source Selection
0	96MHz Relaxation Oscillator (divided by 2)
1	96Mhz Relaxation Oscillator
others	(Other values are reserved for internal test and should not be used.)

In addition to supplying certain blocks directly, this clock is also used as a basis for a number of secondary "module clocks" or "peripheral clocks". These secondary clocks allow certain blocks or peripherals (such as pulse trains, I²C, UART, or SPI) to run at frequency that is lower than the main system clock source frequency. This is done by configuring clock scaling blocks that generate local module clocks by dividing down the system clock source.

For more information, refer to the register descriptions in the CLKMAN module. The clock scaling registers (for the clocks derived from the system clock source) are all named along the lines "CLKMAN_SYS_CLK_CTRL_<n>_<module>".

4.4.2.2 Cryptographic Clock Configuration

The cryptographic and security functions (housed within the TPU) on the **MAX32620** use the TPU relaxation oscillator output as their clock source. Similar to the system clock scaling controls described above, each module that uses the TPU relaxation oscillator output as a clock source has the option to divide the clock source down to generate a modified local module clock.

- **AES**: Module clock is enabled/configured using CLKMAN_CRYPT_CLK_CTRL_0_AES.
- **MAA**: Module clock is enabled/configured using CLKMAN_CRYPT_CLK_CTRL_1_MAA.
- **PRNG**: Module clock is enabled/configured using CLKMAN_CRYPT_CLK_CTRL_2_PRNG.

Each of the cryptographic clocks above are disabled by default and must be enabled before the related module/function can be used.

4.4.2.3 ADC Clock Configuration

The ADC on the **MAX32620** runs at a fixed frequency of 8MHz. The clock source for the ADC is fixed and is derived directly from the 96MHz system relaxation oscillator output by dividing this oscillator output by 12.

Before using the ADC, the ADC oscillator output must be enabled by setting CLKMAN_CLK_CTRL.adc_clock_enable to 1.

4.4.3 Registers (CLKMAN)

Address	Register	Access	Description	Reset By
0x40000400	CLKMAN_CLK_CONFIG	R/W	System Clock Configuration	Sys
0x40000404	CLKMAN_CLK_CTRL	R/W	System Clock Controls	Sys POR
0x40000408	CLKMAN_INTFL	W1C	Interrupt Flags	Sys
0x4000040C	CLKMAN_INTEN	R/W	Interrupt Enable/Disable Controls	Sys
0x40000410	CLKMAN_TRIM_CALC	***	Trim Calculation Controls	Sys
0x40000414	CLKMAN_I2C_TIMER_CTRL	R/W	I2C Timer Control	Sys
0x40000418	CLKMAN_CM4_START_CLK_EN0	R/W	CM4 Start Clock on Interrupt Enable 0	Sys
0x4000041C	CLKMAN_CM4_START_CLK_EN1	R/W	CM4 Start Clock on Interrupt Enable 1	Sys
0x40000440	CLKMAN_SYS_CLK_CTRL_0_CM4	R/W	Control Settings for CLK0 - Cortex M4 Clock	Sys
0x40000444	CLKMAN_SYS_CLK_CTRL_1_SYNC	R/W	Control Settings for CLK1 - Synchronizer Clock	Sys
0x40000448	CLKMAN_SYS_CLK_CTRL_2_SPIX	R/W	Control Settings for CLK2 - SPI XIP Clock	Sys
0x4000044C	CLKMAN_SYS_CLK_CTRL_3_PRNG	R/W	Control Settings for CLK3 - PRNG Clock	Sys
0x40000450	CLKMAN_SYS_CLK_CTRL_4_WDT0	R/W	Control Settings for CLK4 - Watchdog Timer 0	Sys
0x40000454	CLKMAN_SYS_CLK_CTRL_5_WDT1	R/W	Control Settings for CLK5 - Watchdog Timer 1	Sys
0x40000458	CLKMAN_SYS_CLK_CTRL_6_GPIO	R/W	Control Settings for CLK6 - Clock for GPIO Ports	Sys
0x4000045C	CLKMAN_SYS_CLK_CTRL_7_PT	R/W	Control Settings for CLK7 - Source Clock for All Pulse Trains	Sys
0x40000460	CLKMAN_SYS_CLK_CTRL_8_UART	R/W	Control Settings for CLK8 - Source Clock for All UARTs	Sys
0x40000464	CLKMAN_SYS_CLK_CTRL_9_I2CM	R/W	Control Settings for CLK9 - Source Clock for All I2C Masters	Sys
0x40000468	CLKMAN_SYS_CLK_CTRL_10_I2CS	R/W	Control Settings for CLK10 - Source Clock for I2C Slave	Sys
0x4000046C	CLKMAN_SYS_CLK_CTRL_11_SPI0	R/W	Control Settings for CLK11 - SPI Master 0	Sys
0x40000470	CLKMAN_SYS_CLK_CTRL_12_SPI1	R/W	Control Settings for CLK12 - SPI Master 1	Sys
0x40000474	CLKMAN_SYS_CLK_CTRL_13_SPI2	R/W	Control Settings for CLK13 - SPI Master 2	Sys

Address	Register	Access	Description	Reset By
0x40000478	CLKMAN_SYS_CLK_CTRL_14_SPIB	R/W	Control Settings for CLK14 - SPI Bridge Clock	Sys
0x4000047C	CLKMAN_SYS_CLK_CTRL_15_OWM	R/W	Control Settings for CLK15 - 1-Wire Master Clock	Sys
0x40000500	CLKMAN_CRYPT_CLK_CTRL_0_AES	R/W	Control Settings for Crypto Clock 0 - AES	Sys
0x40000504	CLKMAN_CRYPT_CLK_CTRL_1_MAA	R/W	Control Settings for Crypto Clock 1 - MAA	Sys
0x40000508	CLKMAN_CRYPT_CLK_CTRL_2_PRNG	R/W	Control Settings for Crypto Clock 2 - PRNG	Sys
0x40000540	CLKMAN_CLK_GATE_CTRL0	R/W	Dynamic Clock Gating Control Register 0	Sys
0x40000544	CLKMAN_CLK_GATE_CTRL1	R/W	Dynamic Clock Gating Control Register 1	Sys
0x40000548	CLKMAN_CLK_GATE_CTRL2	R/W	Dynamic Clock Gating Control Register 2	Sys

4.4.3.1 CLKMAN_CLK_CONFIG

CLKMAN_CLK_CONFIG.crypto_enable

Field	Bits	Sys Reset	Access	Description
crypto_enable	0	0	R/W	Cryptographic (TPU) Relaxation Oscillator Enable

- 0: Disabled
- 1: Crypto/TPU relaxation oscillator is enabled.

CLKMAN_CLK_CONFIG.crypto_stability_count

Field	Bits	Sys Reset	Access	Description
crypto_stability_count	7:4	5	R/W	Crypto Oscillator Stability Select

Defines terminal count for crypto oscillator stable indicator in terms of crypto clocks as (# of clocks) = 2^(field value + 8)

- 0: 256 clocks (2^8)
- 1: 512 clocks (2^9)
- 2: 1024 clocks (2^10)
- 3: 2048 clocks (2^11)
- 4: 4096 clocks (2^12)
- 5: 8192 clocks (2^13)
- 6: 16384 clocks (2^14)
- 7: 32768 clocks (2^15)
- 8: 65536 clocks (2^16)
- 9: 131072 clocks (2^17)
- 10: 262144 clocks (2^18)
- 11: 524288 clocks (2^19)
- 12: 1048576 clocks (2^20)
- 13: 2097152 clocks (2^21)
- 14: 4194304 clocks (2^22)
- 15: 8388608 clocks (2^23)

4.4.3.2 CLKMAN_CLK_CTRL

CLKMAN_CLK_CTRL.system_source_select

Field	Bits	Sys Reset	Alt Reset	Access	Description
system_source_select	1:0	no effect	POR:0	R/W	System Clock Source Select

- 0: 96MHz Relaxation Osc output (divided by 2)
- 1: 96MHz Relaxation Osc output

Read value is actual mux select, and may lag value written by several clocks due to glitchless clock switching circuit.

CLKMAN_CLK_CTRL.usb_clock_enable

Field	Bits	Sys Reset	Access	Description
usb_clock_enable	4	0	R/W	USB Clock Enable

- 0: Clock to USB is gated off.
- 1: Clock to USB is enabled.

CLKMAN_CLK_CTRL.usb_clock_select

Field	Bits	Sys Reset	Access	Description
usb_clock_select	5	0	R/W	USB Clock Select

- 0: 96MHz Relaxation Osc output (divided by 2)
- 1: Reserved (Internal Maxim test only, do not use)

CLKMAN_CLK_CTRL.crypto_clock_enable

Field	Bits	Sys Reset	Access	Description
crypto_clock_enable	8	0	R/W	Crypto Clock Enable

- 0: Crypto Clock source is gated off.
- 1: Crypto Clock source is enabled for use by other modules.

CLKMAN_CLK_CTRL.rtos_mode

Field	Bits	Sys Reset	Access	Description
rtos_mode	12	0	R/W	Enable RTOS Mode for SysTick Timers

- 0: In this mode (default), when the device enters LP2 or LP1 mode, the free-running clock (FCLK) to the ARM core will be shut off. This has two effects. First, the input clock to the SysTick timer will be gated off, which means that SysTick cannot be used to wake up the device. The SysTick timer will not receive clock pulses during LP2 or LP1 even if the 32kHz clock has been selected as the SysTick clock source. Second, the ARM core will not be accessible via the debug port during LP2 or LP1.
- 1: In this mode, the FCLK will continue running to the ARM core even during LP2 or LP1, which results in additional power consumption during these low power modes. With FCLK always free-running, the ARM CPU can still be accessed using the debug port even in LP2 or LP1. If the SysTick timer has been selected to use STCLK (the 32kHz clock) as its input clock source, then in this mode, SysTick will continue running normally in LP2 or LP1. This means that SysTick can be used to wake the device from these low power modes, and also that SysTick can be used to accurately time a period's duration even if the time period includes entry into and/or exit from LP2/LP1.

CLKMAN_CLK_CTRL.wdt0_clock_enable

Field	Bits	Sys Reset	Alt Reset	Access	Description
wdt0_clock_enable	16	no effect	POR:0	R/W	Watchdog 0 Clock Enable

- 0: Clock to WDT0 is gated off.
- 1: Clock to WDT0 is enabled.

CLKMAN_CLK_CTRL.wdt0_clock_select

Field	Bits	Sys Reset	Alt Reset	Access	Description
wdt0_clock_select	18:17	no effect	POR:0	R/W	Watchdog 0 Clock Source Select

Selects the source for the watchdog 0 clock:

- 0: Scaled Sys Clock Source (as set by SYS_CLK_CTRL_4_WDT0)
- 1: 32.768 kHz RTC oscillator output
- 2: 96MHz ring oscillator
- 3: Nano-ring oscillator

Read value is actual mux select, and may lag value written by several clocks due to glitchless clock switching circuit.

CLKMAN_CLK_CTRL.wdt1_clock_enable

Field	Bits	Sys Reset	Alt Reset	Access	Description
wdt1_clock_enable	20	no effect	POR:0	R/W	Watchdog 1 Clock Enable

- 0: Clock to WDT1 is gated off.
- 1: Clock to WDT1 is enabled.

CLKMAN_CLK_CTRL.wdt1_clock_select

Field	Bits	Sys Reset	Alt Reset	Access	Description
wdt1_clock_select	22:21	no effect	POR:0	R/W	Watchdog 1 Clock Source Select

Selects the source for the watchdog 1 clock:

- 0: Scaled Sys Clock Source (as set by SYS_CLK_CTRL_5_WDT1)
- 1: 32.768 kHz RTC oscillator output
- 2: 96MHz ring oscillator
- 3: Nano-ring oscillator

Read value is actual mux select, and may lag value written by several clocks due to glitchless clock switching circuit.

CLKMAN_CLK_CTRL.adc_clock_enable

Field	Bits	Sys Reset	Access	Description
adc_clock_enable	24	0	R/W	ADC Clock Enable

- 0: Clock to ADC is gated off.
- 1: Clock to ADC is enabled.

4.4.3.3 CLKMAN_INTFL

CLKMAN_INTFL.crypto_stable

Field	Bits	Sys Reset	Access	Description
crypto_stable	0	0	W1C	Crypto Oscillator Stable Interrupt Flag

Write 1 to clear.

Set to 1 by hardware when the crypto oscillator is considered stable.

4.4.3.4 CLKMAN_INTEN

CLKMAN_INTEN.crypto_stable

Field	Bits	Sys Reset	Access	Description
crypto_stable	0	0	R/W	Crypto Oscillator Stable Interrupt Enable

- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

4.4.3.5 CLKMAN_TRIM_CALC

CLKMAN_TRIM_CALC.trim_clk_sel

Field	Bits	Sys Reset	Access	Description
trim_clk_sel	0	0	R/W	Trim Clock Select

Selects which trimmable clock to measure for the calculation.

- 0: 24MHz ring oscillator
- 1: Crypto ring oscillator (divided by 2)

CLKMAN_TRIM_CALC.trim_calc_start

Field	Bits	Sys Reset	Access	Description
trim_calc_start	1	0	R/W	Start Trim Calculation

Write to 1 to start a new trim calculation; self clearing.

CLKMAN_TRIM_CALC.trim_calc_completed

Field	Bits	Sys Reset	Access	Description
trim_calc_completed	2	0	R/O	Trim Calculation Completed

Status bit; this bit is set to 1 by hardware when a trim calculation is completed, and is cleared to 0 by hardware when a new trim calculation is started.

CLKMAN_TRIM_CALC.trim_enable

Field	Bits	Sys Reset	Access	Description
trim_enable	3	0	R/W	Trim Logic Enable

- 0: Clear to put trim calculation logic in low-power state.
- 1: Set to this value before using trim calculation.

CLKMAN_TRIM_CALC.trim_calc_results

Field	Bits	Sys Reset	Access	Description
trim_calc_results	25:16	n/a	R/O	Trim Calculation Results

Results of trim calculation, valid when the Trim Calculation Completed bit is set to 1. This result consists of the number of trimmable clock (selected by bit 0) edges seen during the time taken for 256 HFX clock periods to elapse.

4.4.3.6 CLKMAN_I2C_TIMER_CTRL

CLKMAN_I2C_TIMER_CTRL.i2c_1ms_timer_en

Field	Bits	Sys Reset	Access	Description
i2c_1ms_timer_en	0	0	R/W	I2C 1ms Timer Enable

1:Enables the general purpose timer used by the I2C blocks for determining timeout events.

4.4.3.7 CLKMAN_CM4_START_CLK_EN0

CLKMAN_CM4_START_CLK_EN0.ints

Field	Bits	Sys Reset	Access	Description
ints	31:0	FFFFFFFFh	R/W	Interrupt Sources 0-31

Setting a bit to 1 causes the CM4 clock to start when the corresponding interrupt IRQ source is asserted.

4.4.3.8 CLKMAN_CM4_START_CLK_EN1

CLKMAN_CM4_START_CLK_EN1.ints

Field	Bits	Sys Reset	Access	Description
ints	31:0	FFFFFFFFh	R/W	Interrupt Sources 32-49

Setting a bit to 1 causes the CM4 clock to start when the corresponding interrupt IRQ source is asserted.

4.4.3.9 CLKMAN_SYS_CLK_CTRL_0_CM4

CLKMAN_SYS_CLK_CTRL_0_CM4.cm4_clk_scale

Field	Bits	Sys Reset	Access	Description
cm4_clk_scale	3:0	1	R/W	Control Settings for CLK0 - Cortex M4 Clock

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)

- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is enabled by default following reset.

4.4.3.10 CLKMAN_SYS_CLK_CTRL_1_SYNC

CLKMAN_SYS_CLK_CTRL_1_SYNC.sync_clk_scale

Field	Bits	Sys Reset	Access	Description
sync_clk_scale	3:0	0000b	R/W	Control Settings for CLK1 - Synchronizer Clock

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.11 CLKMAN_SYS_CLK_CTRL_2_SPIX

CLKMAN_SYS_CLK_CTRL_2_SPIX.spix_clk_scale

Field	Bits	Sys Reset	Access	Description
spix_clk_scale	3:0	0000b	R/W	Control Settings for CLK2 - SPI XIP Clock

- 0: CLK is Disabled

- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.12 CLKMAN_SYS_CLK_CTRL_3_PRNG

CLKMAN_SYS_CLK_CTRL_3_PRNG.prng_clk_scale

Field	Bits	Sys Reset	Access	Description
prng_clk_scale	3:0	0000b	R/W	Control Settings for CLK3 - PRNG Clock

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.13 CLKMAN_SYS_CLK_CTRL_4_WDT0

CLKMAN_SYS_CLK_CTRL_4_WDT0.watchdog0_clk_scale

Field	Bits	Sys Reset	Access	Description
watchdog0_clk_scale	3:0	0000b	R/W	Control Settings for CLK4 - Watchdog Timer 0

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

Note If this clock is selected as the watchdog clock source for WDT0, then any system reset will halt that watchdog timer.

4.4.3.14 CLKMAN_SYS_CLK_CTRL_5_WDT1

CLKMAN_SYS_CLK_CTRL_5_WDT1.watchdog1_clk_scale

Field	Bits	Sys Reset	Access	Description
watchdog1_clk_scale	3:0	0000b	R/W	Control Settings for CLK5 - Watchdog Timer 1

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

Note If this clock is selected as the watchdog clock source for WDT1, then any system reset will halt that watchdog timer.

4.4.3.15 CLKMAN_SYS_CLK_CTRL_6_GPIO

CLKMAN_SYS_CLK_CTRL_6_GPIO.gpio_clk_scale

Field	Bits	Sys Reset	Access	Description
gpio_clk_scale	3:0	0000b	R/W	Control Settings for CLK6 - Clock for GPIO Ports

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.16 CLKMAN_SYS_CLK_CTRL_7_PT

CLKMAN_SYS_CLK_CTRL_7_PT.pulse_train_clk_scale

Field	Bits	Sys Reset	Access	Description
pulse_train_clk_scale	3:0	0000b	R/W	Control Settings for CLK7 - Source Clock for All Pulse Trains

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)

- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.17 CLKMAN_SYS_CLK_CTRL_8_UART

CLKMAN_SYS_CLK_CTRL_8_UART.uart_clk_scale

Field	Bits	Sys Reset	Access	Description
uart_clk_scale	3:0	0000b	R/W	Control Settings for CLK8 - Source Clock for All UARTs

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.18 CLKMAN_SYS_CLK_CTRL_9_I2CM

CLKMAN_SYS_CLK_CTRL_9_I2CM.i2cm_clk_scale

Field	Bits	Sys Reset	Access	Description
i2cm_clk_scale	3:0	0000b	R/W	Control Settings for CLK9 - Source Clock for All I2C Masters

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)

- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.19 CLKMAN_SYS_CLK_CTRL_10_I2CS

CLKMAN_SYS_CLK_CTRL_10_I2CS.i2cs_clk_scale

Field	Bits	Sys Reset	Access	Description
i2cs_clk_scale	3:0	0000b	R/W	Control Settings for CLK10 - Source Clock for I2C Slave

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.20 CLKMAN_SYS_CLK_CTRL_11_SPI0

CLKMAN_SYS_CLK_CTRL_11_SPI0.spi0_clk_scale

Field	Bits	Sys Reset	Access	Description
spi0_clk_scale	3:0	0000b	R/W	Control Settings for CLK11 - SPI Master 0

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.21 CLKMAN_SYS_CLK_CTRL_12_SPI1

CLKMAN_SYS_CLK_CTRL_12_SPI1.spi1_clk_scale

Field	Bits	Sys Reset	Access	Description
spi1_clk_scale	3:0	0000b	R/W	Control Settings for CLK12 - SPI Master 1

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.22 CLKMAN_SYS_CLK_CTRL_13_SPI2

CLKMAN_SYS_CLK_CTRL_13_SPI2.spi2_clk_scale

Field	Bits	Sys Reset	Access	Description
spi2_clk_scale	3:0	0000b	R/W	Control Settings for CLK13 - SPI Master 2

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.23 CLKMAN_SYS_CLK_CTRL_14_SPIB

CLKMAN_SYS_CLK_CTRL_14_SPIB.spib_clk_scale

Field	Bits	Sys Reset	Access	Description
spib_clk_scale	3:0	0000b	R/W	Control Settings for CLK14 - SPI Bridge Clock

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.24 CLKMAN_SYS_CLK_CTRL_15_OWM

CLKMAN_SYS_CLK_CTRL_15_OWM.owm_clk_scale

Field	Bits	Sys Reset	Access	Description
owm_clk_scale	3:0	0000b	R/W	Control Settings for CLK15 - 1-Wire Master Clock

- 0: CLK is Disabled
- 1: CLK = (System Clock Source / 1)
- 2: CLK = (System Clock Source / 2)
- 3: CLK = (System Clock Source / 4)
- 4: CLK = (System Clock Source / 8)
- 5: CLK = (System Clock Source / 16)
- 6: CLK = (System Clock Source / 32)
- 7: CLK = (System Clock Source / 64)
- 8: CLK = (System Clock Source / 128)
- 9: CLK = (System Clock Source / 256)

This clock is disabled by default following reset.

4.4.3.25 CLKMAN_CRYPT_CLK_CTRL_0_AES

CLKMAN_CRYPT_CLK_CTRL_0_AES.aes_clk_scale

Field	Bits	Sys Reset	Access	Description
aes_clk_scale	3:0	0000b	R/W	Control Settings for Crypto Clock 0 - AES

- 0000b: CLK is Disabled
- 0001b: CLK = (Crypto Clock Source / 1)
- 0010b: CLK = (Crypto Clock Source / 2)
- 0011b: CLK = (Crypto Clock Source / 4)
- 0100b: CLK = (Crypto Clock Source / 8)
- 0101b: CLK = (Crypto Clock Source / 16)
- 0110b: CLK = (Crypto Clock Source / 32)
- 0111b: CLK = (Crypto Clock Source / 64)
- 1000b: CLK = (Crypto Clock Source / 128)
- 1001b: CLK = (Crypto Clock Source / 256)
- other: CLK = (Crypto Clock Source / 1)

This clock is disabled by default following reset.

4.4.3.26 CLKMAN_CRYPT_CLK_CTRL_1_MAA

CLKMAN_CRYPT_CLK_CTRL_1_MAA.maa_clk_scale

Field	Bits	Sys Reset	Access	Description
maa_clk_scale	3:0	0000b	R/W	Control Settings for Crypto Clock 1 - MAA

- 0000b: CLK is Disabled
- 0001b: CLK = (Crypto Clock Source / 1)
- 0010b: CLK = (Crypto Clock Source / 2)
- 0011b: CLK = (Crypto Clock Source / 4)
- 0100b: CLK = (Crypto Clock Source / 8)
- 0101b: CLK = (Crypto Clock Source / 16)
- 0110b: CLK = (Crypto Clock Source / 32)
- 0111b: CLK = (Crypto Clock Source / 64)
- 1000b: CLK = (Crypto Clock Source / 128)
- 1001b: CLK = (Crypto Clock Source / 256)
- other: CLK = (Crypto Clock Source / 1)

This clock is disabled by default following reset.

4.4.3.27 CLKMAN_CRYPT_CLK_CTRL_2_PRNG

CLKMAN_CRYPT_CLK_CTRL_2_PRNG.prng_clk_scale

Field	Bits	Sys Reset	Access	Description
prng_clk_scale	3:0	0000b	R/W	Control Settings for Crypto Clock 2 - PRNG

- 0000b: CLK is Disabled
- 0001b: CLK = (Crypto Clock Source / 1)
- 0010b: CLK = (Crypto Clock Source / 2)
- 0011b: CLK = (Crypto Clock Source / 4)
- 0100b: CLK = (Crypto Clock Source / 8)
- 0101b: CLK = (Crypto Clock Source / 16)
- 0110b: CLK = (Crypto Clock Source / 32)
- 0111b: CLK = (Crypto Clock Source / 64)
- 1000b: CLK = (Crypto Clock Source / 128)
- 1001b: CLK = (Crypto Clock Source / 256)

- other: CLK = (Crypto Clock Source / 1)

This clock is disabled by default following reset.

4.4.3.28 CLKMAN_CLK_GATE_CTRL0

CLKMAN_CLK_GATE_CTRL0.cm4_clk_gater

Field	Bits	Sys Reset	Access	Description
cm4_clk_gater	1:0	01b	R/W	Clock Gating Control for CM4 CPU

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.ahb32_clk_gater

Field	Bits	Sys Reset	Access	Description
ahb32_clk_gater	3:2	01b	R/W	Clock Gating Control for AHB32

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.icache_clk_gater

Field	Bits	Sys Reset	Access	Description
icache_clk_gater	5:4	01b	R/W	Clock Gating Control for Instruction Cache

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.flash_clk_gater

Field	Bits	Sys Reset	Access	Description
flash_clk_gater	7:6	01b	R/W	Clock Gating Control for Flash Memory

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.sram_clk_gater

Field	Bits	Sys Reset	Access	Description
sram_clk_gater	9:8	01b	R/W	Clock Gating Control for SRAM

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.apb_bridge_clk_gater

Field	Bits	Sys Reset	Access	Description
apb_bridge_clk_gater	11:10	01b	R/W	Clock Gating Control for AHB-to-APB Bridge

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.sysman_clk_gater

Field	Bits	Sys Reset	Access	Description
sysman_clk_gater	13:12	01b	R/W	Clock Gating Control for Clkman/Pwrman/IOman

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.ptp_clk_gater

Field	Bits	Sys Reset	Access	Description
ptp_clk_gater	15:14	01b	R/W	Clock Gating Control for PTP Logic

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.ssb_mux_clk_gater

Field	Bits	Sys Reset	Access	Description
ssb_mux_clk_gater	17:16	01b	R/W	Clock Gating Control for SSB Mux

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.pad_clk_gater

Field	Bits	Sys Reset	Access	Description
pad_clk_gater	19:18	01b	R/W	Clock Gating Control for Pad Mode Filter

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.spix_clk_gater

Field	Bits	Sys Reset	Access	Description
spix_clk_gater	21:20	01b	R/W	Clock Gating Control for SPI XIP

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.pmu_clk_gater

Field	Bits	Sys Reset	Access	Description
pmu_clk_gater	23:22	01b	R/W	Clock Gating Control for PMU

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.usb_clk_gater

Field	Bits	Sys Reset	Access	Description
usb_clk_gater	25:24	01b	R/W	Clock Gating Control for USB

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.crc_clk_gater

Field	Bits	Sys Reset	Access	Description
crc_clk_gater	27:26	01b	R/W	Clock Gating Control for CRC

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.tpu_clk_gater

Field	Bits	Sys Reset	Access	Description
tpu_clk_gater	29:28	01b	R/W	Clock Gating Control for TPU

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL0.watchdog0_clk_gater

Field	Bits	Sys Reset	Access	Description
watchdog0_clk_gater	31:30	01b	R/W	Clock Gating Control for Watchdog Timer 0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

4.4.3.29 CLKMAN_CLK_GATE_CTRL1

CLKMAN_CLK_GATE_CTRL1.watchdog1_clk_gater

Field	Bits	Sys Reset	Access	Description
watchdog1_clk_gater	1:0	01b	R/W	Clock Gating Control for Watchdog Timer 1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.gpio_clk_gater

Field	Bits	Sys Reset	Access	Description
gpio_clk_gater	3:2	01b	R/W	Clock Gating Control for GPIO Ports

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.timer0_clk_gater

Field	Bits	Sys Reset	Access	Description
timer0_clk_gater	5:4	01b	R/W	Clock Gating Control for Timer/Counter Module 0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.timer1_clk_gater

Field	Bits	Sys Reset	Access	Description
timer1_clk_gater	7:6	01b	R/W	Clock Gating Control for Timer/Counter Module 1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.timer2_clk_gater

Field	Bits	Sys Reset	Access	Description
timer2_clk_gater	9:8	01b	R/W	Clock Gating Control for Timer/Counter Module 2

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.timer3_clk_gater

Field	Bits	Sys Reset	Access	Description
timer3_clk_gater	11:10	01b	R/W	Clock Gating Control for Timer/Counter Module 3

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.timer4_clk_gater

Field	Bits	Sys Reset	Access	Description
timer4_clk_gater	13:12	01b	R/W	Clock Gating Control for Timer/Counter Module 4

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.timer5_clk_gater

Field	Bits	Sys Reset	Access	Description
timer5_clk_gater	15:14	01b	R/W	Clock Gating Control for Timer/Counter Module 5

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.pulsetrain_clk_gater

Field	Bits	Sys Reset	Access	Description
pulsetrain_clk_gater	17:16	01b	R/W	Clock Gating Control for Pulse Train Generators

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.uart0_clk_gater

Field	Bits	Sys Reset	Access	Description
uart0_clk_gater	19:18	01b	R/W	Clock Gating Control for UART 0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.uart1_clk_gater

Field	Bits	Sys Reset	Access	Description
uart1_clk_gater	21:20	01b	R/W	Clock Gating Control for UART 1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.uart2_clk_gater

Field	Bits	Sys Reset	Access	Description
uart2_clk_gater	23:22	01b	R/W	Clock Gating Control for UART 2

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.uart3_clk_gater

Field	Bits	Sys Reset	Access	Description
uart3_clk_gater	25:24	01b	R/W	Clock Gating Control for UART 3

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.i2cm0_clk_gater

Field	Bits	Sys Reset	Access	Description
i2cm0_clk_gater	27:26	01b	R/W	Clock Gating Control for I2C Master 0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.i2cm1_clk_gater

Field	Bits	Sys Reset	Access	Description
i2cm1_clk_gater	29:28	01b	R/W	Clock Gating Control for I2C Master 1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL1.i2cm2_clk_gater

Field	Bits	Sys Reset	Access	Description
i2cm2_clk_gater	31:30	01b	R/W	Clock Gating Control for I2C Master 2

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

4.4.3.30 CLKMAN_CLK_GATE_CTRL2

CLKMAN_CLK_GATE_CTRL2.i2cs_clk_gater

Field	Bits	Sys Reset	Access	Description
i2cs_clk_gater	1:0	01b	R/W	Clock Gating Control for I2C Slave

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL2.spi0_clk_gater

Field	Bits	Sys Reset	Access	Description
spi0_clk_gater	3:2	01b	R/W	Clock Gating Control for SPI Master 0

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL2.spi1_clk_gater

Field	Bits	Sys Reset	Access	Description
spi1_clk_gater	5:4	01b	R/W	Clock Gating Control for SPI Master 1

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL2.spi2_clk_gater

Field	Bits	Sys Reset	Access	Description
spi2_clk_gater	7:6	01b	R/W	Clock Gating Control for SPI Master 2

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL2.spi_bridge_clk_gater

Field	Bits	Sys Reset	Access	Description
spi_bridge_clk_gater	9:8	01b	R/W	Clock Gating Control for SPI Bridge

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL2.owm_clk_gater

Field	Bits	Sys Reset	Access	Description
owm_clk_gater	11:10	01b	R/W	Clock Gating Control for 1-Wire Master (OWM)

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

CLKMAN_CLK_GATE_CTRL2.adc_clk_gater

Field	Bits	Sys Reset	Access	Description
adc_clk_gater	13:12	01b	R/W	Clock Gating Control for ADC

Dynamic clock gating control.

- 00b: Clock Off
- 01b: Dynamic Clock Gating Enabled
- 1xb: Clock On

4.5 Windowed Watchdog Timers

4.5.1 Overview

The **MAX32620** features two Windowed Watchdog Timers (WDT) that protect against corrupt or unreliable software, power faults, and other system-level problems that may place the **MAX32620** into unsuitable operating states. When the application is working correctly, application software will periodically reset the active watchdog counter(s) within a specific window of time. If the watchdog timer interrupt is enabled and the software does not reset the counter within the interrupt time period (int_period), the watchdog timer will generate a watchdog timer interrupt. If the watchdog timer reset is enabled and the software does not reset the counter within the reset time period (rst_period), the watchdog timer will generate a system reset. Additionally, a watchdog timer pre-window interrupt can be configured to enforce a minimum duration between one watchdog counter reset event and the next.

4.5.2 Clock Source Selection and Gating

The **MAX32620** supports multiple clock source selection options for the watchdog timers. This enables the use of both watchdog timers to ensure that in the event a specific clock source fails, the second watchdog timer will still catch the failure.

Each Watchdog Timer on the **MAX32620** supports clock source selection from one of three available clocks. To select the clock source, refer to the table below and write to either the wdt0_clock_select or wdt1_clock_select field.

Note See Watchdog Timer value explanation below.

Watchdog Source Selection	Clock Source
00b	Watchdog Module Clock (default)
01b	RTC oscillator
10b	96MHz relaxation oscillator
11b	Nano-ring oscillator

Once a clock source is set up for the watchdog timer, the clock must be turned on to the Watchdog Timer. By default, the clock is gated off to the block. To turn on the clock and enable the Watchdog Timer Peripheral, write a 1 to the WDTn_CTRL.en_clock register.

The default source of the watchdog timer is the system clock and is scalable via the CLKMAN_SYS_CLK_CTRL_4_WDT0.watchdog0_clk_scale and CLKMAN_SYS_CLK_CTRL_5_WDT1.watchdog1_clk_scale registers.

Note This does not start the watchdog operation, but enables the selected clock to clock the Watchdog Timer once the Timer is configured and firmware enables the Watchdog. The configuration procedure is below.

4.5.3 Watchdog Timer Configuration

Each watchdog timer supports independent settings for three independent time delay periods:

- **Pre-window period**: minimum time interval required between watchdog timer clear events
- **Interrupt period**: time from watchdog timer clear until an interrupt is triggered by the watchdog timer
- **Reset period**: time from watchdog timer clear until the system is reset

The Pre-Window period (if it is used at all) must be shorter than the Interrupt Period. If both the Interrupt Period and the Reset Period are used, the Interrupt Period must be shorter than the Reset Period. This allows application firmware to detect that the watchdog counter has not been reset (using the watchdog interrupt) during the appropriate time frame. If after receiving the interrupt, the application firmware still does not reset the watchdog counter, or if the watchdog interrupt event is missed or mishandled for some reason, the watchdog timer will generate a system reset once the Reset Period elapses.

There are 16 choices of duration for each of the three watchdog timer delay periods: 2^{16} through 2^{31} clock cycles of the selected clock. The time delay for a specific clock source is calculated as follows:

- **Pre-window period** = Clock Source Period × wait_period
- **Interrupt period** = Clock Source Period × int_period
- **Reset period** = Clock Source Period × rst_period

An illustration of different watchdog timer int_period values is shown in the table below.

Clock Source	Interrupt Period	Time Period
96MHz	1111(2^{16})	0.68 ms
96MHz	1011(2^{20})	10.93 ms
96MHz	0000(2^{31})	22.25 seconds

4.5.3.1 Locking and Unlocking the Watchdog Timer Configuration

Once the Watchdog Timer is configured, the Watchdog Timer Control Register can be locked by firmware. This is used to protect the watchdog timer settings against unintended actions of runaway firmware or system failure. To do this, write the value 0x24 to the WDTn_LOCK_CTRL.wdlock register field.

If changes to the configuration need to be made by firmware under certain circumstances and the WDTn_LOCK_CTRL register is already set, it may be unlocked. To unlock the configuration, write the value 0x42 to the WDTn_LOCK_CTRL.wdlock register field.

4.5.3.2 Enabling and Disabling the Watchdog Timer Counter

The application software must set the WDTn_CTRL.en_timer to a 1 to start the Watchdog Timer. The Watchdog Timer is free-running; the following procedure must be followed when enabling to prevent an unintended reset during the enable process.

- Write 0xA5 to WDTn_CLEAR
- Write 0x5A to WDTn_CLEAR
- Set WDTn_CTRL.en_timer bit

The watchdog timer can be disabled in multiple ways:

- The application software can clear WDTn_CTRL.en_timer to 0.
- A POR will clear WDTn_CTRL.en_timer to 0.
- The interrupt and reset signals from the watchdog timer can also be enabled or disabled independently through the WDTn_FLAGS.timeout field.

The WDTn `[n]` value has the following significance:

- **WDT0** is a "system reset" watchdog. If the reset output from this watchdog is asserted, this will result in a system reset. The WDT0 control and configuration registers are reset on any system reset as well. This means that if the WDT0 watchdog triggers a system reset, it will also reset and disable itself.
- **WDT1** is a "POR reset" watchdog. If the reset output from this watchdog is asserted, this will result in a Power-On Reset event. The WDT1 control and configuration registers are reset on any Power-On Reset as well. This means that if the WDT1 watchdog triggers a POR, it will also reset and disable itself.
- *Note*: If WDT0 issues a system reset, WDT1 will continue to function because its configuration has not been reset.

4.5.4 Watchdog Timer Operation

Once the Watchdog Timer is enabled and begins counting, firmware is responsible for both periodically "feeding" the watchdog (that is, clearing the watchdog timer to zero) by writing the appropriate sequence to the WDTn_CLEAR register. Application software needs to ensure this action is taken within the defined time window.

4.5.5 Registers (WDT)

Address	Register	Access	Description	Reset By
0x40008000	WDT0_CTRL	R/W	Watchdog Timer 0 Control Register	Sys
0x40008004	WDT0_CLEAR	R/W	Watchdog Timer 0 Clear Register (Feed Dog)	Sys
0x40008008	WDT0_FLAGS	W1C	Watchdog Timer 0 Interrupt and Reset Flags	POR
0x4000800C	WDT0_ENABLE	R/W	Watchdog Timer 0 Interrupt/Reset Enable/Disable Controls	Sys
0x40008014	WDT0_LOCK_CTRL	R/W	Watchdog Timer 0 Register Setting Lock for Control Register	Sys
0x40009000	WDT1_CTRL	R/W	Watchdog Timer 1 Control Register	Sys
0x40009004	WDT1_CLEAR	R/W	Watchdog Timer 1 Clear Register (Feed Dog)	Sys
0x40009008	WDT1_FLAGS	W1C	Watchdog Timer 1 Interrupt and Reset Flags	POR
0x4000900C	WDT1_ENABLE	R/W	Watchdog Timer 1 Interrupt/Reset Enable/Disable Controls	Sys
0x40009014	WDT1_LOCK_CTRL	R/W	Watchdog Timer 1 Register Setting Lock for Control Register	Sys

4.5.5.1 WDTn_CTRL

WDTn_CTRL.int_period

Field	Bits	Sys Reset	Access	Description
int_period	3:0	special	R/W	Period from WDT Clear to Interrupt Flag Set

This field sets the duration of the watchdog interrupt period, which is the time period from the beginning of the watchdog timer count (the count resets to zero when the watchdog timer is first enabled, as well as each time the watchdog timer is cleared by writing to WDTn_CLEAR) until the Watchdog Timeout Interrupt Flag (wait_period) is set.

Defined in terms of a number of watchdog clocks, with the number of clocks given by 2^N, N=(31 - field value), e.g.

- 0h: 2^{31}
- 1h: 2^{30}
- 2h: 2^{29}
- ....
- Eh: 2^{17}
- Fh: 2^{16}

This field is reset under the following conditions:

- For WDT0: Cleared to 0 on Sys_Reset.
- For WDT1: Cleared to 0 on POR_Reset.

WDTn_CTRL.rst_period

Field	Bits	Sys Reset	Access	Description
rst_period	7:4	special	R/W	Period from WDT Clear to Reset Flag Set

Reset Period - the time period from the beginning of the watchdog timer count (WDT is cleared to zero by being enabled or when the watchdog timer is cleared by writing to the CLEAR register) until the Watchdog Reset Flag is set.

Defined in terms of a number of watchdog clocks, with the number of clocks given by 2^N, N=(31 - field value), e.g.

- 0h: 2^{31}
- 1h: 2^{30}
- 2h: 2^{29}
- ....
- Eh: 2^{17}
- Fh: 2^{16}

This field is reset under the following conditions:

- For WDT0: Cleared to 0 on Sys_Reset.
- For WDT1: Cleared to 0 on POR_Reset.

WDTn_CTRL.en_timer

Field	Bits	Sys Reset	Access	Description
en_timer	8	special	R/W	Watchdog Timer Enable

- 0: Watchdog timer is disabled and the counter is held at zero.
- 1: Watchdog timer is enabled and counting.

This bit is reset under the following conditions:

- For WDT0: Cleared to 0 on Sys_Reset.
- For WDT1: Cleared to 0 on POR_Reset.

WDTn_CTRL.en_clock

Field	Bits	Sys Reset	Access	Description
en_clock	9	special	R/W	Watchdog Clock Gate

This control bit enables the watchdog timer module clock which allows flags to be set/cleared (not the same as en_timer which allows the timer to increment).

- 0: Clock to WDT is disabled.
- 1: Clock to WDT is enabled.

This bit is reset under the following conditions:

- For WDT0: Cleared to 0 on Sys_Reset.
- For WDT1: Cleared to 0 on POR_Reset.

WDTn_CTRL.wait_period

Field	Bits	Sys Reset	Access	Description
wait_period	15:12	special	R/W	Period from WDT Clear to Clear Window Begin

This field defines the pre-window period, or the time period the WDT waits (following WDT enable or reset/feed) before allowing the watchdog to be reset without triggering an out-of-window interrupt.

Defined in terms of a number of watchdog clocks, with the number of clocks given by 2^N, N=(31 - field value), e.g.

- 0h: 2^{31}
- 1h: 2^{30}
- 2h: 2^{29}
- ....
- Eh: 2^{17}
- Fh: 2^{16}

This field is reset under the following conditions:

- For WDT0: Cleared to 0 on Sys_Reset.
- For WDT1: Cleared to 0 on POR_Reset.

4.5.5.2 WDTn_CLEAR

WDT0_CLEAR

Sys Reset	Access	Description
n/a	R/W	Watchdog Timer Clear Register (Feed Dog)

Write 0xA5,0x5A sequence to clear watchdog timer (feed watchdog)

Reads always return zero.

4.5.5.3 WDTn_FLAGS

WDTn_FLAGS.timeout

Field	Bits	Sys Reset	Alt Reset	Access	Description
timeout	0	X	POR:0	W1C	Watchdog Timeout Interrupt Flag

Hardware sets this flag to 1 when the watchdog timer reaches the end of the interrupt period without being cleared.

Write to 1 to clear this flag.

WDTn_FLAGS.pre_win

Field	Bits	Sys Reset	Alt Reset	Access	Description
pre_win	1	X	POR:0	W1C	Watchdog Pre-Window Clear Interrupt Flag

Hardware sets this flag to 1 when the watchdog timer is cleared (by firmware writing to WDTn_CLEAR) before the end of the pre-window period.

Write to 1 to clear this flag.

WDTn_FLAGS.reset_out

Field	Bits	Sys Reset	Alt Reset	Access	Description
reset_out	2	X	POR:0	W1C	Watchdog Reset Flag

Hardware sets this flag to 1 when the watchdog timer reaches the end of the reset period without being cleared.

Write to 1 to clear this flag.

4.5.5.4 WDTn_ENABLE

WDTn_ENABLE.timeout

Field	Bits	Sys Reset	Access	Description
timeout	0	special	R/W	Enable Watchdog Interrupt

- 0: No interrupt will be triggered when the Watchdog Interrupt Flag is set.
- 1: An interrupt will be triggered by the WDT when the Watchdog Interrupt Flag is set to 1.

This bit is reset under the following conditions:

- For WDT0: Cleared to 0 on Sys_Reset.
- For WDT1: Cleared to 0 on POR_Reset.

WDTn_ENABLE.pre_win

Field	Bits	Sys Reset	Access	Description
pre_win	1	special	R/W	Enable Watchdog Pre-Window Reset Interrupt

- 0: No pre-window reset interrupt will be triggered.
- 1: An interrupt will be triggered when the Pre-Window Reset Interrupt Flag is set to 1.

This bit is reset under the following conditions:

- For WDT0: Cleared to 0 on Sys_Reset.
- For WDT1: Cleared to 0 on POR_Reset.

WDTn_ENABLE.reset_out

Field	Bits	Sys Reset	Access	Description
reset_out	2	special	R/W	Enable Watchdog Reset Output

- 0: No reset will be triggered by this watchdog.
- 1: A system reset (for WDT0) or system reboot (for WDT1) will be triggered when the Watchdog Reset Flag is set to 1.

This bit is reset under the following conditions:

- For WDT0: Cleared to 0 on Sys_Reset.
- For WDT1: Cleared to 0 on POR_Reset.

4.5.5.5 WDTn_LOCK_CTRL

WDTn_LOCK_CTRL.wdlock

Field	Bits	Sys Reset	Access	Description
wdlock	7:0	special	R/W	Lock for Control Register

- If this field reads 0, the Control register may be written to.
- If this field reads 1, the Control register is read-only.
- Writing 0x24 to this field will set bit 0 to 1 (Lock).
- Writing 0x42 to this field will clear bit 0 to 0 (Unlock).

This field is reset under the following conditions:

- For WDT0: Cleared to 0 on Sys_Reset.
- For WDT1: Cleared to 0 on POR_Reset.

5 Pin Configurations, Packages, and Special Function Multiplexing

5.1 Pin Layout

WLP Pin Layout

The **MAX32620** is offered in a 3.9mm x 3.9mm 81-bump Wafer Level Package (WLP) and provides a WLP, low-cost and robust package solution.

TQFP Pin Layout

Future package option.

5.2 Pin Function Mapping

The Port Function Muxing tables below show the peripheral function options available on each port. The functions are shown in order of priority (highest to lowest from left-to-right) for the specific port. In the event that more than one function is enabled on a given port/pin, the highest priority function takes precedence.

The lowest priority function for each General-Purpose I/O (GPIO) pin is the GPIO function itself, which consists of either direct firmware control over the state of the port pin, or linking the pin to a GPIO function such as a pulse train output or a 32-bit timer input or output. GPIO functionality is enabled by default for any GPIO pin that does not have any other function selected.

All GPIO pins can be enabled for external interrupt mode and/or wakeup detection mode.

Note Each peripheral function is shown in different shades of the same base color (e.g., blue represents SPI peripherals). For each peripheral function, multiple mappings to port pins may be available. These are indicated in the function muxing tables with alpha characters in parentheses. For example, I2CS0 has three possible mappings; these are shown as I2CS0 (A), I2CS0 (B), and I2CS0 (C). In addition, some peripherals support multiple pin options for specific functionality. These are shown in the tables using `[]`. For example, SPI supports up to four slave select (SS) signals, which are labeled SS[0], SS[1], SS[2], and SS[3]. These should be used in order as numbered. For one slave select line, SS[0] would be used. For two slave select lines, use SS[0] and SS[1].

5.2.1 GPIO Function Mapping

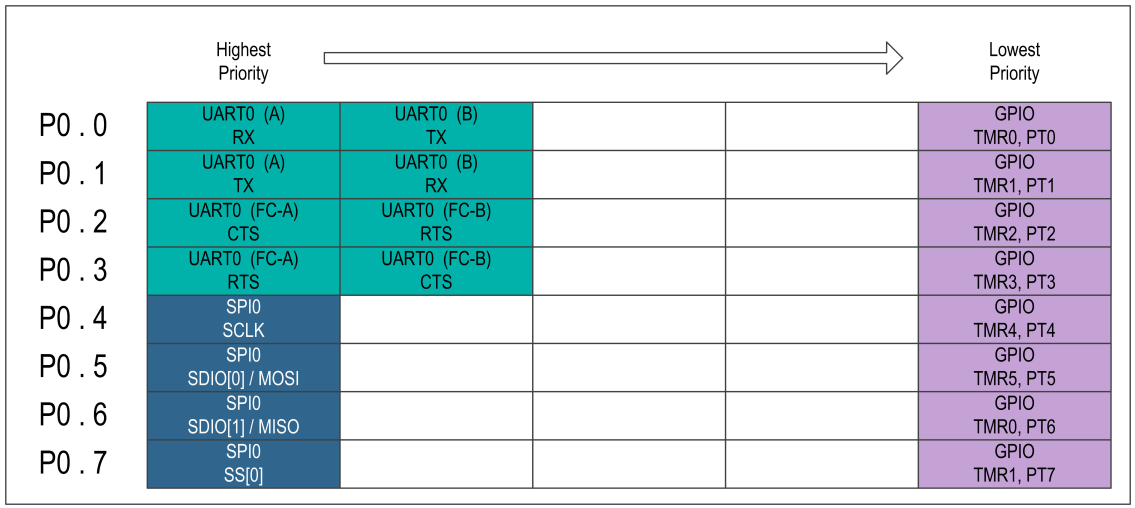

Figure 5.1: GPIO Port 0 IO Function Muxing

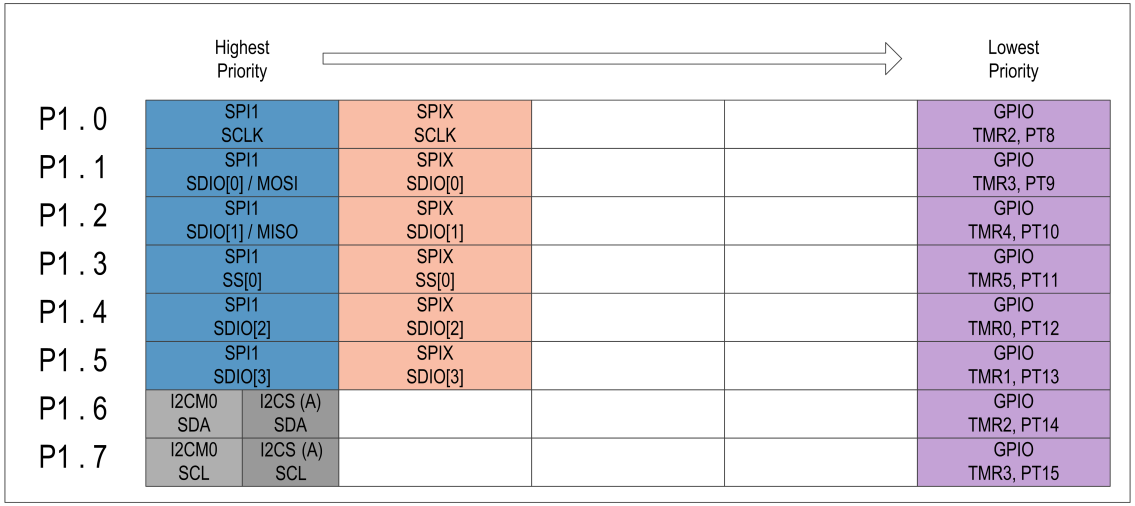

Figure 5.2: GPIO Port 1 IO Function Muxing

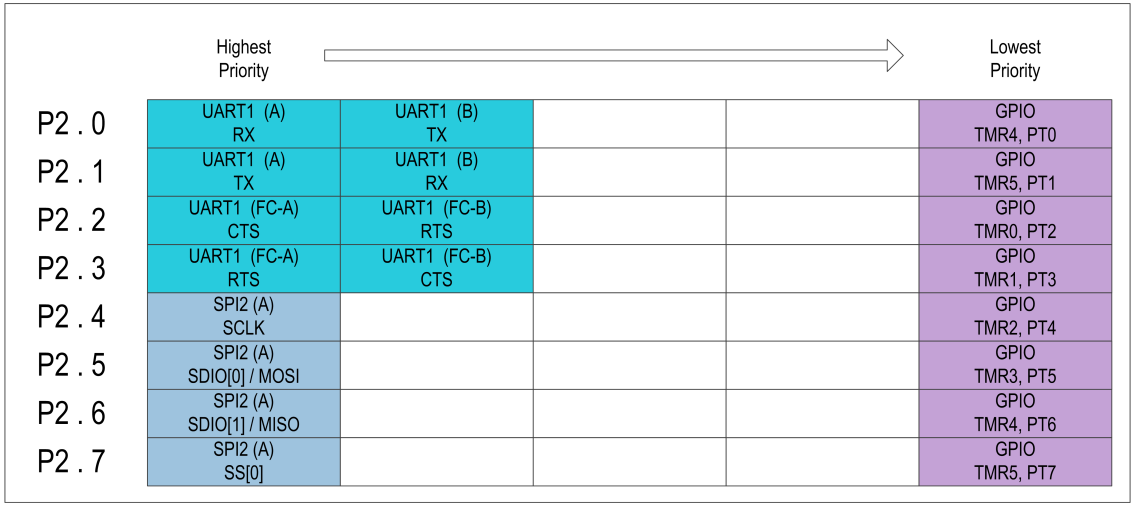

Figure 5.3: GPIO Port 2 IO Function Muxing

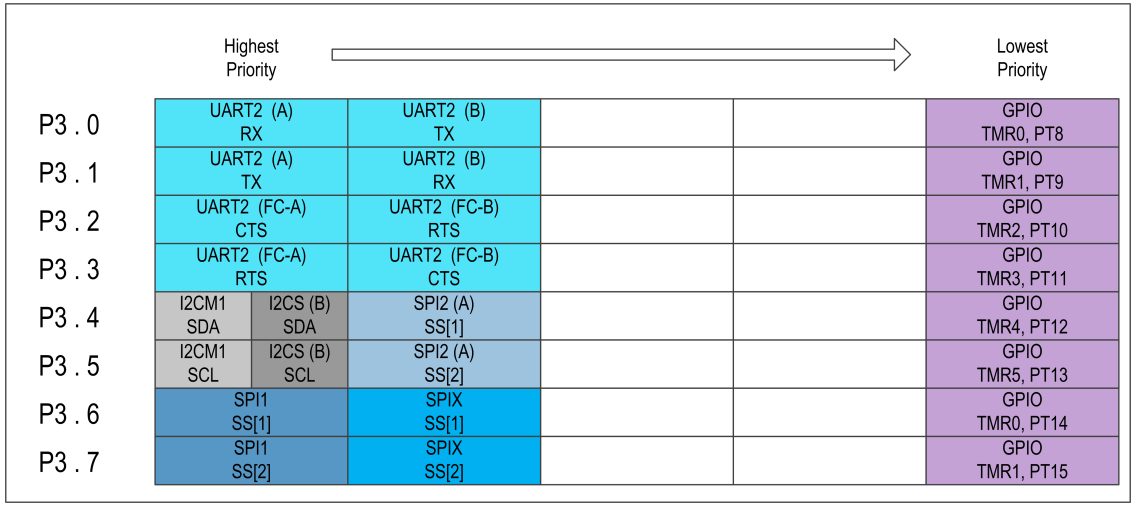

Figure 5.4: GPIO Port 3 IO Function Muxing

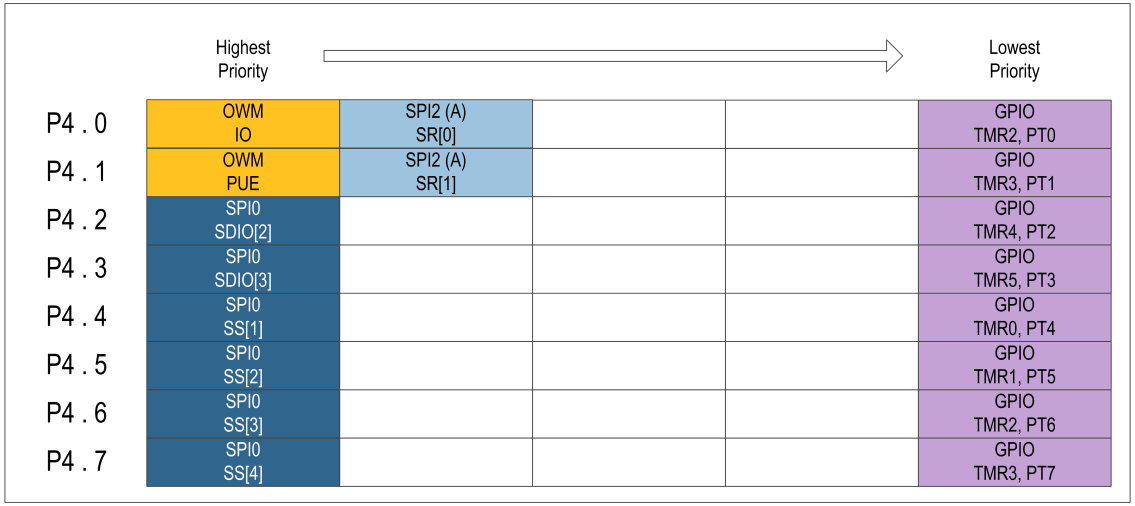

Figure 5.5: GPIO Port 4 IO Function Muxing

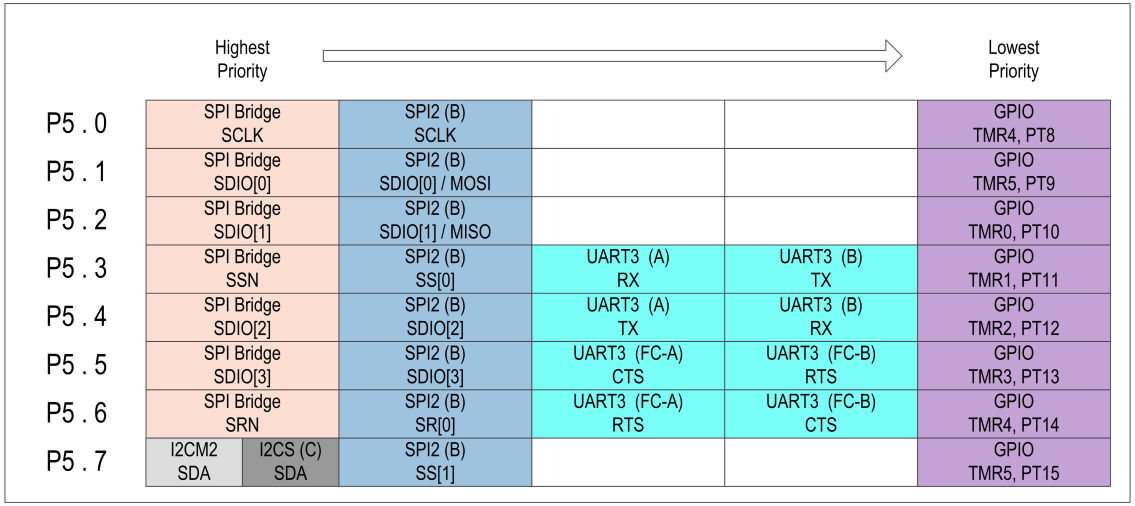

Figure 5.6: GPIO Port 5 IO Function Muxing

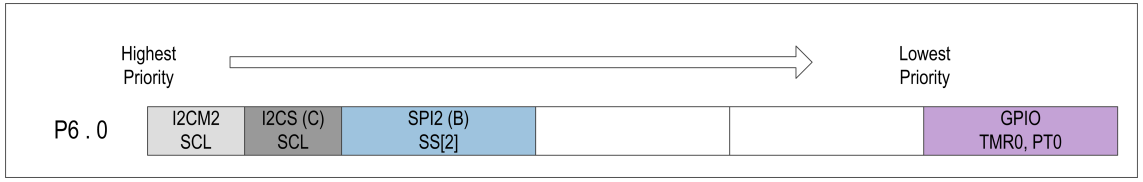

Figure 5.7: GPIO Port 6 IO Function Muxing

5.3 General-Purpose I/O

The **MAX32620** includes 49 general-purpose I/O (GPIO) pins which can be controlled directly by firmware or indirectly by other hardware functions (such as output peripherals). These GPIO pins are logically divided into seven GPIO ports, each port consisting of eight pins with the exception of P6 which only has one pin. The ports are named as follows: P0, P1, P2, P3, P4, P5, and P6. A single GPIO pin within a port is typically referred to by the notation (port).(pin), where the pin within each port is numbered from 0 to 7. For example, the third pin in Port 2 is referred to as P2.2.

Features supported by the GPIO module include the following:

- Functions are selectable on a per-pin basis, with GPIO operation (meaning that the firmware controls the state of the pin) being the lowest-priority function type.
- When a pin is using GPIO functionality, firmware can directly control the drive strength and output type of the pin (e.g., normal drive vs. open-drain or high-impedance).
- Regardless of the selected function for a given pin, firmware can always monitor the current logic level of the pin using the appropriate registers.
  - All GPIO pins support interrupt detection based on input signals external to the device.
  - Interrupts may be detected on rising or falling edges or high/low levels.
  - Interrupts to the CPU are generated on a per-port basis; i.e., any interrupts originating from Port 0 are assigned to a single interrupt vector, interrupts from Port 1 are assigned to a different vector, and so on.
- When in GPIO operation, simple output-only functions can be assigned to a GPIO pin:
  - Output from Pulse Trains (0 through 15)
  - Output from Timers running in 32-bit mode
  - In this case, output drive strength and type will still be determined by firmware

The GPIO drivers on the **MAX32620** support 1.8V input and output drive levels only. Refer to the datasheet for more details on the GPIO driver electrical characteristics.

5.3.1 Device Pins

When not overridden by a higher-priority function, the GPIO module can be used to control the I/O state of any of the 49 port pins:

- P0.0 through P0.7
- P1.0 through P1.7
- P2.0 through P2.7
- P3.0 through P3.7
- P4.0 through P4.7
- P5.0 through P5.7
- P6.0

5.3.2 Interrupts

The GPIO module reports interrupts to the CPU core using the following interrupt vector channels.

Interrupt Number	Vector	Description
0x1F	0x07C	External interrupt triggered on one or more pins of GPIO Port 0
0x20	0x080	External interrupt triggered on one or more pins of GPIO Port 1
0x21	0x084	External interrupt triggered on one or more pins of GPIO Port 2

Interrupt Number	Vector	Description
0x22	0x088	External interrupt triggered on one or more pins of GPIO Port 3
0x23	0x08C	External interrupt triggered on one or more pins of GPIO Port 4
0x24	0x090	External interrupt triggered on one or more pins of GPIO Port 5
0x25	0x094	External interrupt triggered on GPIO Port 6 pin 0

5.3.3 Firmware Control

Firmware control of the output state is handled through two control registers: GPIO_OUT_MODE_Pn which defines 16 modes of operation, while the GPIO_OUT_VAL_Pn defines which of the two output states (0 or 1) is enabled for that mode.

The following table defines all the combinations of mode and value with their defined drive state and pad control states.

gpio_out_mode	(LOW) gpio_out_val == 0	(HIGH) gpio_out_val == 1
0	High impedance with input buffer enabled	Weak pullup
1	Open drain drive LOW	High impedance with input buffer enabled
2	Open drain drive LOW	Weak pullup
3	RESERVED	RESERVED
4, 6, 8	High impedance with input buffer enabled	High impedance with input buffer enabled
5	Normal drive LOW	Normal drive HIGH
7	Slow drive LOW	Slow drive HIGH
9	Fast drive LOW	Fast drive HIGH
10	Weak pulldown	High impedance with input buffer enabled
11	High impedance with input buffer enabled	Open source drive HIGH
12	Weak pulldown	Open source drive HIGH
13	RESERVED	RESERVED
14	RESERVED	RESERVED
15	High impedance with input buffer DISABLED	High impedance with input buffer DISABLED

Note Modes 3, 13-14 are reserved and will be treated as if Mode 0 were selected.

Descriptions of the GPIO_OUT_MODE_Pn output states as defined by GPIO_OUT_VAL_Pn:

- **High impedance**: HiZ, No connection
  - Impedance >> 1 GOhm for (0 < Vpad < 1.8V)
  - Impedance >> 1 MOhm for (1.8V < Vpad < 3.6V)
- **Weak pullup**: 25K resistor pullup HIGH to V_{18}
- **Open drain**: Transistor drive LOW to GND, standard current capacity
- **Normal drive high**: Transistor drive HIGH to V_{18}, 2mA
- **Normal drive low**: Transistor drive LOW to GND, 4mA
- **Slow drive high/low**: As per normal drive, except that it uses negative feedback to slow edge rates
- **Fast drive high**: Transistor drive HIGH to V18, 8mA
- **Fast drive low**: Transistor drive LOW to GND, 24mA
- **Weak pulldown**: 25K resistor pulldown to GND
- **Open source**: Transistor drive HIGH to V18, 2mA

In addition to firmware monitoring of I/O input state, logic is provided to monitor inputs for certain events and to generate interrupts to the processor on detection of these events.

The user can enable interrupt generation on rising edge detection, falling edge detection, or all edges detected. Support for interrupt generate on detection of a high or low logic level is provided as well. Each interrupt status event has a status bit and an enable register. The enable can mask the interrupt from being asserted to the processor. This mask does not affect the setting of the interrupt status bit. Status bits and enable registers are organized along port boundaries (P0 to P6), with each port having its own interrupt vector.

In addition to interrupt generation, inputs can also be monitored for wakeup event generation to wake the system up from a low power state.

5.3.4 Pullup/Pulldown Control (1 MOhm)

The 1 MEG pullup/pulldown control operates in parallel with GPIO_OUT_MODE; the 25K pullup/pulldown, when active, overrides the 1 MEG pullup/pulldown.

Note Driver is high impedance during 1 MEG configuration

By default, the 1MOhm / "super-weak" pullup/pulldown function is disabled on all pads. Possible states for this function are:

- Disabled
- 1MOhm pullup HIGH to V_{18}
- 1MOhm pulldown LOW to GND

The 1MOhm pullup/pulldown is configured independently from the GPIO_OUT_MODE and operates in parallel with the input/output modes shown in the GPIO_OUT_MODE table; however, any GPIO_OUT_MODE setting other than high-impedance will effectively override the 1MOhm pullup/pulldown.

Before either of these procedures, the PWRMAN_WUD_CTRL.ctrl_enable field must be set to 1 (to enable WUD configuration changes) if it is not set to this value

already.

Procedure to enable 1MOhm pullup HIGH (for a single GPIO pin):

- Set the appropriate bit of IOMAN_WUD_REQ0 (for pins P0[7:0], P1[7:0], P2[7:0], or P3[7:0]) or IOMAN_WUD_REQ1 (for pins P4[7:0], P5[7:0], or P6[0]) to 1 to set the GPIO pad to Wakeup Detect Mode. This overrides the GPIO_OUT_MODE setting, forcing the pad to high impedance, and allows the WUD logic on the pad to be configured using the pwrman interface.
- Set PWRMAN_WUD_CTRL.pad_select to the number of the GPIO pad to be configured, where 0..7 is P0[0..7], 8..15 is P1[0..7], and so on up to 48 = P6[0].
- Set PWRMAN_WUD_CTRL.pad_mode to 2 to enable configuration of Weak Hi / Weak Lo setting.
- Write 1 to PWRMAN_WUD_PULSE0 to set the 1MOhm weak pullup HIGH enable.
- Optionally, reset the IOMAN_WUD_REQ0/IOMAN_WUD_REQ1 bit that was set in step 1 back to zero to allow the GPIO_OUT_MODE setting to be used again (if this is needed).

Procedure to enable 1MOhm pullup LOW (for a single GPIO pin):

Same as the above procedure, except step 4 changes to:

- Write 1 to PWRMAN_WUD_PULSE1 to set the 1MOhm weak pullup LOW enable.

5.4 GPIO Pins and Peripheral Mode Functions

For all GPIO pins, the GPIO operation is the lowest priority functionality. This mode will be enabled by default for any GPIO pins that have not been requested for use as part of a higher-priority peripheral function.

When GPIO mode is active, all pins in each GPIO port have access to a common set of low-level peripheral functions that can use GPIO pins as inputs or outputs:

- Output signal streams from Pulse Trains 0 through 15
- Input/output signals (depending on configured timer mode) for the 32-bit Timer (0, 1, 2, 3, 4 and 5) modules

5.4.1 P0/P1 GPIO Function Options

Function	P0.0	P0.1	P0.2	P0.3	P0.4	P0.5	P0.6	P0.7	P1.0	P1.1	P1.2	P1.3	P1.4	P1.5	P1.6	P1.7
PT0	X															
PT1		X														
PT2			X													
PT3				X												
PT4					X											
PT5						X										

Function	P0.0	P0.1	P0.2	P0.3	P0.4	P0.5	P0.6	P0.7	P1.0	P1.1	P1.2	P1.3	P1.4	P1.5	P1.6	P1.7
PT6							X									
PT7								X								
PT8									X							
PT9										X						
PT10											X					
PT11												X				
PT12													X			
PT13														X		
PT14															X	
PT15																X
Timer 0	X						X						X			
Timer 1		X						X						X		
Timer 2			X						X						X	
Timer 3				X						X						X
Timer 4					X						X					
Timer 5						X						X				

5.4.2 P2/P3 GPIO Function Options

Function	P2.0	P2.1	P2.2	P2.3	P2.4	P2.5	P2.6	P2.7	P3.0	P3.1	P3.2	P3.3	P3.4	P3.5	P3.6	P3.7
PT0	X															
PT1		X														
PT2			X													
PT3				X												
PT4					X											

Function	P2.0	P2.1	P2.2	P2.3	P2.4	P2.5	P2.6	P2.7	P3.0	P3.1	P3.2	P3.3	P3.4	P3.5	P3.6	P3.7
PT5						X										
PT6							X									
PT7								X								
PT8									X							
PT9										X						
PT10											X					
PT11												X				
PT12													X			
PT13														X		
PT14															X	
PT15																X
Timer 0			X						X						X	
Timer 1				X						X						X
Timer 2					X						X					
Timer 3						X						X				
Timer 4	X						X						X			
Timer 5		X						X						X		

5.4.3 P4/P5/P6 GPIO Function Options

Function	P4.0	P4.1	P4.2	P4.3	P4.4	P4.5	P4.6	P4.7	P5.0	P5.1	P5.2	P5.3	P5.4	P5.5	P5.6	P5.7	P6.0
PT0	X																X
PT1		X															
PT2			X														
PT3				X													

Function	P4.0	P4.1	P4.2	P4.3	P4.4	P4.5	P4.6	P4.7	P5.0	P5.1	P5.2	P5.3	P5.4	P5.5	P5.6	P5.7	P6.0
PT4					X												
PT5						X											
PT6							X										
PT7								X									
PT8									X								
PT9										X							
PT10											X						
PT11												X					
PT12													X				
PT13														X			
PT14															X		
PT15																X	
Timer 0					X						X						X
Timer 1						X						X					
Timer 2	X						X						X				
Timer 3		X						X						X			
Timer 4			X						X						X		
Timer 5				X						X						X	

5.5 Registers (GPIO)

Address	Register	Access	Description	Reset By
0x4000A000	GPIO_RST_MODE_P0	R/W	Port P0 Default (Power-On Reset) Output Drive Mode	POR
0x4000A004	GPIO_RST_MODE_P1	R/W	Port P1 Default (Power-On Reset) Output Drive Mode	POR
0x4000A008	GPIO_RST_MODE_P2	R/W	Port P2 Default (Power-On Reset) Output Drive Mode	POR
0x4000A00C	GPIO_RST_MODE_P3	R/W	Port P3 Default (Power-On Reset) Output Drive Mode	POR
0x4000A010	GPIO_RST_MODE_P4	R/W	Port P4 Default (Power-On Reset) Output Drive Mode	POR
0x4000A014	GPIO_RST_MODE_P5	R/W	Port P5 Default (Power-On Reset) Output Drive Mode	POR
0x4000A018	GPIO_RST_MODE_P6	R/W	Port P6 Default (Power-On Reset) Output Drive Mode	POR
0x4000A040	GPIO_FREE_P0	R/O	Port P0 Free for GPIO Operation Flags	Sys
0x4000A044	GPIO_FREE_P1	R/O	Port P1 Free for GPIO Operation Flags	Sys
0x4000A048	GPIO_FREE_P2	R/O	Port P2 Free for GPIO Operation Flags	Sys
0x4000A04C	GPIO_FREE_P3	R/O	Port P3 Free for GPIO Operation Flags	Sys
0x4000A050	GPIO_FREE_P4	R/O	Port P4 Free for GPIO Operation Flags	Sys
0x4000A054	GPIO_FREE_P5	R/O	Port P5 Free for GPIO Operation Flags	Sys
0x4000A058	GPIO_FREE_P6	R/O	Port P6 Free for GPIO Operation Flags	Sys
0x4000A080	GPIO_OUT_MODE_P0	R/W	Port P0 GPIO Output Drive Mode	Sys
0x4000A084	GPIO_OUT_MODE_P1	R/W	Port P1 GPIO Output Drive Mode	Sys
0x4000A088	GPIO_OUT_MODE_P2	R/W	Port P2 GPIO Output Drive Mode	Sys
0x4000A08C	GPIO_OUT_MODE_P3	R/W	Port P3 GPIO Output Drive Mode	Sys
0x4000A090	GPIO_OUT_MODE_P4	R/W	Port P4 GPIO Output Drive Mode	Sys
0x4000A094	GPIO_OUT_MODE_P5	R/W	Port P5 GPIO Output Drive Mode	Sys
0x4000A098	GPIO_OUT_MODE_P6	R/W	Port P6 GPIO Output Drive Mode	Sys
0x4000A0C0	GPIO_OUT_VAL_P0	R/W	Port P0 GPIO Output Value	Sys

Address	Register	Access	Description	Reset By
0x4000A0C4	GPIO_OUT_VAL_P1	R/W	Port P1 GPIO Output Value	Sys
0x4000A0C8	GPIO_OUT_VAL_P2	R/W	Port P2 GPIO Output Value	Sys
0x4000A0CC	GPIO_OUT_VAL_P3	R/W	Port P3 GPIO Output Value	Sys
0x4000A0D0	GPIO_OUT_VAL_P4	R/W	Port P4 GPIO Output Value	Sys
0x4000A0D4	GPIO_OUT_VAL_P5	R/W	Port P5 GPIO Output Value	Sys
0x4000A0D8	GPIO_OUT_VAL_P6	R/W	Port P6 GPIO Output Value	Sys
0x4000A100	GPIO_FUNC_SEL_P0	R/W	Port P0 GPIO Function Select	Sys
0x4000A104	GPIO_FUNC_SEL_P1	R/W	Port P1 GPIO Function Select	Sys
0x4000A108	GPIO_FUNC_SEL_P2	R/W	Port P2 GPIO Function Select	Sys
0x4000A10C	GPIO_FUNC_SEL_P3	R/W	Port P3 GPIO Function Select	Sys
0x4000A110	GPIO_FUNC_SEL_P4	R/W	Port P4 GPIO Function Select	Sys
0x4000A114	GPIO_FUNC_SEL_P5	R/W	Port P5 GPIO Function Select	Sys
0x4000A118	GPIO_FUNC_SEL_P6	R/W	Port P6 GPIO Function Select	Sys
0x4000A140	GPIO_IN_MODE_P0	R/W	Port P0 GPIO Input Monitoring Mode	Sys
0x4000A144	GPIO_IN_MODE_P1	R/W	Port P1 GPIO Input Monitoring Mode	Sys
0x4000A148	GPIO_IN_MODE_P2	R/W	Port P2 GPIO Input Monitoring Mode	Sys
0x4000A14C	GPIO_IN_MODE_P3	R/W	Port P3 GPIO Input Monitoring Mode	Sys
0x4000A150	GPIO_IN_MODE_P4	R/W	Port P4 GPIO Input Monitoring Mode	Sys
0x4000A154	GPIO_IN_MODE_P5	R/W	Port P5 GPIO Input Monitoring Mode	Sys
0x4000A158	GPIO_IN_MODE_P6	R/W	Port P6 GPIO Input Monitoring Mode	Sys
0x4000A180	GPIO_IN_VAL_P0	R/O	Port P0 GPIO Input Value	Sys
0x4000A184	GPIO_IN_VAL_P1	R/O	Port P1 GPIO Input Value	Sys
0x4000A188	GPIO_IN_VAL_P2	R/O	Port P2 GPIO Input Value	Sys
0x4000A18C	GPIO_IN_VAL_P3	R/O	Port P3 GPIO Input Value	Sys
0x4000A190	GPIO_IN_VAL_P4	R/O	Port P4 GPIO Input Value	Sys

Address	Register	Access	Description	Reset By
0x4000A194	GPIO_IN_VAL_P5	R/O	Port P5 GPIO Input Value	Sys
0x4000A198	GPIO_IN_VAL_P6	R/O	Port P6 GPIO Input Value	Sys
0x4000A1C0	GPIO_INT_MODE_P0	R/W	Port P0 Interrupt Detection Mode	Sys
0x4000A1C4	GPIO_INT_MODE_P1	R/W	Port P1 Interrupt Detection Mode	Sys
0x4000A1C8	GPIO_INT_MODE_P2	R/W	Port P2 Interrupt Detection Mode	Sys
0x4000A1CC	GPIO_INT_MODE_P3	R/W	Port P3 Interrupt Detection Mode	Sys
0x4000A1D0	GPIO_INT_MODE_P4	R/W	Port P4 Interrupt Detection Mode	Sys
0x4000A1D4	GPIO_INT_MODE_P5	R/W	Port P5 Interrupt Detection Mode	Sys
0x4000A1D8	GPIO_INT_MODE_P6	R/W	Port P6 Interrupt Detection Mode	Sys
0x4000A200	GPIO_INTFL_P0	W1C	Port P0 Interrupt Flags	Sys
0x4000A204	GPIO_INTFL_P1	W1C	Port P1 Interrupt Flags	Sys
0x4000A208	GPIO_INTFL_P2	W1C	Port P2 Interrupt Flags	Sys
0x4000A20C	GPIO_INTFL_P3	W1C	Port P3 Interrupt Flags	Sys
0x4000A210	GPIO_INTFL_P4	W1C	Port P4 Interrupt Flags	Sys
0x4000A214	GPIO_INTFL_P5	W1C	Port P5 Interrupt Flags	Sys
0x4000A218	GPIO_INTFL_P6	W1C	Port P6 Interrupt Flags	Sys
0x4000A240	GPIO_INTEN_P0	R/W	Port P0 Interrupt Enables	Sys
0x4000A244	GPIO_INTEN_P1	R/W	Port P1 Interrupt Enables	Sys
0x4000A248	GPIO_INTEN_P2	R/W	Port P2 Interrupt Enables	Sys
0x4000A24C	GPIO_INTEN_P3	R/W	Port P3 Interrupt Enables	Sys
0x4000A250	GPIO_INTEN_P4	R/W	Port P4 Interrupt Enables	Sys
0x4000A254	GPIO_INTEN_P5	R/W	Port P5 Interrupt Enables	Sys
0x4000A258	GPIO_INTEN_P6	R/W	Port P6 Interrupt Enables	Sys

5.5.1 GPIO_RST_MODE_Pn

GPIO_RST_MODE_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Sys Reset	Alt Reset	Access	Description
pin0	2:0	X	POR:4	R/W	Pn.0 Default Output Drive Mode
pin1	6:4	X	POR:4	R/W	Pn.1 Default Output Drive Mode
pin2	10:8	X	POR:4	R/W	Pn.2 Default Output Drive Mode
pin3	14:12	X	POR:4	R/W	Pn.3 Default Output Drive Mode
pin4	18:16	X	POR:4	R/W	Pn.4 Default Output Drive Mode
pin5	22:20	X	POR:4	R/W	Pn.5 Default Output Drive Mode
pin6	26:24	X	POR:4	R/W	Pn.6 Default Output Drive Mode
pin7	30:28	X	POR:4	R/W	Pn.7 Default Output Drive Mode

This field determines the default output drive mode and output drive value for the associated GPIO pin following system reset.

- 0: GPIO defaults to Drive 0 mode following system reset.
- 1: GPIO defaults to weak pulldown mode following system reset.
- 2: GPIO defaults to weak pullup mode following system reset.
- 3: GPIO defaults to Drive 1 mode following system reset.
- 4: GPIO defaults to high impedance state following system reset.

5.5.2 GPIO_FREE_Pn

GPIO_FREE_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Sys Reset	Access	Description
pin0	0	s	R/O	Pn.0 GPIO Mode Acknowledge
pin1	1	s	R/O	Pn.1 GPIO Mode Acknowledge
pin2	2	s	R/O	Pn.2 GPIO Mode Acknowledge
pin3	3	s	R/O	Pn.3 GPIO Mode Acknowledge
pin4	4	s	R/O	Pn.4 GPIO Mode Acknowledge
pin5	5	s	R/O	Pn.5 GPIO Mode Acknowledge
pin6	6	s	R/O	Pn.6 GPIO Mode Acknowledge
pin7	7	s	R/O	Pn.7 GPIO Mode Acknowledge

Each bit determines the availability of the associated port pin for GPIO use. If a pin is not available for GPIO use, this means that it has been requested for use by a higher-priority function. Another description would be that these bits act as acknowledge bits for GPIO mode requests (similar to the ACK bits in the I/O Manager module), with the difference that the GPIO mode 'request' is always active; this is not a problem since GPIO is the lowest priority mode that can be requested for pins.

- 0: Port pin is not available for GPIO use.
- 1: Port pin is available for GPIO use.

Note All GPIO registers of this type (GPIO_FREE_Px) follow this same format.

5.5.3 GPIO_OUT_MODE_Pn

GPIO_OUT_MODE_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Sys Reset	Access	Description
pin0	3:0	special	R/W	Pn.0 Output Drive Mode
pin1	7:4	special	R/W	Pn.1 Output Drive Mode
pin2	11:8	special	R/W	Pn.2 Output Drive Mode
pin3	15:12	special	R/W	Pn.3 Output Drive Mode
pin4	19:16	special	R/W	Pn.4 Output Drive Mode
pin5	23:20	special	R/W	Pn.5 Output Drive Mode
pin6	27:24	special	R/W	Pn.6 Output Drive Mode
pin7	31:28	special	R/W	Pn.7 Output Drive Mode

- 00: out_val=0: High impedance, out_val=1: Weak pullup
- 01: out_val=0: Open drain, out_val=1: High impedance
- 02: out_val=0: Open drain, out_val=1: Weak pullup
- 03: out_val=0: High impedance, out_val=1: Weak pullup
- 04: out_val=X: High impedance
- 05: out_val=0: Drive LOW, out_val=1: Drive HIGH
- 06: out_val=X: High impedance
- 07: out_val=0: Slow drive LOW, out_val=1: Slow drive HIGH
- 08: out_val=X: High impedance
- 09: out_val=0: Fast drive LOW, out_val=1: Fast drive HIGH
- 10: out_val=0: Weak pulldown, out_val=1: High impedance
- 11: out_val=0: High impedance, out_val=1: Drive HIGH
- 12: out_val=0: Weak pulldown, out_val=1: Drive HIGH
- 13: out_val=0: High impedance, out_val=1: Weak pullup
- 14: out_val=0: High impedance, out_val=1: Weak pullup
- 15: out_val=X: High impedance with input buffer disabled

5.5.4 GPIO_OUT_VAL_Pn

GPIO_OUT_VAL_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Sys Reset	Access	Description
pin0	0	1	R/W	Pn.0 GPIO Output Drive Value
pin1	1	1	R/W	Pn.1 GPIO Output Drive Value
pin2	2	1	R/W	Pn.2 GPIO Output Drive Value
pin3	3	1	R/W	Pn.3 GPIO Output Drive Value
pin4	4	1	R/W	Pn.4 GPIO Output Drive Value
pin5	5	1	R/W	Pn.5 GPIO Output Drive Value
pin6	6	1	R/W	Pn.6 GPIO Output Drive Value
pin7	7	1	R/W	Pn.7 GPIO Output Drive Value

In GPIO mode, selects output drive state for pin.

5.5.5 GPIO_FUNC_SEL_P0

GPIO_FUNC_SEL_P0.pin0

Field	Bits	Sys Reset	Access	Description
pin0	3:0	0000b	R/W	P0.0 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 0
- 2: 32-bit Timer 0 I/O

GPIO_FUNC_SEL_P0.pin1

Field	Bits	Sys Reset	Access	Description
pin1	7:4	0000b	R/W	P0.1 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 1

- 2: 32-bit Timer 1 I/O

GPIO_FUNC_SEL_P0.pin2

Field	Bits	Sys Reset	Access	Description
pin2	11:8	0000b	R/W	P0.2 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 2
- 2: 32-bit Timer 2 I/O

GPIO_FUNC_SEL_P0.pin3

Field	Bits	Sys Reset	Access	Description
pin3	15:12	0000b	R/W	P0.3 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 3
- 2: 32-bit Timer 3 I/O

GPIO_FUNC_SEL_P0.pin4

Field	Bits	Sys Reset	Access	Description
pin4	19:16	0000b	R/W	P0.4 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 4
- 2: 32-bit Timer 4 I/O

GPIO_FUNC_SEL_P0.pin5

Field	Bits	Sys Reset	Access	Description
pin5	23:20	0000b	R/W	P0.5 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 5
- 2: 32-bit Timer 5 I/O

GPIO_FUNC_SEL_P0.pin6

Field	Bits	Sys Reset	Access	Description
pin6	27:24	0000b	R/W	P0.6 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 6
- 2: 32-bit Timer 0 I/O

GPIO_FUNC_SEL_P0.pin7

Field	Bits	Sys Reset	Access	Description
pin7	31:28	0000b	R/W	P0.7 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 7
- 2: 32-bit Timer 1 I/O

5.5.6 GPIO_FUNC_SEL_P1

GPIO_FUNC_SEL_P1.pin0

Field	Bits	Sys Reset	Access	Description
pin0	3:0	0000b	R/W	P1.0 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 8

- 2: 32-bit Timer 2 I/O

GPIO_FUNC_SEL_P1.pin1

Field	Bits	Sys Reset	Access	Description
pin1	7:4	0000b	R/W	P1.1 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 9
- 2: 32-bit Timer 3 I/O

GPIO_FUNC_SEL_P1.pin2

Field	Bits	Sys Reset	Access	Description
pin2	11:8	0000b	R/W	P1.2 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 10
- 2: 32-bit Timer 4 I/O

GPIO_FUNC_SEL_P1.pin3

Field	Bits	Sys Reset	Access	Description
pin3	15:12	0000b	R/W	P1.3 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 11
- 2: 32-bit Timer 5 I/O

GPIO_FUNC_SEL_P1.pin4

Field	Bits	Sys Reset	Access	Description
pin4	19:16	0000b	R/W	P1.4 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 12
- 2: 32-bit Timer 0 I/O

GPIO_FUNC_SEL_P1.pin5

Field	Bits	Sys Reset	Access	Description
pin5	23:20	0000b	R/W	P1.5 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 13
- 2: 32-bit Timer 1 I/O

GPIO_FUNC_SEL_P1.pin6

Field	Bits	Sys Reset	Access	Description
pin6	27:24	0000b	R/W	P1.6 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 14
- 2: 32-bit Timer 2 I/O

GPIO_FUNC_SEL_P1.pin7

Field	Bits	Sys Reset	Access	Description
pin7	31:28	0000b	R/W	P1.7 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 15
- 2: 32-bit Timer 3 I/O

5.5.7 GPIO_FUNC_SEL_P2

GPIO_FUNC_SEL_P2.pin0

Field	Bits	Sys Reset	Access	Description
pin0	3:0	0000b	R/W	P2.0 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 0
- 2: 32-bit Timer 4 I/O

GPIO_FUNC_SEL_P2.pin1

Field	Bits	Sys Reset	Access	Description
pin1	7:4	0000b	R/W	P2.1 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 1
- 2: 32-bit Timer 5 I/O

GPIO_FUNC_SEL_P2.pin2

Field	Bits	Sys Reset	Access	Description
pin2	11:8	0000b	R/W	P2.2 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 2

- 2: 32-bit Timer 0 I/O

GPIO_FUNC_SEL_P2.pin3

Field	Bits	Sys Reset	Access	Description
pin3	15:12	0000b	R/W	P2.3 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 3
- 2: 32-bit Timer 1 I/O

GPIO_FUNC_SEL_P2.pin4

Field	Bits	Sys Reset	Access	Description
pin4	19:16	0000b	R/W	P2.4 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 4
- 2: 32-bit Timer 2 I/O

GPIO_FUNC_SEL_P2.pin5

Field	Bits	Sys Reset	Access	Description
pin5	23:20	0000b	R/W	P2.5 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 5
- 2: 32-bit Timer 3 I/O

GPIO_FUNC_SEL_P2.pin6

Field	Bits	Sys Reset	Access	Description
pin6	27:24	0000b	R/W	P2.6 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 6
- 2: 32-bit Timer 4 I/O

GPIO_FUNC_SEL_P2.pin7

Field	Bits	Sys Reset	Access	Description
pin7	31:28	0000b	R/W	P2.7 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 7
- 2: 32-bit Timer 5 I/O

5.5.8 GPIO_FUNC_SEL_P3

GPIO_FUNC_SEL_P3.pin0

Field	Bits	Sys Reset	Access	Description
pin0	3:0	0000b	R/W	P3.0 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 8
- 2: 32-bit Timer 0 I/O

GPIO_FUNC_SEL_P3.pin1

Field	Bits	Sys Reset	Access	Description
pin1	7:4	0000b	R/W	P3.1 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 9

- 2: 32-bit Timer 1 I/O

GPIO_FUNC_SEL_P3.pin2

Field	Bits	Sys Reset	Access	Description
pin2	11:8	0000b	R/W	P3.2 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 10
- 2: 32-bit Timer 2 I/O

GPIO_FUNC_SEL_P3.pin3

Field	Bits	Sys Reset	Access	Description
pin3	15:12	0000b	R/W	P3.3 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 11
- 2: 32-bit Timer 3 I/O

GPIO_FUNC_SEL_P3.pin4

Field	Bits	Sys Reset	Access	Description
pin4	19:16	0000b	R/W	P3.4 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 12
- 2: 32-bit Timer 4 I/O

GPIO_FUNC_SEL_P3.pin5

Field	Bits	Sys Reset	Access	Description
pin5	23:20	0000b	R/W	P3.5 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 13
- 2: 32-bit Timer 5 I/O

GPIO_FUNC_SEL_P3.pin6

Field	Bits	Sys Reset	Access	Description
pin6	27:24	0000b	R/W	P3.6 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 14
- 2: 32-bit Timer 0 I/O

GPIO_FUNC_SEL_P3.pin7

Field	Bits	Sys Reset	Access	Description
pin7	31:28	0000b	R/W	P3.7 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 15
- 2: 32-bit Timer 1 I/O

5.5.9 GPIO_FUNC_SEL_P4

GPIO_FUNC_SEL_P4.pin0

Field	Bits	Sys Reset	Access	Description
pin0	3:0	0000b	R/W	P4.0 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 0

- 2: 32-bit Timer 2 I/O

GPIO_FUNC_SEL_P4.pin1

Field	Bits	Sys Reset	Access	Description
pin1	7:4	0000b	R/W	P4.1 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 1
- 2: 32-bit Timer 3 I/O

GPIO_FUNC_SEL_P4.pin2

Field	Bits	Sys Reset	Access	Description
pin2	11:8	0000b	R/W	P4.2 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 2
- 2: 32-bit Timer 4 I/O

GPIO_FUNC_SEL_P4.pin3

Field	Bits	Sys Reset	Access	Description
pin3	15:12	0000b	R/W	P4.3 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 3
- 2: 32-bit Timer 5 I/O

GPIO_FUNC_SEL_P4.pin4

Field	Bits	Sys Reset	Access	Description
pin4	19:16	0000b	R/W	P4.4 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 4
- 2: 32-bit Timer 0 I/O

GPIO_FUNC_SEL_P4.pin5

Field	Bits	Sys Reset	Access	Description
pin5	23:20	0000b	R/W	P4.5 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 5
- 2: 32-bit Timer 1 I/O

GPIO_FUNC_SEL_P4.pin6

Field	Bits	Sys Reset	Access	Description
pin6	27:24	0000b	R/W	P4.6 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 6
- 2: 32-bit Timer 2 I/O

GPIO_FUNC_SEL_P4.pin7

Field	Bits	Sys Reset	Access	Description
pin7	31:28	0000b	R/W	P4.7 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 7
- 2: 32-bit Timer 3 I/O

5.5.10 GPIO_FUNC_SEL_P5

GPIO_FUNC_SEL_P5.pin0

Field	Bits	Sys Reset	Access	Description
pin0	3:0	0000b	R/W	P5.0 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 8
- 2: 32-bit Timer 4 I/O

GPIO_FUNC_SEL_P5.pin1

Field	Bits	Sys Reset	Access	Description
pin1	7:4	0000b	R/W	P5.1 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 9
- 2: 32-bit Timer 5 I/O

GPIO_FUNC_SEL_P5.pin2

Field	Bits	Sys Reset	Access	Description
pin2	11:8	0000b	R/W	P5.2 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 10

- 2: 32-bit Timer 0 I/O

GPIO_FUNC_SEL_P5.pin3

Field	Bits	Sys Reset	Access	Description
pin3	15:12	0000b	R/W	P5.3 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 11
- 2: 32-bit Timer 1 I/O

GPIO_FUNC_SEL_P5.pin4

Field	Bits	Sys Reset	Access	Description
pin4	19:16	0000b	R/W	P5.4 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 12
- 2: 32-bit Timer 2 I/O

GPIO_FUNC_SEL_P5.pin5

Field	Bits	Sys Reset	Access	Description
pin5	23:20	0000b	R/W	P5.5 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 13
- 2: 32-bit Timer 3 I/O

GPIO_FUNC_SEL_P5.pin6

Field	Bits	Sys Reset	Access	Description
pin6	27:24	0000b	R/W	P5.6 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 14
- 2: 32-bit Timer 4 I/O

GPIO_FUNC_SEL_P5.pin7

Field	Bits	Sys Reset	Access	Description
pin7	31:28	0000b	R/W	P5.7 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 15
- 2: 32-bit Timer 5 I/O

5.5.11 GPIO_FUNC_SEL_P6

GPIO_FUNC_SEL_P6.pin0

Field	Bits	Sys Reset	Access	Description
pin0	3:0	0000b	R/W	P6.0 Output Function Select

- 0: Firmware control (with OUT_VAL)
- 1: Pulse train 0
- 2: 32-bit Timer 0 I/O

5.5.12 GPIO_IN_MODE_Pn

GPIO_IN_MODE_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Sys Reset	Access	Description
pin0	1:0	10b	R/W	Pn.0 Input Monitoring Mode
pin1	5:4	10b	R/W	Pn.1 Input Monitoring Mode
pin2	9:8	10b	R/W	Pn.2 Input Monitoring Mode
pin3	13:12	10b	R/W	Pn.3 Input Monitoring Mode
pin4	17:16	10b	R/W	Pn.4 Input Monitoring Mode
pin5	21:20	10b	R/W	Pn.5 Input Monitoring Mode
pin6	25:24	10b	R/W	Pn.6 Input Monitoring Mode
pin7	29:28	10b	R/W	Pn.7 Input Monitoring Mode

Determines how corresponding GPIO Input Value bit is calculated; also affects input signal for interrupt detection on this pin (if enabled).

- 00b: Normal input
- 01b: Inverted input
- 10b: Always returns 0 regardless of pin logic level (default)
- 11b: Always returns 1 regardless of logic level at pin

5.5.13 GPIO_IN_VAL_Pn

GPIO_IN_VAL_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Sys Reset	Access	Description
pin0	0	s	R/O	Pn.0 Input Value
pin1	1	s	R/O	Pn.1 Input Value
pin2	2	s	R/O	Pn.2 Input Value
pin3	3	s	R/O	Pn.3 Input Value
pin4	4	s	R/O	Pn.4 Input Value
pin5	5	s	R/O	Pn.5 Input Value
pin6	6	s	R/O	Pn.6 Input Value
pin7	7	s	R/O	Pn.7 Input Value

Returns current input value on this pin, as modified by the corresponding Input Monitoring Mode setting.

5.5.14 GPIO_INT_MODE_Pn

GPIO_INT_MODE_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Sys Reset	Access	Description
pin0	2:0	000b	R/W	Pn.0 GPIO Interrupt Detection Mode
pin1	6:4	000b	R/W	Pn.1 GPIO Interrupt Detection Mode
pin2	10:8	000b	R/W	Pn.2 GPIO Interrupt Detection Mode
pin3	14:12	000b	R/W	Pn.3 GPIO Interrupt Detection Mode
pin4	18:16	000b	R/W	Pn.4 GPIO Interrupt Detection Mode
pin5	22:20	000b	R/W	Pn.5 GPIO Interrupt Detection Mode
pin6	26:24	000b	R/W	Pn.6 GPIO Interrupt Detection Mode
pin7	30:28	000b	R/W	Pn.7 GPIO Interrupt Detection Mode

Sets interrupt detection condition for this pin.

- 000b: Disabled
- 001b: Detect interrupt on falling edge
- 010b: Detect interrupt on rising edge
- 011b: Detect interrupt on rising or falling edge
- 100b: Active low state interrupt detection
- 101b: Active high state interrupt detection
- 11xb: Disabled

5.5.15 GPIO_INTFL_Pn

GPIO_INTFL_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Sys Reset	Access	Description
pin0	0	0	W1C	Pn.0 External Interrupt Flags
pin1	1	0	W1C	Pn.1 External Interrupt Flags
pin2	2	0	W1C	Pn.2 External Interrupt Flags
pin3	3	0	W1C	Pn.3 External Interrupt Flags
pin4	4	0	W1C	Pn.4 External Interrupt Flags
pin5	5	0	W1C	Pn.5 External Interrupt Flags
pin6	6	0	W1C	Pn.6 External Interrupt Flags
pin7	7	0	W1C	Pn.7 External Interrupt Flags

Write a 1 value to clear this bit; writes to 0 have no effect.

Set to 1 by hardware when an interrupt has been detected on this pin.

5.5.16 GPIO_INTEN_Pn

GPIO_INTEN_Pn.[pin0, pin1, pin2, pin3, pin4, pin5, pin6, pin7]

Field	Bits	Sys Reset	Access	Description
pin0	0	0	R/W	Pn.0 External Interrupt Enable
pin1	1	0	R/W	Pn.1 External Interrupt Enable
pin2	2	0	R/W	Pn.2 External Interrupt Enable
pin3	3	0	R/W	Pn.3 External Interrupt Enable
pin4	4	0	R/W	Pn.4 External Interrupt Enable
pin5	5	0	R/W	Pn.5 External Interrupt Enable
pin6	6	0	R/W	Pn.6 External Interrupt Enable
pin7	7	0	R/W	Pn.7 External Interrupt Enable

Enables interrupt to be triggered for this pin when the corresponding Interrupt Flag bit is set.

5.6 Registers (IOMAN)

Address	Register	Access	Description	Reset By
0x40000C00	IOMAN_WUD_REQ0	R/W	Wakeup Detect Mode Request Register 0 (P0/P1/P2/P3)	Sys
0x40000C04	IOMAN_WUD_REQ1	R/W	Wakeup Detect Mode Request Register 1 (P4/P5/P6)	Sys
0x40000C08	IOMAN_WUD_ACK0	R/O	Wakeup Detect Mode Acknowledge Register 0 (P0/P1/P2/P3)	Sys
0x40000C0C	IOMAN_WUD_ACK1	R/O	Wakeup Detect Mode Acknowledge Register 1 (P4/P5/P6)	Sys
0x40000C10	IOMAN_ALI_REQ0	R/W	Analog Input Request Register 0 (P0/P1/P2/P3)	Sys
0x40000C14	IOMAN_ALI_REQ1	R/W	Analog Input Request Register 1 (P4/P5/P6)	Sys
0x40000C18	IOMAN_ALI_ACK0	R/O	Analog Input Acknowledge Register 0 (P0/P1/P2/P3)	Sys
0x40000C1C	IOMAN_ALI_ACK1	R/O	Analog Input Acknowledge Register 1 (P4/P5/P6)	Sys
0x40000C20	IOMAN_ALI_CONNECT0	R/W	Analog I/O Connection Control Register 0	Sys
0x40000C24	IOMAN_ALI_CONNECT1	R/W	Analog I/O Connection Control Register 1	Sys
0x40000C28	IOMAN_SPIX_REQ	R/W	SPIX I/O Mode Request	Sys
0x40000C2C	IOMAN_SPIX_ACK	R/O	SPIX I/O Mode Acknowledge	Sys
0x40000C30	IOMAN_UART0_REQ	R/W	UART0 I/O Mode Request	Sys
0x40000C34	IOMAN_UART0_ACK	R/O	UART0 I/O Mode Acknowledge	Sys
0x40000C38	IOMAN_UART1_REQ	R/W	UART1 I/O Mode Request	Sys
0x40000C3C	IOMAN_UART1_ACK	R/O	UART1 I/O Mode Acknowledge	Sys
0x40000C40	IOMAN_UART2_REQ	R/W	UART2 I/O Mode Request	Sys
0x40000C44	IOMAN_UART2_ACK	R/O	UART2 I/O Mode Acknowledge	Sys
0x40000C48	IOMAN_UART3_REQ	R/W	UART3 I/O Mode Request	Sys
0x40000C4C	IOMAN_UART3_ACK	R/O	UART3 I/O Mode Acknowledge	Sys
0x40000C50	IOMAN_I2CM0_REQ	R/W	I2C Master 0 I/O Request	Sys
0x40000C54	IOMAN_I2CM0_ACK	R/O	I2C Master 0 I/O Acknowledge	Sys

Address	Register	Access	Description	Reset By
0x40000C58	IOMAN_I2CM1_REQ	R/W	I2C Master 1 I/O Request	Sys
0x40000C5C	IOMAN_I2CM1_ACK	R/O	I2C Master 1 I/O Acknowledge	Sys
0x40000C60	IOMAN_I2CM2_REQ	R/W	I2C Master 2 I/O Request	Sys
0x40000C64	IOMAN_I2CM2_ACK	R/O	I2C Master 2 I/O Acknowledge	Sys
0x40000C68	IOMAN_I2CS_REQ	R/W	I2C Slave I/O Request	Sys
0x40000C6C	IOMAN_I2CS_ACK	R/O	I2C Slave I/O Acknowledge	Sys
0x40000C70	IOMAN_SPI0_REQ	R/W	SPI Master 0 I/O Mode Request	Sys
0x40000C74	IOMAN_SPI0_ACK	R/O	SPI Master 0 I/O Mode Acknowledge	Sys
0x40000C78	IOMAN_SPI1_REQ	R/W	SPI Master 1 I/O Mode Request	Sys
0x40000C7C	IOMAN_SPI1_ACK	R/O	SPI Master 1 I/O Mode Acknowledge	Sys
0x40000C80	IOMAN_SPI2_REQ	R/W	SPI Master 2 I/O Mode Request	Sys
0x40000C84	IOMAN_SPI2_ACK	R/O	SPI Master 2 I/O Mode Acknowledge	Sys
0x40000C88	IOMAN_SPIB_REQ	R/O	SPI Bridge I/O Mode Request	Sys
0x40000C8C	IOMAN_SPIB_ACK	R/O	SPI Bridge I/O Mode Acknowledge	Sys
0x40000C90	IOMAN_OWM_REQ	R/W	1-Wire Master I/O Mode Request	Sys
0x40000C94	IOMAN_OWM_ACK	R/W	1-Wire Master I/O Mode Acknowledge	Sys

5.6.1 IOMAN_WUD_REQ0

IOMAN_WUD_REQ0.wud_req_p0

Field	Bits	Sys Reset	Access	Description
wud_req_p0	7:0	00000000b	R/W	Wakeup Detect Request Mode: P0[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ0.wud_req_p1

Field	Bits	Sys Reset	Access	Description
wud_req_p1	15:8	00000000b	R/W	Wakeup Detect Request Mode: P1[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ0.wud_req_p2

Field	Bits	Sys Reset	Access	Description
wud_req_p2	23:16	00000000b	R/W	Wakeup Detect Request Mode: P2[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ0.wud_req_p3

Field	Bits	Sys Reset	Access	Description
wud_req_p3	31:24	00000000b	R/W	Wakeup Detect Request Mode: P3[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

5.6.2 IOMAN_WUD_REQ1

IOMAN_WUD_REQ1.wud_req_p4

Field	Bits	Sys Reset	Access	Description
wud_req_p4	7:0	00000000b	R/W	Wakeup Detect Request Mode: P4[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ1.wud_req_p5

Field	Bits	Sys Reset	Access	Description
wud_req_p5	15:8	00000000b	R/W	Wakeup Detect Request Mode: P5[7:0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

IOMAN_WUD_REQ1.wud_req_p6

Field	Bits	Sys Reset	Access	Description
wud_req_p6	16	0b	R/W	Wakeup Detect Request Mode: P6[0]

- 0:No effect
- 1:Requests enable of wakeup detect mode on the associated GPIO pin.

5.6.3 IOMAN_WUD_ACK0

IOMAN_WUD_ACK0.wud_ack_p0

Field	Bits	Sys Reset	Access	Description
wud_ack_p0	7:0	00000000b	R/O	WUD Mode Acknowledge: P0[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK0.wud_ack_p1

Field	Bits	Sys Reset	Access	Description
wud_ack_p1	15:8	00000000b	R/O	WUD Mode Acknowledge: P1[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK0.wud_ack_p2

Field	Bits	Sys Reset	Access	Description
wud_ack_p2	23:16	00000000b	R/O	WUD Mode Acknowledge: P2[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK0.wud_ack_p3

Field	Bits	Sys Reset	Access	Description
wud_ack_p3	31:24	00000000b	R/O	WUD Mode Acknowledge: P3[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

5.6.4 IOMAN_WUD_ACK1

IOMAN_WUD_ACK1.wud_ack_p4

Field	Bits	Sys Reset	Access	Description
wud_ack_p4	7:0	00000000b	R/O	WUD Mode Acknowledge: P4[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK1.wud_ack_p5

Field	Bits	Sys Reset	Access	Description
wud_ack_p5	15:8	00000000b	R/O	WUD Mode Acknowledge: P5[7:0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

IOMAN_WUD_ACK1.wud_ack_p6

Field	Bits	Sys Reset	Access	Description
wud_ack_p6	16	0b	R/O	WUD Mode Acknowledge: P6[0]

A '1' value indicates that the associated pin has been enabled for wakeup detection mode.

5.6.5 IOMAN_ALI_REQ0

IOMAN_ALI_REQ0.ali_req_p0

Field	Bits	Sys Reset	Access	Description
ali_req_p0	7:0	00000000b	R/W	Analog Input Mode Request: P0[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ0.ali_req_p1

Field	Bits	Sys Reset	Access	Description
ali_req_p1	15:8	00000000b	R/W	Analog Input Mode Request: P1[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ0.ali_req_p2

Field	Bits	Sys Reset	Access	Description
ali_req_p2	23:16	00000000b	R/W	Analog Input Mode Request: P2[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ0.ali_req_p3

Field	Bits	Sys Reset	Access	Description
ali_req_p3	31:24	00000000b	R/W	Analog Input Mode Request: P3[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

5.6.6 IOMAN_ALI_REQ1

IOMAN_ALI_REQ1.ali_req_p4

Field	Bits	Sys Reset	Access	Description
ali_req_p4	7:0	00000000b	R/W	Analog Input Mode Request: P4[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ1.ali_req_p5

Field	Bits	Sys Reset	Access	Description
ali_req_p5	15:8	00000000b	R/W	Analog Input Mode Request: P5[7:0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

IOMAN_ALI_REQ1.ali_req_p6

Field	Bits	Sys Reset	Access	Description
ali_req_p6	16	0b	R/W	Analog Input Mode Request: P6[0]

- 0:No effect
- 1:Requests analog input mode on the associated pin.

5.6.7 IOMAN_ALI_ACK0

IOMAN_ALI_ACK0.ali_ack_p0

Field	Bits	Sys Reset	Access	Description
ali_ack_p0	7:0	00000000b	R/O	Analog In Mode Acknowledge: P0[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK0.ali_ack_p1

Field	Bits	Sys Reset	Access	Description
ali_ack_p1	15:8	00000000b	R/O	Analog In Mode Acknowledge: P1[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK0.ali_ack_p2

Field	Bits	Sys Reset	Access	Description
ali_ack_p2	23:16	00000000b	R/O	Analog In Mode Acknowledge: P2[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK0.ali_ack_p3

Field	Bits	Sys Reset	Access	Description
ali_ack_p3	31:24	00000000b	R/O	Analog In Mode Acknowledge: P3[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

5.6.8 IOMAN_ALI_ACK1

IOMAN_ALI_ACK1.ali_ack_p4

Field	Bits	Sys Reset	Access	Description
ali_ack_p4	7:0	00000000b	R/O	Analog In Mode Acknowledge: P4[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK1.ali_ack_p5

Field	Bits	Sys Reset	Access	Description
ali_ack_p5	15:8	00000000b	R/O	Analog In Mode Acknowledge: P5[7:0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

IOMAN_ALI_ACK1.ali_ack_p6

Field	Bits	Sys Reset	Access	Description
ali_ack_p6	16	0b	R/O	Analog In Mode Acknowledge: P6[0]

A '1' value indicates that the associated pin has been enabled for analog input mode.

5.6.9 IOMAN_ALI_CONNECT0

IOMAN_ALI_CONNECT0

Sys Reset	Access	Description
00000000h	R/W	Analog I/O Connection Control Register 0

Selects analog connection input for GPIO 0 through 31.

5.6.10 IOMAN_ALI_CONNECT1

IOMAN_ALI_CONNECT1

Sys Reset	Access	Description
00000000h	R/W	Analog I/O Connection Control Register 1

Selects analog connection input for GPIO 32 through 48.

5.6.11 IOMAN_SPIX_REQ

IOMAN_SPIX_REQ.core_io_req

Field	Bits	Sys Reset	Access	Description
core_io_req	4	0	R/W	SPIX Core I/O Request

1:Requests SPIX mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPIX_REQ.ss0_io_req

Field	Bits	Sys Reset	Access	Description
ss0_io_req	8	0	R/W	SPIX SS[0] I/O Request

1:Requests SPIX mode for SS[0].

IOMAN_SPIX_REQ.ss1_io_req

Field	Bits	Sys Reset	Access	Description
ss1_io_req	9	0	R/W	SPIX SS[1] I/O Request

1:Requests SPIX mode for SS[1].

IOMAN_SPIX_REQ.ss2_io_req

Field	Bits	Sys Reset	Access	Description
ss2_io_req	10	0	R/W	SPIX SS[2] I/O Request

1:Requests SPIX mode for SS[2].

IOMAN_SPIX_REQ.quad_io_req

Field	Bits	Sys Reset	Access	Description
quad_io_req	12	0	R/W	SPIX Quad I/O Request

1:Requests SPIX mode for SDIO[2] and SDIO[3].

IOMAN_SPIX_REQ.fast_mode

Field	Bits	Sys Reset	Access	Description
fast_mode	16	0	R/W	SPIX Fast Mode Request

1:Enables faster pad output transitions.

5.6.12 IOMAN_SPIX_ACK

IOMAN_SPIX_ACK.core_io_ack

Field	Bits	Sys Reset	Access	Description
core_io_ack	4	0	R/O	SPIX Core I/O Acknowledge

1:Acknowledges SPIX mode selected for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPIX_ACK.ss0_io_ack

Field	Bits	Sys Reset	Access	Description
ss0_io_ack	8	0	R/O	SPIX SS[0] I/O Acknowledge

1:Acknowledges SPIX mode selected for SS[0].

IOMAN_SPIX_ACK.ss1_io_ack

Field	Bits	Sys Reset	Access	Description
ss1_io_ack	9	0	R/O	SPIX SS[1] I/O Acknowledge

1:Acknowledges SPIX mode selected for SS[1].

IOMAN_SPIX_ACK.ss2_io_ack

Field	Bits	Sys Reset	Access	Description
ss2_io_ack	10	0	R/O	SPIX SS[2] I/O Acknowledge

1:Acknowledges SPIX mode selected for SS[2].

IOMAN_SPIX_ACK.quad_io_ack

Field	Bits	Sys Reset	Access	Description
quad_io_ack	12	0	R/O	SPIX Quad I/O Acknowledge

1:Acknowledges SPIX mode selected for SDIO[2] and SDIO[3].

IOMAN_SPIX_ACK.fast_mode

Field	Bits	Sys Reset	Access	Description
fast_mode	16	0	R/O	SPIX Fast Mode Acknowledge

1:Mirror of SPIX Fast Mode Request bit

5.6.13 IOMAN_UART0_REQ

IOMAN_UART0_REQ.io_map

Field	Bits	Sys Reset	Access	Description
io_map	0	0	R/W	UART0 TX/RX I/O Mapping Select

- 0: Select pin mapping A for UART0 TX and RX pins
- 1: Select pin mapping B for UART0 TX and RX pins

IOMAN_UART0_REQ.cts_map

Field	Bits	Sys Reset	Access	Description
cts_map	1	0	R/W	UART0 CTS I/O Mapping Select

- 0: Select pin mapping A for UART0 CTS pin
- 1: Select pin mapping B for UART0 CTS pin

IOMAN_UART0_REQ.rts_map

Field	Bits	Sys Reset	Access	Description
rts_map	2	0	R/W	UART0 RTS I/O Mapping Select

- 0: Select pin mapping A for UART0 RTS pin
- 1: Select pin mapping B for UART0 RTS pin

IOMAN_UART0_REQ.io_req

Field	Bits	Sys Reset	Access	Description
io_req	4	0	R/W	UART0 TX/RX I/O Request

1:Requests UART0 mode for TX and RX pins.

IOMAN_UART0_REQ.cts_io_req

Field	Bits	Sys Reset	Access	Description
cts_io_req	5	0	R/W	UART0 CTS I/O Request

1:Requests UART0 mode for CTS pin.

IOMAN_UART0_REQ.rts_io_req

Field	Bits	Sys Reset	Access	Description
rts_io_req	6	0	R/W	UART0 RTS I/O Request

1:Requests UART0 mode for RTS pin.

5.6.14 IOMAN_UART0_ACK

IOMAN_UART0_ACK.io_map

Field	Bits	Sys Reset	Access	Description
io_map	0	0	R/O	UART0 TX/RX I/O Mapping Acknowledge

Mirror of UART0 TX/RX I/O Mapping Select

IOMAN_UART0_ACK.cts_map

Field	Bits	Sys Reset	Access	Description
cts_map	1	0	R/O	UART0 CTS I/O Mapping Acknowledge

Mirror of UART0 CTS I/O Mapping Select

IOMAN_UART0_ACK.rts_map

Field	Bits	Sys Reset	Access	Description
rts_map	2	0	R/O	UART0 RTS I/O Mapping Acknowledge

Mirror of UART0 RTS I/O Mapping Select

IOMAN_UART0_ACK.io_ack

Field	Bits	Sys Reset	Access	Description
io_ack	4	0	R/O	UART0 TX/RX I/O Acknowledge

1:Acknowledges UART0 mode selected for TX and RX.

IOMAN_UART0_ACK.cts_io_ack

Field	Bits	Sys Reset	Access	Description
cts_io_ack	5	0	R/O	UART0 CTS I/O Acknowledge

1:Acknowledges UART0 mode selected for CTS.

IOMAN_UART0_ACK.rts_io_ack

Field	Bits	Sys Reset	Access	Description
rts_io_ack	6	0	R/O	UART0 RTS I/O Acknowledge

1:Acknowledges UART0 mode selected for RTS.

5.6.15 IOMAN_UART1_REQ

IOMAN_UART1_REQ.io_map

Field	Bits	Sys Reset	Access	Description
io_map	0	0	R/W	UART1 TX/RX I/O Mapping Select

- 0: Select pin mapping A for UART1 TX and RX pins
- 1: Select pin mapping B for UART1 TX and RX pins

IOMAN_UART1_REQ.cts_map

Field	Bits	Sys Reset	Access	Description
cts_map	1	0	R/W	UART1 CTS I/O Mapping Select

- 0: Select pin mapping A for UART1 CTS pin
- 1: Select pin mapping B for UART1 CTS pin

IOMAN_UART1_REQ.rts_map

Field	Bits	Sys Reset	Access	Description
rts_map	2	0	R/W	UART1 RTS I/O Mapping Select

- 0: Select pin mapping A for UART1 RTS pin
- 1: Select pin mapping B for UART1 RTS pin

IOMAN_UART1_REQ.io_req

Field	Bits	Sys Reset	Access	Description
io_req	4	0	R/W	UART1 TX/RX I/O Request

1:Requests UART1 mode for TX and RX pins.

IOMAN_UART1_REQ.cts_io_req

Field	Bits	Sys Reset	Access	Description
cts_io_req	5	0	R/W	UART1 CTS I/O Request

1:Requests UART1 mode for CTS pin.

IOMAN_UART1_REQ.rts_io_req

Field	Bits	Sys Reset	Access	Description
rts_io_req	6	0	R/W	UART1 RTS I/O Request

1:Requests UART1 mode for RTS pin.

5.6.16 IOMAN_UART1_ACK

IOMAN_UART1_ACK.io_map

Field	Bits	Sys Reset	Access	Description
io_map	0	0	R/O	UART1 TX/RX I/O Mapping Acknowledge

Mirror of UART1 TX/RX I/O Mapping Select

IOMAN_UART1_ACK.cts_map

Field	Bits	Sys Reset	Access	Description
cts_map	1	0	R/O	UART1 CTS I/O Mapping Acknowledge

Mirror of UART1 CTS I/O Mapping Select

IOMAN_UART1_ACK.rts_map

Field	Bits	Sys Reset	Access	Description
rts_map	2	0	R/O	UART1 RTS I/O Mapping Acknowledge

Mirror of UART1 RTS I/O Mapping Select

IOMAN_UART1_ACK.io_ack

Field	Bits	Sys Reset	Access	Description
io_ack	4	0	R/O	UART1 TX/RX I/O Acknowledge

1:Acknowledges UART1 mode selected for TX and RX.

IOMAN_UART1_ACK.cts_io_ack

Field	Bits	Sys Reset	Access	Description
cts_io_ack	5	0	R/O	UART1 CTS I/O Acknowledge

1:Acknowledges UART1 mode selected for CTS.

IOMAN_UART1_ACK.rts_io_ack

Field	Bits	Sys Reset	Access	Description
rts_io_ack	6	0	R/O	UART1 RTS I/O Acknowledge

1:Acknowledges UART1 mode selected for RTS.

5.6.17 IOMAN_UART2_REQ

IOMAN_UART2_REQ.io_map

Field	Bits	Sys Reset	Access	Description
io_map	0	0	R/W	UART2 TX/RX I/O Mapping Select

- 0: Select pin mapping A for UART2 TX and RX pins
- 1: Select pin mapping B for UART2 TX and RX pins

IOMAN_UART2_REQ.cts_map

Field	Bits	Sys Reset	Access	Description
cts_map	1	0	R/W	UART2 CTS I/O Mapping Select

- 0: Select pin mapping A for UART2 CTS pin
- 1: Select pin mapping B for UART2 CTS pin

IOMAN_UART2_REQ.rts_map

Field	Bits	Sys Reset	Access	Description
rts_map	2	0	R/W	UART2 RTS I/O Mapping Select

- 0: Select pin mapping A for UART2 RTS pin
- 1: Select pin mapping B for UART2 RTS pin

IOMAN_UART2_REQ.io_req

Field	Bits	Sys Reset	Access	Description
io_req	4	0	R/W	UART2 TX/RX I/O Request

1:Requests UART2 mode for TX and RX pins.

IOMAN_UART2_REQ.cts_io_req

Field	Bits	Sys Reset	Access	Description
cts_io_req	5	0	R/W	UART2 CTS I/O Request

1:Requests UART2 mode for CTS pin.

IOMAN_UART2_REQ.rts_io_req

Field	Bits	Sys Reset	Access	Description
rts_io_req	6	0	R/W	UART2 RTS I/O Request

1:Requests UART2 mode for RTS pin.

5.6.18 IOMAN_UART2_ACK

IOMAN_UART2_ACK.io_map

Field	Bits	Sys Reset	Access	Description
io_map	0	0	R/O	UART2 TX/RX I/O Mapping Acknowledge

Mirror of UART2 TX/RX I/O Mapping Select

IOMAN_UART2_ACK.cts_map

Field	Bits	Sys Reset	Access	Description
cts_map	1	0	R/O	UART2 CTS I/O Mapping Acknowledge

Mirror of UART2 CTS I/O Mapping Select

IOMAN_UART2_ACK.rts_map

Field	Bits	Sys Reset	Access	Description
rts_map	2	0	R/O	UART2 RTS I/O Mapping Acknowledge

Mirror of UART2 RTS I/O Mapping Select

IOMAN_UART2_ACK.io_ack

Field	Bits	Sys Reset	Access	Description
io_ack	4	0	R/O	UART2 TX/RX I/O Acknowledge

1:Acknowledges UART2 mode selected for TX and RX.

IOMAN_UART2_ACK.cts_io_ack

Field	Bits	Sys Reset	Access	Description
cts_io_ack	5	0	R/O	UART2 CTS I/O Acknowledge

1:Acknowledges UART2 mode selected for CTS.

IOMAN_UART2_ACK.rts_io_ack

Field	Bits	Sys Reset	Access	Description
rts_io_ack	6	0	R/O	UART2 RTS I/O Acknowledge

1:Acknowledges UART2 mode selected for RTS.

5.6.19 IOMAN_UART3_REQ

IOMAN_UART3_REQ.io_map

Field	Bits	Sys Reset	Access	Description
io_map	0	0	R/W	UART3 TX/RX I/O Mapping Select

- 0: Select pin mapping A for UART3 TX and RX pins
- 1: Select pin mapping B for UART3 TX and RX pins

IOMAN_UART3_REQ.cts_map

Field	Bits	Sys Reset	Access	Description
cts_map	1	0	R/W	UART3 CTS I/O Mapping Select

- 0: Select pin mapping A for UART3 CTS pin
- 1: Select pin mapping B for UART3 CTS pin

IOMAN_UART3_REQ.rts_map

Field	Bits	Sys Reset	Access	Description
rts_map	2	0	R/W	UART3 RTS I/O Mapping Select

- 0: Select pin mapping A for UART3 RTS pin
- 1: Select pin mapping B for UART3 RTS pin

IOMAN_UART3_REQ.io_req

Field	Bits	Sys Reset	Access	Description
io_req	4	0	R/W	UART3 TX/RX I/O Request

1:Requests UART3 mode for TX and RX pins.

IOMAN_UART3_REQ.cts_io_req

Field	Bits	Sys Reset	Access	Description
cts_io_req	5	0	R/W	UART3 CTS I/O Request

1:Requests UART3 mode for CTS pin.

IOMAN_UART3_REQ.rts_io_req

Field	Bits	Sys Reset	Access	Description
rts_io_req	6	0	R/W	UART3 RTS I/O Request

1:Requests UART3 mode for RTS pin.

5.6.20 IOMAN_UART3_ACK

IOMAN_UART3_ACK.io_map

Field	Bits	Sys Reset	Access	Description
io_map	0	0	R/O	UART3 TX/RX I/O Mapping Acknowledge

Mirror of UART3 TX/RX I/O Mapping Select

IOMAN_UART3_ACK.cts_map

Field	Bits	Sys Reset	Access	Description
cts_map	1	0	R/O	UART3 CTS I/O Mapping Acknowledge

Mirror of UART3 CTS I/O Mapping Select

IOMAN_UART3_ACK.rts_map

Field	Bits	Sys Reset	Access	Description
rts_map	2	0	R/O	UART3 RTS I/O Mapping Acknowledge

Mirror of UART3 RTS I/O Mapping Select

IOMAN_UART3_ACK.io_ack

Field	Bits	Sys Reset	Access	Description
io_ack	4	0	R/O	UART3 TX/RX I/O Acknowledge

1:Acknowledges UART3 mode selected for TX and RX.

IOMAN_UART3_ACK.cts_io_ack

Field	Bits	Sys Reset	Access	Description
cts_io_ack	5	0	R/O	UART3 CTS I/O Acknowledge

1:Acknowledges UART3 mode selected for CTS.

IOMAN_UART3_ACK.rts_io_ack

Field	Bits	Sys Reset	Access	Description
rts_io_ack	6	0	R/O	UART3 RTS I/O Acknowledge

1:Acknowledges UART3 mode selected for RTS.

5.6.21 IOMAN_I2CM0_REQ

IOMAN_I2CM0_REQ.mapping_req

Field	Bits	Sys Reset	Access	Description
mapping_req	4	0	R/W	I2C Master 0 I/O Request

1:Requests I2C Master 0 mode for SCL and SDA.

5.6.22 IOMAN_I2CM0_ACK

IOMAN_I2CM0_ACK.mapping_ack

Field	Bits	Sys Reset	Access	Description
mapping_ack	4	0	R/O	I2C Master 0 I/O Acknowledge

1:Acknowledges I2C Master 0 mode selected for SCL and SDA.

5.6.23 IOMAN_I2CM1_REQ

IOMAN_I2CM1_REQ.mapping_req

Field	Bits	Sys Reset	Access	Description
mapping_req	4	0	R/W	I2C Master 1 I/O Request

1:Requests I2C Master 1 mode for SCL and SDA.

5.6.24 IOMAN_I2CM1_ACK

IOMAN_I2CM1_ACK.mapping_ack

Field	Bits	Sys Reset	Access	Description
mapping_ack	4	0	R/O	I2C Master 1 I/O Acknowledge

1:Acknowledges I2C Master 1 mode selected for SCL and SDA.

5.6.25 IOMAN_I2CM2_REQ

IOMAN_I2CM2_REQ.mapping_req

Field	Bits	Sys Reset	Access	Description
mapping_req	4	0	R/W	I2C Master 2 I/O Request

1:Requests I2C Master 2 mode for SCL and SDA.

5.6.26 IOMAN_I2CM2_ACK

IOMAN_I2CM2_ACK.mapping_ack

Field	Bits	Sys Reset	Access	Description
mapping_ack	4	0	R/O	I2C Master 2 I/O Acknowledge

1:Acknowledges I2C Master 2 mode selected for SCL and SDA.

5.6.27 IOMAN_I2CS_REQ

IOMAN_I2CS_REQ.io_sel

Field	Bits	Sys Reset	Access	Description
io_sel	1:0	00b	R/W	I2C Slave I/O Mapping Select

- 00b: Select pin mapping A for I2C Slave SCL and SDA pins.
- 01b: Select pin mapping B for I2C Slave SCL and SDA pins.
- 10b: Select pin mapping C for I2C Slave SCL and SDA pins.
- 11b: Reserved.

IOMAN_I2CS_REQ.mapping_req

Field	Bits	Sys Reset	Access	Description
mapping_req	4	0	R/W	I2C Slave I/O Request

1:Requests I2C Slave mode for SCL and SDA pins.

5.6.28 IOMAN_I2CS_ACK

IOMAN_I2CS_ACK.io_sel

Field	Bits	Sys Reset	Access	Description
io_sel	1:0	00b	R/O	I2C Slave I/O Mapping Acknowledge

Mirror of I/O mapping select bits from I2CS_REQ.

IOMAN_I2CS_ACK.mapping_ack

Field	Bits	Sys Reset	Access	Description
mapping_ack	4	0	R/O	I2C Slave I/O Acknowledge

1:Acknowledges I2C Slave mode selected for SCL/SDA

5.6.29 IOMAN_SPI0_REQ

IOMAN_SPI0_REQ.core_io_req

Field	Bits	Sys Reset	Access	Description
core_io_req	4	0	R/W	SPI Master 0 Core I/O Request

1:Requests SPI Master 0 mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI0_REQ.ss0_io_req

Field	Bits	Sys Reset	Access	Description
ss0_io_req	8	0	R/W	SPI Master 0 SS[0] I/O Request

1:Requests SPI Master 0 mode for SS[0].

IOMAN_SPI0_REQ.ss1_io_req

Field	Bits	Sys Reset	Access	Description
ss1_io_req	9	0	R/W	SPI Master 0 SS[1] I/O Request

1:Requests SPI Master 0 mode for SS[1].

IOMAN_SPI0_REQ.ss2_io_req

Field	Bits	Sys Reset	Access	Description
ss2_io_req	10	0	R/W	SPI Master 0 SS[2] I/O Request

1:Requests SPI Master 0 mode for SS[2].

IOMAN_SPI0_REQ.ss3_io_req

Field	Bits	Sys Reset	Access	Description
ss3_io_req	11	0	R/W	SPI Master 0 SS[3] I/O Request

1:Requests SPI Master 0 mode for SS[3].

IOMAN_SPI0_REQ.ss4_io_req

Field	Bits	Sys Reset	Access	Description
ss4_io_req	12	0	R/W	SPI Master 0 SS[4] I/O Request

1:Requests SPI Master 0 mode for SS[4].

IOMAN_SPI0_REQ.quad_io_req

Field	Bits	Sys Reset	Access	Description
quad_io_req	20	0	R/W	SPI Master 0 Quad I/O Request

1:Requests SPI Master 0 mode for SDIO[2] and SDIO[3].

IOMAN_SPI0_REQ.fast_mode

Field	Bits	Sys Reset	Access	Description
fast_mode	24	0	R/W	SPI Master 0 Fast Mode Request

1:Enables faster pad output transitions.

5.6.30 IOMAN_SPI0_ACK

IOMAN_SPI0_ACK.core_io_ack

Field	Bits	Sys Reset	Access	Description
core_io_ack	4	0	R/O	SPI Master 0 Core I/O Acknowledge

1:Acknowledges SPI Master 0 mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI0_ACK.ss0_io_ack

Field	Bits	Sys Reset	Access	Description
ss0_io_ack	8	0	R/O	SPI Master 0 SS[0] I/O Acknowledge

1:Acknowledges SPI Master 0 mode for SS[0].

IOMAN_SPI0_ACK.ss1_io_ack

Field	Bits	Sys Reset	Access	Description
ss1_io_ack	9	0	R/O	SPI Master 0 SS[1] I/O Acknowledge

1:Acknowledges SPI Master 0 mode for SS[1].

IOMAN_SPI0_ACK.ss2_io_ack

Field	Bits	Sys Reset	Access	Description
ss2_io_ack	10	0	R/O	SPI Master 0 SS[2] I/O Acknowledge

1:Acknowledges SPI Master 0 mode for SS[2].

IOMAN_SPI0_ACK.ss3_io_ack

Field	Bits	Sys Reset	Access	Description
ss3_io_ack	11	0	R/O	SPI Master 0 SS[3] I/O Acknowledge

1:Acknowledges SPI Master 0 mode for SS[3].

IOMAN_SPI0_ACK.ss4_io_ack

Field	Bits	Sys Reset	Access	Description
ss4_io_ack	12	0	R/O	SPI Master 0 SS[4] I/O Acknowledge

1:Acknowledges SPI Master 0 mode for SS[4].

IOMAN_SPI0_ACK.quad_io_ack

Field	Bits	Sys Reset	Access	Description
quad_io_ack	20	0	R/O	SPI Master 0 Quad I/O Acknowledge

1:Acknowledges SPI Master 0 mode for SDIO[2] and SDIO[3].

IOMAN_SPI0_ACK.fast_mode

Field	Bits	Sys Reset	Access	Description
fast_mode	24	0	R/O	SPI Master 0 Fast Mode Acknowledge

Mirror of SPI Master 0 Fast Mode Request.

5.6.31 IOMAN_SPI1_REQ

IOMAN_SPI1_REQ.core_io_req

Field	Bits	Sys Reset	Access	Description
core_io_req	4	0	R/W	SPI Master 1 Core I/O Request

1:Requests SPI Master 1 mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI1_REQ.ss0_io_req

Field	Bits	Sys Reset	Access	Description
ss0_io_req	8	0	R/W	SPI Master 1 SS[0] I/O Request

1:Requests SPI Master 1 mode for SS[0].

IOMAN_SPI1_REQ.ss1_io_req

Field	Bits	Sys Reset	Access	Description
ss1_io_req	9	0	R/W	SPI Master 1 SS[1] I/O Request

1:Requests SPI Master 1 mode for SS[1].

IOMAN_SPI1_REQ.ss2_io_req

Field	Bits	Sys Reset	Access	Description
ss2_io_req	10	0	R/W	SPI Master 1 SS[2] I/O Request

1:Requests SPI Master 1 mode for SS[2].

IOMAN_SPI1_REQ.quad_io_req

Field	Bits	Sys Reset	Access	Description
quad_io_req	20	0	R/W	SPI Master 1 Quad I/O Request

1:Requests SPI Master 1 mode for SDIO[2] and SDIO[3].

IOMAN_SPI1_REQ.fast_mode

Field	Bits	Sys Reset	Access	Description
fast_mode	24	0	R/W	SPI Master 1 Fast Mode Request

1:Enables faster pad output transitions.

5.6.32 IOMAN_SPI1_ACK

IOMAN_SPI1_ACK.core_io_ack

Field	Bits	Sys Reset	Access	Description
core_io_ack	4	0	R/O	SPI Master 1 Core I/O Acknowledge

1:Acknowledges SPI Master 1 mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI1_ACK.ss0_io_ack

Field	Bits	Sys Reset	Access	Description
ss0_io_ack	8	0	R/O	SPI Master 1 SS[0] I/O Acknowledge

1:Acknowledges SPI Master 1 mode for SS[0].

IOMAN_SPI1_ACK.ss1_io_ack

Field	Bits	Sys Reset	Access	Description
ss1_io_ack	9	0	R/O	SPI Master 1 SS[1] I/O Acknowledge

1:Acknowledges SPI Master 1 mode for SS[1].

IOMAN_SPI1_ACK.ss2_io_ack

Field	Bits	Sys Reset	Access	Description
ss2_io_ack	10	0	R/O	SPI Master 1 SS[2] I/O Acknowledge

1:Acknowledges SPI Master 1 mode for SS[2].

IOMAN_SPI1_ACK.quad_io_ack

Field	Bits	Sys Reset	Access	Description
quad_io_ack	20	0	R/O	SPI Master 1 Quad I/O Acknowledge

1:Acknowledges SPI Master 1 mode for SDIO[2] and SDIO[3].

IOMAN_SPI1_ACK.fast_mode

Field	Bits	Sys Reset	Access	Description
fast_mode	24	0	R/O	SPI Master 1 Fast Mode Acknowledge

Mirror of SPI Master 1 Fast Mode Request.

5.6.33 IOMAN_SPI2_REQ

IOMAN_SPI2_REQ.mapping_req

Field	Bits	Sys Reset	Access	Description
mapping_req	0	0	R/W	SPI Master 2 I/O Mapping Select

- 0: Select pin mapping A for SPI Master 2 pins
- 1: Select pin mapping B for SPI Master 2 pins

IOMAN_SPI2_REQ.core_io_req

Field	Bits	Sys Reset	Access	Description
core_io_req	4	0	R/W	SPI Master 2 Core I/O Request

1:Requests SPI Master 2 mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI2_REQ.ss0_io_req

Field	Bits	Sys Reset	Access	Description
ss0_io_req	8	0	R/W	SPI Master 2 SS[0] I/O Request

1:Requests SPI Master 2 mode for SS[0].

IOMAN_SPI2_REQ.ss1_io_req

Field	Bits	Sys Reset	Access	Description
ss1_io_req	9	0	R/W	SPI Master 2 SS[1] I/O Request

1:Requests SPI Master 2 mode for SS[1].

IOMAN_SPI2_REQ.ss2_io_req

Field	Bits	Sys Reset	Access	Description
ss2_io_req	10	0	R/W	SPI Master 2 SS[2] I/O Request

1:Requests SPI Master 2 mode for SS[2].

IOMAN_SPI2_REQ.sr0_io_req

Field	Bits	Sys Reset	Access	Description
sr0_io_req	16	0	R/W	SPI Master 2 SR[0] I/O Request

1:Requests SPI Master 2 mode for SR[0].

IOMAN_SPI2_REQ.sr1_io_req

Field	Bits	Sys Reset	Access	Description
sr1_io_req	17	0	R/W	SPI Master 2 SR[1] I/O Request

1:Requests SPI Master 2 mode for SR[1].

IOMAN_SPI2_REQ.quad_io_req

Field	Bits	Sys Reset	Access	Description
quad_io_req	20	0	R/W	SPI Master 2 Quad I/O Request

1:Requests SPI Master 2 mode for SDIO[2] and SDIO[3].

IOMAN_SPI2_REQ.fast_mode

Field	Bits	Sys Reset	Access	Description
fast_mode	24	0	R/W	SPI Master 2 Fast Mode Request

1:Enables faster pad output transitions.

5.6.34 IOMAN_SPI2_ACK

IOMAN_SPI2_ACK.mapping_ack

Field	Bits	Sys Reset	Access	Description
mapping_ack	0	0	R/O	SPI Master 2 I/O Mapping Acknowledge

Mirror of SPI Master 2 I/O Mapping Select value.

IOMAN_SPI2_ACK.core_io_ack

Field	Bits	Sys Reset	Access	Description
core_io_ack	4	0	R/O	SPI Master 2 Core I/O Acknowledge

1:Acknowledges SPI Master 2 mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPI2_ACK.ss0_io_ack

Field	Bits	Sys Reset	Access	Description
ss0_io_ack	8	0	R/O	SPI Master 2 SS[0] I/O Acknowledge

1:Acknowledges SPI Master 2 mode for SS[0].

IOMAN_SPI2_ACK.ss1_io_ack

Field	Bits	Sys Reset	Access	Description
ss1_io_ack	9	0	R/O	SPI Master 2 SS[1] I/O Acknowledge

1:Acknowledges SPI Master 2 mode for SS[1].

IOMAN_SPI2_ACK.ss2_io_ack

Field	Bits	Sys Reset	Access	Description
ss2_io_ack	10	0	R/O	SPI Master 2 SS[2] I/O Acknowledge

1:Acknowledges SPI Master 2 mode for SS[2].

IOMAN_SPI2_ACK.sr0_io_req

Field	Bits	Sys Reset	Access	Description
sr0_io_req	16	0	R/O	SPI Master 2 SR[0] I/O Acknowledge

1:Acknowledges SPI Master 2 mode for SR[0].

IOMAN_SPI2_ACK.sr1_io_req

Field	Bits	Sys Reset	Access	Description
sr1_io_req	17	0	R/O	SPI Master 2 SR[1] I/O Acknowledge

1:Acknowledges SPI Master 2 mode for SR[1].

IOMAN_SPI2_ACK.quad_io_ack

Field	Bits	Sys Reset	Access	Description
quad_io_ack	20	0	R/O	SPI Master 2 Quad I/O Acknowledge

1:Acknowledges SPI Master 2 mode for SDIO[2] and SDIO[3].

IOMAN_SPI2_ACK.fast_mode

Field	Bits	Sys Reset	Access	Description
fast_mode	24	0	R/O	SPI Master 2 Fast Mode Acknowledge

Mirror of SPI Master 2 Fast Mode Request.

5.6.35 IOMAN_SPIB_REQ

IOMAN_SPIB_REQ.core_io_req

Field	Bits	Sys Reset	Access	Description
core_io_req	4	0	R/O	SPI Bridge Core I/O Request

1:Requests SPI Bridge mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPIB_REQ.quad_io_req

Field	Bits	Sys Reset	Access	Description
quad_io_req	8	0	R/O	SPI Bridge Quad I/O Request

1:Requests SPI Bridge mode for SDIO[2] and SDIO[3].

IOMAN_SPIB_REQ.fast_mode

Field	Bits	Sys Reset	Access	Description
fast_mode	12	0	R/O	SPI Bridge Fast Mode Request

1:Enables faster pad output transitions.

5.6.36 IOMAN_SPIB_ACK

IOMAN_SPIB_ACK.core_io_ack

Field	Bits	Sys Reset	Access	Description
core_io_ack	4	0	R/O	SPI Bridge Core I/O Acknowledge

1:Acknowledges SPI Bridge mode for SCLK, SDIO[0] and SDIO[1].

IOMAN_SPIB_ACK.quad_io_ack

Field	Bits	Sys Reset	Access	Description
quad_io_ack	8	0	R/O	SPI Bridge Quad I/O Acknowledge

1:Acknowledges SPI Bridge mode for SDIO[2] and SDIO[3].

IOMAN_SPIB_ACK.fast_mode

Field	Bits	Sys Reset	Access	Description
fast_mode	12	0	R/O	SPI Bridge Fast Mode Acknowledge

Mirror of SPI Bridge Fast Mode Request.

5.6.37 IOMAN_OWM_REQ

IOMAN_OWM_REQ.mapping_req

Field	Bits	Sys Reset	Access	Description
mapping_req	4	0	R/W	1-Wire Line I/O Request

1:Requests 1-Wire line mode for OWD.

IOMAN_OWM_REQ.epu_io_req

Field	Bits	Sys Reset	Access	Description
epu_io_req	5	0	R/W	External Pullup Control Line I/O Request

1:Requests 1-Wire external pullup control mode for EPU.

5.6.38 IOMAN_OWM_ACK

IOMAN_OWM_ACK.mapping_ack

Field	Bits	Sys Reset	Access	Description
mapping_ack	4	0	R/W	1-Wire Line I/O Acknowledge

1:Acknowledges 1-Wire line mode for OWD.

IOMAN_OWM_ACK.epu_io_ack

Field	Bits	Sys Reset	Access	Description
epu_io_ack	5	0	R/W	External Pullup Control Line I/O Acknowledge

1:Acknowledges 1-Wire external pullup control mode for EPU.

6 Peripheral Management Unit (PMU)

6.1 Overview

The Peripheral Management Unit (PMU) on the **MAX32620** is a DMA-based linked list processing engine. The PMU can perform operations and data transfers involving memory and/or peripherals in the Advanced Peripheral Bus (APB) and Advanced High-performance Bus (AHB) peripheral memory spaces. These operations can be performed while the main CPU is in a sleep state. This allows low-overhead peripheral operations - for which intensive CPU resources are not required - to be performed without the CPU, significantly reducing overall power consumption.

Key features of the PMU engine include:

- Six independent channels with round-robin scheduling to allow multiple parallel operations without requiring use of the CPU
- Suited for moving data to/from memory and peripherals
- Co-processor-like state machine within the **MAX32620**
- Integrated AHB bus master
- Programmed using SRAM-based PMU opcodes
- Interrupt conditions from peripherals to initiate PMU actions without requiring actual CPU-based interrupt handling
- Trigger interrupts to the CPU if needed
- Operation with CPU in sleep state reduces power consumption

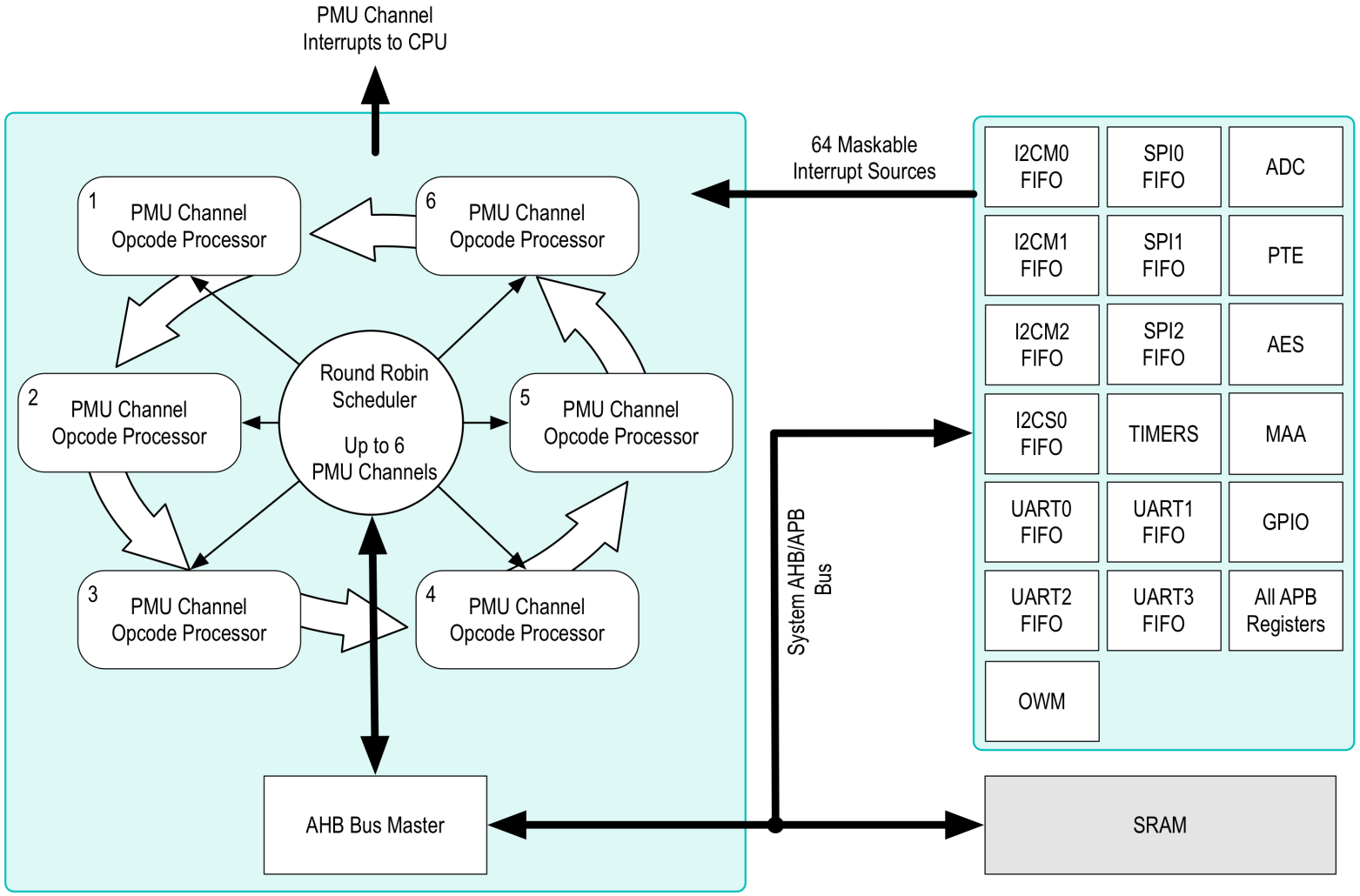

Figure 6.1: PMU Block Diagram

6.2 PMU Operation

The PMU is a six-channel, programmable state machine that executes user-defined PMU descriptor sequences. These descriptor sequences reside in the **MAX32620** memory space. Acting as an AHB master, the PMU can access any memory address in either the AHB or APB memory spaces. Each of the six channels runs independently and shares accesses to the AHB/APB address space using an internal round-robin arbiter, on a descriptor-by-descriptor basis, with priority encoding. Priority ranges from channel one with the highest priority and channel six with the lowest priority. The PMU runs from a fixed-source clock and is clocked at the same frequency as the main **MAX32620** system clock (scaling configured by CLKMAN_SYS_CLK_CTRL_0_CM4).

6.2.1 PMU Operation Descriptors

Each active PMU channel independently executes a PMU descriptor sequence specific to that channel. A descriptor sequence is made up of one or more PMU descriptors, with each descriptor consisting of a 32-bit op code doubleword and one or more operand doublewords, as described below. When configuring a PMU channel for operation, the user specifies the address of the first PMU descriptor in the descriptor sequence. With the exception of descriptors that may change the next descriptor address (such as JUMP, LOOP and BRANCH), the PMU channel executes descriptors sequentially.

Descriptor execution for a given PMU channel automatically halts when a descriptor containing (STOP == 1) is executed or an error condition is encountered. A PMU channel can also be halted manually by clearing the enable bit in the channel's PMUn_CFG register. There is no limit to the number of descriptors that can be executed in a descriptor sequence.

Each PMU channel has two control registers that control the operation of the channel. The descriptor address register PMUn_DSCADR is used to set the 32-bit address of the first descriptor in the descriptor sequence. The channel configuration register PMUn_CFG is used to start and stop descriptor execution for that channel. This register also reports status and error information.

6.2.2 Setup and PMU Channel Start

Before using a PMU channel, the user must specify an area of memory, which may be RAM or Flash memory, to hold the PMU descriptor sequence. Once the descriptor sequence has been written to memory, the user must set the starting address of the first descriptor using the PMUn_DSCADR register. To start channel execution, set the enable bit in the channel's PMUn_CFG register to 1. The PMU channel engine will then fetch the first descriptor in the sequence and begin processing.

Upon completion of the first PMU descriptor, subsequent descriptors are fetched sequentially from memory unless the descriptor address is changed by a JUMP, LOOP, or BRANCH descriptor. As each PMU descriptor is executed, the PMUn_DSCADR register is updated with the address of the next descriptor to be executed by the channel.

The end of the descriptor sequence is defined by a descriptor with the STOP bit set. Once the PMU channel executes a descriptor with (STOP == 1), the PMU channel will halt execution. When this occurs, the enable bit will be cleared, and the ll_stopped bit in the PMUn_CFG will be set to 1 by hardware to indicate the reason why the channel stopped executing.

6.2.3 PMU Channel Arbitration

The PMU is an AHB bus master, and allows each of the channels access to the AHB/APB memory space. Arbitration between PMU channels for access to the bus master is performed by a round-robin scheduler, on a descriptor-by-descriptor basis, with priority encoding.

For example: If three PMU channels are active, execution would start with channel 1, descriptor 1; then proceed to channel 2, descriptor 1; channel 3, descriptor 1; channel 1, descriptor 2; etc. If a channel is temporarily blocked due to a flow control condition, execution will move to the next active channel. For example, execution of a WAIT, POLL, or TRANSFER descriptor may cause a channel to stay at the same execution address for some time while the descriptor operation completes. But this does not block the round-robin arbitration for other active channels; these channels will continue executing in turn while the first channel completes its long operation.

6.3 PMU Descriptor Details

There are eight descriptor types which can be included in a PMU descriptor sequence. Each descriptor consists of one 32-bit descriptor op code doubleword and from one to four 32-bit descriptor parameter doublewords. The number of parameters varies according to the descriptor type, but all descriptors of a given type will always have the same number of parameters.

The available descriptor types and the total length of each type (in doublewords) is as follows:

- MOVE : 3 doublewords
- WRITE : 4 doublewords
- WAIT : 4 doublewords
- JUMP : 2 doublewords
- LOOP : 2 doublewords
- BRANCH : 5 doublewords
- POLL : 5 doublewords
- TRANSFER: 4 doublewords

6.3.1 MOVE Descriptor

The MOVE descriptor is used to copy a user-specified number of bytes from a read address location to a write address location. Read and write addresses may be in either the AHB or APB memory space. The read address and write address are specified (independently from each other) to be either fixed or incrementing. A fixed read/write address is used when reading from or writing to a FIFO or similar access point where the address does not need to change. An incrementing read/write address is used when reading from or writing to a standard memory structure such as the internal SRAM, or an internal working memory for a peripheral such as the MAA. AHB accesses may be 8-bit, 16-bit or 32-bit width as supported by the particular area of memory or the peripheral FIFO being accessed. All APB accesses are restricted to 32-bit width only.

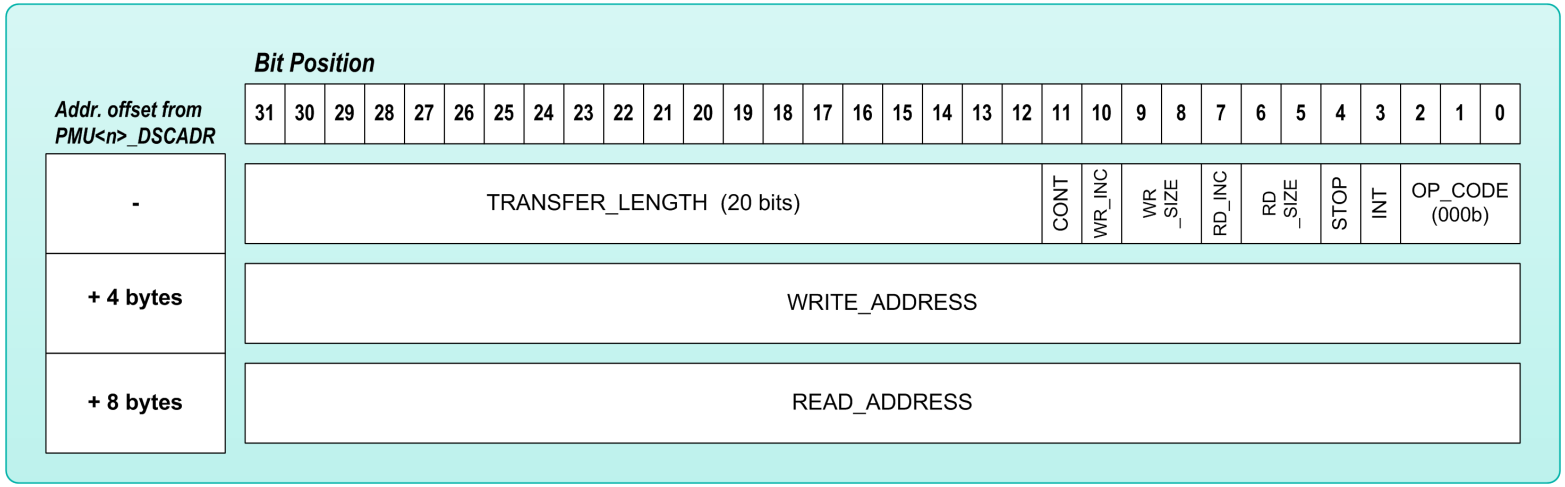

Figure 6.2: PMU MOVE Descriptor

6.3.1.1 OP_CODE

This three-bit field is set to 000b to define the descriptor as a MOVE descriptor.

6.3.1.2 INT

If this bit is set to 1, the interrupt flag for the current channel will be set to 1 after the channel completes execution of this descriptor.

6.3.1.3 STOP

If this bit is set to 1, once the PMU channel completes execution of this descriptor, the PMU channel will halt execution. When this occurs, the ll_stopped flag for the current channel will be set to 1 by hardware to indicate that the PMU descriptor sequence has completed execution.

6.3.1.4 RD_SIZE, WR_SIZE

These fields determine the width of memory access that will be used on the read side (source) and write side (destination) of the data transfer. Note that any access to APB register space requires the width to be 32-bit only; use of 8-bit or 16-bit width when accessing APB space will result in a bus error. When accessing AHB

areas, the widths for the read and write sides can be different; the PMU channel controller will automatically handle data packing/unpacking as needed.

RD SIZE	WR SIZE	Operation
00b	00b	Perform 8-bit reads and 8-bit writes
00b	01b	Perform 8-bit reads and pack data into 16-bit writes
00b	10b	Perform 8-bit reads and pack data into 32-bit writes
01b	00b	Perform 16-bit reads and unpack into 8-bit writes
01b	01b	Perform 16-bit reads and writes
01b	10b	Perform 16-bit reads and pack data into 32-bit writes
10b	00b	Perform 32-bit reads and unpack data into 8-bit writes
10b	01b	Perform 32-bit reads and unpack data into 16-bit writes
10b	10b	Perform 32-bit reads and writes
XX	11b	(Reserved)
11b	XX	(Reserved)

6.3.1.5 RD_INC

If this bit is set to 0, then read address auto-incrementing is disabled. This means that the same read address value will be used for all reads in this data transfer sequence. This is a useful setting when reading from FIFOs, where a single address location must be read repeatedly to unload data.

If this bit is set to 1, then read address auto-incrementing is enabled. This means that every read access in the data transfer will cause the read address pointer to automatically post-increment, based on the RD_SIZE width. For example, if RD_SIZE is set to 01b, then following each 16-bit read access in the data transfer sequence, the read address pointer will be incremented by 2. This is a useful setting when accessing SRAM or memory register files, where a different address is used to access each location in the memory area.

Note When executing a MOVE descriptor with (CONT == 1), the RD_INC bit must be set to the same value as the RD_INC bit from the previously-executed MOVE descriptor with (CONT == 0).

6.3.1.6 WR_INC

If this bit is set to 0, then write address auto-incrementing is disabled. This means that the same write address value will be used for all writes in this data transfer sequence. This is a useful setting when writing to FIFOs or similar locations, where a single address location must be written to repeatedly to load or process data.

If this bit is set to 1, then write address auto-incrementing is enabled. This means that every write access in the data transfer will cause the write address pointer to automatically post-increment, based on the WR_SIZE width. For example, if WR_SIZE is set to 00b, then following each 8-bit write access in the data transfer sequence, the write address pointer will be incremented by 1. This is a useful setting when accessing SRAM or memory register files, where a different address is used to access each location in the memory area.

Note When executing a MOVE descriptor with (CONT == 1), the WR_INC bit must be set to the same value as the WR_INC bit from the previously-executed MOVE descriptor with (CONT == 0).

6.3.1.7 CONT

Setting this bit to 1 allows a subsequent MOVE descriptor to continue operation using the read address, write address, and RD_INC and WR_INC values defined by a previous MOVE descriptor. Before a MOVE descriptor with (CONT == 1) can be executed, a previous MOVE descriptor with (CONT == 0) must be executed to initialize READ_ADDRESS, WRITE_ADDRESS, RD_INC and WR_INC.

If this bit is set to 0, the MOVE descriptor will not rely on any existing read and write settings from a previous MOVE descriptor and instead will use its own values for these fields.

6.3.1.8 TRANSFER_LENGTH

Total length of transfer (in bytes) from the read address to the write address.

6.3.1.9 WRITE_ADDRESS

This field specifies the write address used for the data transfer (or if WR_INC == 1, the starting write address).

Note When executing a MOVE descriptor with (CONT == 1), the WRITE_ADDRESS field must be set to the same value as the WRITE_ADDRESS field from the previously-executed MOVE descriptor with (CONT == 0). In this case, the contents of the WRITE_ADDRESS field in the MOVE descriptor with (CONT == 1) will not be used, but the two fields must still be set to identical values for proper operation.

6.3.1.10 READ_ADDRESS

This field specifies the read address used for the data transfer (or if RD_INC == 1, the starting read address).

Note When executing a MOVE descriptor with (CONT == 1), the READ_ADDRESS field must be set to the same value as the READ_ADDRESS field from the previously-executed MOVE descriptor with (CONT == 0). In this case, the contents of the READ_ADDRESS field in the MOVE descriptor with (CONT == 1) will not be used, but the two fields must still be set to identical values for proper operation.

6.3.2 WRITE Descriptor

The WRITE descriptor will cause the write value to be written to the write address location. Only bits set in the write mask field will be modified. This allows the user to set or clear individual bits in a 32-bit field. Write accesses may be in either the AHB or APB memory space. All accesses are restricted to 32-bit accesses.

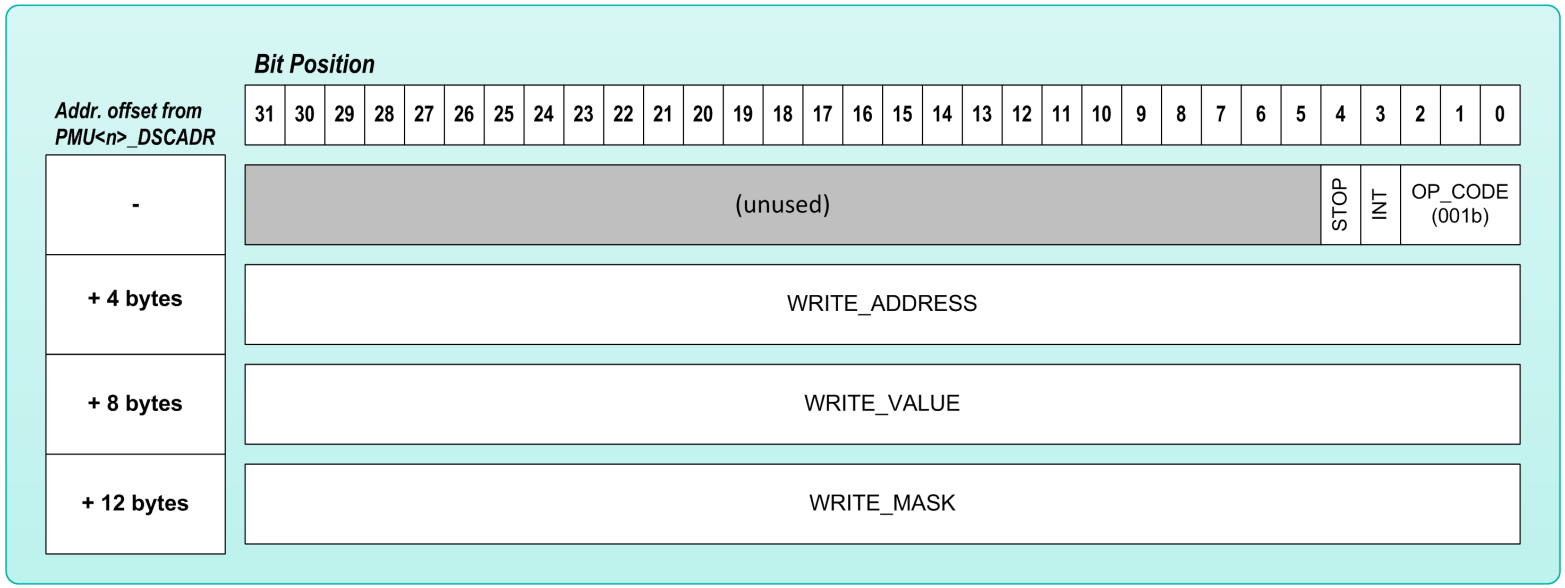

Figure 6.3: PMU WRITE Descriptor

6.3.2.1 OP_CODE

This three-bit field is set to 001b to define the descriptor as a WRITE descriptor.

6.3.2.2 INT

If this bit is set to 1, the interrupt flag for the current channel will be set to 1 after the channel completes execution of this descriptor.

6.3.2.3 STOP

If this bit is set to 1, once the PMU channel completes execution of this descriptor, the PMU channel will halt execution. When this occurs, the ll_stopped flag for the current channel will be set to 1 by hardware to indicate that the PMU descriptor sequence has completed execution.

6.3.2.4 WRITE_ADDRESS

Address of location to be written to.

6.3.2.5 WRITE_VALUE

Value that will be written to the write location (only where the matching bits in the WRITE_MASK are also set to 1).

6.3.2.6 WRITE_MASK

This field determines which bits in WRITE_VALUE will be written to the destination. Bits in WRITE_MASK set to 1 indicate positions where new data will be written; bits set to 0 indicate positions where the data will not be changed.

6.3.3 WAIT Descriptor

The WAIT descriptor will cause the PMU channel to pause execution (remaining at the WAIT descriptor location) until one or more of the interrupt sources selected by the interrupt mask are set. If all bits in INT_MASK are zero, the WAIT descriptor will complete immediately (following any delay specified by WAIT + WAIT_COUNT) and the PMU channel will continue on as normal.

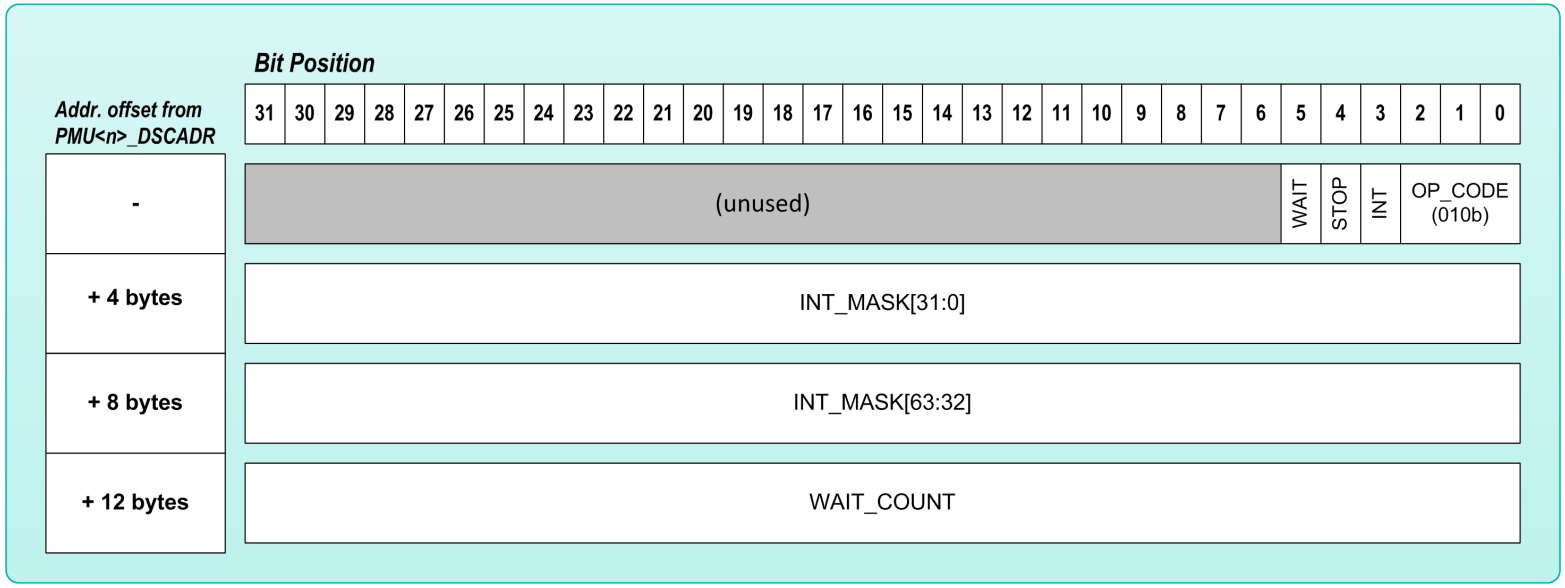

Figure 6.4: PMU WAIT Descriptor

6.3.3.1 OP_CODE

This three-bit field is set to 010b to define the descriptor as a WAIT descriptor.

6.3.3.2 INT

If this bit is set to 1, the interrupt flag for the current channel will be set to 1 after the channel completes execution of this descriptor.

6.3.3.3 STOP

If this bit is set to 1, once the PMU channel completes execution of this descriptor, the PMU channel will halt execution. When this occurs, the ll_stopped flag for the current channel will be set to 1 by hardware to indicate that the PMU descriptor sequence has completed execution.

6.3.3.4 WAIT

If this bit is set to 1, after the condition specified by INT_MASK has occurred, the PMU channel will delay for the number of PMU module clock specified by WAIT_COUNT before continuing on.

6.3.3.5 INT_MASK

The interrupt mask field corresponds to the following interrupt sources available to the PMU.

The following interrupts are generated by the Peripheral FIFOs automatically as listed below based on FIFO configuration registers. These interrupts *do not* require the user to clear or enable the interrupt source; they are self-clearing and enabled by the appropriate FIFO configuration registers as shown in the table below.

PMU FIFO Interrupt Sources for WAIT - Peripheral FIFO Generated

Bit	Source	Interrupt Set Condition	Interrupt Clear Condition
0	UART0 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in UART0_TX_FIFO_CTRL.fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
1	UART0 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in UART0_RX_FIFO_CTRL.fifo_af_lvl	Self-clears when RX FIFO level falls below this threshold
2	UART1 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in UART1_TX_FIFO_CTRL.fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
3	UART1 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in UART1_RX_FIFO_CTRL.fifo_af_lvl	Self-clears when RX FIFO level falls below this threshold
4	UART2 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in UART2_TX_FIFO_CTRL.fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
5	UART2 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in UART2_RX_FIFO_CTRL.fifo_af_lvl	Self-clears when RX FIFO level falls below this threshold
6	UART3 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in UART3_TX_FIFO_CTRL.fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
7	UART3 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in UART3_RX_FIFO_CTRL.fifo_af_lvl	Self-clears when RX FIFO level falls below this threshold
8	SPI0 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in SPI0_FIFO_CTRL.tx_fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
9	SPI0 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in SPI0_FIFO_CTRL.rx_fifo_af_lvl	Self-clears when the RX FIFO level falls below this threshold

Bit	Source	Interrupt Set Condition	Interrupt Clear Condition
10	SPI1 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in SPI1_FIFO_CTRL.tx_fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
11	SPI1 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in SPI1_FIFO_CTRL.rx_fifo_af_lvl	Self-clears when the RX FIFO level falls below this threshold
12	SPI2 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in SPI2_FIFO_CTRL.tx_fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
13	SPI2 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in SPI2_FIFO_CTRL.rx_fifo_af_lvl	Self-clears when the RX FIFO level falls below this threshold
14	I2CM0 TX FIFO Empty	Set when no data is in the TX FIFO	Self-clears when the TX FIFO contains at least one byte of data
15	I2CM0 RX FIFO Not Empty	Set when the RX FIFO contains at least one byte of data	Self-clears when no data is in the RX FIFO
16	I2CM1 TX FIFO Empty	Set when no data is in the TX FIFO	Self-clears when the TX FIFO contains at least one byte of data
17	I2CM1 RX FIFO Not Empty	Set when the RX FIFO contains at least one byte of data	Self-clears when no data is in the RX FIFO
18	I2CM2 TX FIFO Empty	Set when no data is in the TX FIFO	Self-clears when the TX FIFO contains at least one byte of data
19	I2CM2 RX FIFO Not Empty	Set when the RX FIFO contains at least one byte of data	Self-clears when no data is in the RX FIFO

The remaining interrupts, shown below, *must be enabled* through the appropriate peripheral register and also *cleared* after the interrupt becomes set.

PMU Interrupt Event Table for WAIT Descriptor

Interrupt Bit	Source	Enable and Clear
20	SPI0 rx_stalled OR tx_ready OR tx_stalled	SPI0_INTEN.rx_stalled, SPI0_INTEN.tx_stalled or SPI0_INTEN.tx_ready to enable; SPI0_INTFL.rx_stalled, SPI0_INTFL.tx_stalled or SPI0_INTFL.tx_readyto clear (W1C)
21	SPI1 rx_stalled OR tx_ready OR tx_stalled	SPI1_INTEN.rx_stalled, SPI1_INTEN.tx_stalled or SPI1_INTEN.tx_ready to enable; SPI1_INTFL.rx_stalled, SPI1_INTFL.tx_stalled or SPI1_INTFL.tx_readyto clear (W1C)
22	SPI2 rx_stalled OR tx_ready OR tx_stalled	SPI2_INTEN.rx_stalled, SPI2_INTEN.tx_stalled or SPI2_INTEN.tx_ready to enable; SPI2_INTFL.rx_stalled, SPI2_INTFL.tx_stalled or SPI2_INTFL.tx_readyto clear (W1C)
23	SPI Bridge Interrupt	Internal use only
24	I2CM0 TX Finished	I2CM0_INTEN.tx_done to enable, I2CM0_INTFL.tx_done to clear (W1C)

Interrupt Bit	Source	Enable and Clear
25	I2CM1 TX Finished	I2CM1_INTEN.tx_done to enable, I2CM1_INTFL.tx_done to clear (W1C)
26	I2CM2 TX Finished	I2CM2_INTEN.tx_done to enable, I2CM2_INTFL.tx_done to clear (W1C)
27	I2CS Byte(s) Updated	Set one or more bits in I2CS_INTEN to enable, clear corresponding flags in I2CS_INTFL when set (W1C)
28	ADC Done	ADC_INTR.adc_done_ie to enable, ADC_INTR.adc_done_if to clear (W1C)
29	ADC Ready	ADC_INTR.adc_ref_ready_ie to enable, ADC_INTR.adc_ref_ready_if to clear (W1C)
30	ADC Sample Above High Limit	ADC_INTR.adc_hi_limit_ie to enable, ADC_INTR.adc_hi_limit_if to clear (W1C)
31	ADC Sample Below Low Limit	ADC_INTR.adc_lo_limit_ie to enable, ADC_INTR.adc_lo_limit_if to clear (W1C)
32	RTC Timer Match (Comparator 0)	RTCTMR_INTEN.comp0 to enable, RTCTMR_FLAGS.comp0 to clear (W1C)
33	RTC Timer Match (Comparator 1)	RTCTMR_INTEN.comp1 to enable, RTCTMR_FLAGS.comp1 to clear (W1C)
34	RTC Prescaler Interval	RTCTMR_INTEN.prescale_comp to enable, RTCTMR_FLAGS.prescale_comp to clear (W1C)
35	RTC Timer Overflow	RTCTMR_INTEN.overflow to enable, RTCTMR_FLAGS.overflow to clear (W1C)
36	Pulse Train 0 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 0 is disabled.
37	Pulse Train 1 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 1 is disabled.
38	Pulse Train 2 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 2 is disabled.
39	Pulse Train 3 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 3 is disabled.
40	Pulse Train 4 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 4 is disabled.
41	Pulse Train 5 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 5 is disabled.
42	Pulse Train 6 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 6 is disabled.
43	Pulse Train 7 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 7 is disabled.
44	Pulse Train 8 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 8 is disabled.
45	Pulse Train 9 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 9 is disabled.
46	Pulse Train 10 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 10 is disabled.
47	Pulse Train 11 Disabled	Does not require enable. This PMU interrupt source is set to 1 whenever pulse train 11 is disabled.
48	Timer0 32-bit or 16-bit interrupts	TMR0_INTEN.timer0 to enable, TMR0_INTFL.timer0 to clear
49	Timer1 32-bit or 16-bit interrupts	TMR1_INTEN.timer0 to enable, TMR1_INTFL.timer0 to clear

Interrupt Bit	Source	Enable and Clear
50	Timer2 32-bit or 16-bit interrupts	TMR2_INTEN.timer0 to enable, TMR2_INTFL.timer0 to clear
51	Timer3 32-bit or 16-bit interrupts	TMR3_INTEN.timer0 to enable, TMR3_INTFL.timer0 to clear
52	Timer4 32-bit or 16-bit interrupts	TMR4_INTEN.timer0 to enable, TMR4_INTFL.timer0 to clear
53	Timer5 32-bit or 16-bit interrupts	TMR5_INTEN.timer0 to enable, TMR5_INTFL.timer0 to clear
54	Interrupt(s) on Port 0 GPIO	GPIO_INT_MODE_P0.pin[7:0] to enable, GPIO_INTFL_P0.pin[7:0] to clear (W1C)
55	Interrupt(s) on Port 1 GPIO	GPIO_INT_MODE_P1.pin[7:0] to enable, GPIO_INTFL_P1.pin[7:0] to clear (W1C)
56	Interrupt(s) on Port 2 GPIO	GPIO_INT_MODE_P2.pin[7:0] to enable, GPIO_INTFL_P2.pin[7:0] to clear (W1C)
57	Interrupt(s) on Port 3 GPIO	GPIO_INT_MODE_P3.pin[7:0] to enable, GPIO_INTFL_P3.pin[7:0] to clear (W1C)
58	Interrupt(s) on Port 4 GPIO	GPIO_INT_MODE_P4.pin[7:0] to enable, GPIO_INTFL_P4.pin[7:0] to clear (W1C)
59	Interrupt(s) on Port 5 GPIO	GPIO_INT_MODE_P5.pin[7:0] to enable, GPIO_INTFL_P5.pin[7:0] to clear (W1C)
60	Interrupt(s) on Port 6 GPIO	GPIO_INT_MODE_P6.pin[7:0] to enable, GPIO_INTFL_P6.pin[7:0] to clear (W1C)
61	AES done	AES_CTRL.inten to enable, AES_CTRL.intfl to clear (W1C)
62	MAA done (**MAX32621** only)	MAA_CTRL.inten to enable, MAA_CTRL.if_done to clear (W1C)
63	1-Wire Master Interrupt (Reset Done)	OWM_INTEN.ow_reset_done to enable, OWM_INTFL.ow_reset_done to clear (W1C)
63	1-Wire Master Interrupt (TX Data Empty)	OWM_INTEN.tx_data_empty to enable, OWM_INTFL.tx_data_empty to clear (W1C)
63	1-Wire Master Interrupt (RX Data Ready)	OWM_INTEN.rx_data_ready to enable, OWM_INTFL.rx_data_ready to clear (W1C)
63	1-Wire Master Interrupt (Line Short)	OWM_INTEN.line_short to enable, OWM_INTFL.line_short to clear (W1C)
63	1-Wire Master Interrupt (Line Low)	OWM_INTEN.line_low to enable, OWM_INTFL.line_low to clear (W1C)

6.3.3.6 WAIT_COUNT

When (WAIT == 1), this field specifies the number of PMU module clocks for the PMU channel to delay after the INT_MASK condition has been satisfied.

6.3.4 JUMP Descriptor

The JUMP op code will allow the PMU engine to fetch the next op code at the specified location in the operand field rather than sequentially in memory.

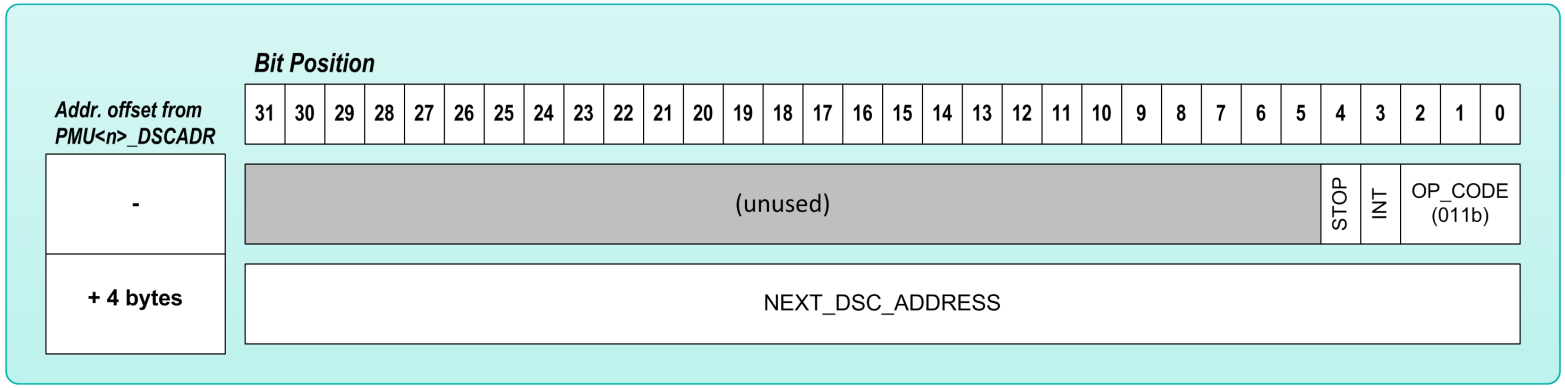

Figure 6.5: PMU JUMP Descriptor

6.3.4.1 OP_CODE

This three-bit field is set to 011b to define the descriptor as a JUMP descriptor.

6.3.4.2 INT

If this bit is set to 1, the interrupt flag for the current channel will be set to 1 after the channel completes execution of this descriptor.

6.3.4.3 STOP

If this bit is set to 1, once the PMU channel completes execution of this descriptor, the PMU channel will halt execution. When this occurs, the ll_stopped flag for the current channel will be set to 1 by hardware to indicate that the PMU descriptor sequence has completed execution.

6.3.4.4 NEXT_DSC_ADDRESS

This field specifies the jump destination, which is the next descriptor address that will be fetched by the PMU channel.

6.3.5 LOOP Descriptor

The LOOP descriptor will cause the PMU engine to fetch the next op code at the specified location rather than sequentially in memory until the specified loop counter (initialized from the value counter0 or counter1, in PMUn_LOOP) decrements to zero. The specified loop counter will decrement by one each time this op code is executed by the PMU engine. When the specified loop counter reaches zero, the next sequential op code will be fetched. The zero check is performed after the specified counter is decremented.

The INT and STOP fields are not checked until the specified counter reaches zero.

The loop counters are sticky counters and do not need to be reloaded when a new LOOP descriptor is fetched unless a new counter value is needed.

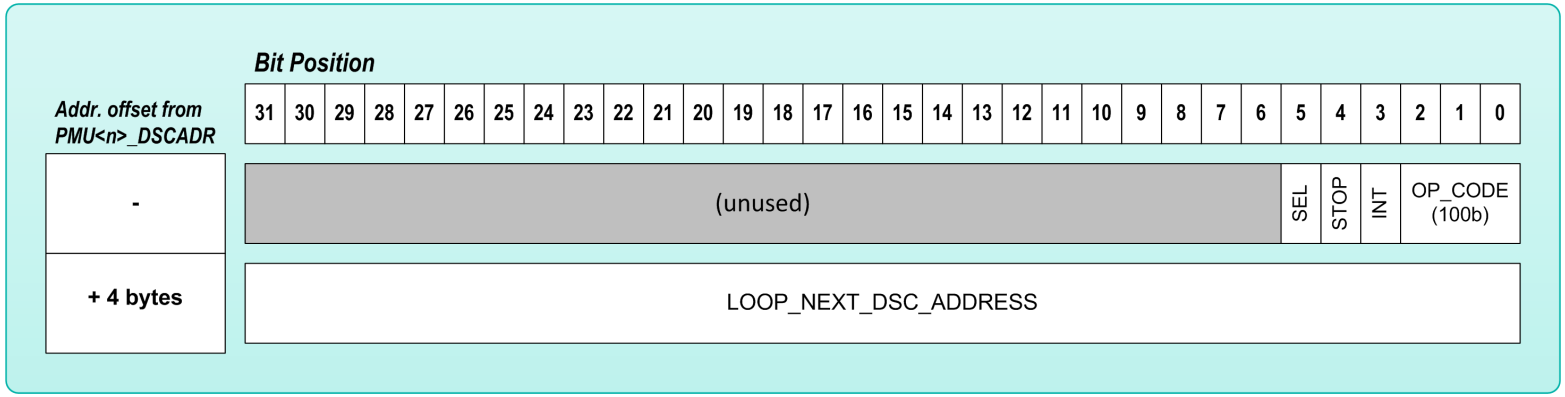

Figure 6.6: PMU LOOP Descriptor

6.3.5.1 OP_CODE

This three-bit field is set to 100b to define the descriptor as a LOOP descriptor.

6.3.5.2 INT

If this bit is set to 1, the interrupt flag for the current channel will be set to 1 after the channel completes execution of this descriptor.

6.3.5.3 STOP

If this bit is set to 1, once the PMU channel completes execution of this descriptor, the PMU channel will halt execution. When this occurs, the ll_stopped flag for the current channel will be set to 1 by hardware to indicate that the PMU descriptor sequence has completed execution.

6.3.5.4 SEL

This bit selects whether loop counter 0 (SEL == 0) or loop counter 1 (SEL == 1) is used when executing this descriptor.

6.3.5.5 NEXT_DSC_ADDRESS

This field specifies the loop destination; if the specified loop counter has not reached zero, the next descriptor will be fetched from this address.

6.3.6 POLL Descriptor

The POLL descriptor will cause the PMU engine to pause, wait for the specified polling interval to occur, and then read the specified address location. Read accesses may be in either the AHB or APB memory space. When the bit(s) set in the data mask match those in the expected data field, the execution of this descriptor will terminate. The polling interval is specified in system clocks.

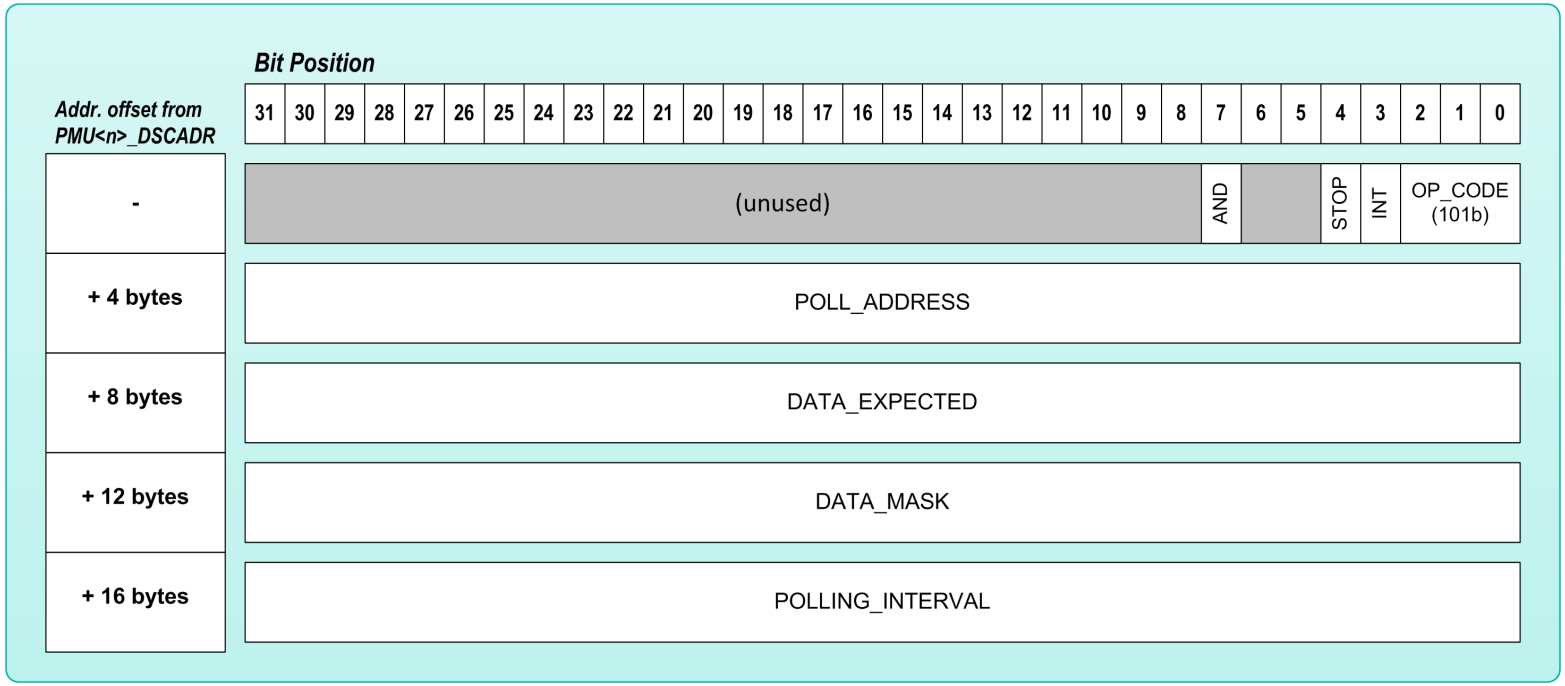

Figure 6.7: PMU POLL Descriptor

6.3.6.1 OP_CODE

This three-bit field is set to 101b to define the descriptor as a POLL descriptor.

6.3.6.2 INT

If this bit is set to 1, the interrupt flag for the current channel will be set to 1 after the channel completes execution of this descriptor.

6.3.6.3 STOP

If this bit is set to 1, once the PMU channel completes execution of this descriptor, the PMU channel will halt execution. When this occurs, the ll_stopped flag for the current channel will be set to 1 by hardware to indicate that the PMU descriptor sequence has completed execution.

6.3.6.4 AND

This bit, when set to 1, will require the data read to match the expected data in all positions that have bits set in the data mask. Otherwise, polling will complete when the data read matches the expected data in any bit position that has a bit set in the data mask.

6.3.6.5 POLL_ADDRESS

Address location which will be read to retrieve data.

6.3.6.6 DATA_EXPECTED

For each bit position in DATA_MASK set to 1, the corresponding bit in this field will be compared against the same bit of data retrieved from POLL_ADDRESS.

6.3.6.7 DATA_MASK

Determines which locations in DATA_EXPECTED will be checked against the data read from POLL_ADDRESS. Only bits which have corresponding bits in DATA_MASK set to 1 will be checked.

6.3.6.8 POLLING_INTERVAL

Number of system clocks for the PMU channel to delay between polling cycles.

6.3.7 BRANCH Descriptor

The BRANCH op code will cause the PMU engine to read the specified address location and fetch the next op code from the specified location when the branch condition is met using the specified bits in the data mask. If the branch condition is either $\leq$, $\geq$, $<$, or $>$ the AND bit is ignored. When the branch condition is not met, the next opcode is fetched sequentially. Read accesses may be in either the AHB or APB memory space.

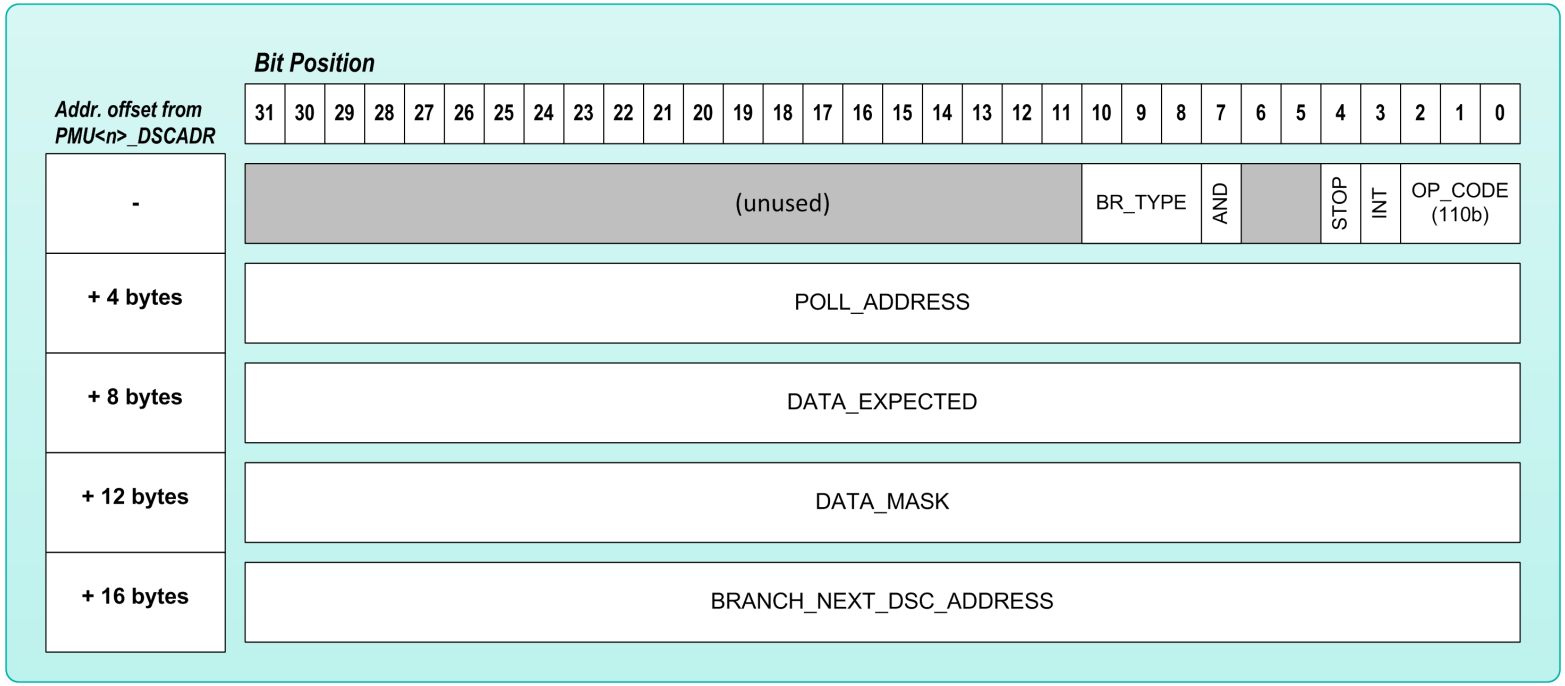

Figure 6.8: PMU BRANCH Descriptor

6.3.7.1 OP_CODE

This three-bit field is set to 110b to define the descriptor as a BRANCH descriptor.

6.3.7.2 INT

If this bit is set to 1, the interrupt flag for the current channel will be set to 1 after the channel completes execution of this descriptor.

6.3.7.3 STOP

If this bit is set to 1, once the PMU channel completes execution of this descriptor, the PMU channel will halt execution. When this occurs, the ll_stopped flag for the current channel will be set to 1 by hardware to indicate that the PMU descriptor sequence has completed execution.

6.3.7.4 AND

Ignored if BRANCH_TYPE is $\leq$, $\geq$, $<$, or $>$.

This bit, when set to 1, will require that all bits set in the data mask match the expected value in the data read from the poll address in order to avoid a branch. Otherwise, a branch will be avoided when any of the bits set in the data mask match the expected value.

Branch Type	AND Bit	Action
Branch Not Equal	0	Using the bits specified in the data mask, if all of the bits in the polled data are not equal to the expected data, the branch will be taken.
	1	Using the bits specified in the data mask, if any of the bits in the polled data are not equal to the expected data, the branch will be taken.
Branch Equal	0	Using the bits specified in the data mask, if any of the bits in the polled data are equal to the expected data, the branch will be taken.
	1	Using the bits specified in the data mask, if all of the bits in the polled data are equal to the expected data, the branch will be taken.

6.3.7.5 BR_TYPE

Branch Type	Description	Branch Taken
000	Branch Not Equal	Polled data is not equal to expected data
001	Branch Equal	Polled data equals expected data
010	Branch Less Than or Equal	Polled data is less than or equal to expected data
011	Branch Greater Than or Equal	Polled data is greater than or equal to expected data
100	Branch Less Than	Polled data is less than expected data
101	Branch Greater Than	Polled data is greater than expected data
11X	Reserved	

6.3.7.6 POLL_ADDRESS

Address location which will be read to retrieve data.

6.3.7.7 DATA_EXPECTED

For each bit position in DATA_MASK set to 1, the corresponding bit in this field will be compared against the same bit of data retrieved from POLL_ADDRESS.

6.3.7.8 DATA_MASK

Determines which locations in DATA_EXPECTED will be checked against the data read from POLL_ADDRESS. Only bits which have corresponding bits in DATA_MASK set to 1 will be checked.

6.3.7.9 BRANCH_NEXT_DSC_ADDRESS

This field specifies the destination branch address, which is the address of the next descriptor that will be fetched by the PMU channel if the branch is taken.

6.3.8 TRANSFER Descriptor

The TRANSFER descriptor is used to move a user specified total block of data (in bytes) from the read address location to the write address location in burst size blocks (in bytes) as specified by the user. Transfers are only performed while the specified bit(s) in the interrupt mask are set.

The TRANSFER op code will continue to execute until the total number of bytes has been transferred, as specified by TRANSFER_LENGTH. This op code combines the functionality of the WAIT, MOVE, and JUMP op codes into a single op code and is particularly useful for servicing the FIFOs on various peripherals. Read and write accesses may be in either the AHB or APB memory space. APB accesses are restricted to 32-bit accesses. The descriptor bit settings and operands are shown below.

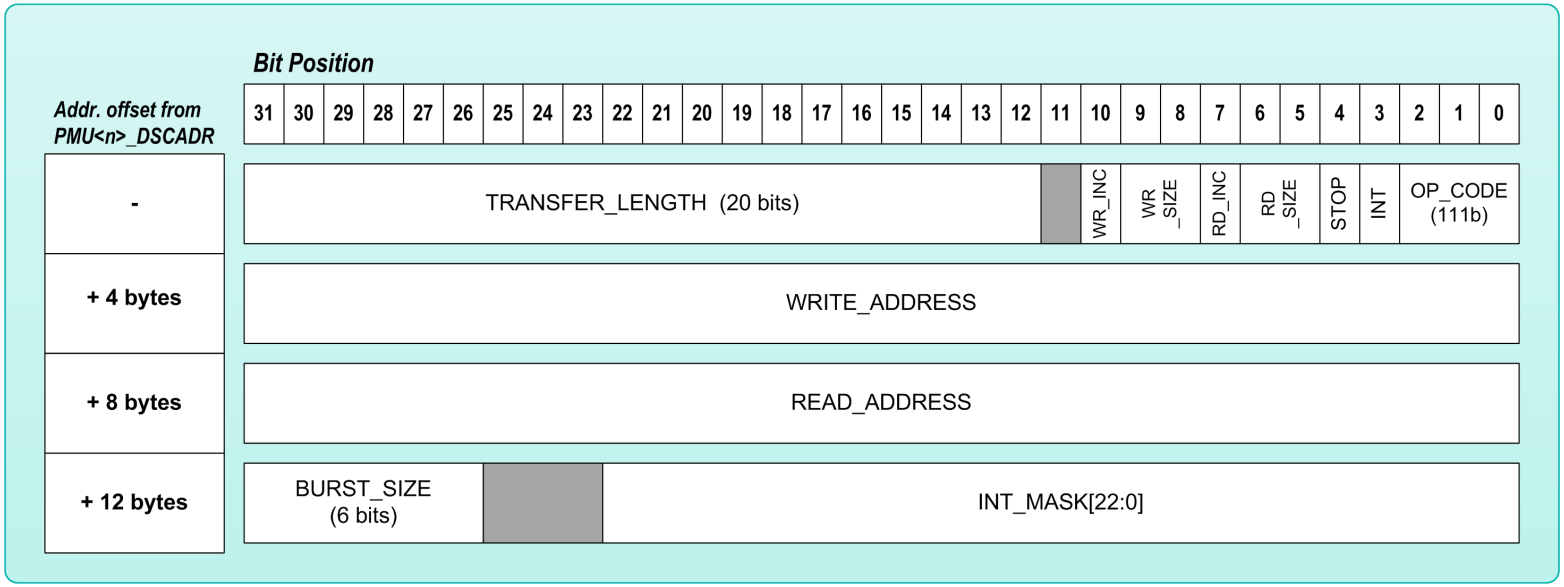

Figure 6.9: PMU TRANSFER Descriptor

6.3.8.1 OP_CODE

This three-bit field is set to 111b to define the descriptor as a TRANSFER descriptor.

6.3.8.2 INT

If this bit is set to 1, the interrupt flag for the current channel will be set to 1 after the channel completes execution of this descriptor.

6.3.8.3 STOP

If this bit is set to 1, once the PMU channel completes execution of this descriptor, the PMU channel will halt execution. When this occurs, the ll_stopped flag for the current channel will be set to 1 by hardware to indicate that the PMU descriptor sequence has completed execution.

6.3.8.4 RD_SIZE, WR_SIZE

These fields determine the width of memory access that will be used on the read side (source) and write side (destination) of the data transfer. Note that any access to APB register space requires the width to be 32-bit only; use of 8-bit or 16-bit width when accessing APB space will result in a bus error. When accessing AHB areas, the widths for the read and write sides can be different; the PMU channel controller will automatically handle data packing/unpacking as needed.

RD SIZE	WR SIZE	Operation
00b	00b	Perform 8-bit reads and 8-bit writes
00b	01b	Perform 8-bit reads and pack data into 16-bit writes
00b	10b	Perform 8-bit reads and pack data into 32-bit writes
01b	00b	Perform 16-bit reads and unpack into 8-bit writes
01b	01b	Perform 16-bit reads and writes
01b	10b	Perform 16-bit reads and pack data into 32-bit writes
10b	00b	Perform 32-bit reads and unpack data into 8-bit writes
10b	01b	Perform 32-bit reads and unpack data into 16-bit writes
10b	10b	Perform 32-bit reads and writes
XX	11b	(Reserved)
11b	XX	(Reserved)

6.3.8.5 RD_INC

If this bit is set to 0, then read address auto-incrementing is disabled. This means that the same read address value will be used for all reads in this data transfer sequence. This is a useful setting when reading from FIFOs, where a single address location must be read repeatedly to unload data.

If this bit is set to 1, then read address auto-incrementing is enabled. This means that every read access in the data transfer will cause the read address pointer to automatically post-increment, based on the RD_SIZE width. For example, if RD_SIZE is set to 01b, then following each 16-bit read access in the data transfer sequence, the read address pointer will be incremented by 2. This is a useful setting when accessing SRAM or memory register files, where a different address is used to access each location in the memory area.

6.3.8.6 WR_INC

If this bit is set to 0, then write address auto-incrementing is disabled. This means that the same write address value will be used for all writes in this data transfer sequence. This is a useful setting when writing to FIFOs or similar locations, where a single address location must be written to repeatedly to load or process data.

If this bit is set to 1, then write address auto-incrementing is enabled. This means that every write access in the data transfer will cause the write address pointer to automatically post-increment, based on the WR_SIZE width. For example, if WR_SIZE is set to 00b, then following each 8-bit write access in the data transfer sequence, the write address pointer will be incremented by 1. This is a useful setting when accessing SRAM or memory register files, where a different address is used to access each location in the memory area.

6.3.8.7 TRANSFER_LENGTH

Length of total transfer (in bytes) from read address to write address.

6.3.8.8 WRITE_ADDRESS

This field specifies the write address used for the data transfer (or if WR_INC == 1, the starting write address).

6.3.8.9 READ_ADDRESS

This field specifies the read address used for the data transfer (or if WR_INC == 1, the starting read address).

6.3.8.10 INT_MASK

Determines which interrupt source(s) will be checked to gate the data transfer. Bit positions set to 1 will cause the corresponding interrupt source to be checked; interrupt sources with bit positions set to 0 will be disregarded for this descriptor.

The interrupt mask field corresponds to the following interrupt sources available to the PMU, which are generated by peripheral FIFO conditions automatically. These interrupts *do not* require the user to clear or enable the interrupt source; they are self-clearing and are based on FIFO configuration register settings as shown in the table below.

PMU FIFO Interrupt Sources for TRANSFER - Peripheral FIFO Generated

Bit	Source	Interrupt Set Condition	Interrupt Clear Condition
0	UART0 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in UART0_TX_FIFO_CTRL.fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold

Bit	Source	Interrupt Set Condition	Interrupt Clear Condition
1	UART0 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in UART0_RX_FIFO_CTRL.fifo_af_lvl	Self-clears when RX FIFO level falls below this threshold
2	UART1 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in UART1_TX_FIFO_CTRL.fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
3	UART1 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in UART1_RX_FIFO_CTRL.fifo_af_lvl	Self-clears when RX FIFO level falls below this threshold
4	UART2 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in UART2_TX_FIFO_CTRL.fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
5	UART2 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in UART2_RX_FIFO_CTRL.fifo_af_lvl	Self-clears when RX FIFO level falls below this threshold
6	UART3 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in UART3_TX_FIFO_CTRL.fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
7	UART3 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in UART3_RX_FIFO_CTRL.fifo_af_lvl	Self-clears when RX FIFO level falls below this threshold
8	SPI0 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in SPI0_FIFO_CTRL.tx_fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
9	SPI0 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in SPI0_FIFO_CTRL.rx_fifo_af_lvl	Self-clears when the RX FIFO level falls below this threshold
10	SPI1 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in SPI1_FIFO_CTRL.tx_fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
11	SPI1 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in SPI1_FIFO_CTRL.rx_fifo_af_lvl	Self-clears when the RX FIFO level falls below this threshold
12	SPI2 TX FIFO Almost Empty	Set when the TX FIFO level falls below the user defined threshold in SPI2_FIFO_CTRL.tx_fifo_ae_lvl	Self-clears when the TX FIFO level is above this threshold
13	SPI2 RX FIFO Almost Full	Set when the RX FIFO level is above the user defined threshold in SPI2_FIFO_CTRL.rx_fifo_af_lvl	Self-clears when the RX FIFO level falls below this threshold
14	I2CM0 TX FIFO Empty	Set when no data is in the TX FIFO	Self-clears when the TX FIFO contains at least one byte of data
15	I2CM0 RX FIFO Not Empty	Set when the RX FIFO contains at least one byte of data	Self-clears when no data is in the RX FIFO
16	I2CM0 RX FIFO Full	Set when there is no space remaining in the RX FIFO	Self-clears when there is at least one byte free in the RX FIFO

Bit	Source	Interrupt Set Condition	Interrupt Clear Condition
17	I2CM1 TX FIFO Empty	Set when no data is in the TX FIFO	Self-clears when the TX FIFO contains at least one byte of data
18	I2CM1 RX FIFO Not Empty	Set when the RX FIFO contains at least one byte of data	Self-clears when no data is in the RX FIFO
19	I2CM1 RX FIFO Full	Set when there is no space remaining in the RX FIFO	Self-clears when there is at least one byte free in the RX FIFO
20	I2CM2 TX FIFO Empty	Set when no data is in the TX FIFO	Self-clears when the TX FIFO contains at least one byte of data
21	I2CM2 RX FIFO Not Empty	Set when the RX FIFO contains at least one byte of data	Self-clears when no data is in the RX FIFO
22	I2CM2 RX FIFO Full	Set when there is no space remaining in the RX FIFO	Self-clears when there is at least one byte free in the RX FIFO

6.3.8.11 BURST_SIZE

Maximum burst length of transfer (in bytes) from read address to write address when specified interrupt source is detected.

6.4 Registers (PMU)

Address	Register	Access	Description	Reset By
0x40005000	PMU0_DSCADR	R/W	PMU Channel 0 Next Descriptor Address	Sys
0x40005004	PMU0_CFG	***	PMU Channel 0 Configuration	Sys
0x40005008	PMU0_LOOP	R/W	PMU Channel 0 Loop Counters	Sys
0x4000500C	PMU0_OP	R/W	PMU Channel 0 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]	Sys
0x40005010	PMU0_DSC1	R/W	PMU Channel 0 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]	Sys
0x40005014	PMU0_DSC2	R/W	PMU Channel 0 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]	Sys
0x40005018	PMU0_DSC3	R/W	PMU Channel 0 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]	Sys
0x4000501C	PMU0_DSC4	R/W	PMU Channel 0 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]	Sys
0x40005020	PMU1_DSCADR	R/W	PMU Channel 1 Next Descriptor Address	Sys
0x40005024	PMU1_CFG	***	PMU Channel 1 Configuration	Sys
0x40005028	PMU1_LOOP	R/W	PMU Channel 1 Loop Counters	Sys
0x4000502C	PMU1_OP	R/W	PMU Channel 1 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]	Sys
0x40005030	PMU1_DSC1	R/W	PMU Channel 1 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]	Sys
0x40005034	PMU1_DSC2	R/W	PMU Channel 1 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]	Sys
0x40005038	PMU1_DSC3	R/W	PMU Channel 1 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]	Sys
0x4000503C	PMU1_DSC4	R/W	PMU Channel 1 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]	Sys
0x40005040	PMU2_DSCADR	R/W	PMU Channel 2 Next Descriptor Address	Sys
0x40005044	PMU2_CFG	***	PMU Channel 2 Configuration	Sys
0x40005048	PMU2_LOOP	R/W	PMU Channel 2 Loop Counters	Sys
0x4000504C	PMU2_OP	R/W	PMU Channel 2 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]	Sys
0x40005050	PMU2_DSC1	R/W	PMU Channel 2 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]	Sys
0x40005054	PMU2_DSC2	R/W	PMU Channel 2 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]	Sys

Address	Register	Access	Description	Reset By
0x40005058	PMU2_DSC3	R/W	PMU Channel 2 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]	Sys
0x4000505C	PMU2_DSC4	R/W	PMU Channel 2 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]	Sys
0x40005060	PMU3_DSCADR	R/W	PMU Channel 3 Next Descriptor Address	Sys
0x40005064	PMU3_CFG	***	PMU Channel 3 Configuration	Sys
0x40005068	PMU3_LOOP	R/W	PMU Channel 3 Loop Counters	Sys
0x4000506C	PMU3_OP	R/W	PMU Channel 3 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]	Sys
0x40005070	PMU3_DSC1	R/W	PMU Channel 3 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]	Sys
0x40005074	PMU3_DSC2	R/W	PMU Channel 3 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]	Sys
0x40005078	PMU3_DSC3	R/W	PMU Channel 3 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]	Sys
0x4000507C	PMU3_DSC4	R/W	PMU Channel 3 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]	Sys
0x40005080	PMU4_DSCADR	R/W	PMU Channel 4 Next Descriptor Address	Sys
0x40005084	PMU4_CFG	***	PMU Channel 4 Configuration	Sys
0x40005088	PMU4_LOOP	R/W	PMU Channel 4 Loop Counters	Sys
0x4000508C	PMU4_OP	R/W	PMU Channel 4 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]	Sys
0x40005090	PMU4_DSC1	R/W	PMU Channel 4 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]	Sys
0x40005094	PMU4_DSC2	R/W	PMU Channel 4 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]	Sys
0x40005098	PMU4_DSC3	R/W	PMU Channel 4 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]	Sys
0x4000509C	PMU4_DSC4	R/W	PMU Channel 4 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]	Sys
0x400050A0	PMU5_DSCADR	R/W	PMU Channel 5 Next Descriptor Address	Sys
0x400050A4	PMU5_CFG	***	PMU Channel 5 Configuration	Sys
0x400050A8	PMU5_LOOP	R/W	PMU Channel 5 Loop Counters	Sys
0x400050AC	PMU5_OP	R/W	PMU Channel 5 Current Descriptor DWORD 0 [INTERNAL TEST ONLY]	Sys
0x400050B0	PMU5_DSC1	R/W	PMU Channel 5 Current Descriptor DWORD 1 [INTERNAL TEST ONLY]	Sys
0x400050B4	PMU5_DSC2	R/W	PMU Channel 5 Current Descriptor DWORD 2 [INTERNAL TEST ONLY]	Sys
0x400050B8	PMU5_DSC3	R/W	PMU Channel 5 Current Descriptor DWORD 3 [INTERNAL TEST ONLY]	Sys

Address	Register	Access	Description	Reset By
0x400050BC	PMU5_DSC4	R/W	PMU Channel 5 Current Descriptor DWORD 4 [INTERNAL TEST ONLY]	Sys

6.4.1 PMUn_DSCADR

PMU0_DSCADR

Sys Reset	Access	Description
00000000h	R/W	PMU Channel Next Descriptor Address

This register contains a byte address which points to the first 32-bit word of the next PMU descriptor which will be fetched by this PMU channel. Before the PMU channel is started, this register must be set to the address of the first PMU descriptor to be fetched.

As each PMU descriptor is fetched and executed, this register is automatically updated to point to the first 32-bit word of the next PMU descriptor to be executed by the PMU channel controller. In order for a PMU descriptor sequence to be rerun (after the PMU channel has stopped), this register must be reloaded with the address of the first PMU descriptor in the sequence.

6.4.2 PMUn_CFG

PMUn_CFG.enable

Field	Bits	Sys Reset	Access	Description
enable	0	0	R/W	PMU Channel Enable

- 0: Disabled/stopped (this channel only)
- 1: Enabled/running (this channel only)

To begin a PMU descriptor sequence, firmware must set this bit to 1. It is possible to halt the PMU channel before the end of the descriptor sequence has been reached by manually clearing this bit to 0.

This bit will be automatically cleared to 0 by hardware when the PMU channel stops running. This will occur whenever the PMU channel controller executes a descriptor with the STOP bit set (which will also cause ll_stopped to be set to 1), or whenever either the Bus Error Flag or the Bus Timeout Flag is set to 1.

PMUn_CFG.ll_stopped

Field	Bits	Sys Reset	Access	Description
ll_stopped	2	0	W1C	PMU Channel Stopped Flag

This bit is set to 1 by hardware when the PMU channel completes execution of a descriptor which has a STOP bit set to 1. This bit is not set if the PMU channel halts due to a bus error or timeout, or if the PMU channel is halted manually by clearing the enable bit.

In order to clear this bit, firmware must write this bit to 1. This bit will not clear automatically when the PMU channel is started. Firmware should always clear this bit before starting PMU channel execution, so that the (STOP == 1) channel halt condition can be properly detected.

PMUn_CFG.manual

Field	Bits	Sys Reset	Access	Description
manual	3	0	R/W	Manual Mode Enable [INTERNAL USE ONLY]

This bit is intended for Maxim internal test use only.

This bit should not be used by application firmware. This bit must remain set to the default value (0) for proper PMU operation.

PMUn_CFG.bus_error

Field	Bits	Sys Reset	Access	Description
bus_error	4	0	W1C	AHB Bus Error Flag

This bit is set to 1 by hardware when an AHB bus error occurs during channel descriptor execution. When this occurs, the PMU channel will also be halted, and the enable bit will be cleared to zero by hardware automatically. Other active PMU channels will continue execution normally, however.

In order to clear this bit, firmware must write this bit to 1. This bit will not clear automatically when the PMU channel is started. Firmware should always clear this bit before starting PMU channel execution, so that the AHB bus error channel halt condition can be properly detected.

PMUn_CFG.to_stat

Field	Bits	Sys Reset	Access	Description
to_stat	6	0	W1C	AHB Bus Timeout Flag

This bit is set to 1 by hardware when a bus timeout occurs during channel descriptor execution. When this occurs, the PMU channel will also be halted, and the enable bit will be cleared to zero by hardware automatically. Other active PMU channels will continue execution normally, however.

Conditions for the bus timeout are determined by the time out interval select and prescale select fields in this register.

In order to clear this bit, firmware must write this bit to 1. This bit will not clear automatically when the PMU channel is started. Firmware should always clear this bit before starting PMU channel execution, so that the bus timeout channel halt condition can be properly detected.

PMUn_CFG.to_sel

Field	Bits	Sys Reset	Access	Description
to_sel	13:11	000b	R/W	Time Out Interval Select

This field selects the bus timeout period that is used for this PMU channel. If a single AHB bus operation exceeds this period, a bus timeout condition will occur, causing the PMU channel to halt and causing the bus timeout flag to be set to 1.

Note that the setting of this field has no effect if the ps_sel field is set to 0 (which disables the AHB bus timeout counter).

The timeout interval selected by this field is derived from the base period selected by the ps_sel field (when ps_sel != 0).

- 0: 4 x (ps_sel period)
- 1: 8 x (ps_sel period)
- 2: 16 x (ps_sel period)
- 3: 32 x (ps_sel period)
- 4: 64 x (ps_sel period)
- 5: 128 x (ps_sel period)
- 6: 256 x (ps_sel period)
- 7: 512 x (ps_sel period)

PMUn_CFG.ps_sel

Field	Bits	Sys Reset	Access	Description
ps_sel	15:14	00b	R/W	Time Out Interval Prescale Select

This field selects the base time period used for the AHB bus timeout interval, based on the PMU module clock. The value of this field is combined with the to_sel field to determine the total AHB bus timeout interval.

- 0: Disabled (default)
- 1: (PMU module clock) divided by 256
- 2: (PMU module clock) divided by 65,536 (2^16)
- 3: (PMU module clock) divided by 16,777,216 (2^24)

PMUn_CFG.interrupt

Field	Bits	Sys Reset	Access	Description
interrupt	16	0	W1C	Descriptor Interrupt Flag

This interrupt flag is set to 1 by hardware when the PMU channel completes execution of a descriptor which has (INT == 1).

When this interrupt flag is set to 1, an interrupt will be triggered by the PMU module if int_en is also set to 1.

This interrupt flag must be cleared by firmware when the PMU interrupt is serviced. To clear this flag, write the flag to 1.

PMUn_CFG.int_en

Field	Bits	Sys Reset	Access	Description
int_en	17	0	R/W	PMU Channel Interrupt Enable

CPU interrupt enable/disable for this PMU channel.

- 0: No interrupts will be generated for this channel.
- 1: The PMU will generate an interrupt when the Descriptor Interrupt flag is set. AHB bus error or bus timeout conditions will cause a halt to PMU operation.

PMUn_CFG.burst_size

Field	Bits	Sys Reset	Access	Description
burst_size	28:24	16	R/W	AHB Maximum Burst Size

This field controls the maximum size of the PMU transfer bursts in bytes.

6.4.3 PMUn_LOOP

PMUn_LOOP.counter_0

Field	Bits	Sys Reset	Access	Description
counter_0	15:0	0000h	R/W	PMU Channel Loop Counter 0

This field contains the initial value that will be loaded into the PMU channel internal loop counter 0 when the PMU executes a LOOP descriptor and the internal loop counter 0 is at zero.

Note that the internal loop counter (which is the counter that is decremented each time the LOOP descriptor is executed) is not accessed through this field. This field contains a 'sticky' loop counter and its contents will not change as the internal loop counter 0 is decremented.

This means that it is possible to have more than one LOOP descriptor in a PMU descriptor sequence that uses loop counter 0.

PMUn_LOOP.counter_1

Field	Bits	Sys Reset	Access	Description
counter_1	31:16	0000h	R/W	PMU Channel Loop Counter 1

This field contains the initial value that will be loaded into the PMU channel internal loop counter 1 when the PMU executes a LOOP descriptor and the internal loop counter 1 is at zero.

Note that the internal loop counter (which is the counter that is decremented each time the LOOP descriptor is executed) is not accessed through this field. This field contains a 'sticky' loop counter and its contents will not change as the internal loop counter 1 is decremented.

This means that it is possible to have more than one LOOP descriptor in a PMU descriptor sequence that uses loop counter 1.

6.4.4 PMUn_OP

PMU0_OP

Sys Reset	Access	Description
00000000h	R/W	PMU Channel Current Descriptor DWORD 0 [INTERNAL TEST ONLY]

This register is intended for Maxim internal test use only.

Application software should not use this register. This register must remain set to its default value for proper PMU operation.

6.4.5 PMUn_DSC1

PMU0_DSC1

Sys Reset	Access	Description
00000000h	R/W	PMU Channel Current Descriptor DWORD 1 [INTERNAL TEST ONLY]

This register is intended for Maxim internal test use only.

Application software should not use this register. This register must remain set to its default value for proper PMU operation.

6.4.6 PMUn_DSC2

PMU0_DSC2

Sys Reset	Access	Description
00000000h	R/W	PMU Channel Current Descriptor DWORD 2 [INTERNAL TEST ONLY]

This register is intended for Maxim internal test use only.

Application software should not use this register. This register must remain set to its default value for proper PMU operation.

6.4.7 PMUn_DSC3

PMU0_DSC3

Sys Reset	Access	Description
00000000h	R/W	PMU Channel Current Descriptor DWORD 3 [INTERNAL TEST ONLY]

This register is intended for Maxim internal test use only.

Application software should not use this register. This register must remain set to its default value for proper PMU operation.

6.4.8 PMUn_DSC4

PMU0_DSC4

Sys Reset	Access	Description
00000000h	R/W	PMU Channel Current Descriptor DWORD 4 [INTERNAL TEST ONLY]

This register is intended for Maxim internal test use only.

Application software should not use this register. This register must remain set to its default value for proper PMU operation.

7 Communication Peripherals

7.1 1-Wire Master

7.1.1 1-Wire Master Overview

The **MAX32620** provides a 1-Wire® master (OWM) which can be used to communicate with one or more external 1-Wire slave devices using a single-signal combined clock and data protocol. This 1-Wire master is contained in the OWM module. The OWM module handles the lower-level details (including timing and drive modes) required by the 1-Wire protocol, allowing the CPU to communicate over the 1-Wire bus at a logical data level. Features provided by the OWM module include:

- Flexible 1-Wire timing generation (required 1MHz timing base) using the OWM module clock frequency, which is in turn derived from the current system clock source. Optional prescaling of the OWM module clock can be used to allow proper 1-Wire timing generation using a range of base frequencies.
- Automatic generation of proper 1-Wire time slots for both standard and overdrive timing modes.
- Flexible configuration for 1-Wire line pullup modes: options for internal pullup, external fixed pullup, and optional external strong pullup are available.
- Long line compensation and bit banging (direct firmware drive) modes.
- 1-Wire reset generation and presence pulse detection.
- Generation of 1-Wire read and write time slots for single bit and 8-bit byte transmissions.
- Search ROM Accelerator (SRA) mode simplifies generation of multiple bit time slots and discrepancy resolution required when performing the Search ROM function to determine the IDs of multiple unknown 1-Wire slaves on the bus.
- Interrupts available for transmit data completion, received data available, presence pulse detection, and 1-Wire line error conditions.

References For more information on the Maxim 1-Wire protocol and supporting devices, refer to the following sources:

- AN937: *The Book of iButton® Standards* (http://www.maximintegrated.com/en/appnotes/index.mvp/id/937)
- AN1796: *Overview of 1-Wire Technology and Its Use* (http://www.maximintegrated.com/en/app-notes/index.mvp/id/1796)
- AN187: *1-Wire Search Algorithm* (http://www.maximintegrated.com/en/appnotes/index.mvp/id/187)

7.1.2 OWM Pin Configuration

OWM Interface: IO

The IO signal is a bidirectional I/O that is used to directly drive the external 1-Wire bus. As described in the 1-Wire interface specification, this I/O is generally driven as an open drain output. The 1-Wire bus requires a common pullup to return the 1-Wire bus line to an idle high state when no master or slave device is actively driving the line low. This pullup may consist of a fixed resistor pullup (connected to the 1-Wire bus outside the **MAX32620**), an internal pullup enabled by setting OWM_CFG.int_pullup_enable to 1, or an external pullup enabled by setting both OWM_CFG.ext_pullup_mode and OWM_CFG.ext_pullup_enable to 1.

OWM Interface: PUE

The PUE signal is an active high output that is used to enable an optional external pullup on the 1-Wire bus. This pullup is intended to provide a stronger (lower impedance) pullup on the 1-Wire bus under certain circumstances, such as during Overdrive mode.

OWM Pin Mapping

Logic Signal	Port and Pin
IO	P4.0
PUE	P4.1

7.1.3 OWM Clock Selection and Clock Gating

The OWM in the **MAX32620** operates from a module clock derived from the main system clock source. This clock source is configured using the register CLKMAN_SYS_CLK_CTRL_15_OWM. By default, the OWM module clock is disabled; the owm_clk_scale field in this register is used both to enable the module clock and to select the divide down rate (if any) used when deriving the module clock from the system clock source.

OWM Module Clock Control

owm_clk_scale	OWM Module Clock Frequency
0	OWM Module Clock is disabled
1	System Clock Source / 1
2	System Clock Source / 2
3	System Clock Source / 4
4	System Clock Source / 8
5	System Clock Source / 16
6	System Clock Source / 32
7	System Clock Source / 64
8	System Clock Source / 128
9	System Clock Source / 256
other	Reserved

In order to optimize power consumption, the OWM module uses a dynamic clock gating scheme which automatically turns off the local module clock whenever the OWM is inactive. Normally, this dynamic mode (which is the default) should remain set. If there is any reason to modify the module clock gating mode, this can be done by setting CLKMAN_CLK_GATE_CTRL2.owm_clk_gater as shown in the table below.

OWM Module Clock Gating Control

owm_clk_gater	Setting
0	**Clock forced off** - OWM clock is disabled
1	**Dynamic Clock Gating Enabled** - OWM clock active only when the OWM module is in use
2, 3	**Clock forced on** - OWM clock is enabled at all times

7.1.3.1 1-Wire Time Slot Period Generation

In order to correctly generate time slots as required by the 1-Wire protocol in Standard or Overdrive timing modes, the OWM module must be configured to generate a standardized 1MHz clock based on the currently configured module clock frequency. This clock is used when generating both Standard and Overdrive timing, so this setting does not need to be adjusted when transitioning from Standard to Overdrive mode or vice versa.

To configure the generation of the standardized 1MHz clock, the OWM_CLK_DIV_1US.divisor field should be set to the currently configured value of the OWM module clock in MHz. For example, if the system clock is running at 96MHz, and CLKMAN_SYS_CLK_CTRL_15_OWM.owm_clk_scale is set to 6, which would result in an OWM module clock of (96MHz / 32) or 3MHz, then the OWM_CLK_DIV_1US.divisor field should be set to 3, which will cause the OWM module clock to be divided by 3 in order to generate the standardized 1MHz 1-Wire timing slot clock.

7.1.4 1-Wire Protocol

The general timing and communications protocols used by the 1-Wire master interface are those standardized for the 1-Wire network, as defined by the references at the beginning of this section.

Since the 1-Wire interface is a master interface, it performs the task of initiating and timing all communications on the 1-Wire bus. Except for presence pulse generation when a device first connects to the 1-Wire bus, 1-Wire slave devices perform 1-Wire bus communication only as directed by the 1-Wire bus master. From a firmware application's perspective, the lowest-level timing and electrical details of how the 1-Wire network operates are unimportant, and the application can simply concern itself with configuring the OWM module properly and directing it to perform low-level operations such as reset and read and write bit/byte operations. Thus, the OWM module on the **MAX32620** is designed to interface to the 1-Wire bus at a fairly low level; however, understanding of how all layers of the 1-Wire network operate is helpful for application development.

7.1.4.1 Networking Layers

As discussed in the *Book of iButton Standards*, the 1-Wire communications protocol can be described in terms of the standard ISO Open System Interconnection (OSI) layered network model. Network layers which apply to this description are the Physical, Link, Network and Transport layers. The Presentation layer would correspond to higher-level application software functions (such as library layers) which implement communications protocols using the 1-Wire layers as a basic foundation.

7.1.4.2 Bus Interface (Physical Layer)

As described in the *Book of iButton Standards*, the 1-Wire communications bus consists of a single data/power line plus ground. Devices (either master or slave) which interface to the 1-Wire communications bus do so using an open drain (active low) connection, which means that the 1-Wire bus normally idles in a high state. An external pullup resistor is used to pull the 1-Wire line high when no master or slave device is driving the line. This means that 1-Wire devices do not actively drive the 1-Wire line high; instead, they either drive the line low or release it (set outputs to high impedance) to allow the external resistor to pull the line high. This allows the 1-Wire bus to operate in a wired-AND manner and avoids bus contention in the event that more than one device attempts to drive the 1-Wire bus at the same time.

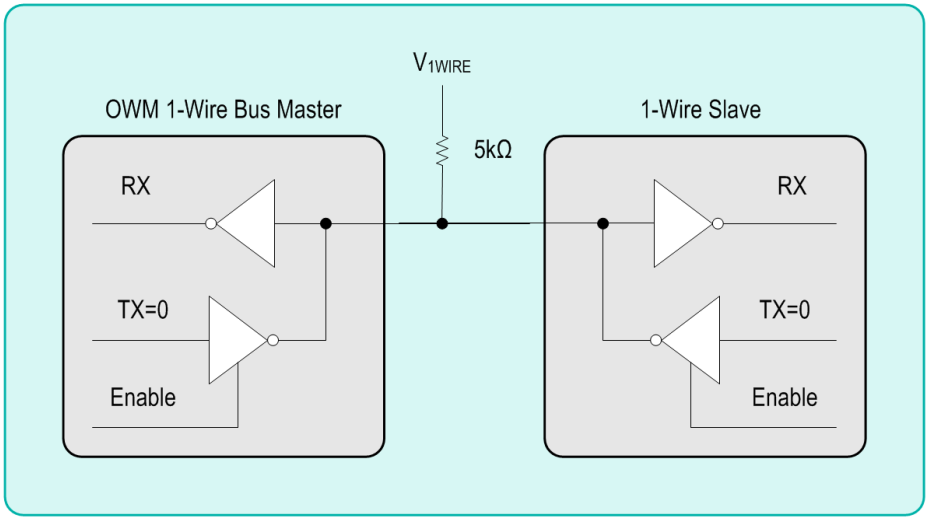

Figure 7.1: 1-Wire Bus Driver Interface

7.1.4.3 Reset, Presence Detect and Data Transfer (Link Layer)

The 1-Wire Bus supports a single master and one or more slave devices (multidrop). Slave devices may connect to and disconnect from the 1-Wire Bus dynamically (as is typically the case with iButtons that operate using an intermittent touch contact interface), which means that it is the master's responsibility to poll the bus as needed to determine the number and types of 1-Wire devices that are connected to the bus.

All communications sequences on the 1-Wire bus are initiated by the master device. The master determines when a 1-Wire time slot begins, as well as the overall communications speed that will be used. There are three different communications speeds supported by the 1-Wire specification - standard speed, overdrive speed, and hyperdrive speed. However, only standard speed and overdrive speed are supported by the OWM 1-Wire master.

Reset and Presence Detect

When a 1-Wire device is first connected to the bus, it generates a presence pulse to let the bus master know that it is present on the line and waiting for communication from the master.

The 1-Wire master begins each communications sequence by sending a reset pulse. This pulse resets all 1-Wire slave devices on the line to their initial states and causes them all to begin monitoring the line for a ROM-layer command from the 1-Wire master. Each 1-Wire device on the line responds to the reset pulse by sending out a presence pulse; these pulses from multiple 1-Wire slave devices are combined in wired-AND fashion, resulting in a pulse whose length is determined by the slowest 1-Wire slave device on the bus.

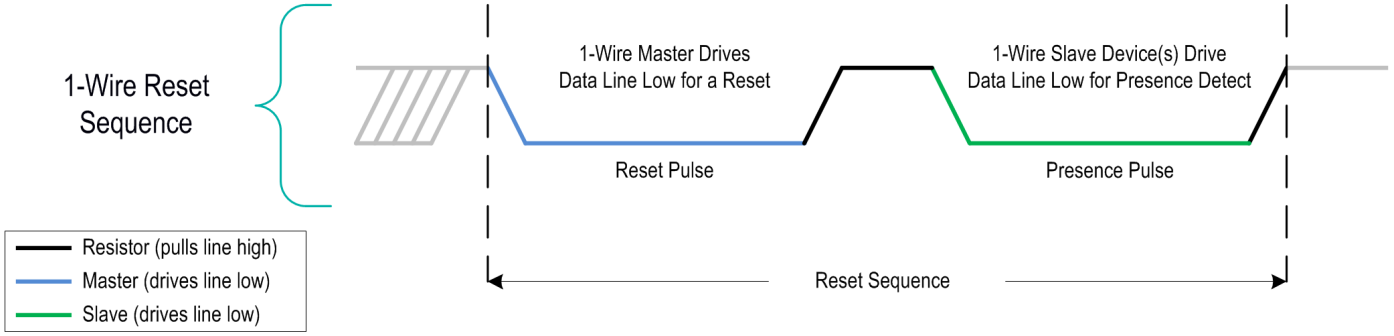

Figure 7.2: 1-Wire Reset Sequence

In general, the 1-Wire line must idle in a high state when communications is not taking place. It is possible for the master to pause communications in between time slots; there is no overall 'time-out' period that will cause a slave to revert to the reset state if the master takes too long between one time slot and the next.

The 1-Wire communications protocol relies on the fact that the maximum allowable length for a bit transfer (write 0/1 or read bit) time slot is less than the minimum length for a 1-Wire reset. At any time, if the 1-Wire line is held low (by the master or by any slave device) for more than the minimum reset pulse time, all slave devices on the line will interpret this as a 1-Wire reset pulse.

7.1.4.4 Reading and Writing Bits

Writing Bits (From Master to Slave)

All 1-Wire bit time slots are initiated by the 1-Wire bus master and begin with a single falling edge. There is no indication given by the beginning of a time slot whether a read bit or write bit operation is intended, as the time slots all begin in the same manner. Rather, the 1-Wire command protocol enforces agreement between the 1-Wire master and slave as to which time slots are used for bit writes and which time slots are used for bit reads.

When multiple bits of a value are transmitted (or read) in sequence, the least significant bit of the value is always sent or received first. The 1-Wire bus is a half-duplex bus, so data may be transmitted in only one direction (from master to slave or from slave to master) at any given time.

As can be seen in the figure below, the time slots for writing a zero bit and writing a one bit begin identically, with the falling edge and a minimum-width low pulse sent by the master. To write a one bit, the master releases the line after the minimum low pulse, allowing it to be pulled high. To write a zero bit, the master continues to hold the line low until the end of the time slot.

From the slave's point of view, the initial falling edge of the time slot triggers the start of an internal timer, and when the proper amount of time has passed, the slave samples the 1-Wire line which is being driven by the master. This sampling point is in between the end of the minimum-width low pulse and the end of the time slot.

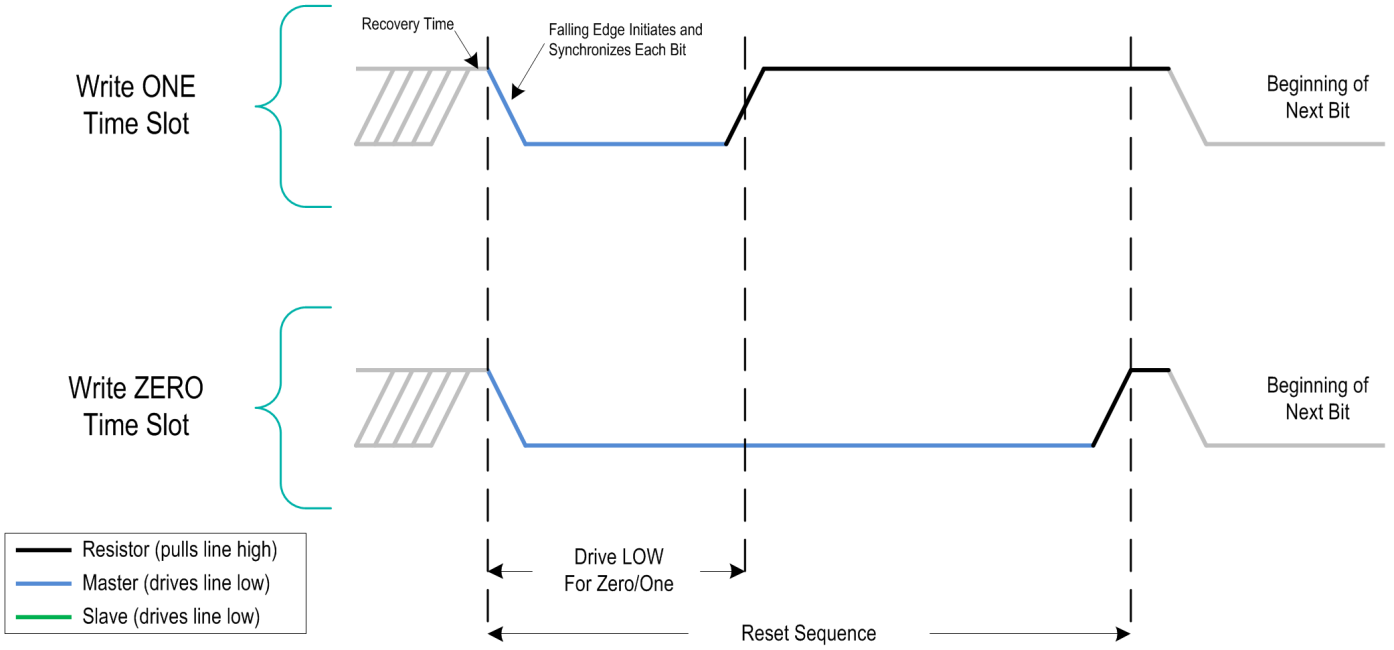

Figure 7.3: 1-Wire Write Bit Sequence

Reading Bits (From Slave to Master)

As with all 1-Wire transactions, the master initiates all bit read time slots. Like the bit write time slots, the bit read time slot begins with a falling edge. From the master's point of view, this time slot is transmitted identically to the "Write 1 Bit" time slot shown above. The master begins by transmitting a falling edge, holds the line low for a minimum-width period, and then releases the line.

The difference here is that instead of the slave sampling the line, the slave begins transmitting either a 0 (by holding the line low) or a 1 (by leaving the line to float high) after the initial falling edge. The master then samples the line to read the bit value that is being transmitted by the slave device.

As an example, the figure below shows a Read ROM sequence in which the slave device transmits its 64-bit ROM ID back to the 1-Wire bus master upon request. Note that in order to transmit a 1 bit, the slave device does not need to do anything; it simply leaves the line alone (to float high) and waits for the next time slot. To

transmit a 0 bit, the slave device holds the line low until the end of the time slot.

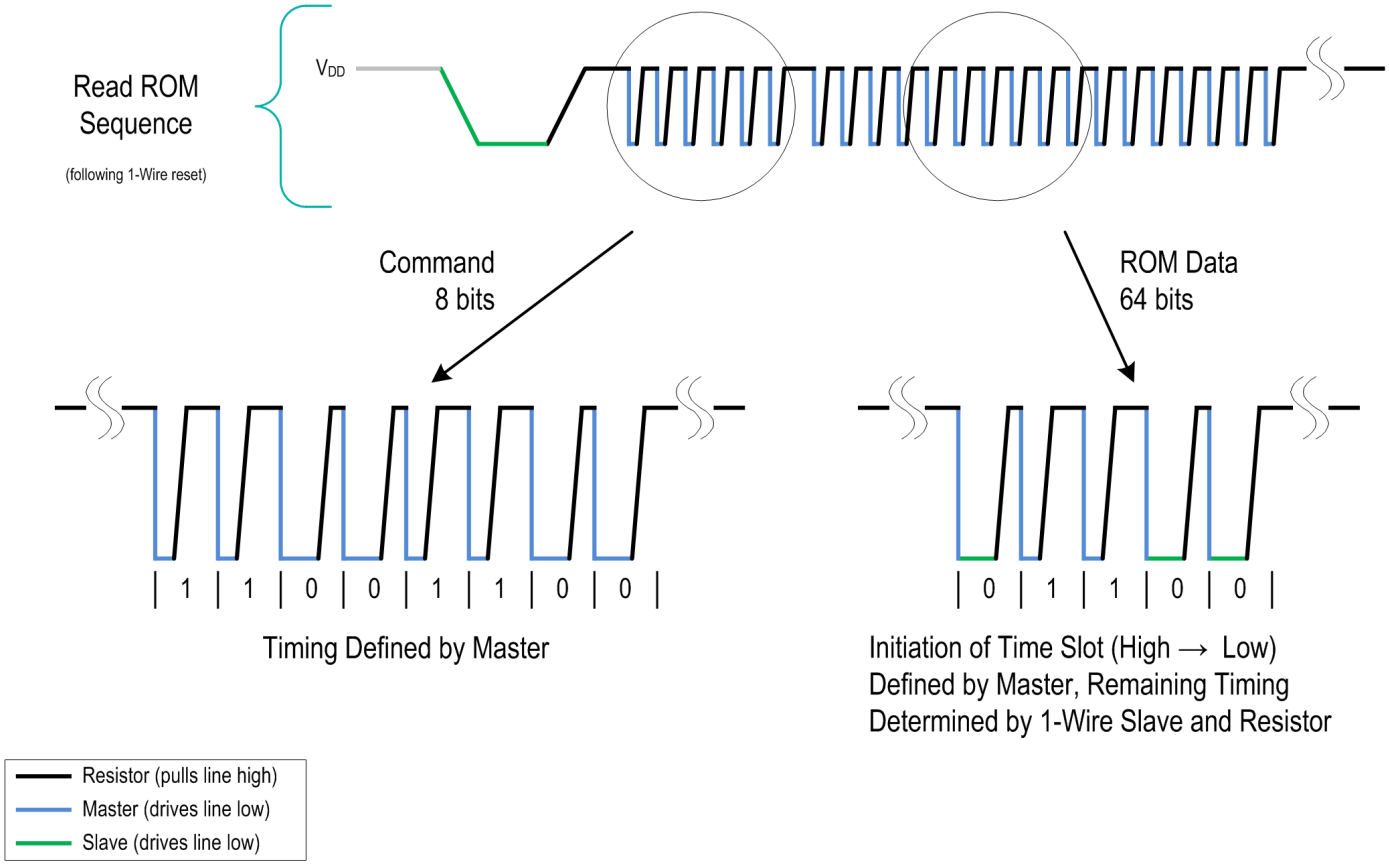

Figure 7.4: 1-Wire Read Sequence

7.1.4.5 Standard Speed and Overdrive Speed

By default, all 1-Wire communications following reset begin at the lowest rate of speed (i.e., standard speed). For 1-Wire devices that support it, it is possible for the 1-Wire master to increase the rate of communications from standard speed to overdrive speed by sending the appropriate ROM layer (network layer) command.

The protocols and time slots operate identically for standard and overdrive speeds; the difference comes in the widths of the time slots and pulses, as shown below.

If a 1-Wire slave device receives a standard-speed reset pulse, it will reset and revert to standard speed communications. If the device is already communicating in overdrive mode, and it receives a pulse at the overdrive speed, it will reset but will remain in overdrive mode.

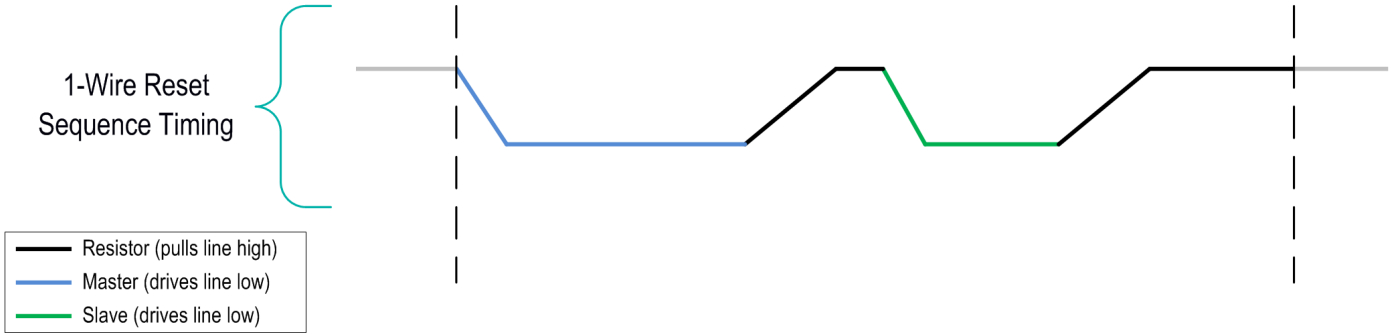

Figure 7.5: 1-Wire Reset Sequence Timing

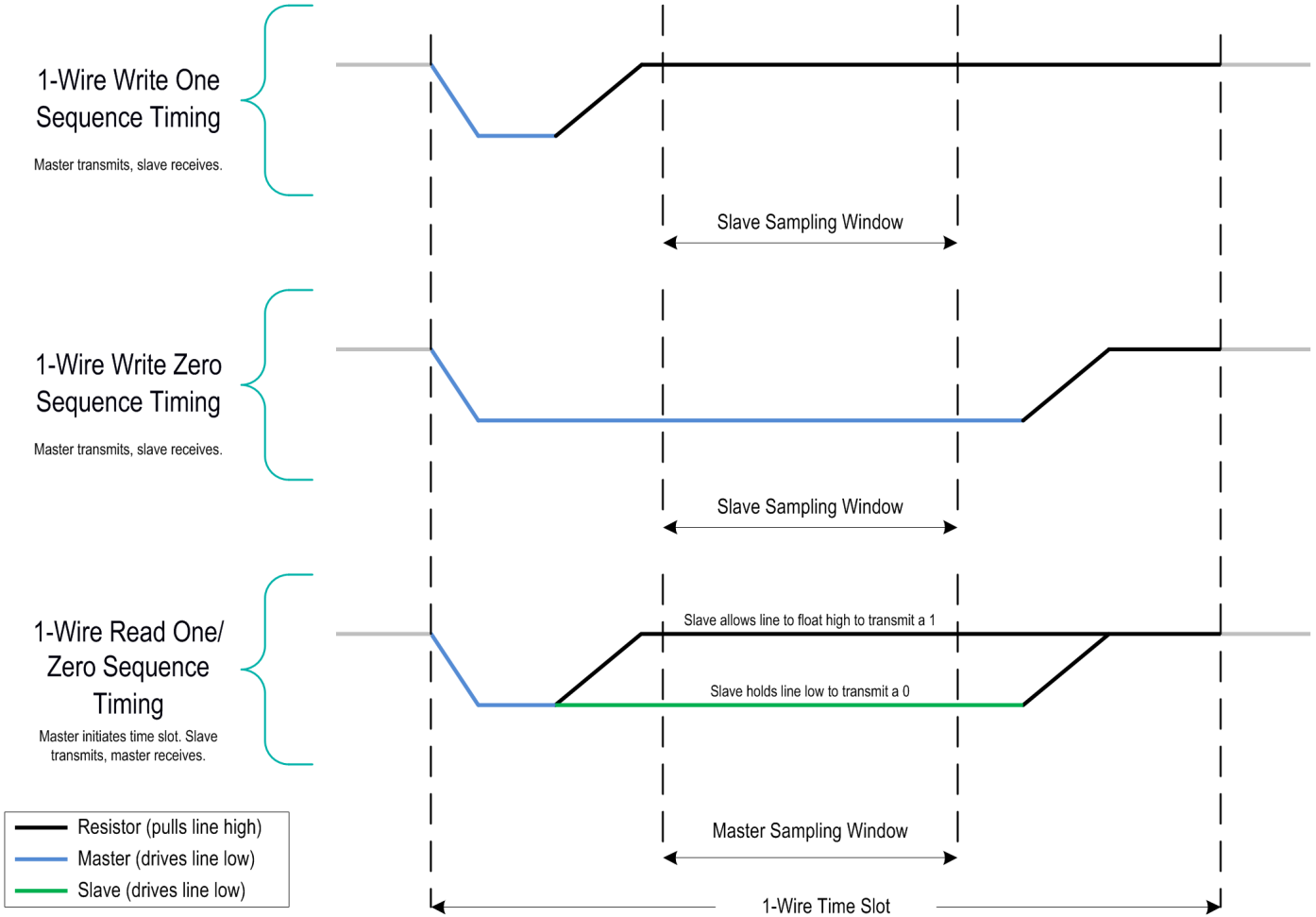

Figure 7.6: Sequence Timings

7.1.4.6 ROM Commands (Network Layer)

Following the initial 1-Wire reset pulse on the bus, all slave 1-Wire devices are active, which means that they are monitoring the bus for commands. Since the 1-Wire bus may have multiple slave devices present on the bus at any time, the 1-Wire master must go through a process (defined by the 1-Wire command protocol) to activate only the 1-Wire slave device that it intends to communicate with, and deactivate all others. This is the purpose of the commands described in the network layer of the 1-Wire protocol, also known as the ROM layer commands.

The ROM command layer relies on the fact that all 1-Wire slave devices are assigned a globally unique 64-bit ROM ID. This ROM ID value is factory programmed to ensure that no two 1-Wire slave devices have the same value.

Typical 1-Wire Slave Device ROM ID Fields

Field	Bits	Description
Family Code	0 .. 7	This 8-bit value is used to identify the type of a 1-Wire slave device.
Unique ID	8 .. 55	This 48-bit value is factory-programmed to give each 1-Wire slave device (within a given Family Code group) a globally unique identifier.
CRC	56 .. 63	This is the 8-bit 1-Wire CRC (Note 1) of bits [55:0].

Note (1) As defined in the *Book of iButton Standards*. The CRC is generated using the polynomial ($X^8 + X^5 + X^4 + 1$).

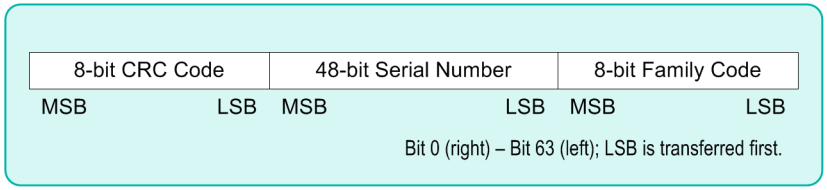

Figure 7.7: Typical 1-Wire Slave Device ROM ID Fields

Note that for certain operations which consist only of writing from the 1-Wire master to the slave, it is technically possible for the master to communicate with more than one slave at a time on the same 1-Wire bus. For this to work, the exact same data must be transmitted to all slave devices, and any values read back from the slaves must either be identical as well or must be disregarded by the master device (since different slaves may attempt to transmit different values). The descriptions below will assume, however, that the master is communicating with only one slave device at a time, since this is the method that is normally used.

As explained above, the ROM ID contents play an important role in addressing and selecting devices on the 1-Wire bus. All devices but one will be in an idle/inactive state after the Match ROM command or the Search ROM command has been executed. They will return to the active state only after receiving a 1-Wire reset pulse.

Devices with overdrive capability can be distinguished from others by their family code and the two additional ROM commands: Overdrive Skip ROM and Overdrive Match ROM. The first transmission of the ROM command itself takes place at the "normal" speed that is understood by all 1-Wire devices. After a device with overdrive capability has been addressed and set into overdrive mode (i.e., after the appropriate ROM command has been received), further communication to that device has to occur at overdrive speed. Since all deselected devices remain in the idle state as long as no reset pulse of regular duration is detected, even multiple overdrive components can reside on the same 1-Wire bus. A reset pulse of regular duration will reset all 1-Wire devices on the bus and simultaneously set all overdrive-capable devices back to the default, standard speed.

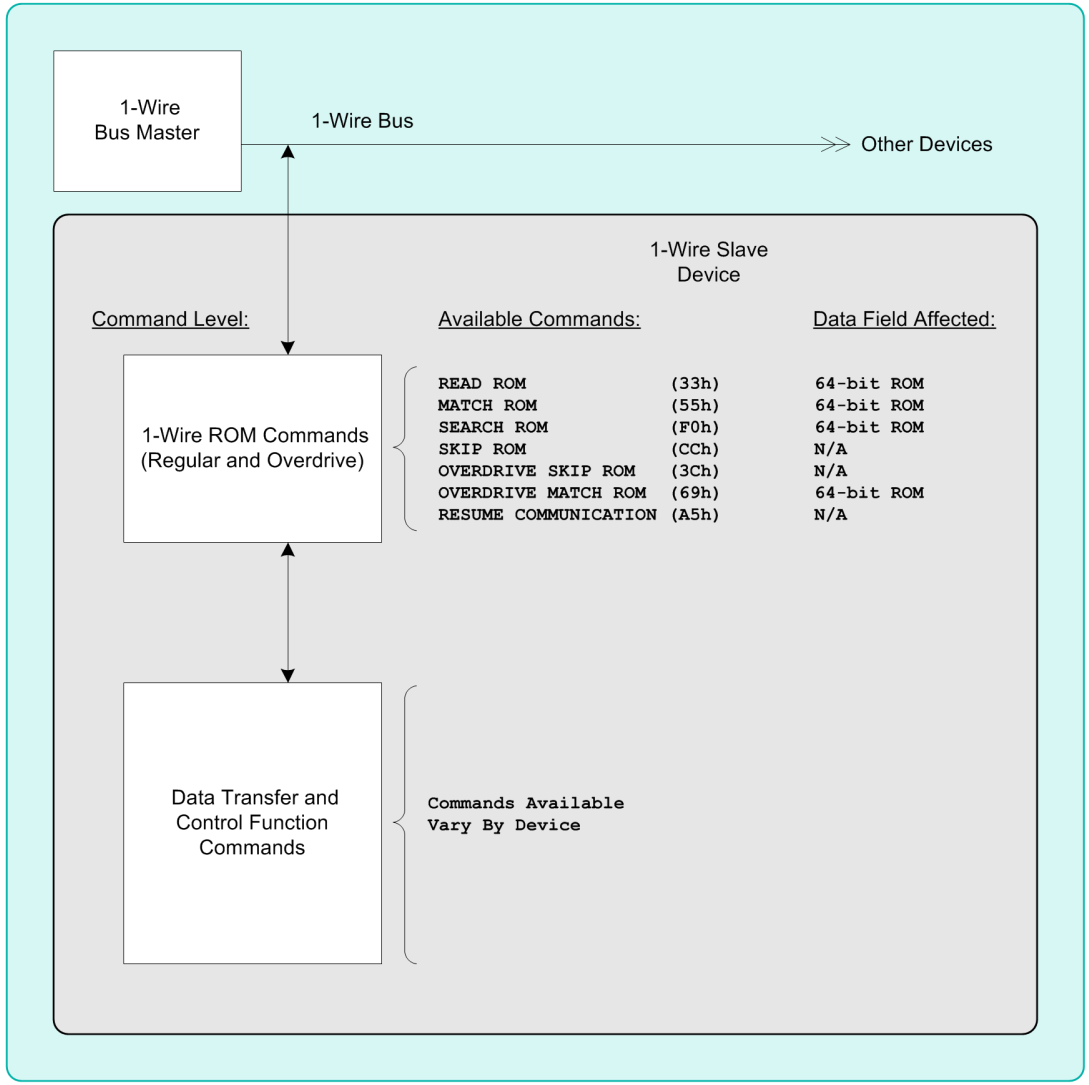

Figure 7.8: Typical 1-Wire Slave Device ROM Commands

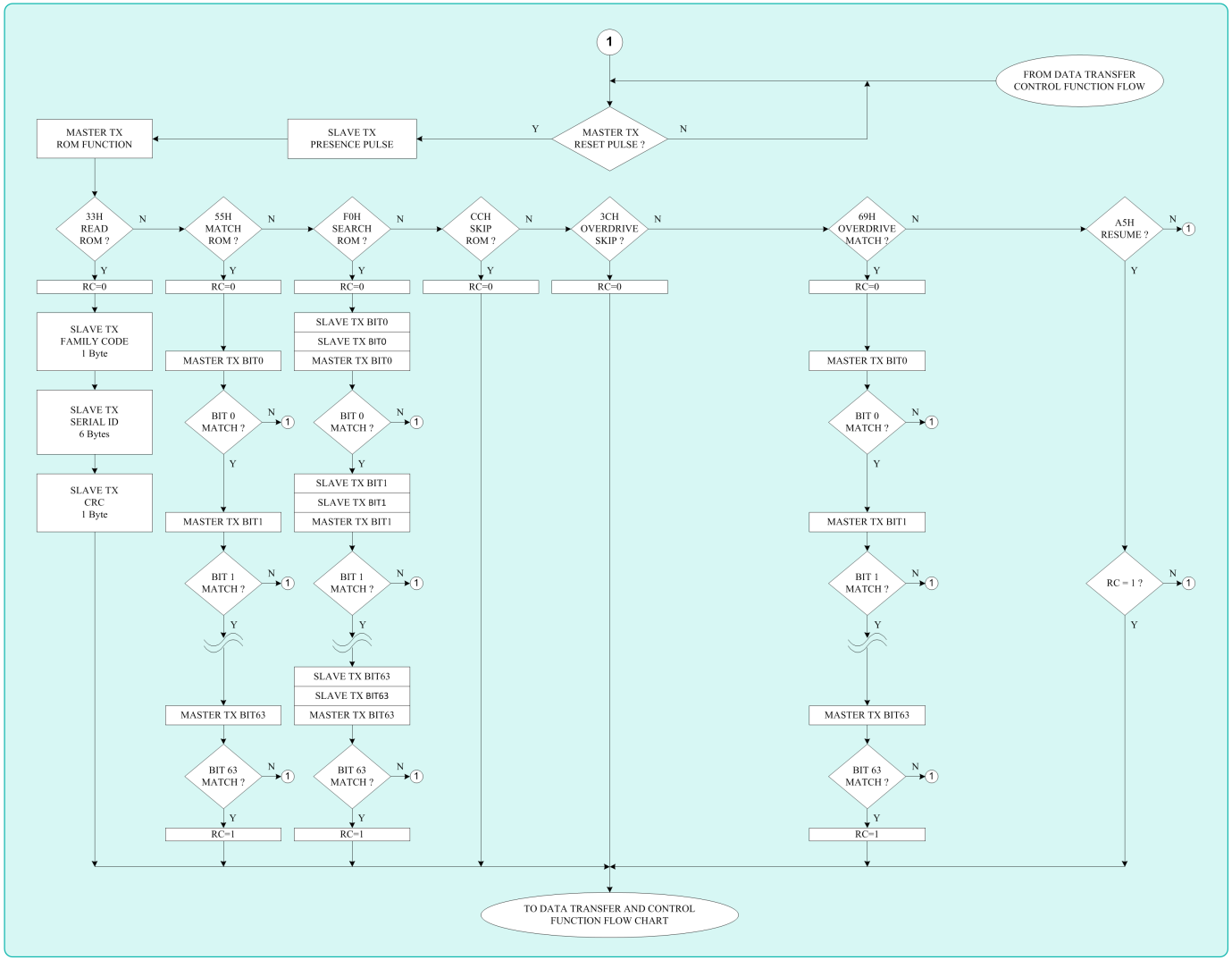

Figure 7.9: ROM Layer Command Operations

Read ROM Command [33h]

The Read ROM command allows the 1-Wire master to obtain the 8-byte ROM ID of any slave device connected to the 1-Wire bus. Each slave device on the bus responds to this command by transmitting all eight bytes of its ROM ID value, starting with the least significant byte (Family Code) and ending with the most significant byte (CRC).

Since this command is addressed to all 1-Wire devices on the bus, if more than one slave is present on the bus there will be a data collision as multiple slaves attempt to transmit their ROM IDs at once. This condition can be detected by the 1-Wire master since the CRC value will not match the ROM ID value received. In this case, the 1-Wire master should reset the 1-Wire bus at this point and select a single slave device on the bus to continue, either by using the Match ROM command (if the ROM ID values are already known) or the Search ROM command (if the master has not yet identified some or all devices on the bus).

After the Read ROM command has completed, all slave devices on the 1-Wire bus are selected or "active", and communications will proceed to the Transport layer.

Skip ROM Command [CCh], Overdrive Skip ROM Command [3Ch]

The Skip ROM command is used to activate all slave devices present on the 1-Wire bus, regardless of their ROM ID. Normally, this command is used when only a single 1-Wire slave device is connected to the bus. After the Skip ROM command has completed, all slave devices on the 1-Wire bus are selected or "active", and communications will proceed to the Transport layer.

The Overdrive Skip ROM command operates in an identical manner, except that executing it also causes the receiving slave device(s) to shift communications speed from standard speed to overdrive speed. The Overdrive Skip ROM command byte itself (3Ch) is transmitted at standard speed; all subsequent communications will then be sent at overdrive speed.

Match ROM Command [55h], Overdrive Match ROM Command [69h]

The Match ROM command is used by the 1-Wire master to select one and only one slave 1-Wire device when the ROM ID of the device has already been determined. When transmitting this command, the master sends the command byte (i.e., 55h for standard speed, 69h for overdrive speed) and then sends the entire 64-bit ROM ID for the device it wishes to select, least significant bit first.

During the transmission of the ROM ID by the master, all slave devices monitor the bus. As each bit is transmitted, each of the slave devices compares it against the corresponding bit of their ROM ID. If the bits match, the slave device continues to monitor the bus. If the bit does not match, the slave device transitions to the inactive state (waiting for a 1-Wire reset) and stops monitoring the bus.

At the end of the transmission, at most one slave 1-Wire device will be active, which will be the slave device whose ROM ID matched the ROM ID that was transmitted. All other slave devices will be inactive. Communications then proceeds to the Transport layer for the device that was selected.

The Overdrive Match ROM command operates in an identical manner, except that executing it also causes the slave device that is selected by the command to shift communications speed from standard speed to overdrive speed. The Overdrive Match ROM command byte (69h) and the 64-bit ROM ID bits are transmitted at standard speed; all subsequent communications will then be sent at overdrive speed.

Search ROM Command [F0h]

The Search ROM command allows the 1-Wire master to determine the ROM ID values of all 1-Wire slave devices connected to the bus using an iterative search process. Each execution of the Search ROM command will reveal the ROM ID of one slave device on the bus.

The operation of the Search ROM command resembles a combination of the Read ROM and Match ROM commands, and works as follows. First, all slaves on the bus transmit the least significant bit (Bit 0) of their ROM IDs. Next, all slaves on the bus transmit a complement of the same bit. By analyzing the two bits received, the master can determine whether the Bit 0 values were 0 for all slaves, 1 for all slaves, or a combination of the two. Next, the master selects which slaves will remain activated for the next step in the Search ROM process, by transmitting the Bit 0 value for the slave(s) it wishes to select. All slaves whose Bit 0 matches the value transmitted by the master remain active, while slaves with a different Bit 0 value go to the inactive state and do not participate in the remainder of the Search ROM command.

Next, the same process is followed for Bit 1, then Bit 2, and so on until the 63rd bit (most significant bit) of the ROM ID has been transmitted. At this point only one slave device will remain active, and the master can either continue on with communications at the Transport layer or issue a 1-Wire reset pulse to go back for another pass at the Search ROM command.

The *Book of iButton Standards* goes into more detail on the process that can be used by the master to obtain ROM IDs of all devices on the 1-Wire bus using multiple executions of the Search ROM command. The algorithm resembles a binary tree search, and can be used regardless of how many devices are on the bus, as long as no devices connect to or disconnect from the bus during the iterative process.

There is no overdrive equivalent version of the Search ROM command.

Resume Communications Command [A5h]

If more than one 1-Wire slave device is on the bus, then the master must specify which one it wishes to communicate with each time a new 1-Wire command (starting with a reset pulse) begins. Using the commands discussed previously, this would normally involve sending the Match ROM command each time, which means the master must explicitly specify the full 64-bit ROM ID of the part it wishes to communicate with for each command.

The Resume Communications command provides a shortcut for this process by allowing the master to repeatedly select the same device for multiple commands without having to transmit the full ROM ID each time. For devices which support this command, the operation of the command is as follows.

When the 1-Wire master selects a single device (using the Match ROM or Search ROM commands) an internal flag called the RC (for Resume Communications) flag is set in the slave device. (Only one device on the bus will have this flag set at any one time; the Skip ROM command will select multiple devices, but the RC flag is not set by the Skip ROM command).

When the master resets the 1-Wire bus, the RC flag will remain set. At this point, it is possible for the master to send the Resume Communications command. This command does not have a ROM ID attached to it, but the device that has the RC flag set will respond to this command by going to the active state, while all other devices will deactivate and drop off the 1-Wire bus.

Issuing any other ROM command will clear the RC flag on all devices. So, for example, if a Match ROM command is issued for device A, its RC flag will be set. The Resume Communications command can then be used repeatedly to send commands to device A. If a Match ROM command is then sent with the ROM ID of device B, the RC flag on device A will clear to 0, and the RC flag on device B will be set.

7.1.5 1-Wire Operation

Once the OWM module has been correctly configured as described previously, then using the OWM module to communicate with the 1-Wire network is simply a matter of directing the OWM 1-Wire master to generate the proper reset, read and write operations in order to communicate with the 1-Wire slave devices used in a given application.

The OWM 1-Wire Master handles the following 1-Wire protocol primitives directly in either Standard or Overdrive mode:

- 1-Wire Reset (including detection of presence pulse from responding slave device/devices)
- Write single bit (one write time slot)
- Write 8-bit byte, least significant bit first (eight write time slots)
- Read single bit (one write-1 time slot)
- Read 8-bit byte, least significant bit first (eight write-1 time slots)
- Search ROM Acceleration Mode, allowing four groups of three time slots (read, read, write) to be generated from a single 4-bit register write, to support the Search ROM command

7.1.5.1 Resetting the OWM 1-Wire Master

The first step in any 1-Wire communications sequence is to reset the 1-Wire bus. To direct the OWM module to perform a 1-Wire reset, the following sequence can be used:

1. Write OWM_CTRL_STAT.start_ow_reset to 1. This will generate a reset pulse and check for a replying presence pulse from any slave device(s) connected to the 1-Wire bus.
2. Once the reset time slot has completed, the OWM_CTRL_STAT.start_ow_reset field will be automatically cleared to zero.
3. At this point, the interrupt flag OWM_INTFL.ow_reset_done will also be set to 1 by hardware. This flag (which will trigger an interrupt from the OWM module if OWM_INTFL.ow_reset_done is also set to 1) must be cleared by firmware, by writing a 1 bit to the flag.
4. If a presence pulse was detected on the 1-Wire bus during the reset sequence (which should normally be the case unless no 1-Wire slave devices are present on the bus), the OWM_CTRL_STAT.presence_detect flag will also be set to 1. This flag does not result in the generation of an interrupt.

7.1.5.2 1-Wire Data Writes

Writing a Single Bit to the 1-Wire Bus

To transmit a single bit of data on the 1-Wire bus, the following procedure should be used.

1. Set OWM_CFG.single_bit_mode to 1. This setting causes the 1-Wire master to transmit/receive a single bit of data at a time, instead of the default of 8 bits.
2. Write the data bit to be transmitted (0 or 1) to OWM_DATA.tx_rx. Only bit 0 of the tx_rx field is used in this instance; the other bits in the field are ignored.

3. Once the single bit transmission has completed, hardware will set the interrupt flag OWM_INTFL.tx_data_empty to 1. This flag (which will trigger an OWM module interrupt if OWM_INTEN.tx_data_empty is also set to 1) must be cleared by firmware by writing a 1 to the flag.

Writing an 8-Bit Byte to the 1-Wire Bus

To transmit an 8-bit byte of data on the 1-Wire bus, the following procedure should be used.

1. Set OWM_CFG.single_bit_mode to 0. This setting causes the 1-Wire master to transmit/receive data 8 bits at a time. The least significant bit (LSB) of the data is always transmitted first.
2. Write the 8-bit value to be transmitted to OWM_DATA.tx_rx.
3. Once the 8-bit transmission has completed, hardware will set the interrupt flag OWM_INTFL.tx_data_empty to 1. This flag (which will trigger an OWM module interrupt if OWM_INTEN.tx_data_empty is also set to 1) must be cleared by firmware by writing a 1 to the flag.

7.1.5.3 1-Wire Data Reads

Reading a Single Bit Value from the 1-Wire Bus

The procedure for reading a single bit is very similar to the procedure for writing a single bit, since from the 1-Wire master's perspective, the operation is performed by writing a high (1) bit which the slave device either leaves high (to transmit a 1 bit) or overdrives by forcing the line low (to transmit a 0 bit).

1. Set OWM_CFG.single_bit_mode to 1. This setting causes the 1-Wire master to transmit/receive a single bit of data at a time, instead of the default of 8 bits.
2. Write OWM_DATA.tx_rx to 1. Only bit 0 of the tx_rx field is used in this instance; the other bits in the field are ignored.
3. Once the single bit transmission has completed, hardware will set the interrupt flag OWM_INTFL.tx_data_empty to 1. This flag (which will trigger an OWM module interrupt if OWM_INTEN.tx_data_empty is also set to 1) must be cleared by firmware by writing a 1 to the flag.
4. As the hardware shifts the bit value out, it also samples the value returned from the slave device. Once this value is ready to be read, the interrupt flag OWM_INTFL.rx_data_ready will be set to 1.
5. Read OWM_DATA.tx_rx (only bit 0 is used) to determine the value returned by the slave device.

Reading an 8-Bit Value from the 1-Wire Bus

The procedure for reading an 8-bit byte is very similar to the procedure for writing an 8-bit byte, since from the 1-Wire master's perspective, the operation is performed by writing eight high (1) bits which the slave device either leaves high (to transmit 1 bits) or overdrives by forcing the line low (to transmit 0 bits).

1. Set OWM_CFG.single_bit_mode to 0. This setting causes the 1-Wire master to transmit/receive in the default 8-bit mode.
2. Write OWM_DATA.tx_rx to 0FFh.
3. Once the 8-bit transmission has completed, hardware will set the interrupt flag OWM_INTFL.tx_data_empty to 1. This flag (which will trigger an OWM module interrupt if OWM_INTEN.tx_data_empty is also set to 1) must be cleared by firmware by writing a 1 to the flag.
4. As the hardware shifts the bit values out, it also samples the values returned from the slave device. Once the full 8-bit value is ready to be read, the interrupt flag OWM_INTFL.rx_data_ready will be set to 1.
5. Read OWM_DATA.tx_rx to determine the 8-bit value returned by the slave device. Note that if no slave devices are present or the slave device(s) are not communicating with the master, the return value (0FFh) will be the same as the transmitted value.

Search ROM Accelerator (SRA) Operation

The Search ROM operation is intended to allow a 1-Wire master to systematically determine each of the 64-bit ROM ID values of all 1-Wire slave devices currently present on the bus. If only a single device is present, the simpler Read ROM command can be used; but for multiple devices, the Search ROM command provides a way to sequentially iterate through an unknown number of slaves with unknown ID values. Refer to the *Book of iButton Standards* for a thorough description of this command and other commands supported at the ROM layer of the 1-Wire protocol.

To allow this command to be processed more quickly, the OWM module provides a special accelerator mode for use with the Search ROM command. This mode is activated by setting OWM_CTRL_STAT.sra_mode to 1.

When this mode is active, ROM IDs being processed by the Search ROM command are broken into 4-bit nibbles, where the current 64-bit ROM ID varies with each pass through the search algorithm. Each 4-bit processing step is initiated by writing the 4-bit value to OWM_DATA.tx_rx. This causes a total of 12 1-Wire time slots to be generated by the 1-Wire master, as each bit in the 4-bit value (starting with the LSB) results in a read of two bits (all active slaves transmitting bit N of their ROM IDs, then all active slaves transmitting the complement of bit N of their ROM ID), and then a write of a single bit by the 1-Wire master.

After the 4-bit processing stage has completed, the return value loaded into OWM_DATA.tx_rx will consist of eight bits divided into two 4-bit nibbles. The low nibble (bits 0 through 3) contains the four discrepancy flags: one for each ID bit processed. If the discrepancy bit is set to 1, it means that either two slaves with differing ID bits in that position both responded (the two bits read were both zero), or that no slaves responded (the two bits read were both 1). If the discrepancy bit is set to 0, then the two bits read were complementary (either 0, 1 or 1, 0) meaning that there was no bus conflict.

The high nibble (bits 7 through 4) contains the four bits that were transmitted by the 1-Wire master. These bit values will either be the bit values read (if there was no bus conflict) or the 'default' bit values specified in the original OWM_DATA.tx_rx value (if there were conflicting slaves driving the bus).

In this way, at each step in the Search ROM command, the master either 'follows' the ID of the responding slave(s), or deselects some of the slaves on the bus in case of a conflict. By the time the end of the 64-bit ROM ID has been reached (which will be the 16th 4-bit group processing step), the combination of all bits from the high nibbles of the received data will be equal to the ROM ID of one of the slaves remaining on the bus. Subsequent passes through the Search ROM algorithm can be used to determine additional slave ROM ID values, until all slaves have been identified. Refer to the *Book of iButton Standards* for a much more in depth explanation of the search function and possible variants of the search algorithm applicable to particular circumstances.

7.1.6 Registers (OWM)

Address	Register	Access	Description	Reset By
0x4001E000	OWM_CFG	R/W	1-Wire Master Configuration	Sys
0x4001E004	OWM_CLK_DIV_1US	R/W	1-Wire Master Clock Divisor	Sys
0x4001E008	OWM_CTRL_STAT	***	1-Wire Master Control/Status	Sys
0x4001E00C	OWM_DATA	R/W	1-Wire Master Data Buffer	Sys
0x4001E010	OWM_INTFL	W1C	1-Wire Master Interrupt Flags	Sys
0x4001E014	OWM_INTEN	R/W	1-Wire Master Interrupt Enables	Sys

7.1.6.1 OWM_CFG

OWM_CFG.long_line_mode

Field	Bits	Sys Reset	Access	Description
long_line_mode	0	0	R/W	Long Line Mode

Adjusts timing parameters to support a long wire.

OWM_CFG.force_pres_det

Field	Bits	Sys Reset	Access	Description
force_pres_det	1	0	R/W	Force Line During Presence Detect

For noisy wires, forces the 1-wire line during presence detection.

OWM_CFG.bit_bang_en

Field	Bits	Sys Reset	Access	Description
bit_bang_en	2	0	R/W	Bit Bang Enable

Enables bit bang control of the 1-Wire line.

OWM_CFG.ext_pullup_mode

Field	Bits	Sys Reset	Access	Description
ext_pullup_mode	3	0	R/W	Provide an extra output to control an external pullup.

OWM_CFG.ext_pullup_enable

Field	Bits	Sys Reset	Access	Description
ext_pullup_enable	4	0	R/W	Enable External FET Pullup

Enable external FET pullup when idle, otherwise allow line to float.

OWM_CFG.single_bit_mode

Field	Bits	Sys Reset	Access	Description
single_bit_mode	5	0	R/W	Enable Single Bit TX/RX Mode

When set, only a single bit at a time will be transmitted and received (LSB of OWM_DATA) rather than the whole byte.

OWM_CFG.overdrive

Field	Bits	Sys Reset	Access	Description
overdrive	6	0	R/W	Enables overdrive speed for 1-Wire operations.

OWM_CFG.int_pullup_enable

Field	Bits	Sys Reset	Access	Description
int_pullup_enable	7	0	R/W	Enable internal pullup.

7.1.6.2 OWM_CLK_DIV_1US

OWM_CLK_DIV_1US.divisor

Field	Bits	Sys Reset	Access	Description
divisor	7:0	00b	R/W	Clock Divisor for 1MHz

Frequency is set to the OWM module clock frequency divided by the value of this divisor field. Off when set to 0.

7.1.6.3 OWM_CTRL_STAT

OWM_CTRL_STAT.start_ow_reset

Field	Bits	Sys Reset	Access	Description
start_ow_reset	0	0	R/W	Start OW Reset

Write to 1 to begin a OW Reset sequence. Hardware clears this bit to 0 automatically when the OW Reset has completed.

OWM_CTRL_STAT.sra_mode

Field	Bits	Sys Reset	Access	Description
sra_mode	1	0	R/W	SRA Mode

Enables SRA mode. This mode is used to identify slaves and their addresses which are attached to the 1-Wire bus.

OWM_CTRL_STAT.bit_bang_oe

Field	Bits	Sys Reset	Access	Description
bit_bang_oe	2	0	R/W	Bit Bang Output Enable

When bit bang mode is enabled, controls the 1-Wire line output enable.

OWM_CTRL_STAT.ow_input

Field	Bits	Sys Reset	Access	Description
ow_input	3	n/a	R/O	OW Input State

Returns the current logic level on the 1-Wire line.

OWM_CTRL_STAT.presence_detect

Field	Bits	Sys Reset	Access	Description
presence_detect	7	0	R/O	Presence Pulse Detected

Set to 1 when a presence pulse was detected from one or more slaves during the OW Reset sequence.

7.1.6.4 OWM_DATA

OWM_DATA.tx_rx

Field	Bits	Sys Reset	Access	Description
tx_rx	7:0	00b	R/W	Tx/Rx Buffer

Writing to this field sets the transmit data for the next 1-Wire data transmit cycle. Reading from this field returns the data received by the master during the last 1-Wire data transmit cycle.

7.1.6.5 OWM_INTFL

OWM_INTFL.ow_reset_done

Field	Bits	Sys Reset	Access	Description
ow_reset_done	0	0	W1C	OW Reset Sequence Completed

OWM_INTFL.tx_data_empty

Field	Bits	Sys Reset	Access	Description
tx_data_empty	1	0	W1C	Tx Data Empty Interrupt Flag

OWM_INTFL.rx_data_ready

Field	Bits	Sys Reset	Access	Description
rx_data_ready	2	0	W1C	Rx Data Ready Interrupt Flag

OWM_INTFL.line_short

Field	Bits	Sys Reset	Access	Description
line_short	3	0	W1C	OW Line Short Detected Interrupt Flag

OWM_INTFL.line_low

Field	Bits	Sys Reset	Access	Description
line_low	4	0	W1C	OW Line Low Detected Interrupt Flag

7.1.6.6 OWM_INTEN

OWM_INTEN.ow_reset_done

Field	Bits	Sys Reset	Access	Description
ow_reset_done	0	0	R/W	OW Reset Sequence Completed

OWM_INTEN.tx_data_empty

Field	Bits	Sys Reset	Access	Description
tx_data_empty	1	0	R/W	Tx Data Empty Interrupt Enable

OWM_INTEN.rx_data_ready

Field	Bits	Sys Reset	Access	Description
rx_data_ready	2	0	R/W	Rx Data Ready Interrupt Enable

OWM_INTEN.line_short

Field	Bits	Sys Reset	Access	Description
line_short	3	0	R/W	OW Line Short Detected Interrupt Enable

OWM_INTEN.line_low

Field	Bits	Sys Reset	Access	Description
line_low	4	0	R/W	OW Line Low Detected Interrupt Enable

7.2 I²C

7.2.1 Overview

The **MAX32620** integrates three I²C bus masters for communication with a wide variety of I²C enabled slaves. The I²C bus is a two-wire, bidirectional bus (i.e., can operate as a master-transmitter or master-receiver) using a ground line and two bus lines: the serial data access line (SDA) and the serial clock line (SCL). Both the SDA and SCL lines must be driven as open-collector/drain outputs. External resistors (R_P) are recommended to pull the lines to a logic-high state; internal pullups can also be used to start I²C buses with low capacitance.

The **MAX32620** supports both the master and slave protocols. The master I²C peripherals have ownership of the I²C bus, drive the clock via the SCL pin, and generate the START and STOP signals. This enables the **MAX32620** to send and receive data to and from a slave as required by the user's application. In slave mode, the **MAX32620** relies on an externally generated clock to drive SCL and responds to data and commands only when requested by the external I²C master device.

7.2.2 Features

The I²C host port is compliant with the I²C Bus Specification with features:

- I²C bus specification version 2.1 compliant (100kHz and 400kHz)
- Programmable for both normal (100 kHz) and fast bus data rates (400kHz)
- Tagged-byte (16-byte depth) FIFOs:
  - Transaction FIFO
  - Results FIFO
- Support for 10-bit device addressing
- Clock synchronization and bus arbitration
- Supports arbitration in a multi-master environment
- Supports I²C bus hold for slow host service
- Transfer status interrupts and flags
- Support DMA data transfer via the Peripheral Management Unit (PMU)

The **MAX32620** I²C slave interface module supports the following features:

- High level I²C protocol interface engine allows an external I²C bus master to write to and read from a set of 32 data byte registers with no action required by the CPU
- Support for 7-bit or 10-bit device addressing
- SCL filter parameters are configurable to match timing values used by the I²C bus
  - Filter rates can be set for Standard speed, Full speed, or Full Plus speed modes
  - Spreadsheet available to compute programming values
- The 32 data byte registers can be set to be either read/write or read-only (by the external bus master) on a register-by-register basis
- An interrupt to the CPU can be triggered when any of the 32 data byte registers is updated by the external bus master; this is configurable on an individual register basis

Speed Categories

The I²C ports support two operating speed categories:

- Standard-mode with a bit rate up to 100Kbps
- Fast-mode with a bit rate up to 400Kbps

Note All interfaces are downward compatible and will operate at a lower bus speed as necessary.

7.2.3 I²C Port and Pin Configurations

Note See Pin Layout for a detailed mapping of **MAX32620** multiplexed function locations. Functional priority distinction is included in the mapping.

Standard Layout (WLP) Configuration

SDA and SCL port pins for the I²C Master ports (I2CM0, I2CM1, I2CM2) and port and pin configuration options for the I²C Slave port (I2CS):

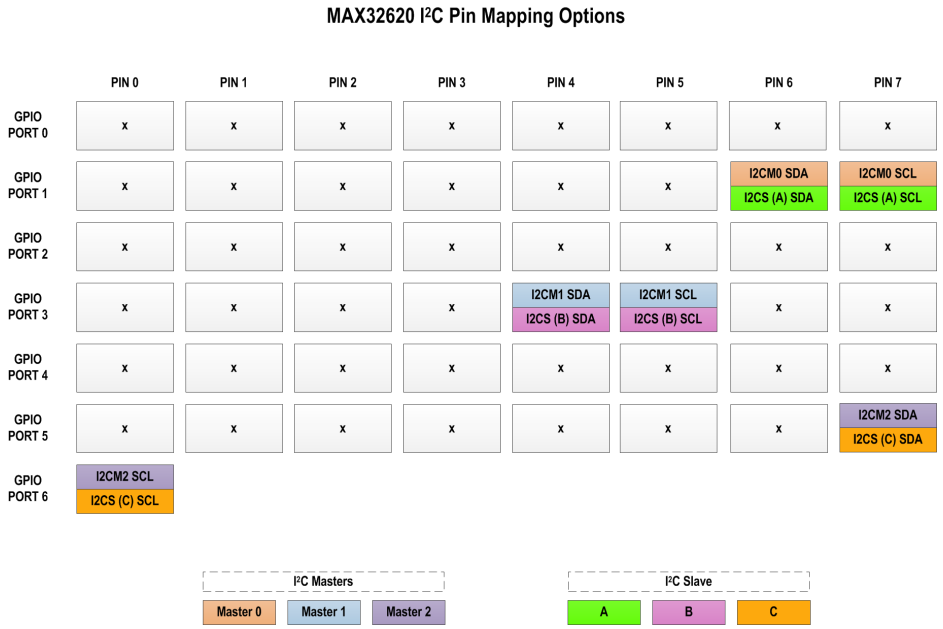

Figure 7.10: Mapping Options

I²C Master Configurations (Standard)

I2CM0

Logic Signal	Port and Pin
SDA	P1.6
SCL	P1.7

I2CM1

Logic Signal	Port and Pin
SDA	P3.4
SCL	P3.5

I2CM2

Logic Signal	Port and Pin
SDA	P5.7
SCL	P6.0

I²C Slave Configurations (Standard)

I2CS

Logic Signal	Port and Pin
SDA	A) P1.6 B) P3.4 C) P5.7
SCL	A) P1.7 B) P3.5 C) P6.0

7.2.4 I²C Master Operation

Firmware defines I²C read and write operations via a transaction packet pushed onto the Master Transaction FIFO. I²C operation results are then read from Master Results FIFO. An unexpected NACK on write data results in system interrupt. In this case, hardware can be opted to automatically issue a Stop under this condition to free the bus. Results FIFO records the ACK/NACK status for each read data value received. Further, a full Results FIFO or empty Transaction FIFO will result in clock stretching by master. There is a timeout feature which will issue a system interrupt if this delay exceeds a firmware controllable value (in mS). Loss of arbitration in a multi-master system will result in a system interrupt.

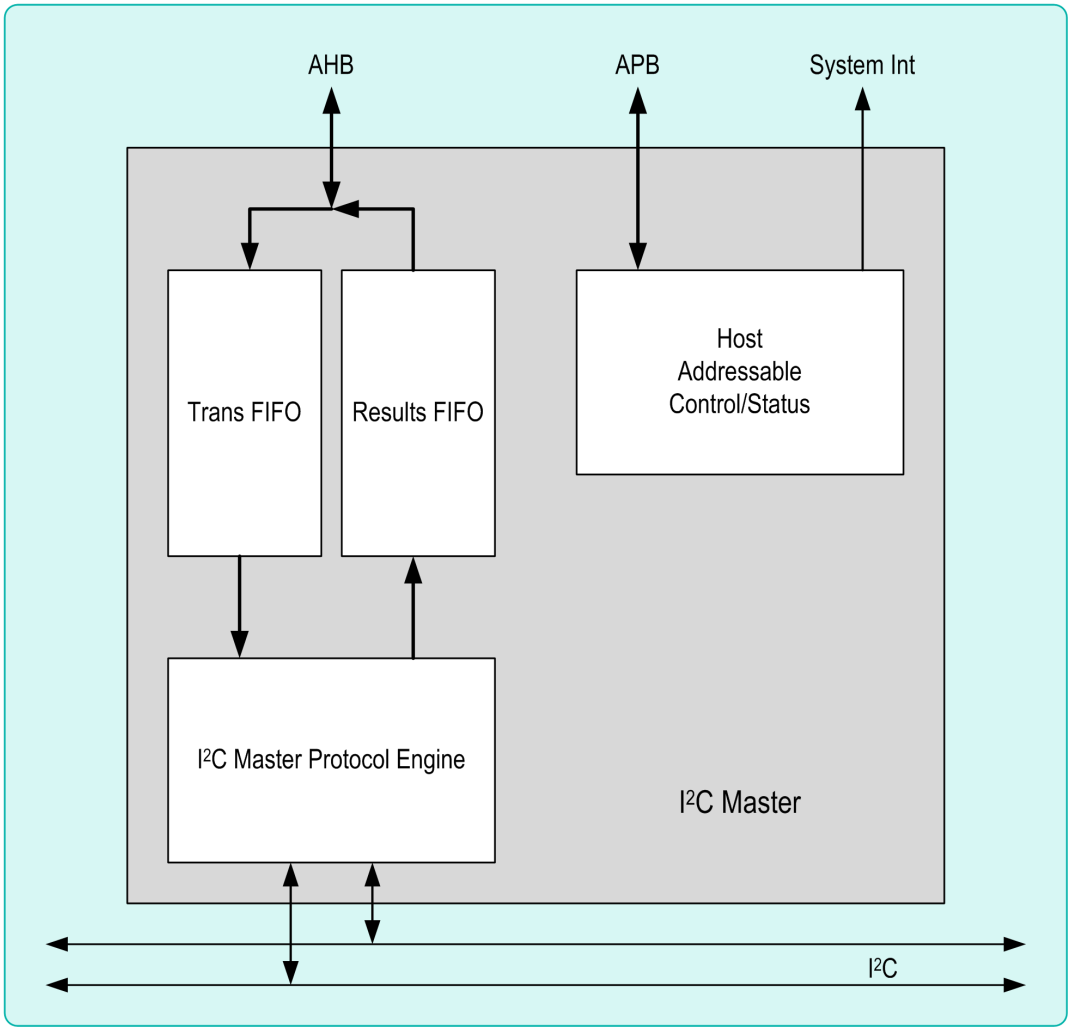

Figure 7.11: I²C Master Block Diagram

7.2.5 Protocol

The I²C protocol communication is a two-wire serial transmission. It is a half-duplex protocol where data can be transferred at various baud rates: up to 100Kbits (100Kbps) in standard mode and up to 400Kbits (400Kbps) in fast mode.

Transfer Protocol

Each transfer is made of a start bit followed by one or more sequences of eight data bits, acknowledge bit(s) sent by the receiver, and a stop bit. Data is sent with the most significant bit (MSB) first. Every byte delivered to the SDA line must be eight bits long; however, the number of bytes that can be transmitted per transfer is unrestricted.

Bit Transfer

Both SDA and SCL lines are bi-directional lines connected to a positive supply voltage via a current source or a pullup resistor. When the bus is free, the lines are in high state. The data on the SDA line must be stable when the SCL line is high.

Communication starts when the SDA line switches from high to low state and the SCL line is high. Communication stops when the SDA line switches from low to high state and the SCL line is high. Only the master generates the start and stop conditions. After the start condition and during communication, the bus is considered busy.

Start and Stop Conditions

A high to low transition on the SDA line while SCL is high defines a start condition; a low to high transition on the SDA line while SCL is high defines a stop condition.

Acknowledge (ACK) and Not Acknowledge (NACK)

The acknowledge takes place after every byte after which the receiver signals the transmitter that the byte was successfully received and another byte may be sent. The acknowledge signal operates as follows: the transmitter releases the SDA line during the acknowledge clock pulse so the receiver can pull the SDA line to the low state (it remains stable in the low state during the high period of this clock pulse on the SCL line). Setup and hold times may affect this behavior.

A NACK signal will occur when the SDA remains high during this ninth clock pulse. The I²C master can then generate either a stop condition to abort the transfer or a repeated start condition to start a new transfer. There are five conditions that lead to NACK signal generation:

1. No receiver is present on the bus with the transmitted address so there is no device to respond with an acknowledge signal.
2. The receiver is unable to receive or transmit because it is performing some real-time function and is not ready to start communication with the master.
3. During the transfer, the receiver gets data or commands that it does not understand.

4. During the transfer, the receiver cannot receive any more data bytes.
5. A master-receiver must signal the end of the transfer to the slave transmitter.

Addressing

The I2CM block runs in master mode only, and in fast or standard mode. The first byte sent defines the type of addressing.

Definition of the First Byte

Definition of the First Byte

7-1 bits	R/W bit	Definition
0000 000	0	General Call Address
0000 000	1	START byte
0000 001	X	Reserved (CBUS address)
0000 010	X	Reserved
0000 011	X	Reserved
0000 1XX	X	Reserved
1111 1XX	X	Reserved

General Call Address

The general call address is for addressing every device connected to the I²C bus at the same time. The General Call Address is 0x00 for the first byte transmitted and addresses all devices on the bus. A device that does require data from the general call address acknowledges the address and behaves as a slave-receiver; a device that does not require any of the data supplied within the general call address does not have to acknowledge the command. A General Call Address is followed by a second byte, which initiates the devices.

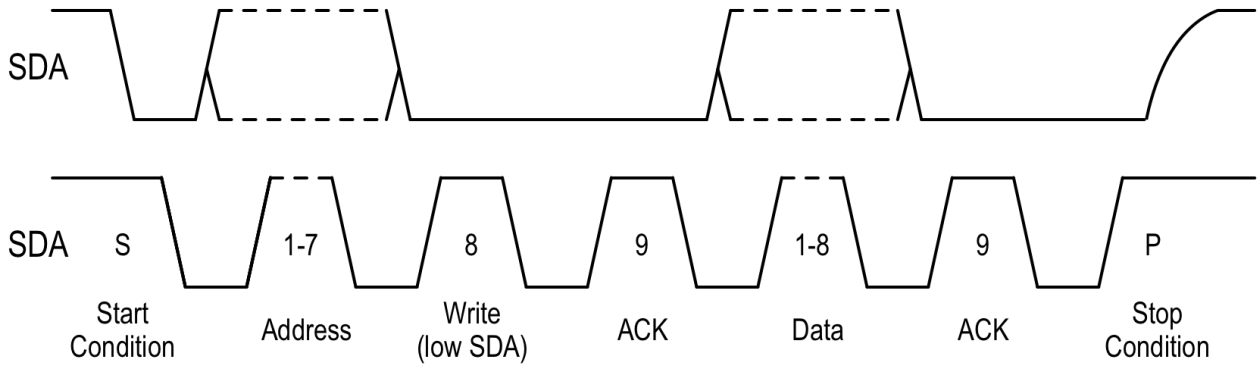

Figure 7.12: I²C Transfer

Read/Write Bit

When the R/W bit is 0, the second byte has the following definition:

- 0x06: Reset and write the programmable part of the slave address by hardware.
- 0x04: Write the programmable part of the slave address by hardware.
- 0x00: This code is not allowed to be used as the second byte.

When the R/W bit is set to 1, the hardware address of the master is written in the second byte.

The General Call Address has no effect on this interface.

START Byte

The START byte is used to initiate communication with slow devices (i.e., one reliant on software polling). The START byte has no effect on this interface.

If two or more slaves are addressed at the same time, only the slave that has the smallest address (described in decimal figures), can exchange data with the master. Communication with the other slaves is cut off.

Clock Synchronization

The I²C protocol accepts multiple masters on the bus. This is the reason why synchronization between the masters' clocks is compulsory. Clock synchronization is performed using the wired-AND connection of SCL.

Bus Arbitration

When the I²C bus is free, the two masters can initiate communication; only one master can make a valid transmission and the other must switch to slave mode. Each master compares the data sent to the SDA line. If the data is different, the master with SDA high state output will switch off its SDA data output. Arbitration occurs until only one master remains.

Master Interrupt

The **MAX32620** I²C Controller contains multiple interrupt sources that are combined into one interrupt request signal to the processor. The source of the interrupt is determined by which bits are set in the I2CMn_INTEN register.

It is recommended to acknowledge an interrupt before enabling it (unmask operation) to avoid a previous event triggering.

7.2.6 Peripheral Clock Selection and Clock Gating

To initialize the I²C master ports, the system clock source and divider must first be configured. Divider selection is dependent on many variables, including: the targeted bus-speed, the system clock frequency, and the dividers of both the system clock frequency and the I²C peripheral clock. The register field CLKMAN_SYS_CLK_CTRL_9_I2CM.i2cm_clk_scale enables the system clock to the I²C master ports and sets the system clock frequency divider. This is a 4-bit field.

Peripheral Clock Selection

i2cm_clk_scale	Description
0000b	CLK is Disabled
0001b	(System Clock Source / 1)
0010b	(System Clock Source / 2)
0011b	(System Clock Source / 4)
0100b	(System Clock Source / 8)
0101b	(System Clock Source / 16)
0110b	(System Clock Source / 32)
0111b	(System Clock Source / 64)
1000b	(System Clock Source / 128)
1001b	(System Clock Source / 256)

i2cm_clk_scale	Description
other	(System Clock Source / 1)

7.2.6.1 Peripheral Clock Frequency Selection

To set up the SCL frequency, it is necessary to set up four configuration registers to achieve proper operation: the I²C Peripheral Clock, Filter Clock Divisor, SCL low time, and SCL high time. The amount of time required for the SCL low time versus SCL high time (Duty Cycle) is dependent on the specific slave device(s) being communicated with. table_i2c_scl_clks shows typical target SCL low versus SLC high times for standard slave I²C devices.

Note Pullup delays, input filtering delays, and multi-master clock synchronization affect the observed SCL frequency on the I²C bus.

SCL Clock Configuration Common Calculations

PCLK	F_{I2C}	$I2C_{DUTY}$	F_{HOLD}	T_{RC}	Filt CLK Divisor	SCL Hi	SCL Lo
96	100	0.67	1	1000	48	164	576
96	400	0.67	0.25	300	12	33	144
96	1000	0.67	0.1	120	5	11	57
48	100	0.67	1	1000	24	80	288
48	400	0.67	0.25	300	6	15	72
48	1000	0.67	0.1	120	2	3	29
24	100	0.67	1	1000	12	38	144
24	400	0.67	0.25	300	3	5	36
12	100	0.67	1	1000	6	17	72
12	400	0.67	0.25	300	2	1	18
6	100	0.67	1	1000	3	7	36
3	100	0.67	1	1000	2	1	18

Note Assuming more typical RC delay, observed I²C frequency could be 8-10% faster than the target.

Explanation of SCL Clock Configuration Common Calculations values:

- PCLK = Peripheral Clock (MHz)

- F_{I2C} = I²C Frequency (KHz)
- I^2C_{DUTY} = I²C Duty Cycle (H/L)
- F_{HOLD} = I²C Hold (μs)
- T_{RC} = RC Rise Time (ns)
- Filt CLK Divisor = fs_filter_clk_div
- SCL Hi = fs_scl_hi_cnt
- SCL Lo = fs_scl_lo_cnt

7.2.7 Communication and Data Transfer

7.2.7.1 FIFO-Based I²C Master

The FIFO-based I²C engine implements a packetized interface to slave devices. This interface supports multi-master environments, 10-bit addressing, and System Management Bus (SMbus) protocol.

Sixteen-byte FIFOs are provided on each master peripheral for both TX and RX. The data is tagged prior to enqueuing to allow decoupling of firmware execution from I²C operation. A full complement of FIFO status is available for firmware monitoring and interrupt generation.

The user must define the filter clock frequency, which is typically two to four times faster than the target SCL frequency. The target SCL frequency and duty cycle are set by defining the low and high system clock limits of the SCL clock (i.e., SCL low time and SCL high time). If operating in a high-speed environment, the user must define two sets of clock control registers for normal-speed as well as high-speed.

Note Pullup delays, input filtering delays, and multi-master clock synchronization affect the observed SCL frequency on the I²C bus.

I²C and SMBus Compliance

SMBus and I²C protocols are essentially the same: an SMBus master is able to control I²C devices and vice versa at the protocol level. The SMBus clock is defined from 10kHz to 100kHz whereas I²C can range from 0Hz to 100 kHz or 0Hz to 400kHz depending on the mode. An I²C bus running at less than 10kHz is not SMBus compliant since the SMBus device(s) may time out (see Timeout Feature below).

Peripheral Management Unit

The I²C supports direct memory access peripheral communication via the Peripheral Management Unit (PMU). This allows the PMU to read and/or write the FIFOs of the I²C device.

To enable the PMU transfers, reference Peripheral Management Unit (PMU). There is no additional configuration for the I²C device: the PMU should be seen as a core (executing software) replacement. The I²C interrupts are not rerouted to the PMU and are still routed to the core (it is up to the software to configure the interrupts according to the PMU usage).

RX FIFO

RX FIFO transfers are 8-byte depth. The RX FIFO is written by hardware when a data has been received from the device. If the RX FIFO is full, all data received from the slave are lost; a new read operation cannot be started with the RX FIFO full. Reading the data register is necessary to read the RX FIFO.

TX FIFO

TX FIFO transfers are 8-byte depth. The TX FIFO is written by software using the data register (a write in the data register writes into the TX FIFO). The TX FIFO is read by hardware only when the acknowledge bit has been received (meaning just before the stop/restart bit in case of NACK or just before the first bit of the next data in case of ACK).

Timeout Feature

The SMBus protocol has a timeout feature which resets devices if communication takes too long. The minimum clock frequency of 10kHz for SMBus peripherals prevents locking up the bus erroneously. The I²C can be a DC bus, meaning that a slave device stretches the master clock when performing some routine while the master is accessing it. This notifies the master that the slave is busy but does not want to lose communication. The slave device will allow continuation of the communication after its task is complete. There is no limit to the delay in the I²C bus protocol; for a SMBus system, the delay is limited to 35ms. SMBus protocol assumes if a communication takes too long, there must be a problem on the bus and all devices must reset in order to clear this mode. This is the timeout feature of the SMBus. Slave devices are not then allowed to hold the clock in the low state too long.

7.2.8 I²C Interrupts

Slave Interrupt

Interrupt sources are available for each of the 32 data bytes to indicate that a new value has been written to that data byte by an external I²C master (data byte updated). The enabled interrupt sources (as determined by bit settings in I2CS_INTEN) are combined into a single interrupt request to the NVIC.

In the firmware interrupt handler, the source of the interrupt can be determined by examining the interrupt flag bits in the I2CS_INTFL register. Once the source of the interrupt has been determined, the interrupt flag can be cleared by writing the flag to 1.

When enabling one or more new interrupts using the I2CS_INTEN register, it is recommended to first clear any previously set interrupt flags by writing 1 bits to those flags in the I2CS_INTFL register. This avoids the generation of unwanted interrupts based on events that occurred before the interrupt or interrupts were enabled.

7.2.9 Module Clock Generation

When initializing the I2CS module, the module clock source must first be enabled and configured. Divider selection is dependent on many variables, including the targeted I²C bus speed and system clock frequency. Since the I2CS is a slave-only module, it does not have direct control over the I²C bus speed, so it must be initialized to support the fastest I²C bus speed that will be used in the application.

To enable the source clock for the I2CS module, the register field CLKMAN_SYS_CLK_CTRL_10_I2CS.i2cs_clk_scale must be written to a non-zero value. The value chosen will determine if the undivided system clock source is used directly to generate the I2CS module clock, or if the system clock source will be divided down to generate a lower frequency I2CS module clock, as shown in the table below.

Module Clock Selection

i2cs_clk_scale	I2CS Clock Setting
0	I2CS clock is Disabled
1	(System Clock Source / 1)
2	(System Clock Source / 2)
3	(System Clock Source / 4)
4	(System Clock Source / 8)
5	(System Clock Source / 16)
6	(System Clock Source / 32)
7	(System Clock Source / 64)
8	(System Clock Source / 128)
9	(System Clock Source / 256)
other	Reserved

The filter clock divisor field must also be set using the same calculations that would be used for the I2CM module running at the same desired bus frequency; refer to the I²C Clock Selection Frequency section for more details.

7.2.10 Communication and Data Transfer

7.2.10.1 I²C Mailbox

The I2CS module is organized using a bidirectional set of 32 'mailbox' data registers. These registers provide a way to pass data back and forth between the **MAX32620** and one (or more) external I²C master devices over the I²C bus interface. Because the protocol for accessing these registers via the I²C bus is supported entirely in hardware logic by the I2CS, when an external I²C bus master writes to or reads from one or more of these data registers, the main CPU does not have to be involved at all.

The 32 mailbox data registers are general-purpose by design. Any organization or special meaning that is given to the register set (including whether all 32 registers are used or only a subset of them) must be determined by the application designer. All of the registers provide identical capabilities and can be read or written from

either side in the same manner.

From the internal (APB bus) side, the 32 mailbox data registers can be read from or written to directly by the CPU or by the PMU by accessing the data_field field within the appropriate I2CS_DATA_BYTE register. From the external (I²C bus) side, the same 32 mailbox data registers can be read from or written to by an external I²C bus master, using the built-in protocol supported by the I2CS module.

7.2.10.2 Slave Addressing

The I²C slave interface implemented in the I2CS module responds to a single 7-bit or 10-bit slave address on the I²C bus. The slave address used is controlled by the DEV_ID.slave_dev_id field; when in 7-bit mode, any bits written to the high 3 bits of this field will not be used. The 7-bit address or 10-bit address mode option is selected using DEV_ID.ten_bit_id_mode.

The I2CS module does not support the General Call function.

7.2.10.3 Writing to a Single Mailbox Register

In order for the external I²C bus master to write to a single mailbox data register in the I2CS module, the following transaction sequence must be used:

1. START
2. Tx: Address byte or bytes for 7-bit or 10-bit ID of the I2CS slave
3. Rx: ACK from I2CS
4. Tx: Index value (from 0 to 31) of the mailbox data register to be accessed
5. Rx: ACK from I2CS
6. Tx: New data byte (8 bits) to be written to the mailbox data register
7. Rx: ACK from I2CS
8. STOP

Note that if the mailbox data register in question has been set (by firmware) to be read-only by setting the corresponding I2CS_DATA_BYTE<n>.read_only_fl flag to 1, if the external bus master attempts to write to that data register using the above protocol, the I2CS module will respond with a NACK after the new data byte is sent by the I²C master.

When the I²C master writes to the mailbox data register, the corresponding I2CS_DATA_BYTE<n>.data_updated_fl will be set to 1 to indicate that the data register has been updated by the external master.

7.2.10.4 Writing to Multiple Mailbox Registers

Once the index value has been sent by the I²C master as part of a write or read transaction sequence, the index value is stored internally by the I2CS module, and it is incremented by 1 following each data byte read or written by the master. This means that it is possible to write to multiple mailbox data registers in a single transaction using the following sequence:

1. START
2. Tx: Address byte or bytes for 7-bit or 10-bit ID of the I2CS slave
3. Rx: ACK from I2CS
4. Tx: Index value N (from 0 to 31)
5. Rx: ACK from I2CS
6. Tx: New data byte (8 bits) to be written to data register N
7. Rx: ACK from I2CS
8. Tx: New data byte (8 bits) to be written to data register (N+1)
9. Rx: ACK from I2CS
10. Tx: New data byte (8 bits) to be written to data register (N+2)
11. Rx: ACK from I2CS
12. Tx: New data byte (8 bits) to be written to data register (N+3)
13. Rx: ACK from I2CS
14. STOP

Note that if *any* of the the mailbox data registers have been set (by firmware) to be read-only by setting the corresponding I2CS_DATA_BYTE<n>.read_only_fl flag to 1, if the external bus master attempts to write to that data register using the above protocol, the I2CS module will respond with a NACK after the new data byte is sent by the I²C master, and the sequence will be interrupted.

When the I²C master writes to each mailbox data register, the corresponding I2CS_DATA_BYTE<n>.data_updated_fl will be set to 1 to indicate that the data register has been updated by the external master.

7.2.10.5 Reading from a Single Mailbox Register

In order for the external I²C bus master to read to a single mailbox data register in the I2CS module, the following transaction sequence must be used:

1. START
2. Tx: Address, with R/nW bit set to 0 (write operation)
3. Rx: ACK from I2CS
4. Tx: Index value (from 0 to 31) of the mailbox data register to be accessed
5. Rx: ACK from I2CS
6. Repeated START
7. Tx: Address, with R/nW bit set to 1 (read operation)
8. Switch to read mode, I2CS will begin transmitting data at this point
9. Rx: Current data byte (8 bits) from the mailbox data register
10. Tx: NACK from I²C master
11. STOP

The end of the read is signalled to the I2CS with the STOP.

7.2.10.6 Reading from Multiple Mailbox Registers

As with write operations, it is possible for the external I²C bus master to read from multiple mailbox data registers in a single transaction sequence, as follows:
In order for the external I²C bus master to read to a single mailbox data register in the I2CS module, the following transaction sequence must be used:

1. START
2. Tx: Address, with R/nW bit set to 0 (write operation)
3. Rx: ACK from I2CS
4. Tx: Index value N (from 0 to 31)
5. Rx: ACK from I2CS

6. Repeated START
7. Tx: Address, with R/nW bit set to 1 (read operation)
8. Switch to read mode, I2CS will begin transmitting data at this point
9. Rx: Current data byte (8 bits) from the mailbox data register N
10. Tx: ACK from I2C master
11. Rx: Current data byte (8 bits) from the mailbox data register (N+1)
12. Tx: ACK from I2C master
13. Rx: Current data byte (8 bits) from the mailbox data register (N+2)
14. Tx: ACK from I2C master
15. Rx: Current data byte (8 bits) from the mailbox data register (N+3)
16. Tx: NACK from I2C master
17. STOP

The end of the read is signalled to the I2CS with the STOP.

7.2.11 Registers (I2CM)

Address	Register	Access	Description	Reset By
0x40016000	I2CM0_FS_CLK_DIV	R/W	I2C Master 0 Full Speed SCL Clock Settings	Sys
0x4001600C	I2CM0_TIMEOUT	R/W	I2C Master 0 Timeout and Auto-Stop Settings	Sys
0x40016010	I2CM0_CTRL	R/W	I2C Master 0 Control Register	Sys
0x40016014	I2CM0_TRANS	***	I2C Master 0 Transaction Start and Status Flags	Sys
0x40016018	I2CM0_INTFL	W1C	I2C Master 0 Interrupt Flags	Sys
0x4001601C	I2CM0_INTEN	R/W	I2C Master 0 Interrupt Enable/Disable Controls	Sys
0x40016028	I2CM0_BB	***	I2C Master 0 Bit-Bang Control Register	Sys
0x40017000	I2CM1_FS_CLK_DIV	R/W	I2C Master 1 Full Speed SCL Clock Settings	Sys
0x4001700C	I2CM1_TIMEOUT	R/W	I2C Master 1 Timeout and Auto-Stop Settings	Sys
0x40017010	I2CM1_CTRL	R/W	I2C Master 1 Control Register	Sys
0x40017014	I2CM1_TRANS	***	I2C Master 1 Transaction Start and Status Flags	Sys
0x40017018	I2CM1_INTFL	W1C	I2C Master 1 Interrupt Flags	Sys
0x4001701C	I2CM1_INTEN	R/W	I2C Master 1 Interrupt Enable/Disable Controls	Sys
0x40017028	I2CM1_BB	***	I2C Master 1 Bit-Bang Control Register	Sys
0x40018000	I2CM2_FS_CLK_DIV	R/W	I2C Master 2 Full Speed SCL Clock Settings	Sys
0x4001800C	I2CM2_TIMEOUT	R/W	I2C Master 2 Timeout and Auto-Stop Settings	Sys
0x40018010	I2CM2_CTRL	R/W	I2C Master 2 Control Register	Sys
0x40018014	I2CM2_TRANS	***	I2C Master 2 Transaction Start and Status Flags	Sys
0x40018018	I2CM2_INTFL	W1C	I2C Master 2 Interrupt Flags	Sys
0x4001801C	I2CM2_INTEN	R/W	I2C Master 2 Interrupt Enable/Disable Controls	Sys
0x40018028	I2CM2_BB	***	I2C Master 2 Bit-Bang Control Register	Sys
0x40107000	I2CM0_FIFO_TRANS	R/W	I2C Master 0 Transaction FIFO	Sys

Address	Register	Access	Description	Reset By
0x40107800	I2CM0_FIFO_RSLTS	R/W	I2C Master 0 Results FIFO	Sys
0x40108000	I2CM1_FIFO_TRANS	R/W	I2C Master 1 Transaction FIFO	Sys
0x40108800	I2CM1_FIFO_RSLTS	R/W	I2C Master 1 Results FIFO	Sys
0x40109000	I2CM2_FIFO_TRANS	R/W	I2C Master 2 Transaction FIFO	Sys
0x40109800	I2CM2_FIFO_RSLTS	R/W	I2C Master 2 Results FIFO	Sys

7.2.11.1 I2CMn_FS_CLK_DIV

I2CMn_FS_CLK_DIV.fs_filter_clk_div

Field	Bits	Sys Reset	Access	Description
fs_filter_clk_div	7:0	00000001b	R/W	Full Speed Filter Clock Divisor

Filter frequency = (I2C Master Module Clock)/fs_filter_clk_div. Setting this value to 1 will disable the I2C Master.

I2CMn_FS_CLK_DIV.fs_scl_lo_cnt

Field	Bits	Sys Reset	Access	Description
fs_scl_lo_cnt	19:8	12'b0	R/W	Full Speed SCL Low Count

Number of clocks to hold SCL low for clock output

I2CMn_FS_CLK_DIV.fs_scl_hi_cnt

Field	Bits	Sys Reset	Access	Description
fs_scl_hi_cnt	31:20	12'b0	R/W	Full Speed SCL High Count

Number of clocks to hold SCL high for clock output

7.2.11.2 I2CMn_TIMEOUT

I2CMn_TIMEOUT.tx_timeout

Field	Bits	Sys Reset	Access	Description
tx_timeout	23:16	8'b0	R/W	Transaction Timeout Limit

Timeout limit (in mS) for a given transaction; when this timeout expires, master releases the bus and triggers an interrupt.

I2CMn_TIMEOUT.auto_stop_en

Field	Bits	Sys Reset	Access	Description
auto_stop_en	24	0	R/W	Auto-Stop Enable

If 1, master automatically issues a Stop when a timeout or unexpected Nack occurs.

7.2.11.3 I2CMn_CTRL

I2CMn_CTRL.tx_fifo_en

Field	Bits	Sys Reset	Access	Description
tx_fifo_en	2	0	R/W	Master Transaction FIFO Enable

0:Disabled; 1:Enabled

I2CMn_CTRL.rx_fifo_en

Field	Bits	Sys Reset	Access	Description
rx_fifo_en	3	0	R/W	Master Results FIFO Enable

0:Disabled; 1:Enabled

I2CMn_CTRL.mstr_reset_en

Field	Bits	Sys Reset	Access	Description
mstr_reset_en	7	0	R/W	Master Reset

1:Held in reset

7.2.11.4 I2CMn_TRANS

I2CMn_TRANS.tx_start

Field	Bits	Sys Reset	Access	Description
tx_start	0	0	R/W	Start Transaction

Write to 1 to begin a master transaction.

I2CMn_TRANS.tx_in_progress

Field	Bits	Sys Reset	Access	Description
tx_in_progress	1	0	R/O	Transaction In Progress

Set to 1 by hardware when a new transaction is started; cleared when transaction ends.

I2CMn_TRANS.tx_done

Field	Bits	Sys Reset	Access	Description
tx_done	2	0	R/O	Transaction Done

Set to 1 by hardware when a transaction completes; cleared to 0 on transaction Start.

I2CMn_TRANS.tx_nacked

Field	Bits	Sys Reset	Access	Description
tx_nacked	3	0	R/O	Transaction Nacked

Set to 1 when a transaction completes due to an unexpected NACK; cleared to 0 on Start.

I2CMn_TRANS.tx_lost_arbitr

Field	Bits	Sys Reset	Access	Description
tx_lost_arbitr	4	0	R/O	Transaction Lost Arbitration

Set to 1 when a transaction halts due to an arbitration failure; cleared to 0 on Start.

I2CMn_TRANS.tx_timeout

Field	Bits	Sys Reset	Access	Description
tx_timeout	5	0	R/O	Transaction Timed Out

Set to 1 when a transaction halts due to a timeout; cleared to 0 on Start.

7.2.11.5 I2CMn_INTFL

I2CMn_INTFL.tx_done

Field	Bits	Sys Reset	Access	Description
tx_done	0	0	W1C	Transaction Done Int Status

Write 1 to clear.

Set to 1 by hardware when a transaction completes.

I2CMn_INTFL.tx_nacked

Field	Bits	Sys Reset	Access	Description
tx_nacked	1	0	W1C	Transaction NACKed Int Status

Write 1 to clear.

Set to 1 by hardware when a transaction is interrupted due to an unexpected NACK response.

I2CMn_INTFL.tx_lost_arbitr

Field	Bits	Sys Reset	Access	Description
tx_lost_arbitr	2	0	W1C	Transaction Lost Arbitration Int Status

Write 1 to clear.

Set to 1 by hardware when a transaction is interrupted due to a lost arbitration failure.

I2CMn_INTFL.tx_timeout

Field	Bits	Sys Reset	Access	Description
tx_timeout	3	0	W1C	Transaction Timed Out Int Status

Write 1 to clear.

Set to 1 by hardware when a transaction times out.

I2CMn_INTFL.tx_fifo_empty

Field	Bits	Sys Reset	Access	Description
tx_fifo_empty	4	0	W1C	Transaction FIFO Empty Int Status

Write 1 to clear.

Set to 1 by hardware when the transaction FIFO is empty.

I2CMn_INTFL.tx_fifo_3q_empty

Field	Bits	Sys Reset	Access	Description
tx_fifo_3q_empty	5	0	W1C	Transaction FIFO 3Q Empty Int Status

Write 1 to clear.

Set to 1 by hardware when the transaction FIFO is three-quarters empty.

I2CMn_INTFL.rx_fifo_not_empty

Field	Bits	Sys Reset	Access	Description
rx_fifo_not_empty	6	0	W1C	Results FIFO Not Empty Int Status

Write 1 to clear.

Set to 1 by hardware when the results FIFO is not empty.

I2CMn_INTFL.rx_fifo_2q_full

Field	Bits	Sys Reset	Access	Description
rx_fifo_2q_full	7	0	W1C	Results FIFO 2Q Full Int Status

Write 1 to clear.

Set to 1 by hardware when the results FIFO is half full (two quarters).

I2CMn_INTFL.rx_fifo_3q_full

Field	Bits	Sys Reset	Access	Description
rx_fifo_3q_full	8	0	W1C	Results FIFO 3Q Full Int Status

Write 1 to clear.

Set to 1 by hardware when the results FIFO is three-quarters full.

I2CMn_INTFL.rx_fifo_full

Field	Bits	Sys Reset	Access	Description
rx_fifo_full	9	0	W1C	Results FIFO Full Int Status

Write 1 to clear.

Set to 1 by hardware when the results FIFO is completely full.

7.2.11.6 I2CMn_INTEN

I2CMn_INTEN.tx_done

Field	Bits	Sys Reset	Access	Description
tx_done	0	0	R/W	Transaction Done Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.tx_nacked

Field	Bits	Sys Reset	Access	Description
tx_nacked	1	0	R/W	Transaction NACKed Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.tx_lost_arbitr

Field	Bits	Sys Reset	Access	Description
tx_lost_arbitr	2	0	R/W	Transaction Lost Arbitration IntEnable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.tx_timeout

Field	Bits	Sys Reset	Access	Description
tx_timeout	3	0	R/W	Transaction Timed Out Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.tx_fifo_empty

Field	Bits	Sys Reset	Access	Description
tx_fifo_empty	4	0	R/W	Transaction FIFO Empty Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.tx_fifo_3q_empty

Field	Bits	Sys Reset	Access	Description
tx_fifo_3q_empty	5	0	R/W	Transaction FIFO 3Q Empty Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.rx_fifo_not_empty

Field	Bits	Sys Reset	Access	Description
rx_fifo_not_empty	6	0	R/W	Results FIFO Not Empty Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.rx_fifo_2q_full

Field	Bits	Sys Reset	Access	Description
rx_fifo_2q_full	7	0	R/W	Results FIFO 2Q Full Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.rx_fifo_3q_full

Field	Bits	Sys Reset	Access	Description
rx_fifo_3q_full	8	0	R/W	Results FIFO 3Q Full Int Enable

0:Associated int disabled; 1:Interrupt enabled

I2CMn_INTEN.rx_fifo_full

Field	Bits	Sys Reset	Access	Description
rx_fifo_full	9	0	R/W	Results FIFO Full Int Enable

0:Associated int disabled; 1:Interrupt enabled

7.2.11.7 I2CMn_BB

I2CMn_BB.bb_scl_out

Field	Bits	Sys Reset	Access	Description
bb_scl_out	0	1	R/W	Bit Bang SCL Output

I2CMn_BB.bb_sda_out

Field	Bits	Sys Reset	Access	Description
bb_sda_out	1	1	R/W	Bit Bang SDA Output

I2CMn_BB.bb_scl_in_val

Field	Bits	Sys Reset	Access	Description
bb_scl_in_val	2	s	R/O	Bit Bang SCL Input Value

I2CMn_BB.bb_sda_in_val

Field	Bits	Sys Reset	Access	Description
bb_sda_in_val	3	s	R/O	Bit Bang SCL Input Value

I2CMn_BB.rx_fifo_cnt

Field	Bits	Sys Reset	Access	Description
rx_fifo_cnt	20:16	s	R/O	Results FIFO Data Received Count

7.2.11.8 I2CMn_FIFO_TRANS

I2CM0_FIFO_TRANS

Sys Reset	Access	Description
n/a	R/W	I2C Master 0 Transaction FIFO

Writes to this space result in pushes to the I2C Master Transaction FIFO.

Reads from this space return the FIFO full flag in bit 0, and all other bits are 0.

7.2.11.9 I2CMn_FIFO_RSLTS

I2CM0_FIFO_RSLTS

Sys Reset	Access	Description
n/a	R/W	I2C Master 0 Results FIFO

Reads from this space return the next value which is pulled from the Results FIFO.

Writes to this space are ignored.

7.2.12 Registers (I2CS)

Address	Register	Access	Description	Reset By
0x40019000	I2CS_CLK_DIV	R/W	I2C Slave Clock Divisor Control	Sys
0x40019004	I2CS_DEV_ID	R/W	I2C Slave Device ID Register	Sys
0x40019008	I2CS_INTFL	W1C	I2CS Interrupt Flags	Sys
0x4001900C	I2CS_INTEN	R/W	I2CS Interrupt Enable/Disable Controls	Sys
0x40019010	I2CS_DATA_BYTE	***	I2CS Data Byte	Sys

7.2.12.1 I2CS_CLK_DIV

I2CS_CLK_DIV.fs_filter_clock_div

Field	Bits	Sys Reset	Access	Description
fs_filter_clock_div	7:0	0	R/W	FS Filter Clock Divisor

- 1: I2CS interface is disabled.
- 2..255: Filter frequency is set to (I2CS clock)/(field value), where the I2CS clock is determined by SYS_CLK_CTRL_10_I2CS.
- 0: Filter frequency is set to (I2CS clock)/256, where the I2CS clock is determined by SYS_CLK_CTRL_10_I2CS.

7.2.12.2 I2CS_DEV_ID

I2CS_DEV_ID.slave_dev_id

Field	Bits	Sys Reset	Access	Description
slave_dev_id	9:0	0	R/W	Slave Device ID

This field determines the slave device ID that will be used by the I2CS interface to compare against incoming messages on the I2C bus. If 10-bit ID mode is enabled, then all 10 bits of this field will be used as the device address; otherwise, only the low 7 bits will be used.

I2CS_DEV_ID.ten_bit_id_mode

Field	Bits	Sys Reset	Access	Description
ten_bit_id_mode	12	0	R/W	10-bit ID Mode

- 0: The I2CS interface will operate in 7-bit addressing mode (default).
- 1: The I2CS interface will operate in 10-bit addressing mode.

I2CS_DEV_ID.slave_reset

Field	Bits	Sys Reset	Access	Description
slave_reset	14	0	R/W	Slave Reset

Writing this bit to 1 will cause the I2CS engine to be held in a reset state indefinitely. The I2CS engine will resume normal operation once this bit is written to 0 again. Setting this bit to 1 does not affect the contents of the I2CS module registers.

7.2.12.3 I2CS_INTFL

I2CS_INTFL.byte0

Field	Bits	Sys Reset	Access	Description
byte0	0	0	W1C	Updated Byte 0

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte1

Field	Bits	Sys Reset	Access	Description
byte1	1	0	W1C	Updated Byte 1

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte2

Field	Bits	Sys Reset	Access	Description
byte2	2	0	W1C	Updated Byte 2

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte3

Field	Bits	Sys Reset	Access	Description
byte3	3	0	W1C	Updated Byte 3

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte4

Field	Bits	Sys Reset	Access	Description
byte4	4	0	W1C	Updated Byte 4

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte5

Field	Bits	Sys Reset	Access	Description
byte5	5	0	W1C	Updated Byte 5

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte6

Field	Bits	Sys Reset	Access	Description
byte6	6	0	W1C	Updated Byte 6

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte7

Field	Bits	Sys Reset	Access	Description
byte7	7	0	W1C	Updated Byte 7

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte8

Field	Bits	Sys Reset	Access	Description
byte8	8	0	W1C	Updated Byte 8

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte9

Field	Bits	Sys Reset	Access	Description
byte9	9	0	W1C	Updated Byte 9

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte10

Field	Bits	Sys Reset	Access	Description
byte10	10	0	W1C	Updated Byte 10

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte11

Field	Bits	Sys Reset	Access	Description
byte11	11	0	W1C	Updated Byte 11

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte12

Field	Bits	Sys Reset	Access	Description
byte12	12	0	W1C	Updated Byte 12

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte13

Field	Bits	Sys Reset	Access	Description
byte13	13	0	W1C	Updated Byte 13

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte14

Field	Bits	Sys Reset	Access	Description
byte14	14	0	W1C	Updated Byte 14

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte15

Field	Bits	Sys Reset	Access	Description
byte15	15	0	W1C	Updated Byte 15

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte16

Field	Bits	Sys Reset	Access	Description
byte16	16	0	W1C	Updated Byte 16

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte17

Field	Bits	Sys Reset	Access	Description
byte17	17	0	W1C	Updated Byte 17

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte18

Field	Bits	Sys Reset	Access	Description
byte18	18	0	W1C	Updated Byte 18

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte19

Field	Bits	Sys Reset	Access	Description
byte19	19	0	W1C	Updated Byte 19

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte20

Field	Bits	Sys Reset	Access	Description
byte20	20	0	W1C	Updated Byte 20

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte21

Field	Bits	Sys Reset	Access	Description
byte21	21	0	W1C	Updated Byte 21

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte22

Field	Bits	Sys Reset	Access	Description
byte22	22	0	W1C	Updated Byte 22

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte23

Field	Bits	Sys Reset	Access	Description
byte23	23	0	W1C	Updated Byte 23

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte24

Field	Bits	Sys Reset	Access	Description
byte24	24	0	W1C	Updated Byte 24

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte25

Field	Bits	Sys Reset	Access	Description
byte25	25	0	W1C	Updated Byte 25

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte26

Field	Bits	Sys Reset	Access	Description
byte26	26	0	W1C	Updated Byte 26

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte27

Field	Bits	Sys Reset	Access	Description
byte27	27	0	W1C	Updated Byte 27

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte28

Field	Bits	Sys Reset	Access	Description
byte28	28	0	W1C	Updated Byte 28

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte29

Field	Bits	Sys Reset	Access	Description
byte29	29	0	W1C	Updated Byte 29

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte30

Field	Bits	Sys Reset	Access	Description
byte30	30	0	W1C	Updated Byte 30

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

I2CS_INTFL.byte31

Field	Bits	Sys Reset	Access	Description
byte31	31	0	W1C	Updated Byte 31

Set to 1 by hardware when the associated data byte is written by an external I2C bus master.

7.2.12.4 I2CS_INTEN

I2CS_INTEN.byte0

Field	Bits	Sys Reset	Access	Description
byte0	0	0	R/W	Updated Byte 0

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte1

Field	Bits	Sys Reset	Access	Description
byte1	1	0	R/W	Updated Byte 1

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte2

Field	Bits	Sys Reset	Access	Description
byte2	2	0	R/W	Updated Byte 2

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte3

Field	Bits	Sys Reset	Access	Description
byte3	3	0	R/W	Updated Byte 3

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte4

Field	Bits	Sys Reset	Access	Description
byte4	4	0	R/W	Updated Byte 4

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte5

Field	Bits	Sys Reset	Access	Description
byte5	5	0	R/W	Updated Byte 5

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte6

Field	Bits	Sys Reset	Access	Description
byte6	6	0	R/W	Updated Byte 6

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte7

Field	Bits	Sys Reset	Access	Description
byte7	7	0	R/W	Updated Byte 7

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte8

Field	Bits	Sys Reset	Access	Description
byte8	8	0	R/W	Updated Byte 8

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte9

Field	Bits	Sys Reset	Access	Description
byte9	9	0	R/W	Updated Byte 9

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte10

Field	Bits	Sys Reset	Access	Description
byte10	10	0	R/W	Updated Byte 10

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte11

Field	Bits	Sys Reset	Access	Description
byte11	11	0	R/W	Updated Byte 11

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte12

Field	Bits	Sys Reset	Access	Description
byte12	12	0	R/W	Updated Byte 12

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte13

Field	Bits	Sys Reset	Access	Description
byte13	13	0	R/W	Updated Byte 13

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte14

Field	Bits	Sys Reset	Access	Description
byte14	14	0	R/W	Updated Byte 14

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte15

Field	Bits	Sys Reset	Access	Description
byte15	15	0	R/W	Updated Byte 15

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte16

Field	Bits	Sys Reset	Access	Description
byte16	16	0	R/W	Updated Byte 16

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte17

Field	Bits	Sys Reset	Access	Description
byte17	17	0	R/W	Updated Byte 17

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte18

Field	Bits	Sys Reset	Access	Description
byte18	18	0	R/W	Updated Byte 18

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte19

Field	Bits	Sys Reset	Access	Description
byte19	19	0	R/W	Updated Byte 19

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte20

Field	Bits	Sys Reset	Access	Description
byte20	20	0	R/W	Updated Byte 20

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte21

Field	Bits	Sys Reset	Access	Description
byte21	21	0	R/W	Updated Byte 21

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte22

Field	Bits	Sys Reset	Access	Description
byte22	22	0	R/W	Updated Byte 22

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte23

Field	Bits	Sys Reset	Access	Description
byte23	23	0	R/W	Updated Byte 23

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte24

Field	Bits	Sys Reset	Access	Description
byte24	24	0	R/W	Updated Byte 24

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte25

Field	Bits	Sys Reset	Access	Description
byte25	25	0	R/W	Updated Byte 25

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte26

Field	Bits	Sys Reset	Access	Description
byte26	26	0	R/W	Updated Byte 26

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte27

Field	Bits	Sys Reset	Access	Description
byte27	27	0	R/W	Updated Byte 27

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte28

Field	Bits	Sys Reset	Access	Description
byte28	28	0	R/W	Updated Byte 28

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte29

Field	Bits	Sys Reset	Access	Description
byte29	29	0	R/W	Updated Byte 29

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte30

Field	Bits	Sys Reset	Access	Description
byte30	30	0	R/W	Updated Byte 30

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

I2CS_INTEN.byte31

Field	Bits	Sys Reset	Access	Description
byte31	31	0	R/W	Updated Byte 31

- 0: Interrupt disabled (default)
- 1: An interrupt will be triggered by the I2CS module when the associated Updated Byte N interrupt flag is set.

7.2.12.5 I2CS_DATA_BYTE

I2CS_DATA_BYTE.data_field

Field	Bits	Sys Reset	Access	Description
data_field	7:0	00h	R/W	Data Field

This field contains the 8-bit data byte value which can be read from or written to either internally (over the APB bus) or externally (by an I2C bus master).

I2CS_DATA_BYTE.read_only_fl

Field	Bits	Sys Reset	Access	Description
read_only_fl	8	0	R/W	Read Only Flag

This flag affects only access to the 8-bit data byte from the external I2C master side; it is always possible to read and write the data byte value internally using the APB bus.

- 0: External master has read/write access to the 8-bit data field (default).
- 1: External master has read-only access to the 8-bit data field. An attempt by the external master to write to the data field will result in a NACK response from the I2CS.

I2CS_DATA_BYTE.data_updated_fl

Field	Bits	Sys Reset	Access	Description
data_updated_fl	9	0	R/O	Byte Updated Flag

This flag is set to 1 by hardware when the external I2C bus master writes a new value to the 8-bit data byte value field. Reading this register (over the APB bus) will clear this flag to zero.

7.3 SPI

7.3.1 Overview

The serial peripheral interface (SPI) module of the **MAX32620** microcontroller provides a highly configurable, flexible, and efficient interface to communicate with a wide variety of SPI slave devices. The integrated SPI controllers provide an independent master-mode-only serial communication channel that communicates with peripheral devices in a single- or multiple-slave system. Up to three separate SPI interfaces are available for general use, with the third SPI instance reserved for future Bluetooth module communication.

Note The number of SPI instances available is contingent upon the application usage of other peripherals and GPIO pins.

The three SPI Master ports (SPI0, SPI1, and SPI2) available on the **MAX32620** each support the following features:

- Support of all four SPI modes (0, 1, 2, and 3) for single-bit communication
- 3- or 4-wire mode for single-bit slave device communication
- Full-duplex operation in single-bit, 4-wire mode
- Dual and Quad I/O supported
- Up to five slave select (SS) lines per port with programmable polarity
- Up to two slave ready (SR) lines with programmable polarity
- Programmable interface timing
- Programmable SCK frequency and duty cycle
- Programmable SCK alternate timing
- SS assertion and deassertion timing with respect to leading/trailing SCK edge

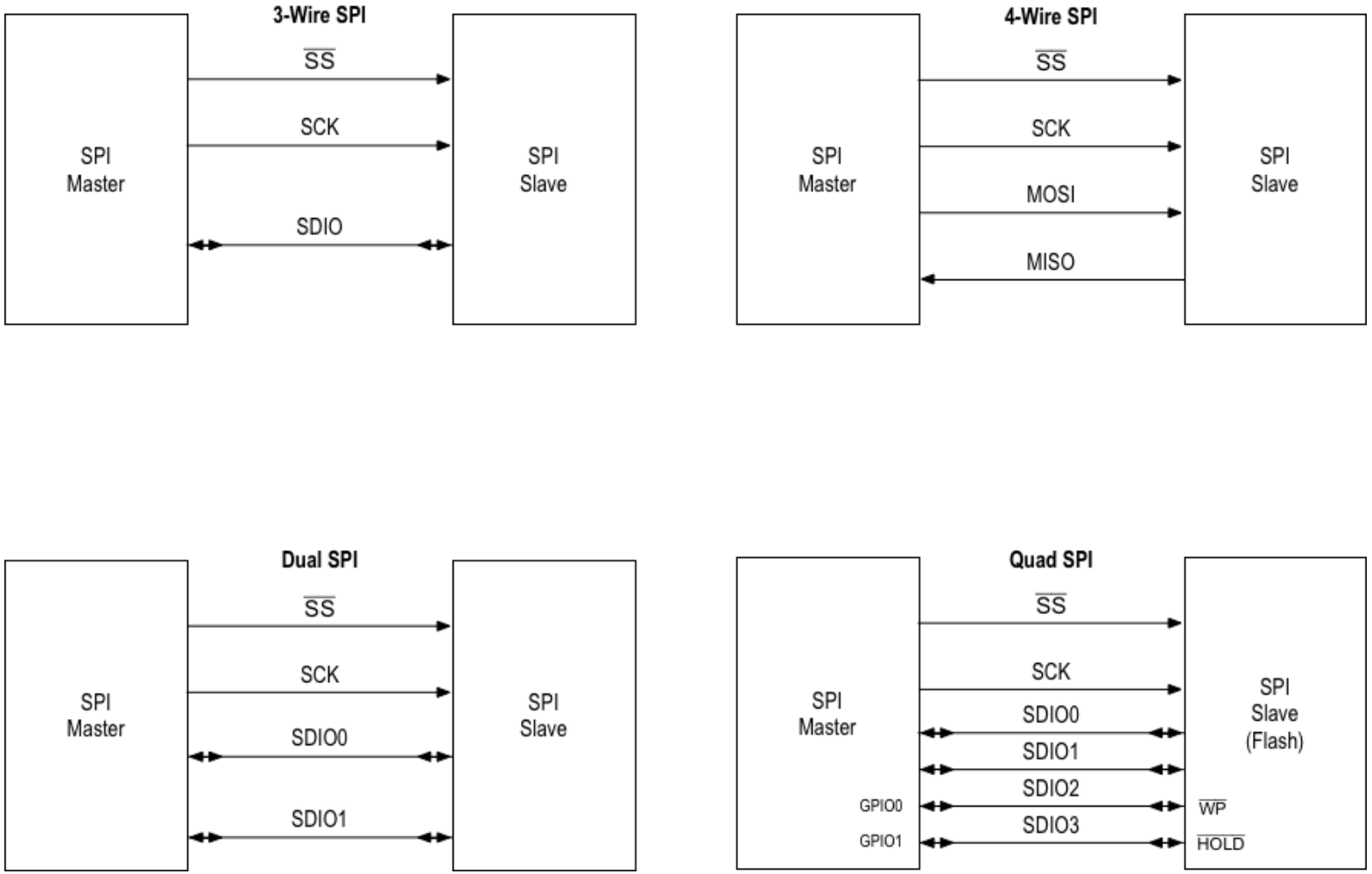

Figure 7.13: Multi I/O SPI Support

7.3.2 SPI Port and Pin Configurations

SPI Wiring Configurations

- **3-Wire SPI**: SS, SCK, SDIO
- **4-Wire SPI**: SS, SCK, MOSI, MISO
- **Dual SPI**: SS, SCK, SDIO0, SDIO1
- **Quad SPI**: SS, SCK, SDIO0, SDIO1, SDIO2, SDIO3

Slave Ready (SR) lines are optional; configuration details can be found in the Static Configuration section.

Note See Pin Layout for a detailed mapping of **MAX32620** multiplexed function locations. Functional priority distinction is included in the mapping.

7.3.2.1 Pin Layout Configuration

The tables below contain the available pin configurations for each of the SPI Master ports (SPI0, SPI1, SPI2):

SPI0

Logic Signal	Port and Pin
SS	P0.7(0), P4.4(1), P4.5(2), P4.6(3), P4.7(4)
SCK	P0.4
SDIO	P4.2(2), P.4.3(3)
SDIO (MOSI)	P0.5(0)
SDIO (MISO)	P0.6(1)

Optional: Slave Ready

Logic Signal	Port and Pin
SR	N/A

SPI1

Logic Signal	Port and Pin
SS	P1.3(0), P3.6(1), P3.7(2)
SCK	P1.0
SDIO	P1.4(2), P1.5(3)
SDIO (MOSI)	P1.1(0)
SDIO (MISO)	P1.2(1)

Optional: Slave Ready

Logic Signal	Port and Pin
SR	N/A

SPI2

Logic Signal	Port and Pin
SS	A) P2.7(0), P3.4(1), P3.5(2) B) P5.3(0), P5.7(1), P6.0(2)
SCK	A) P2.4 B) P5.0
SDIO	A) N/A B) P5.4(2), P5.5(3)
SDIO (MOSI)	A) P2.5(0) B) P5.1(0)
SDIO (MISO)	A) P2.6(1) B) P5.2(1)

Optional: Slave Ready

Logic Signal	Port and Pin
SR	A) P4.0(0), P4.1(1) B) P5.6(0)

7.3.3 Clock Selection and Configuration

The **MAX32620** supports programmable SPI clock rates, which are a divisor of the system clock. Each of the three SPI ports is able to set its clock rate independently. To set the base clock rate, write to the appropriate Clock Control register with the value desired to achieve the ideal clock rate for the slave devices.

SPI Clock Rate Configuration Registers

Port	SPI Clock Configuration Register
SPI0	CLKMAN_SYS_CLK_CTRL_11_SPI0.spi0_clk_scale
SPI1	CLKMAN_SYS_CLK_CTRL_12_SPI1.spi1_clk_scale
SPI2	CLKMAN_SYS_CLK_CTRL_13_SPI2.spi2_clk_scale

Setting the SPI Clock Rate - `REGISTER CLKMAN_SYS_CLK_CTRL_x_SPIy`

[3:0]	SPI Clock Rate CLKMAN_SYS_CLK_CTRL_x_SPIy
0000b	Disabled
0001b	(System Clock Source / 1)
0010b	(System Clock Source / 2)
0011b	(System Clock Source / 4)
0100b	(System Clock Source / 8)
0101b	(System Clock Source / 16)

[3:0]	SPI Clock Rate CLKMAN_SYS_CLK_CTRL_x_SPIy
0110b	(System Clock Source / 32)
0111b	(System Clock Source / 64)
1000b	(System Clock Source / 128)
1001b	(System Clock Source / 256)
other	(System Clock Source / 1)

7.3.4 Clock Gating

Clock gating for the SPI ports is controlled by the register fields CLKMAN_CLK_GATE_CTRL2.spi0_clk_gater, CLKMAN_CLK_GATE_CTRL2.spi1_clk_gater, and CLKMAN_CLK_GATE_CTRL2.spi2_clk_gater. The table below shows the supported settings for the `spi[n]_clk_gater` register fields and their meanings.

SPI[n] Clock Control Value (2b)	Setting
00b	Clock off - SPI[n] disabled
01b	Dynamic Clock Gating Enabled - SPI[n] clock active only when used
10b or 11b	Clock on - SPI[n] enabled at all times

7.3.5 Configuration Modes Overview

Once the main SPI clock is set up for the port, the remainder of the configuration and operation for SPI is mapped into three categories:

- *Static Configuration:* Performed during SPI initial setup and/or when SPI is not active.
  - SPIn_SS_SR_POLARITY
  - SPIn_GEN_CTRL
- *Dynamic Configuration:* Configuration required to communicate with a specific slave device, which may take place while the SPI port is active.
  - SPIn_MSTR_CFG
- *Interrupt Servicing:* Status and Control used by an application either directly or via the Peripheral Management Unit's DMA to efficiently service SPI data transfer.
  - SPIn_FIFO_CTRL
  - SPIn_INTFL
  - SPIn_INTEN

7.3.5.1 Static Configuration

Static configuration should be performed while the SPI port is disabled. Static configuration includes:

- Slave Select signal polarity
- Slave Ready (Flow Control) signal polarity

Slave select polarity is independently configurable for each slave select line for a given SPI port. To set the Slave Select 0 for the SPI port to an Active High State, set the ss_polarity field to 0000001b in the register SPIn_SS_SR_POLARITY. To set additional slave selects for the same port, set the appropriate bit(s) in the ss_polarity field to match the slave select line. By default, the slave selects are set to Active Low.

For the slave ready polarity, each slave ready input signal for a given SPI port is configured in the same manner as the Slave Select. Set the fc_polarity bit in the SPIn_SS_SR_POLARITY register to either a 1 or a 0.

7.3.5.2 Dynamic Configuration

To begin communicating with a given slave device it is necessary to set up several parameters specific to that slave. All of these settings are controlled by the SPIn_MSTR_CFG register.

The SPIn_MSTR_CFG.slave_sel field determines which slave select pin is active during the transaction. A total of five possibilities exist for each SPI Port as determined by the GPIO mapping and the user configuration and design.

If the slave device only supports 3-Wire mode, setting SPIn_MSTR_CFG.three_wire_mode to 1 puts the SPI Port into 3-Wire mode. For this mode, the corresponding output pin SDIO[0] is used for both MOSI and MISO.

7.3.5.3 SPI Mode Selection (Clock Polarity and Phase)

To select one of the four supported SPI Modes of operation, the SPIn_MSTR_CFG.spi_mode field is used. By default, SPI Mode 0 is selected, spi_mode = 00b. The selected SPI Mode affects the SPI clock state (active low or active high) and the clock edge on which the SPI master samples data (rising edge or falling edge).

SPI Modes

SPI Mode	spi_mode xxb	SPI Clock Polarity	SPI Sampling Clock Edge
0	00	Active High	Rising Edge
1	01	Active High	Falling Edge
2	10	Active Low	Falling edge
3	11	Active Low	Rising Edge

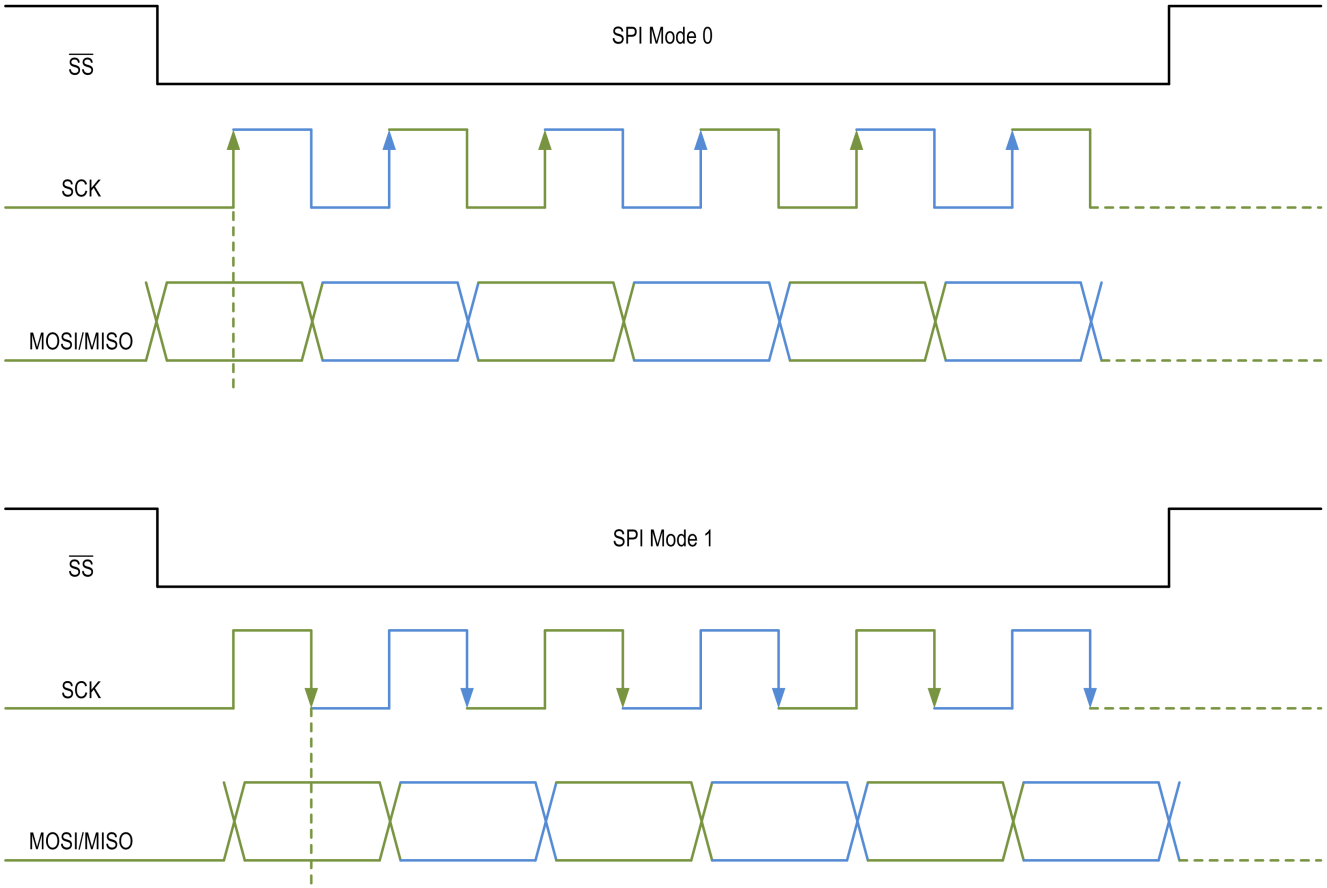

Figure 7.14: SPI Modes 0 and 1 Timing

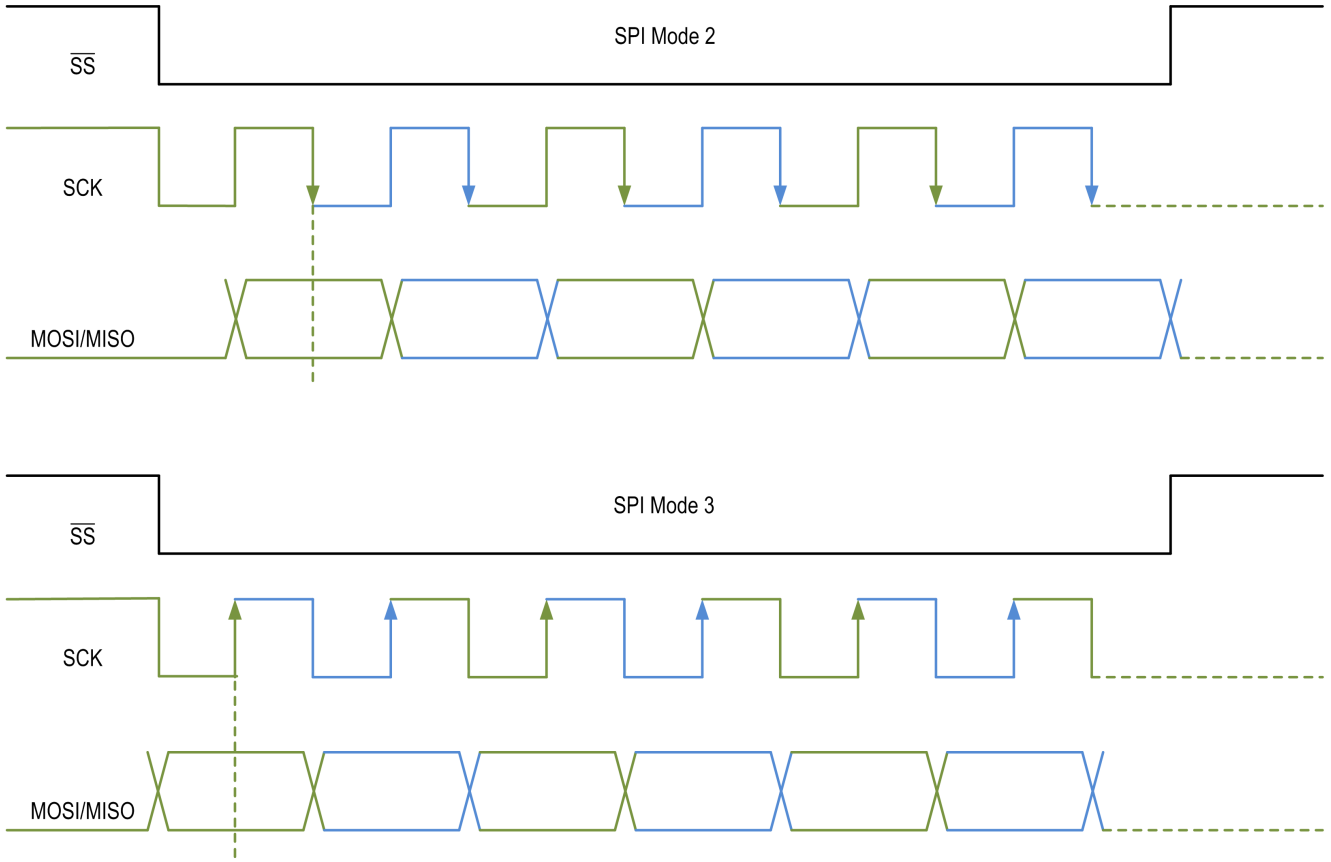

Figure 7.15: SPI Modes 2 and 3 Timing

7.3.5.4 Serial Clock

The number of system clocks SCK will be in either active or inactive states is set with the SPIn_MSTR_CFG.sck_hi_clk and SPIn_MSTR_CFG.sck_lo_clk register fields. These 4-bit fields define clock pulses which synchronize data transmission generated by the master. These settings are further determined by the clock

polarity settings.

Additional configuration options are supported and are detailed in the SPIn_MSTR_CFG register.

Alternate Timing Generation

Alternate timing generation is an optional method of data transmission synchronization. The fields SPIn_MSTR_CFG.alt_sck_hi_clk and SPIn_MSTR_CFG.alt_sck_lo_clk operate in the same manner as the SPIn_MSTR_CFG.sck_hi_clk and SPIn_MSTR_CFG.sck_lo_clk registers. These settings are asserted when alternate timing generation is enabled.

7.3.5.5 Transaction Delay

Various transaction delay behaviors are set using the SPIn_MSTR_CFG.act_delay and SPIn_MSTR_CFG.inact_delay register fields. The act_delay field controls the delay from automatic assertion of slave select lines to the beginning of the SCK clock pulses as well as the delay from the end of the clock pulses until automatic deassertion of SS. Conversely, inact_delay controls the delay between deassertion of SS at the end of a transaction and reassertion of SS to begin the next transaction; this control essentially regulates the delay period between back-to-back SPI transactions.

act_delay	Delay Time
00b	0 system clocks
01b	2 system clocks
10b	4 system clocks
11b	8 system clocks

inact_delay	Delay Time
00b	0 system clocks
01b	2 system clocks
10b	4 system clocks
11b	8 system clocks

7.3.5.6 Page Size

The SPIn_MSTR_CFG.page_size field is used to define the number of bytes per page for transactions defining their length in numbers of pages. The following table shows the writes for the various page sizes.

page_size	Bytes Per Page
00b	4 (Default)
01b	8
10b	16
11b	4

7.3.6 Communication and Data Transfer

Once the SPI has been configured to communicate with a specific slave, SPI transactions are initiated by writing to the SPI Transaction FIFO mapped into the AHB system address map. See SPI: FIFOs for transmit FIFO details for each SPI port. Prior to initiating a transaction, both the TX and RX SPI FIFOs need to be enabled. To enable the Transmit FIFO, set register field SPIn_GEN_CTRL.tx_fifo_en for the specific SPI peripheral being used. The Receive FIFO is enabled by setting SPIn_GEN_CTRL.rx_fifo_en to 1.

The FIFO is 2 bytes wide and expects a 16-bit header followed by an optional payload padded out to a 16-bit boundary up to a total of 32 bytes.

If the transaction generates results data, this data is pushed onto the SPI Results FIFO mapped into the AHB system address map. See (SPI: FIFOs) for receive FIFO details for each SPI port. This FIFO is 8 bits wide and will zero-pad to a byte boundary at the completion of a SPI transaction.

SPI Transaction Header Type 0

The format of the standard transaction header (with bits [15:14] set to 00b) is as follows:

Bit	Mnemonic	Description
[1:0]	Direction	Direction of information transfer with respect to the Master. • 0 = None • 1 = TX • 2 = RX • 3 = Both For headers which do not define a transmission (i.e., Direction = None or Rx), no payload is required. Conversely, headers which do not define a reception (i.e., Direction = None or Tx), result in no data being pushed onto the results FIFO.

Bit	Mnemonic	Description
[3:2]	Size Units	Units used to interpret the size field. • 0 = Bits • 1 = Bytes • 2 = Pages (See SPIn_MSTR_CFG.page_size for page size definition)
[8:4]	Size	Size of transaction in terms of units. 1 = 1, 2 = 2, ... *Note 0 = 32 units*
[10:9]	Width	Number of SDIO I/O to use for the transaction. This has no effect on the size of the transaction, just the time required to complete it. • *Note:* • *0 = 1-bit wide* • *1 = 2-bits wide* • *2 = 4-bits wide*
[11]	Alt Timing	RESERVED
[12]	Flow Control	When set to 1, use selected slave flow control input to moderate traffic movement.
[13]	Deassert SS	When set to 1, deassert selected slave select at the completion of this transaction.
[15:14]	Header Type	Type 0 (00b)

SPI Transaction Header Type 1

If bits [15:14] of the transaction header are set to 1 (01b), then instead of starting a transaction, the header is used to set bits [13:0] in the SPIn_MSTR_CFG register as defined below.

Field	Description
[2:0]	Written to SPIn_MSTR_CFG.slave_sel
[3]	Written to SPIn_MSTR_CFG.three_wire_mode
[5:4]	Written to SPIn_MSTR_CFG.spi_mode
[7:6]	Written to SPIn_MSTR_CFG.page_size
[11:8]	Written to SPIn_MSTR_CFG.sck_hi_clk
[13:12]	Written to SPIn_MSTR_CFG.sck_lo_clk bits [1:0] only
[15:14]	Header Type 1 (01b)

SPI Transaction Header Type 2

If bits [15:14] of the transaction header are set to 2 (10b), then instead of starting a transaction, the header is used to set bits [27:14] in the SPIn_MSTR_CFG register as defined below.

Field	Description
[1:0]	Written to SPIn_MSTR_CFG.sck_lo_clk bits [3:2] only
[3:2]	Written to SPIn_MSTR_CFG.act_delay
[5:4]	Written to SPIn_MSTR_CFG.inact_delay
[9:6]	Written to SPIn_MSTR_CFG.alt_sck_hi_clk
[13:10]	Written to SPIn_MSTR_CFG.alt_sck_lo_clk
[15:14]	Header Type 2 (10b)

7.3.7 Interrupts

Interrupt logic is provided to allow efficient servicing of the SPI Master function by firmware or the PMU engine. Interrupts may be grouped into two categories:

- Keeping the transaction FIFO full.
- Keeping the results FIFO empty. Programmable levels in the FIFO allow interrupt events to be issued if the transaction FIFO falls below a certain level, or if the results FIFO fills above a certain level. See SPIn_FIFO_CTRL for details.

7.3.8 SPI: FIFOs

SPI Master FIFO AHB Address Ranges

FIFO Write Points for Transaction Setup

SPI[n] FIFO	Start Address	End Address
SPI0_FIFO_TRANS	0x4010_0000	0x4010_07FE
SPI1_FIFO_TRANS	0x4010_1000	0x4010_17FE
SPI2_FIFO_TRANS	0x4010_2000	0x4010_27FE

Writes to this space result in pushes to the SPI Master Transaction FIFO. This space supports single accesses as well as burst accesses. Access widths of 8-bit, 16-bit, and 32-bit are supported. Reads from this space always return zeros. The SPI Master Transaction FIFO is 2 bytes wide. 16-bit writes result in a single push to the FIFO. 32-bit writes result in two FIFO pushes, the LSW is pushed first, followed by the MSW. 8-bit writes must occur in even/odd byte address pairs. The odd byte address data is held by the slave until the even byte address data is written, at which point 2 bytes are pushed to the FIFO.

FIFO Read Points for Results Data

SPI[n] FIFO	Start Address	End Address
SPI0_FIFO_RSLTS	0x4010_0800	0x4010_0FFF
SPI1_FIFO_RSLTS	0x4010_1800	0x4010_1FFF
SPI2_FIFO_RSLTS	0x4010_2800	0x4010_2FFF

Reads from this space pull data from the SPI Master Results FIFO. This space supports single accesses as well as burst accesses. Access widths of 8-bit, 16-bit, and 32-bit are supported. Writes to this space are ignored. The SPI Master Results FIFO is 1-bit wide. 8-bit reads result in a single pull from the FIFO. 16- bit reads result in two FIFO pulls, the LSB is pulled first, followed by the MSB. 32-bit reads result in four FIFO pulls, the LSB is pulled first, continuing until the MSB is pulled last.

7.3.9 Registers (SPI)

Address	Register	Access	Description	Reset By
0x4001A000	SPI0_MSTR_CFG	R/W	SPI Master 0 Configuration Register	Sys
0x4001A004	SPI0_SS_SR_POLARITY	R/W	SPI Master 0 Polarity Control for SS and SR Signals	Sys
0x4001A008	SPI0_GEN_CTRL	***	SPI Master 0 General Control Register	Sys
0x4001A00C	SPI0_FIFO_CTRL	***	SPI Master 0 FIFO Control Register	Sys
0x4001A010	SPI0_SPCL_CTRL	R/W	SPI Master 0 Special Mode Controls	Sys
0x4001A014	SPI0_INTFL	W1C	SPI Master 0 Interrupt Flags	Sys
0x4001A018	SPI0_INTEN	R/W	SPI Master 0 Interrupt Enable/Disable Settings	Sys
0x4001B000	SPI1_MSTR_CFG	R/W	SPI Master 1 Configuration Register	Sys
0x4001B004	SPI1_SS_SR_POLARITY	R/W	SPI Master 1 Polarity Control for SS and SR Signals	Sys
0x4001B008	SPI1_GEN_CTRL	***	SPI Master 1 General Control Register	Sys
0x4001B00C	SPI1_FIFO_CTRL	***	SPI Master 1 FIFO Control Register	Sys
0x4001B010	SPI1_SPCL_CTRL	R/W	SPI Master 1 Special Mode Controls	Sys
0x4001B014	SPI1_INTFL	W1C	SPI Master 1 Interrupt Flags	Sys
0x4001B018	SPI1_INTEN	R/W	SPI Master 1 Interrupt Enable/Disable Settings	Sys
0x4001C000	SPI2_MSTR_CFG	R/W	SPI Master 2 Configuration Register	Sys
0x4001C004	SPI2_SS_SR_POLARITY	R/W	SPI Master 2 Polarity Control for SS and SR Signals	Sys
0x4001C008	SPI2_GEN_CTRL	***	SPI Master 2 General Control Register	Sys
0x4001C00C	SPI2_FIFO_CTRL	***	SPI Master 2 FIFO Control Register	Sys
0x4001C010	SPI2_SPCL_CTRL	R/W	SPI Master 2 Special Mode Controls	Sys
0x4001C014	SPI2_INTFL	W1C	SPI Master 2 Interrupt Flags	Sys
0x4001C018	SPI2_INTEN	R/W	SPI Master 2 Interrupt Enable/Disable Settings	Sys
0x4010A000	SPI0_FIFO_TRANS	R/W	SPI Master FIFO Write Space for Transaction Setup	Sys

Address	Register	Access	Description	Reset By
0x4010A800	SPI0_FIFO_RSLTS	R/W	SPI Master FIFO Read Space for Results Data	Sys
0x4010B000	SPI1_FIFO_TRANS	R/W	SPI Master 1 FIFO Write Space for Transaction Setup	Sys
0x4010B800	SPI1_FIFO_RSLTS	R/W	SPI Master 1 FIFO Read Space for Results Data	Sys
0x4010C000	SPI2_FIFO_TRANS	R/W	SPI Master 2 FIFO Write Space for Transaction Setup	Sys
0x4010C800	SPI2_FIFO_RSLTS	R/W	SPI Master 2 FIFO Read Space for Results Data	Sys

7.3.9.1 SPIn_MSTR_CFG

SPIn_MSTR_CFG.slave_sel

Field	Bits	Sys Reset	Access	Description
slave_sel	2:0	000b	R/W	SPI Slave Select

Selects which SS slave select (out of those which are supported) will be asserted during a SPI transaction.

SPIn_MSTR_CFG.three_wire_mode

Field	Bits	Sys Reset	Access	Description
three_wire_mode	3	0	R/W	3-Wire Mode

- 0: Disabled
- 1: Enabled - SDIO[0] will serve as both MOSI and MISO (half duplex mode)

SPIn_MSTR_CFG.spi_mode

Field	Bits	Sys Reset	Access	Description
spi_mode	5:4	00b	R/W	SPI Mode

Defines Clock Polarity (bit 5) and Clock Phase (bit 4), collectively referred to as SPI Mode.

SPIn_MSTR_CFG.page_size

Field	Bits	Sys Reset	Access	Description
page_size	7:6	00b	R/W	Page Size

Defines number of bytes per page, for transactions that define their length in number of pages.

- 00b: 4 bytes per page
- 01b: 8 bytes per page
- 10b: 16 bytes per page
- 11b: 4 bytes per page

SPIn_MSTR_CFG.sck_hi_clk

Field	Bits	Sys Reset	Access	Description
sck_hi_clk	11:8	0000b	R/W	SCK High Clocks

Number of system clocks SCK will be in the active state (determined by clock polarity setting). If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the system clock.

SPIn_MSTR_CFG.sck_lo_clk

Field	Bits	Sys Reset	Access	Description
sck_lo_clk	15:12	0000b	R/W	SCK Low Clocks

Number of system clocks SCK will be in the INactive state (determined by clock polarity setting). If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the system clock.

SPIn_MSTR_CFG.act_delay

Field	Bits	Sys Reset	Access	Description
act_delay	17:16	00b	R/W	SS Active Timing

Controls the delay from automatic assertion of slave select (SS) to the beginning of the SCK clock pulses as well as the delay from the end of the clock pulses until the automatic deassertion of SS.

- 00b: 0 system clocks
- 01b: 2 system clocks
- 10b: 4 system clocks
- 11b: 8 system clocks

SPIn_MSTR_CFG.inact_delay

Field	Bits	Sys Reset	Access	Description
inact_delay	19:18	00b	R/W	SS Inactive Timing

Controls the delay between deassertion of SS at the end of a transaction and reassertion of SS to begin the next transaction, for back-to-back SPI transactions.

- 00b: 0 system clocks
- 01b: 2 system clocks
- 10b: 4 system clocks
- 11b: 8 system clocks

SPIn_MSTR_CFG.alt_sck_hi_clk

Field	Bits	Sys Reset	Access	Description
alt_sck_hi_clk	23:20	0000b	R/W	Alt SCK High Clocks

Number of system clocks SCK will be in the active state (determined by clock polarity setting), when alternate timing generation is enabled. If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the system clock.

SPIn_MSTR_CFG.alt_sck_lo_clk

Field	Bits	Sys Reset	Access	Description
alt_sck_lo_clk	27:24	0000b	R/W	Alt SCK Low Clocks

Number of system clocks SCK will be in the inactive state (determined by clock polarity setting), when alternate timing generation is enabled. If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the system clock.

7.3.9.2 SPIn_SS_SR_POLARITY

SPIn_SS_SR_POLARITY.ss_polarity

Field	Bits	Sys Reset	Access	Description
ss_polarity	7:0	00h	R/W	SS Signal Polarity

Defines polarity of each implemented SS slave select signal, where 0=active low, 1=active high.

SPIn_SS_SR_POLARITY.fc_polarity

Field	Bits	Sys Reset	Access	Description
fc_polarity	15:8	00h	R/W	SR Signal Polarity [FC Polarity]

Defines polarity of each implemented SR flow control signal, where 0=active low, 1=active high.

7.3.9.3 SPIn_GEN_CTRL

SPIn_GEN_CTRL.spi_mstr_en

Field	Bits	Sys Reset	Access	Description
spi_mstr_en	0	0	R/W	Enable/Disable SPI Master

- 0: SPI is disabled and forced to reset state
- 1: SPI is enabled.

SPIn_GEN_CTRL.tx_fifo_en

Field	Bits	Sys Reset	Access	Description
tx_fifo_en	1	0	R/W	Transaction FIFO Enable

- 0: Transaction FIFO is disabled/forced to reset
- 1: Transaction FIFO is enabled

SPIn_GEN_CTRL.rx_fifo_en

Field	Bits	Sys Reset	Access	Description
rx_fifo_en	2	0	R/W	Results FIFO Enable

- 0: Results FIFO is disabled/forced to reset
- 1: Results FIFO is enabled

SPIn_GEN_CTRL.bit_bang_mode

Field	Bits	Sys Reset	Access	Description
bit_bang_mode	3	0	R/W	Bit Bang Mode Enable

- 0: Bit Bang Mode disabled
- 1: Bit Bang Mode enabled

SPIn_GEN_CTRL.bb_ss_in_out

Field	Bits	Sys Reset	Access	Description
bb_ss_in_out	4	0	R/W	Bit Bang SS Input/Output

When written, defines output state of currently selected slave select (Bit Bang Mode only)

When read, returns the current state of the slave select (0=deasserted, 1=asserted)

SPIn_GEN_CTRL.bb_sr_in

Field	Bits	Sys Reset	Access	Description
bb_sr_in	5	0	R/O	Bit Bang SR Input

Writes have no effect.

When read, returns the current state of the flow control (0=deasserted, 1=asserted)

SPIn_GEN_CTRL.bb_sck_in_out

Field	Bits	Sys Reset	Access	Description
bb_sck_in_out	6	0	R/W	Bit Bang SCK Input/Output

When written, defines output state of SPI clock signal SCK (Bit Bang Mode only)

When read, returns the current state of the SCK clock (0=inactive, 1=active)

SPIn_GEN_CTRL.bb_sdio_in

Field	Bits	Sys Reset	Access	Description
bb_sdio_in	11:8	0000b	R/O	Bit Bang SDIO Input

Writes have no effect.

When read, returns the current input state of the SDIO pins

SPIn_GEN_CTRL.bb_sdio_out

Field	Bits	Sys Reset	Access	Description
bb_sdio_out	15:12	0000b	R/W	Bit Bang SDIO Output

Defines output state of SDIO pins (Bit Bang only)

SPIn_GEN_CTRL.bb_sdio_dr_en

Field	Bits	Sys Reset	Access	Description
bb_sdio_dr_en	19:16	0000b	R/W	Bit Bang SDIO Drive Enable

Enables output drive of SDIO pins (Bit Bang only)

7.3.9.4 SPIn_FIFO_CTRL

SPIn_FIFO_CTRL.tx_fifo_ae_lvl

Field	Bits	Sys Reset	Access	Description
tx_fifo_ae_lvl	3:0	15	R/W	Transaction FIFO AE Level

Defines number of unused FIFO entries (words) required to assert Almost Empty flag. FIFO depth is 16 entries.

SPIn_FIFO_CTRL.tx_fifo_used

Field	Bits	Sys Reset	Access	Description
tx_fifo_used	12:8	n/a	R/O	Transaction FIFO Used

Returns number of currently used entries in the Transaction FIFO

SPIn_FIFO_CTRL.rx_fifo_af_lvl

Field	Bits	Sys Reset	Access	Description
rx_fifo_af_lvl	20:16	31	R/W	Results FIFO AF Level

Defines number of used FIFO entries (bytes) required to assert Almost Full flag. FIFO depth is 32 bytes.

SPIn_FIFO_CTRL.rx_fifo_used

Field	Bits	Sys Reset	Access	Description
rx_fifo_used	29:24	n/a	R/O	Results FIFO Used

Returns number of currently used byte entries in the Results FIFO

7.3.9.5 SPIn_SPCL_CTRL

SPIn_SPCL_CTRL.ss_sample_mode

Field	Bits	Sys Reset	Access	Description
ss_sample_mode	0	0	R/W	SS Sample Mode

Enables (when set to 1) the ability to drive SDIO outputs prior to the assertion of Slave Select. This bit should be set when the SPI bus is idle and the transaction FIFO is empty; it will auto- clear when the next slave select assertion occurs.

SPIn_SPCL_CTRL.miso_fc_en

Field	Bits	Sys Reset	Access	Description
miso_fc_en	1	0	R/W	SDIO(1) to SR(0) Mode

When set to 1, routes SDIO(1) input to SR0 for use with devices that use MISO for flow control during writes. Must be cleared to perform reads.

SPIn_SPCL_CTRL.ss_sa_sdio_out

Field	Bits	Sys Reset	Access	Description
ss_sa_sdio_out	7:4	0000b	R/W	SDIO Active Output Value

Defines output mode of SDIO when SS Sample Mode is active.

SPIn_SPCL_CTRL.ss_sa_sdio_dr_en

Field	Bits	Sys Reset	Access	Description
ss_sa_sdio_dr_en	11:8	0000b	R/W	SDIO Active Drive Mode

Defines output drive mode of SDIO when SS Sample Mode is active.

7.3.9.6 SPIn_INTFL

SPIn_INTFL.tx_stalled

Field	Bits	Sys Reset	Access	Description
tx_stalled	0	0	W1C	Transaction Stalled Int Status

Write 1 to clear.

0:Int not active, 1:Interrupt has been triggered Set when transaction FIFO is empty and selected Slave Select is asserted.

SPIn_INTFL.rx_stalled

Field	Bits	Sys Reset	Access	Description
rx_stalled	1	0	W1C	Results Stalled Int Status

Write 1 to clear.

0:Int not active, 1:Interrupt has been triggered Set when results FIFO is full and selected Slave Select is asserted.

SPIn_INTFL.tx_ready

Field	Bits	Sys Reset	Access	Description
tx_ready	2	0	W1C	Transaction Ready Int Status

Write 1 to clear.

0:Int not active, 1:Interrupt has been triggered Set when transaction FIFO is empty and selected Slave Select is deasserted.

SPIn_INTFL.rx_done

Field	Bits	Sys Reset	Access	Description
rx_done	3	0	W1C	Results Done Int Status

Write 1 to clear.

0:Int not active, 1:Interrupt has been triggered Set when results FIFO is not empty and selected Slave Select is deasserted.

SPIn_INTFL.tx_fifo_ae

Field	Bits	Sys Reset	Access	Description
tx_fifo_ae	4	0	W1C	TXFIFO Almost Empty Int Status

Write 1 to clear.

0:Int not active, 1:Interrupt has been triggered Set when transaction FIFO is in the 'almost empty' state.

SPIn_INTFL.rx_fifo_af

Field	Bits	Sys Reset	Access	Description
rx_fifo_af	5	0	W1C	RXFIFO Almost Full Int Status

Write 1 to clear.

0:Int not active, 1:Interrupt has been triggered Set when the results FIFO is in the 'almost full' state.

7.3.9.7 SPIn_INTEN

SPIn_INTEN.tx_stalled

Field	Bits	Sys Reset	Access	Description
tx_stalled	0	0	R/W	Transaction Stalled Int Enable

0: Interrupt source disabled; 1:Interrupt enabled.

SPIn_INTEN.rx_stalled

Field	Bits	Sys Reset	Access	Description
rx_stalled	1	0	R/W	Results Stalled Int Enable

0: Interrupt source disabled; 1:Interrupt enabled.

SPIn_INTEN.tx_ready

Field	Bits	Sys Reset	Access	Description
tx_ready	2	0	R/W	Transaction Ready Int Enable

0: Interrupt source disabled; 1:Interrupt enabled.

SPIn_INTEN.rx_done

Field	Bits	Sys Reset	Access	Description
rx_done	3	0	R/W	Results Done Int Enable

0: Interrupt source disabled; 1:Interrupt enabled.

SPIn_INTEN.tx_fifo_ae

Field	Bits	Sys Reset	Access	Description
tx_fifo_ae	4	0	R/W	TXFIFO Almost Empty Int Enable

0: Interrupt source disabled; 1:Interrupt enabled.

SPIn_INTEN.rx_fifo_af

Field	Bits	Sys Reset	Access	Description
rx_fifo_af	5	0	R/W	RXFIFO Almost Full Int Enable

0: Interrupt source disabled; 1:Interrupt enabled.

7.3.9.8 SPIn_FIFO_TRANS

SPI0_FIFO_TRANS

Sys Reset	Access	Description
n/a	R/W	SPI Master FIFO Write Space for Transaction Setup

Writes to this space result in pushes to the SPI Master Transaction FIFO. This write space supports single accesses as well as burst accesses; access widths of 8-bit, 16-bit and 32-bit are supported. Reads from this space always return zeroes.

The SPI Master Transaction FIFO is 16 bits wide and 16 levels deep. Performing a 16-bit write to this space results in a single 16-bit push to the write end of the FIFO. A 32-bit write results in two 16-bit pushes; the least significant word is pushed first, followed by the most significant word. When two pushes occur from a single write, a busy status will be returned by the FIFO to the AHB bus master until both pushes have completed.

If 8-bit writes are used to load the FIFO, these writes must occur in even/odd address pairs. An even byte address write will be held until the corresponding odd byte address write is received, at which point both bytes will be pushed to the FIFO in a single 16-bit push operation (odd byte: MSB, even byte: LSB)

7.3.9.9 SPIn_FIFO_RSLTS

SPI0_FIFO_RSLTS

Sys Reset	Access	Description
n/a	R/W	SPI Master FIFO Read Space for Results Data

Reads from this space pull data from the SPI Master Results FIFO. This read space supports single accesses as well as burst accesses; access widths of 8-bit, 16-bit and 32-bit are supported. Writes to this space are ignored.

The SPI Master Results FIFO is 8 bits wide and 32 levels deep. Reading an 8-bit value from this space results in a single pull from the read end of the FIFO. Reading a 16-bit value results in two byte pulls (b0 and b1) from the read end of the FIFO; the resulting 16-bit return value is b1:b0 where b1 is the MSB and b0 is the LSB. Reading a 32-bit value results in four byte pulls (b0, b1, b2, b3) from the read end of the FIFO; the resulting 32-bit return value is b3:b2:b1:b0 where b3 is the MSB

and b0 is the LSB.

7.4 UART

7.4.1 Overview

The **MAX32620** provides four UART ports which can be used to communicate with external devices requiring an asynchronous serial protocol. Features of the **MAX32620** UARTs:

- Flexible baud rate generation based on the module clock frequency (equal to the system clock source or a subdivide of the system clock source)
- Programmable word size (5- to 8-bits), stop bits, and parity settings
- Automatic parity and framing error detection
- Automatic flow control can be enabled for RTS and CTS lines
- 32-byte AHB-mapped FIFO in both directions (separate read/RX and write/TX FIFOs)
- AHB FIFOs support 8-bit, 16-bit, or 32-bit burst accesses
- Interrupts can be triggered for transmit operations on TX done, TX clear to send (CTS asserted), and TX FIFO almost empty conditions
- Interrupts can be triggered for receive operations on RX FIFO not empty (data received), RX stalled (RTS output is deasserted) RX FIFO almost full, RX FIFO overflow, RX framing error detected, or RX parity error detected conditions
- Configurable RX FIFO levels for RTS assertion and RX FIFO almost full interrupt trigger
- Configurable TX FIFO level for TX FIFO almost empty interrupt trigger
- Multi-drop mode support

7.4.2 UART Port and Pin Configurations

UART Interface Signals (Per UART Port/Instance)

Each enabled UART instance on the **MAX32620** has from two to four associated interface signals as described in the following sections.

TX and RX

For each enabled UART instance, the TX (transmit) and RX (receive) pin functions must be enabled, although if the UART is being used in a transmit-only or receive-only application, it is not necessary for both pins to be connected. However, the I/O Manager requires that the TX/RX pins be enabled as a unit.

The TX line operates as an output from the UART peripheral and is used to transmit data that has been loaded into the TX FIFO. In a multi-drop application, a UART peripheral configured to operate in multi-drop slave mode will set its TX output to high impedance when it is not being addressed by the bus master, thus allowing another UART slave on the bus to take control of the TX line.

The RX line operates as an input to the UART peripheral. Data received in the proper format on the RX line will be loaded into the RX FIFO where it can be unloaded by application firmware (either directly by reading the RX FIFO read address location, or indirectly using a PMU channel).

CTS and RTS

The CTS (Clear to Send) and RTS (Ready to Send) signals are used for automatic hardware-based flow control. Their use is optional: these signals may be enabled individually (CTS only or RTS only), together (CTS and RTS), or not at all.

The polarities of the CTS and RTS signals are individually configurable to either active low or active high. Both signals default to active low.

The CTS line operates as an input to the UART peripheral and indicates whether the UART device on the other end of the UART bus (whether master or slave) is ready to receive data, indicating that the UART peripheral should begin/continue transmission of any queued data in the TX FIFO on its TX line.

The RTS line operates as an output from the UART peripheral and allows the UART peripheral to indicate that it is ready to receive data, or in other words, that it is ready for the UART on the other side of the bus to transmit data.

A UART peripheral configured to operate in multi-drop mode as a slave UART will set its RTS output to high impedance whenever it is not being addressed by the UART bus master.

7.4.2.1 UART 0 Pin Configurations

The tables below show the pin configuration/mapping options for UART 0. Note that while the register fields in the I/O Manager have separate settings for the RTS and CTS mapping option, the available pin mappings on the **MAX32620** require these two fields to be set to identical values (both to A or both to B) to avoid collisions between the RTS and CTS functions.

TX and RX Pin Mapping for UART 0

Logic Signal	Mapping Option A	Mapping Option B
RX	P0.0	P0.1
TX	P0.1	P0.0

CTS and RTS Pin Mapping for UART 0

Logic Signal	Mapping Option A	Mapping Option B
CTS	P0.2	P0.3
RTS	P0.3	P0.2

7.4.2.2 UART 1 Pin Configurations

The tables below show the pin configuration/mapping options for UART 1. Note that while the register fields in the I/O Manager have separate settings for the RTS and CTS mapping option, the available pin mappings on the **MAX32620** require these two fields to be set to identical values (both to A or both to B) to avoid collisions between the RTS and CTS functions.

TX and RX Pin Mapping for UART 1

Logic Signal	Mapping Option A	Mapping Option B
RX	P2.0	P2.1
TX	P2.1	P2.0

CTS and RTS Pin Mapping for UART 1

Logic Signal	Mapping Option A	Mapping Option B
CTS	P2.2	P2.3
RTS	P2.3	P2.2

7.4.2.3 UART 2 Pin Configurations

The tables below show the pin configuration/mapping options for UART 2. Note that while the register fields in the I/O Manager have separate settings for the RTS and CTS mapping option, the available pin mappings on the **MAX32620** require these two fields to be set to identical values (both to A or both to B) to avoid collisions between the RTS and CTS functions.

TX and RX Pin Mapping for UART 2

Logic Signal	Mapping Option A	Mapping Option B
RX	P3.0	P3.1
TX	P3.1	P3.0

CTS and RTS Pin Mapping for UART 2

Logic Signal	Mapping Option A	Mapping Option B
CTS	P3.2	P3.3
RTS	P3.3	P3.2

7.4.2.4 UART 3 Pin Configurations

The tables below show the pin configuration/mapping options for UART 3. Note that while the register fields in the I/O Manager have separate settings for the RTS and CTS mapping option, the available pin mappings on the **MAX32620** require these two fields to be set to identical values (both to A or both to B) to avoid collisions between the RTS and CTS functions.

TX and RX Pin Mapping for UART 3

Logic Signal	Mapping Option A	Mapping Option B
RX	P5.3	P5.4
TX	P5.4	P5.3

CTS and RTS Pin Mapping for UART 3

Logic Signal	Mapping Option A	Mapping Option B
CTS	P5.5	P5.6
RTS	P5.6	P5.5

7.4.3 UART Clock Configuration

UART Common Clock Basis and Scaling

All active UARTs in the **MAX32620** use a single common clock as a basis for logic operation and baud rate generation. This clock is derived from the currently selected system clock source (nominally either 48MHz or 96MHz, as selected by CLKMAN_CLK_CTRL.system_source_select.

To enable the common UART clock, CLKMAN_SYS_CLK_CTRL_8_UART must be written to a nonzero value. Additionally, the common UART clock can optionally be derived from a scaled-down version of the system clock source as shown below.

CLKMAN_SYS_CLK_CTRL_8_UART.uart_clk_scale (4b)	Setting
0	UART Clock Disabled
1	UART Clock = (System Clock Source / 1)
2	UART Clock = (System Clock Source / 2)
3	UART Clock = (System Clock Source / 4)
4	UART Clock = (System Clock Source / 8)
5	UART Clock = (System Clock Source / 16)
6	UART Clock = (System Clock Source / 32)
7	UART Clock = (System Clock Source / 64)
8	UART Clock = (System Clock Source / 128)
9	UART Clock = (System Clock Source / 256)
All Other Settings	Reserved

UART Clock Gating Controls (Per Instance)

Each UART peripheral instance has a separate clock gating control. By default, these clock gating controls are set to dynamically gate the clock to the UART instance. This means that the UART peripheral instance will not be clocked unless it is being accessed (either over AHB or APB) or it is in the process of transmitting or receiving data. This allows power consumption to be reduced when the UART is in a waiting or idle state, while allowing it to remain ready for operations when needed.

UART Instance	Clock Gating Control
UART 0	CLKMAN_CLK_GATE_CTRL1.uart0_clk_gater
UART 1	CLKMAN_CLK_GATE_CTRL1.uart1_clk_gater
UART 2	CLKMAN_CLK_GATE_CTRL1.uart2_clk_gater
UART 3	CLKMAN_CLK_GATE_CTRL1.uart3_clk_gater

If necessary, the clock gating control can be set for any given UART peripheral instance to statically gate the clock on or off regardless of the current activity or inactivity of the UART or APB/AHB bus access.

uart(n)_clk_gater (2b)	Setting
00b	Force clock off - UART instance disabled
01b	Dynamic clock gating - UART clock active only when used/active
1xb	Force clock on - UART enabled at all times

7.4.4 UART Register Interface

7.4.4.1 UART APB Registers

Each UART instance is controlled by a block of APB registers assigned to that UART; all blocks contain identical control, interrupt, and status registers. Address locations for each APB register are shown in the tables below.

APB Registers for UART 0

Address	Register	Details
0x40012000	UART0_CTRL	UART Control Register
0x40012004	UART0_BAUD	UART Baud Control Register
0x40012008	UART0_TX_FIFO_CTRL	UART TX Fifo Control Register
0x4001200C	UART0_RX_FIFO_CTRL	UART RX Fifo Control Register

Address	Register	Details
0x40012010	UART0_MD_CTRL	UART Multidrop Control Register
0x40012014	UART0_INTFL	UART Interrupt Flags
0x40012018	UART0_INTEN	UART Interrupt Enable/Disable Controls

APB Registers for UART 1

Address	Register	Details
0x40013000	UART1_CTRL	UART Control Register
0x40013004	UART1_BAUD	UART Baud Control Register
0x40013008	UART1_TX_FIFO_CTRL	UART TX Fifo Control Register
0x4001300C	UART1_RX_FIFO_CTRL	UART RX Fifo Control Register
0x40013010	UART1_MD_CTRL	UART Multidrop Control Register
0x40013014	UART1_INTFL	UART Interrupt Flags
0x40013018	UART1_INTEN	UART Interrupt Enable/Disable Controls

APB Registers for UART 2

Address	Register	Details
0x40014000	UART2_CTRL	UART Control Register
0x40014004	UART2_BAUD	UART Baud Control Register
0x40014008	UART2_TX_FIFO_CTRL	UART TX Fifo Control Register
0x4001400C	UART2_RX_FIFO_CTRL	UART RX Fifo Control Register
0x40014010	UART2_MD_CTRL	UART Multidrop Control Register
0x40014014	UART2_INTFL	UART Interrupt Flags
0x40014018	UART2_INTEN	UART Interrupt Enable/Disable Controls

APB Registers for UART 3

Address	Register	Details
0x40015000	UART3_CTRL	UART Control Register
0x40015004	UART3_BAUD	UART Baud Control Register
0x40015008	UART3_TX_FIFO_CTRL	UART TX Fifo Control Register
0x4001500C	UART3_RX_FIFO_CTRL	UART RX Fifo Control Register
0x40015010	UART3_MD_CTRL	UART Multidrop Control Register
0x40015014	UART3_INTFL	UART Interrupt Flags
0x40015018	UART3_INTEN	UART Interrupt Enable/Disable Controls

7.4.4.2 UART AHB FIFOs

In addition to the control, interrupt and status registers accessed via the lower-speed APB bus, access points to the RX FIFO (read-only) and TX FIFO (write-only) are mapped to the AHB bus for each UART peripheral instance. These FIFO address locations allow the RX and TX FIFOs to be unloaded and loaded using high-speed AHB bus transactions including burst accesses up to the FIFO depth (32 bytes).

AHB UART FIFO TX and RX Addresses

Address	Register	Details
0x40103000	UART0_FIFO_TX	UART 0 - FIFO Write Point for Data to Transmit
0x40103800	UART0_FIFO_RX	UART 0 - FIFO Read Point for Received Data
0x40104000	UART1_FIFO_TX	UART 1 - FIFO Write Point for Data to Transmit
0x40104800	UART1_FIFO_RX	UART 1 - FIFO Read Point for Received Data
0x40105000	UART2_FIFO_TX	UART 2 - FIFO Write Point for Data to Transmit
0x40105800	UART2_FIFO_RX	UART 2 - FIFO Read Point for Received Data
0x40106000	UART3_FIFO_TX	UART 3 - FIFO Write Point for Data to Transmit
0x40106800	UART3_FIFO_RX	UART 3 - FIFO Read Point for Received Data

7.4.5 Format and Baud Rate Selection

The overall baud rate for each UART is based on the common UART module clock frequency. This baud rate is set using UARTn_BAUD, where the baud rate is given by the following formula.

$$\text{Baud Rate} = \frac{UARTCommonClockFrequency}{UARTn\_BAUD * 16}$$

Setting UARTn_BAUD to zero will shut off the baud clock generator, effectively preventing operation for the UART instance.

The number of stop bits used and parity calculation mode are all set by writing to the appropriate bits and bit fields in the UARTn_CTRL register.

- UARTn_CTRL.parity is used to enable/disable parity as well as to determine if parity calculation is based on number of 0s or number of 1s in the character transmitted/received.
- UARTn_CTRL.extra_stop determines whether an additional stop bit should be transmitted as well as verified in incoming characters.

To set the transfer size (character length), the UARTn_CTRL.data_size field must be set.

UART Character Size (2b)	Setting
00b	5-bit character
01b	6-bit character
01b	7-bit character
11b	8-bit character

7.4.6 Transmitting and Receiving Data

Data to be transmitted must be written to the TX FIFO by writing to the UARTn_FIFO_TX register (either directly or using a PMU channel). This data will be transmitted by the UART hardware automatically a character at a time in the order that it was written to the TX FIFO. The status flags and associated UART interrupts can be used to monitor the TX FIFO status and determine when the transfer cycle or cycles have completed.

If the amount of data to be transmitted by the UART is more than will fit in the TX FIFO at one time (that is, more than 32 characters), then the TX FIFO Almost Empty flag can be used to monitor when the data in the TX FIFO has dropped below a user-specified level. This can be used to give the CPU (or the PMU) time to load more data into the TX FIFO before the TX FIFO goes empty, ensuring that the data is transferred as quickly as possible over the UART interface with no unnecessary gaps between portions of the data.

As data is received by the UART, it is loaded into the RX FIFO, and it can then be unloaded by reading from the UARTn_FIFO_RX register. If the amount of data in the RX FIFO exceeds the user-defined RX FIFO Almost Full level, the associated flag can be used to notify application firmware or trigger an interrupt. This ensures that the CPU or the PMU will be notified in time to unload enough data from the RX FIFO to prevent an RX FIFO overflow condition.

7.4.7 Interrupts

Interrupts can be generated by each UART instance for one or more of the following conditions:

- All queued data in the TX FIFO has been transmitted (TX FIFO is empty and current transmit cycle has completed)
- TX is allowed to resume (CTS flow control is enabled, and CTS input transitions from inactive to active state)
- When the number of unused entries in the TX FIFO increases one beyond the TX FIFO Almost Empty Level
- The RX FIFO goes from an empty state to a non-empty state
- The RTS output from the UART is deasserted (when RTS flow control is enabled)
- A framing error occurs on a received character (the expected one or two stop bits were not properly detected).
- The received character has an invalid parity bit according to the chosen parity settings.
- An overflow occurs on the RX FIFO, and the data from the incoming character is lost.
- When the number of used entries in the RX FIFO increases one beyond the RX FIFO Almost Full Level

7.4.8 Hardware Flow Control

Hardware flow control can be enabled for each active UART instance separately for the CTS line, the RTS line, or both lines as needed.

UART CTS

The CTS (Clear To Send) line is an input to the UART that is driven by an external device. It is used by the external device to notify the UART whether or not the external device is ready to receive additional data. With CTS hardware flow control enabled, the UART samples the state of the CTS line just before a new character would be transmitted from the TX FIFO. If the CTS line is asserted at this point, the UART will begin transmitting the new character. If the CTS line is deasserted, the UART waits for CTS to return to the asserted state before continuing transmission.

It is important to note that even if the CTS line transitions from an asserted to a non-asserted state, the UART will continue transmitting the current character if transmission has already begun (that is, the UART will not pause transmission in the middle of the character).

The polarity of the CTS line is determined by the CTS Polarity field; by default, CTS is active high.

To enable the CTS hardware flow control, set bit UARTn_CTRL.cts_en to a 1. The CTS Polarity field is controlled by the UARTn_CTRL.cts_polarity bit. Writing a 1 to this field will set the polarity high and writing a 0 sets it to active low.

UART RTS

The RTS (Ready To Send) line is an output from the UART that notifies an external device whether or not the UART is ready to receive more data. When hardware flow control is enabled, the RTS output is automatically asserted whenever the number of used entries in the RX FIFO exceeds the configurable RTS Level. Once

the number of used entries in the RX FIFO drops below this level (which will occur when the RX FIFO is unloaded sufficiently by the CPU or PMU), then the RTS output will be deasserted, indicating to the external device that it should continue transmitting data.

This automatic assertion/deassertion of the RTS output is separate from the RX FIFO Almost Full interrupt and is triggered based on a separate condition. Flow control using RTS can be enabled to operate whether or not the RX FIFO Almost Full interrupt is enabled. The two are intended to work in parallel, since the RTS output will notify the external device to pause its data transmission, while the RX FIFO Almost Full interrupt will notify the CPU or PMU that it needs to unload data from the RX FIFO to make space for more incoming data.

RTS flow control is enabled by setting the bit UARTn_CTRL.rts_en.

The configurable RTS LTE level is a 6 bit value and defaults to 0x3F indicating anytime data in the receive FIFO drops below 63 bytes the RTS output will be deasserted. To change the RTS LTE level, set the UARTn_CTRL.rts_level to any value $>$ 0 up to 63 (0x3F).

The polarity of the RTS line is determined by the RTS Polarity field; by default, RTS is active low. The RTS Polarity is controlled by the bit UARTn_CTRL.rts_polarity. Setting rts_polarity high changes it to active high.

7.4.9 Multidrop Mode Support

It is possible to define a UART system which supports multiple devices attached to the same set of signal wires. The UARTs on the **MAX32620** can be enabled to support this type of system configuration, also known as Multidrop Mode.

In this mode, one UART device on the bus acts as the master, and the rest (one or more UART devices) act as slaves on the bus. The **MAX32620** UART peripheral can be configured to operate in either a master or slave configuration when multidrop mode is enabled.

The master controls the operation of the bus and initiates each read/write transaction. The master typically communicates with one slave at a time, and is able to transmit and receive data from any of the slaves on the bus. The UART slaves on the bus do not communicate with one another directly, but only communicate with the master and only when they have been addressed.

Each slave on the bus has a unique (at the scope of the bus) slave address, with the slave addresses pre-assigned in some manner not covered here. By default, all slaves are disconnected from the bus. The master is always connected to the bus.

When a UART slave is disconnected from the bus, it sets outputs to high impedance instead of driving them on the bus (TX, and RTS if enabled). This allows these bus lines to float so that another slave on the bus can drive them. When a UART slave is connected to the bus, it drives TX and RTS normally. In either state, the slave device must monitor its RX line to receive data (when connected to the bus) and to check for the next address character (when either connected or disconnected).

When no slaves are connected to the bus, the TX and RTS lines will not be driven by any slave device, which presents an issue with floating input lines on the bus master. To prevent this, a weak pullup must be placed on the common slave TX output line (which is also the RX input for the UART bus master). If the RTS common slave output is being used (the CTS input line on the UART bus master), then a pullup or pulldown must be placed on this line as well, according to the selected CTS/RTS signal polarity, in order to pull CTS/RTS to the asserted (active) state when no device is driving the line.

In multidrop mode, the mark/space parity bit defines whether each character transmitted on the bus is an address character (with the parity bit set) or a data character (with the parity bit cleared).

If an address character is transmitted on the bus, each slave compares that address to its local address value. If the two match, then the slave will connect to the bus

and transmit/receive data as needed until another address character arrives, at which point it will compare the address again.

If the address transmitted does not match the slave address, then the slave will disconnect from the bus and disregard all transmitted data until the next address character arrives, at which point it will compare the address again.

7.4.10 Registers (UART)

Address	Register	Access	Description	Reset By
0x40012000	UART0_CTRL	R/W	UART 0 Control Register	Sys
0x40012004	UART0_BAUD	R/W	UART 0 Baud Control Register	Sys
0x40012008	UART0_TX_FIFO_CTRL	***	UART 0 TX FIFO Control Register	Sys
0x4001200C	UART0_RX_FIFO_CTRL	***	UART 0 RX FIFO Control Register	Sys
0x40012010	UART0_MD_CTRL	R/W	UART 0 Multidrop Control Register	Sys
0x40012014	UART0_INTFL	W1C	UART 0 Interrupt Flags	Sys
0x40012018	UART0_INTEN	R/W	UART 0 Interrupt Enable/Disable Controls	Sys
0x40013000	UART1_CTRL	R/W	UART 1 Control Register	Sys
0x40013004	UART1_BAUD	R/W	UART 1 Baud Control Register	Sys
0x40013008	UART1_TX_FIFO_CTRL	***	UART 1 TX FIFO Control Register	Sys
0x4001300C	UART1_RX_FIFO_CTRL	***	UART 1 RX FIFO Control Register	Sys
0x40013010	UART1_MD_CTRL	R/W	UART 1 Multidrop Control Register	Sys
0x40013014	UART1_INTFL	W1C	UART 1 Interrupt Flags	Sys
0x40013018	UART1_INTEN	R/W	UART 1 Interrupt Enable/Disable Controls	Sys
0x40014000	UART2_CTRL	R/W	UART 2 Control Register	Sys
0x40014004	UART2_BAUD	R/W	UART 2 Baud Control Register	Sys
0x40014008	UART2_TX_FIFO_CTRL	***	UART 2 TX FIFO Control Register	Sys
0x4001400C	UART2_RX_FIFO_CTRL	***	UART 2 RX FIFO Control Register	Sys
0x40014010	UART2_MD_CTRL	R/W	UART 2 Multidrop Control Register	Sys
0x40014014	UART2_INTFL	W1C	UART 2 Interrupt Flags	Sys
0x40014018	UART2_INTEN	R/W	UART 2 Interrupt Enable/Disable Controls	Sys
0x40015000	UART3_CTRL	R/W	UART 3 Control Register	Sys

Address	Register	Access	Description	Reset By
0x40015004	UART3_BAUD	R/W	UART 3 Baud Control Register	Sys
0x40015008	UART3_TX_FIFO_CTRL	***	UART 3 TX FIFO Control Register	Sys
0x4001500C	UART3_RX_FIFO_CTRL	***	UART 3 RX FIFO Control Register	Sys
0x40015010	UART3_MD_CTRL	R/W	UART 3 Multidrop Control Register	Sys
0x40015014	UART3_INTFL	W1C	UART 3 Interrupt Flags	Sys
0x40015018	UART3_INTEN	R/W	UART 3 Interrupt Enable/Disable Controls	Sys
0x40103000	UART0_FIFO_TX	R/W	FIFO Write Point for Data to Transmit	Sys
0x40103800	UART0_FIFO_RX	R/W	FIFO Read Point for Received Data	Sys
0x40104000	UART1_FIFO_TX	R/W	FIFO Write Point for Data to Transmit	Sys
0x40104800	UART1_FIFO_RX	R/W	FIFO Read Point for Received Data	Sys
0x40105000	UART2_FIFO_TX	R/W	FIFO Write Point for Data to Transmit	Sys
0x40105800	UART2_FIFO_RX	R/W	FIFO Read Point for Received Data	Sys
0x40106000	UART3_FIFO_TX	R/W	FIFO Write Point for Data to Transmit	Sys
0x40106800	UART3_FIFO_RX	R/W	FIFO Read Point for Received Data	Sys

7.4.10.1 UARTn_CTRL

UARTn_CTRL.uart_en

Field	Bits	Sys Reset	Access	Description
uart_en	0	0	R/W	UART Enable

Enables UART function when set, resets UART when cleared.

UARTn_CTRL.rx_fifo_en

Field	Bits	Sys Reset	Access	Description
rx_fifo_en	1	0	R/W	RX FIFO Enable

Enables RX FIFO when set, clears/resets RX FIFO when cleared.

UARTn_CTRL.tx_fifo_en

Field	Bits	Sys Reset	Access	Description
tx_fifo_en	2	0	R/W	TX FIFO Enable

Enables TX FIFO when set, clears/resets TX FIFO when cleared.

UARTn_CTRL.data_size

Field	Bits	Sys Reset	Access	Description
data_size	5:4	11b	R/W	Data Size

Number of data bits transferred or received per character.

- 00b: 5 bits
- 01b: 6 bits
- 10b: 7 bits
- 11b: 8 bits

UARTn_CTRL.extra_stop

Field	Bits	Sys Reset	Access	Description
extra_stop	8	0	R/W	Extra Stop Enable

- 0: A single Stop bit will be generated and expected (default).
- 1: Two Stop bits will be generated and expected.

UARTn_CTRL.parity

Field	Bits	Sys Reset	Access	Description
parity	13:12	0	R/W	Parity Mode

- 0: Parity disabled (default).
- 1: Odd parity checking enabled
- 2: Even parity checking enabled
- 3: Enable Mark/Space mode for parity bit (multidrop mode).

UARTn_CTRL.cts_en

Field	Bits	Sys Reset	Access	Description
cts_en	16	0	R/W	CTS Enable

- 0: CTS is disabled (default).
- 1: Enable hardware TX flow control via CTS input.

UARTn_CTRL.cts_polarity

Field	Bits	Sys Reset	Access	Description
cts_polarity	17	0	R/W	CTS Polarity

- 0: CTS is active high (default).
- 1: CTS is active low.

UARTn_CTRL.rts_en

Field	Bits	Sys Reset	Access	Description
rts_en	18	0	R/W	RTS Enable

- 0: RTS is disabled (default).
- 1: Enable hardware RX flow control via RTS output.

UARTn_CTRL.rts_polarity

Field	Bits	Sys Reset	Access	Description
rts_polarity	19	0	R/W	RTS Polarity

- 0: RTS is active high (default).
- 1: RTS is active low.

UARTn_CTRL.rts_level

Field	Bits	Sys Reset	Access	Description
rts_level	25:20	3Fh	R/W	RX FIFO LTE Level for RTS Assert

Assert RTS when RX FIFO level is less than or equal to this value.

7.4.10.2 UARTn_BAUD

UARTn_BAUD.baud_divisor

Field	Bits	Sys Reset	Access	Description
baud_divisor	7:0	00h	R/W	Baud Divisor

Baud rate equals (UART module clock)/(baud_divisor $*$ 16). If this field is set to 0 (default), the UART is disabled.

7.4.10.3 UARTn_TX_FIFO_CTRL

UARTn_TX_FIFO_CTRL.fifo_entry

Field	Bits	Sys Reset	Access	Description
fifo_entry	5:0	0	R/O	TX FIFO Entries

Current number of used entries (bytes) in transmit FIFO.

UARTn_TX_FIFO_CTRL.fifo_ae_lvl

Field	Bits	Sys Reset	Access	Description
fifo_ae_lvl	20:16	1Fh	R/W	TX FIFO AE Level

Almost Empty flag is asserted when the number of unused entries (bytes) in the transmit FIFO exceeds this level. FIFO depth is 32 bytes.

7.4.10.4 UARTn_RX_FIFO_CTRL

UARTn_RX_FIFO_CTRL.fifo_entry

Field	Bits	Sys Reset	Access	Description
fifo_entry	5:0	0	R/O	RX FIFO Entries

Current number of used entries (bytes) in receive FIFO.

UARTn_RX_FIFO_CTRL.fifo_af_lvl

Field	Bits	Sys Reset	Access	Description
fifo_af_lvl	20:16	1Fh	R/W	RX FIFO AF Level

Almost Full flag is asserted when the number of used FIFO entries (bytes) in the receive FIFO exceeds this level. FIFO depth is 32 bytes.

7.4.10.5 UARTn_MD_CTRL

UARTn_MD_CTRL.slave_addr

Field	Bits	Sys Reset	Access	Description
slave_addr	7:0	00h	R/W	Slave Address

When multidrop mode is enabled (by setting the parity mode to mark/space), this field contains an 8-bit slave address that will be compared against incoming marked data (address field) to determine if the following message should be received or ignored.

UARTn_MD_CTRL.slave_addr_msk

Field	Bits	Sys Reset	Access	Description
slave_addr_msk	15:8	00h	R/W	Slave Address Mask

This field is used to mask off slave address bits for comparing.

UARTn_MD_CTRL.md_mstr

Field	Bits	Sys Reset	Access	Description
md_mstr	16	1	R/W	Multidrop Master

This setting has no effect when multidrop mode is not enabled.

- 0: The UART acts as a multidrop slave.
- 1: The UART acts as a multidrop master. Masters do not disable their receivers or turn off their output drives.

UARTn_MD_CTRL.tx_addr_mark

Field	Bits	Sys Reset	Access	Description
tx_addr_mark	17	0	R/W	TX Address Mark

Setting this bit to 1 will cause the next byte transmitted from the FIFO to be marked as an address byte in multidrop mode. As this bit is set independently of the current TX FIFO contents, care must be taken that the next byte to be sent from the FIFO is the one that is intended to be marked.

7.4.10.6 UARTn_INTFL

UARTn_INTFL.tx_done

Field	Bits	Sys Reset	Access	Description
tx_done	0	0	W1C	TX Done Interrupt Flag

UARTn_INTFL.tx_unstalled

Field	Bits	Sys Reset	Access	Description
tx_unstalled	1	0	W1C	TX Unstalled Interrupt Flag

UARTn_INTFL.tx_fifo_ae

Field	Bits	Sys Reset	Access	Description
tx_fifo_ae	2	0	W1C	TX FIFO Almost Empty Interrupt Flag

UARTn_INTFL.rx_fifo_not_empty

Field	Bits	Sys Reset	Access	Description
rx_fifo_not_empty	3	0	W1C	RX FIFO Not Empty Interrupt Flag

UARTn_INTFL.rx_stalled

Field	Bits	Sys Reset	Access	Description
rx_stalled	4	0	W1C	RX Stalled Interrupt Flag

UARTn_INTFL.rx_fifo_af

Field	Bits	Sys Reset	Access	Description
rx_fifo_af	5	0	W1C	RX FIFO Almost Full Interrupt Flag

UARTn_INTFL.rx_fifo_overflow

Field	Bits	Sys Reset	Access	Description
rx_fifo_overflow	6	0	W1C	RX FIFO Overflow Interrupt Flag

UARTn_INTFL.rx_framing_err

Field	Bits	Sys Reset	Access	Description
rx_framing_err	7	0	W1C	RX Framing Error Interrupt Flag

UARTn_INTFL.rx_parity_err

Field	Bits	Sys Reset	Access	Description
rx_parity_err	8	0	W1C	RX Parity Error Interrupt Flag

7.4.10.7 UARTn_INTEN

UARTn_INTEN.tx_done

Field	Bits	Sys Reset	Access	Description
tx_done	0	0	R/W	TX Done Interrupt Enable/Disable

UARTn_INTEN.tx_unstalled

Field	Bits	Sys Reset	Access	Description
tx_unstalled	1	0	R/W	TX Unstalled Interrupt Enable/Disable

UARTn_INTEN.tx_fifo_ae

Field	Bits	Sys Reset	Access	Description
tx_fifo_ae	2	0	R/W	TX FIFO Almost Empty Interrupt Enable/Disable

UARTn_INTEN.rx_fifo_not_empty

Field	Bits	Sys Reset	Access	Description
rx_fifo_not_empty	3	0	R/W	RX FIFO Not Empty Interrupt Enable/Disable

UARTn_INTEN.rx_stalled

Field	Bits	Sys Reset	Access	Description
rx_stalled	4	0	R/W	RX Stalled Interrupt Enable/Disable

UARTn_INTEN.rx_fifo_af

Field	Bits	Sys Reset	Access	Description
rx_fifo_af	5	0	R/W	RX FIFO Almost Full Interrupt Enable/Disable

UARTn_INTEN.rx_fifo_overflow

Field	Bits	Sys Reset	Access	Description
rx_fifo_overflow	6	0	R/W	RX FIFO Overflow Interrupt Enable/Disable

UARTn_INTEN.rx_framing_err

Field	Bits	Sys Reset	Access	Description
rx_framing_err	7	0	R/W	RX Framing Error Interrupt Enable/Disable

UARTn_INTEN.rx_parity_err

Field	Bits	Sys Reset	Access	Description
rx_parity_err	8	0	R/W	RX Parity Error Interrupt Enable/Disable

7.4.10.8 UARTn_FIFO_TX

UART0_FIFO_TX

Sys Reset	Access	Description
n/a	R/W	FIFO Write Point for Data to Transmit

7.4.10.9 UARTn_FIFO_RX

UART0_FIFO_RX

Sys Reset	Access	Description
n/a	R/W	FIFO Read Point for Received Data

7.5 USB Device Interface

7.5.1 Overview

The **MAX32620** includes a Universal Serial Bus(USB) peripheral dedicated to device operation. The peripheral is a USB 2.0 compliant full speed device and includes a Serial Interface Engine (SIE) that connects to the internal USB transceiver and pin drivers.

7.5.2 Operation

The USB device controller supports a total of eight endpoints with programmable configuration.

7.5.2.1 USB Reset Definitions

The USB device is reset under the following conditions:

- **USB Bus Reset**: Host issues a bus reset
- **USB Device Reset**: Firmware changes the USB_CN.usb_en bit from 0 (disabled) to 1 (enabled).
- **System Reset**: System undergoing a reset.

Note In this document a reset condition without any qualifier refers to a System Reset. The term *USB reset* refers to either a USB Bus Reset or a USB Device Reset.

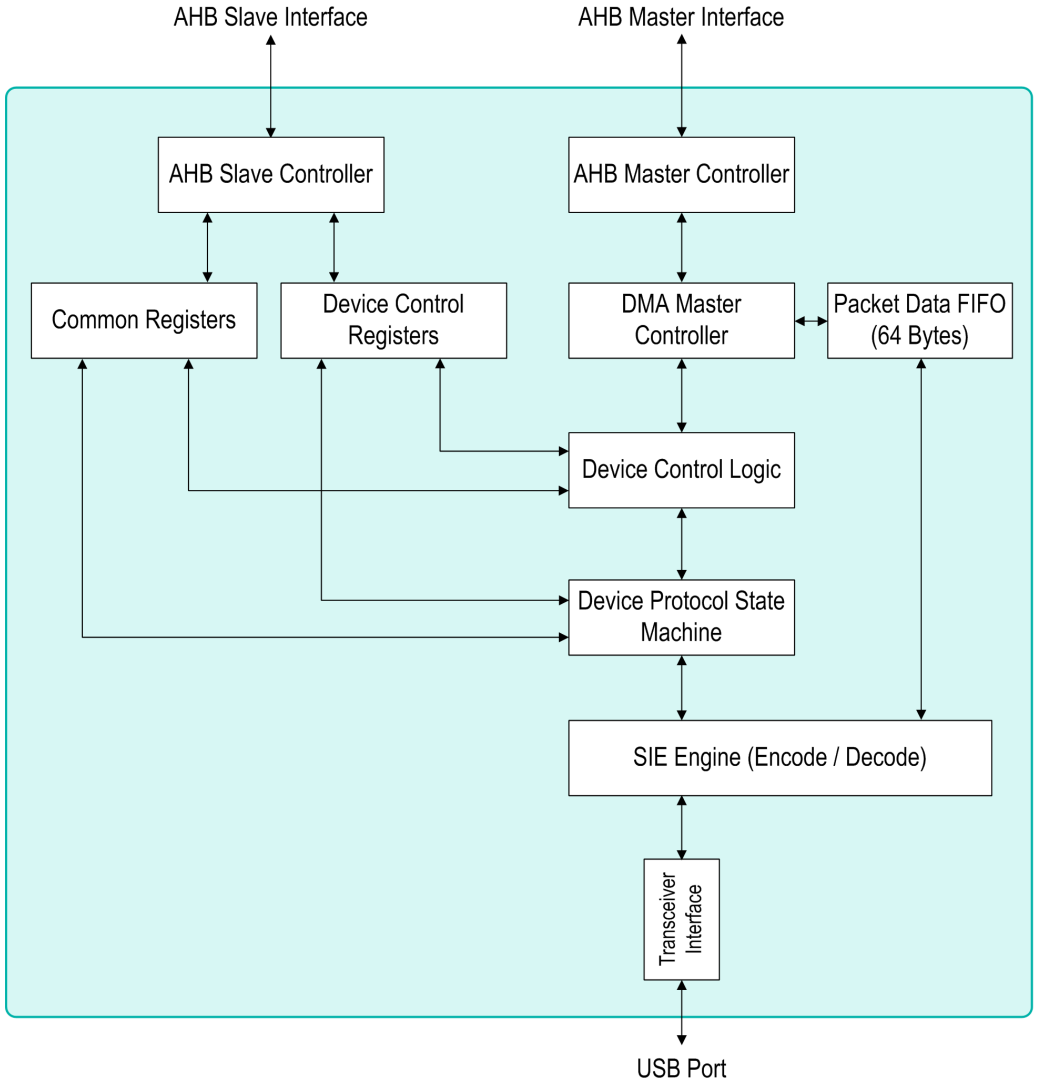

Figure 7.16: USB Device Peripheral Block Diagram

7.5.3 USB Endpoints

The **MAX32620** supports eight USB End Points and each can be individually configured. Each End Point supports:

- Single / Double buffer
- Programmable buffer starting address
- Programmable interrupt generation
- Ability to STALL a packet
- Bulk and Interrupt transfer
- Control transfer (endpoint 0 only)
- Configurable response to Status Stage of Control transfer

Each endpoint includes:

- Endpoint Control Register
- Endpoint Buffer Descriptor
- Endpoint Buffer Address

7.5.3.1 Endpoint Control Register

The Endpoint Control Register for each endpoint (USB_EP) is used to define general characteristics for that endpoint. characteristics.

- Disable an endpoint, ep_dir = "0"
- Direction of packet transfer, ep_dir
  - Endpoint 0 is the only CONTROL mode endpoint
  - All other Endpoints can be set to IN or OUT
- Single or double buffering, ep_buf2
- Reset data buffer to 0 for double buffered streams, ep_dt
- Interrupt generation:
  - Enable interrupt for IN or OUT data transfers, ep_int_en
  - Enable interrupt for NAK handshakes sent to host, ep_nak_en
- Endpoint STALL status, ep_stall
- Send a STALL to host for status stage, ep_st_stall
- Send an ACK to the host for status stage, ep_st_ack

7.5.3.2 Endpoint Buffer Descriptor

The Endpoint Buffer Descriptor is used as a communication port between the application firmware and the SIE in system memory. The Endpoint Buffer Descriptor starting address is specified by the USB Endpoint Descriptor Base Address Register (USB_EP_BASE).

The endpoint data toggle value and the buffer are maintained internally by the SIE. The data toggle value is initialized to DATA0 on USB reset. The buffer will be reset to buffer 0 on USB reset. When an endpoint is double-buffered, the SIE will use the two buffers in ping pong fashion. Firmware can reset the active data buffer

by writing a "1" to the ep_dt.

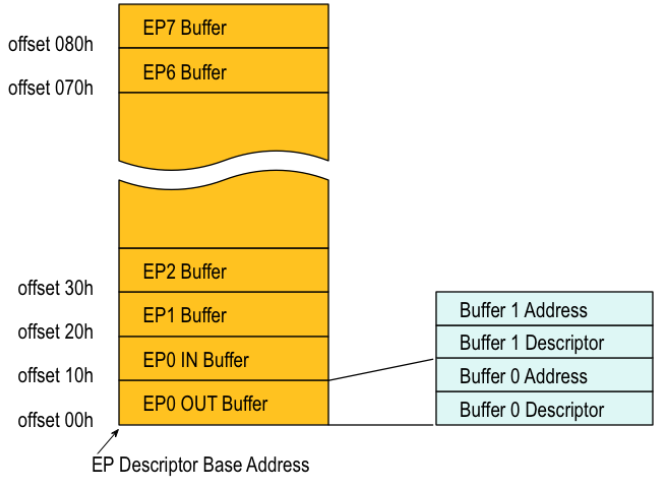

Figure 7.17: Endpoint Buffer Descriptor Memory

7.5.4 Registers (USB)

Address	Register	Access	Description	Reset By
0x40100000	USB_CN	R/W	USB Control Register	Sys
0x40100200	USB_DEV_ADDR	R/O	USB Device Address Register	Sys
0x40100204	USB_DEV_CN	R/W	USB Device Control Register	Sys
0x40100208	USB_DEV_INTFL	***	USB Device Interrupt	Sys
0x4010020C	USB_DEV_INTEN	R/W	USB Device Interrupt Enable	Sys
0x40100220	USB_EP_BASE	R/W	USB Endpoint Descriptor Table Base Address	Sys
0x40100224	USB_CUR_BUF	R/O	USB Current Endpoint Buffer Register	Sys
0x40100228	USB_IN_OWNER	R/W	USB IN Endpoint Buffer Owner Register	Sys
0x4010022C	USB_OUT_OWNER	R/W	USB OUT Endpoint Buffer Owner Register	Sys
0x40100230	USB_IN_INT	W1C	USB IN Endpoint Buffer Available Interrupt	Sys
0x40100234	USB_OUT_INT	W1C	USB OUT Endpoint Data Available Interrupt	Sys
0x40100238	USB_NAK_INT	W1C	USB IN Endpoint NAK Interrupt	Sys
0x4010023C	USB_DMA_ERR_INT	W1C	USB DMA Error Interrupt	Sys
0x40100240	USB_BUF_OVR_INT	W1C	USB Buffer Overflow Interrupt	Sys
0x40100260	USB_SETUP0	R/O	USB SETUP Packet Bytes 0 to 3	Sys
0x40100264	USB_SETUP1	R/O	USB SETUP Packet Bytes 4 to 7	Sys
0x40100280	USB_EP	R/W	USB Endpoint[n] Control Register	Sys

7.5.4.1 USB_CN

USB_CN.usb_en

Field	Bits	Sys Reset	Access	Description
usb_en	0	0	R/W	USB Device Interface Enable

Enabling the USB Device interface (setting this bit from 0 to 1) causes a reset of the USB Device internal state.

- 0: Disabled
- 1: Enabled

USB_CN.host

Field	Bits	Sys Reset	Access	Description
host	1	0	R/W	USB Host Mode Select

Reserved field; only Device mode is implemented.

7.5.4.2 USB_DEV_ADDR

USB_DEV_ADDR.dev_addr

Field	Bits	Sys Reset	Access	Description
dev_addr	6:0	0000000b	R/O	USB Device Address

Set by the USB SIE hardware during host enumeration as the result of a SetAddress() request.

7.5.4.3 USB_DEV_CN

USB_DEV_CN.sigrwu

Field	Bits	Sys Reset	Access	Description
sigrwu	2	0	R/W	USB Signal Remote Wakeup

- 0: Do not send remote wakeup signal
- 1: Send remote wakeup signal to USB bus host.

USB_DEV_CN.connect

Field	Bits	Sys Reset	Access	Description
connect	3	0	R/W	Connect to USB

- 0: Disconnect internal pullup resistor between DPLUS and VBUS.
- 1: Connect internal pullup resistor between DPLUS and VBUS.

USB_DEV_CN.ulpm

Field	Bits	Sys Reset	Access	Description
ulpm	4	0	R/W	USB Low Power Mode

- 0: USB transceiver in normal operation
- 1: USB transceiver will enter a low power mode

USB_DEV_CN.urst

Field	Bits	Sys Reset	Access	Description
urst	5	0	R/W	USB Device Controller Reset

- 0: USB device controller is released to run normally.
- 1: USB device controller is held in reset (until this bit is cleared back to 0 by firmware)

USB_DEV_CN.vbgate

Field	Bits	Sys Reset	Access	Description
vbgate	6	0	R/W	VBUS Gate

- 0: CONNECT operation is independent of VBUS status
- 1: CONNECT operation is performed conditionally depending on presence of VBUS voltage

USB_DEV_CN.oscen

Field	Bits	Sys Reset	Access	Description
oscen	7	0	R/W	OSC En

Reserved for test; do not modify the value of this bit.

USB_DEV_CN.bact_oe

Field	Bits	Sys Reset	Access	Description
bact_oe	8	0	R/W	BACT OE

Reserved for test; do not modify the value of this bit.

USB_DEV_CN.fifo_mode

Field	Bits	Sys Reset	Access	Description
fifo_mode	9	0	R/W	FIFO Mode

This register bit, when set, configures the device controller to respond to an incoming IN request as soon as the device controller's FIFO become non-empty, instead of waiting for the full packet to be received by the FIFO. Setting this bit to 1 reduces the likelihood of returning NAK. This is the recommended setting.

7.5.4.4 USB_DEV_INTFL

USB_DEV_INTFL.dpact

Field	Bits	Sys Reset	Access	Description
dpact	0	0	W1C	DPLUS Activity Interrupt Flag

This interrupt flag indicates that activity has been detected on the DPLUS (D+) USB interface pin.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.rwu_dn

Field	Bits	Sys Reset	Access	Description
rwu_dn	1	0	W1C	Remote Wakeup Done Interrupt Flag

This interrupt flag indicates that the remote wakeup signalling to the USB bus host has been completed.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.bact

Field	Bits	Sys Reset	Access	Description
bact	2	0	W1C	USB Bus Activity Interrupt Flag

This interrupt flag indicates that USB bus activity has been detected (the USB controller has received a SYNC field).

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.brst

Field	Bits	Sys Reset	Access	Description
brst	3	0	W1C	USB Bus Reset In Progress Interrupt Flag

This interrupt flag indicates that a USB bus reset condition has been detected, which causes all USB registers to be cleared to their default states.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.susp

Field	Bits	Sys Reset	Access	Description
susp	4	0	W1C	USB Suspend Interrupt Flag

This interrupt flag indicates that a USB bus suspend condition has been detected.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.no_vbus

Field	Bits	Sys Reset	Access	Description
no_vbus	5	0	W1C	No VBUS Interrupt Flag

This interrupt flag indicates that VBUS has powered down (level transition from 1 to 0).

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.vbus

Field	Bits	Sys Reset	Access	Description
vbus	6	0	W1C	VBUS Detect Interrupt Flag

This interrupt flag indicates that VBUS has powered up (level transition from 0 to 1).

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.brst_dn

Field	Bits	Sys Reset	Access	Description
brst_dn	7	0	W1C	USB Bus Reset Completed Interrupt Flag

This interrupt flag indicates that a USB bus reset condition has completed.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.setup

Field	Bits	Sys Reset	Access	Description
setup	8	0	W1C	Setup Packet Interrupt Flag

This interrupt flag indicates that a SETUP packet has been received by Endpoint 0.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.ep_in

Field	Bits	Sys Reset	Access	Description
ep_in	9	0	W1C	Endpoint IN Interrupt Flag

This interrupt flag indicates that an Endpoint IN Interrupt is pending at one or more endpoints.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.ep_out

Field	Bits	Sys Reset	Access	Description
ep_out	10	0	W1C	Endpoint OUT Interrupt Flag

This interrupt flag indicates that an Endpoint OUT Interrupt is pending at one or more endpoints.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.ep_nak

Field	Bits	Sys Reset	Access	Description
ep_nak	11	0	W1C	Endpoint NAK Interrupt Flag

This interrupt flag indicates that an Endpoint NAK Interrupt is pending at one or more endpoints.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.dma_err

Field	Bits	Sys Reset	Access	Description
dma_err	12	0	W1C	DMA Error Interrupt Flag

This interrupt flag indicates that a DMA error has been detected by one or more endpoints.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.buf_ovr

Field	Bits	Sys Reset	Access	Description
buf_ovr	13	0	W1C	Buffer Overflow Interrupt Flag

This interrupt flag indicates that a buffer overflow error has been detected by one or more endpoints.

Set to 1 by hardware when the associated interrupt condition has been detected. Write 1 to clear.

USB_DEV_INTFL.vbus_st

Field	Bits	Sys Reset	Access	Description
vbus_st	16	0	R/O	VBUS Status

This status flag indicates the current VBUS level (0-low, 1-high).

7.5.4.5 USB_DEV_INTEN

USB_DEV_INTEN.dpact

Field	Bits	Sys Reset	Access	Description
dpact	0	0	R/W	DPLUS Activity Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.rwu_dn

Field	Bits	Sys Reset	Access	Description
rwu_dn	1	0	R/W	Remote Wakeup Done Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.bact

Field	Bits	Sys Reset	Access	Description
bact	2	0	R/W	USB Bus Activity Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.brst

Field	Bits	Sys Reset	Access	Description
brst	3	0	R/W	USB Bus Reset In Progress Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.susp

Field	Bits	Sys Reset	Access	Description
susp	4	0	R/W	USB Suspend Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.no_vbus

Field	Bits	Sys Reset	Access	Description
no_vbus	5	0	R/W	No VBUS Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.vbus

Field	Bits	Sys Reset	Access	Description
vbus	6	0	R/W	VBUS Detect Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.brst_dn

Field	Bits	Sys Reset	Access	Description
brst_dn	7	0	R/W	USB Bus Reset Completed Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.setup

Field	Bits	Sys Reset	Access	Description
setup	8	0	R/W	Setup Packet Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.ep_in

Field	Bits	Sys Reset	Access	Description
ep_in	9	0	R/W	Endpoint IN Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.ep_out

Field	Bits	Sys Reset	Access	Description
ep_out	10	0	R/W	Endpoint OUT Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.ep_nak

Field	Bits	Sys Reset	Access	Description
ep_nak	11	0	R/W	Endpoint NAK Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.dma_err

Field	Bits	Sys Reset	Access	Description
dma_err	12	0	R/W	DMA Error Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

USB_DEV_INTEN.buf_ovr

Field	Bits	Sys Reset	Access	Description
buf_ovr	13	0	R/W	Buffer Overflow Interrupt Enable

- 0: No interrupt will be triggered when the associated interrupt flag is set.
- 1: An interrupt will be reported to the CPU (if not otherwise masked) when the associated interrupt flag is set.

7.5.4.6 USB_EP_BASE

USB_EP_BASE.ep_base

Field	Bits	Sys Reset	Access	Description
ep_base	31:9	23'b0	R/W	USB Endpoint Descriptor Table Base Address

This field represents the base address of the endpoint descriptor table in RAM, with the physical byte address given as (EP_BASE[31:9] $*$ 512).

7.5.4.7 USB_CUR_BUF

USB_CUR_BUF.out_buf

Field	Bits	Sys Reset	Access	Description
out_buf	15:0	00000000h	R/O	OUT Transfer Current Buffers

For double-buffered endpoints, the low 8 bits of this field indicate the current buffer that the USB controller will use for an OUT transfer, as follows:

- bit is set to 0: buffer 0 will be used for this endpoint
- bit is set to 1: buffer 1 will be used for this endpoint

For single-buffered endpoints, the corresponding bit in this field will always be zero.

- bit 0 : current buffer for OUT transfers on Endpoint 0
- bit 1 : current buffer for OUT transfers on Endpoint 1
- bit 2 : current buffer for OUT transfers on Endpoint 2
- bit 3 : current buffer for OUT transfers on Endpoint 3
- bit 4 : current buffer for OUT transfers on Endpoint 4
- bit 5 : current buffer for OUT transfers on Endpoint 5
- bit 6 : current buffer for OUT transfers on Endpoint 6

- bit 7 : current buffer for OUT transfers on Endpoint 7

USB_CUR_BUF.in_buf

Field	Bits	Sys Reset	Access	Description
in_buf	31:16	00000000h	R/O	IN Transfer Current Buffers

For double-buffered endpoints, the low 8 bits of this field indicate the current buffer that the USB controller will use for an IN transfer, as follows:

- bit is set to 0: buffer 0 will be used for this endpoint
- bit is set to 1: buffer 1 will be used for this endpoint

For single-buffered endpoints, the corresponding bit in this field will always be zero.

- bit 0 : current buffer for IN transfers on Endpoint 0
- bit 1 : current buffer for IN transfers on Endpoint 1
- bit 2 : current buffer for IN transfers on Endpoint 2
- bit 3 : current buffer for IN transfers on Endpoint 3
- bit 4 : current buffer for IN transfers on Endpoint 4
- bit 5 : current buffer for IN transfers on Endpoint 5
- bit 6 : current buffer for IN transfers on Endpoint 6
- bit 7 : current buffer for IN transfers on Endpoint 7

7.5.4.8 USB_IN_OWNER

USB_IN_OWNER.buf0_owner

Field	Bits	Sys Reset	Access	Description
buf0_owner	15:0	00000000h	R/W	Owner for IN Buffer 0 for Endpoints

Bits 0 to 7 indicate the owner of the corresponding Endpoint IN buffer 0 as follows:

- 0 - Endpoint buffer is owned by application software
- 1 - Endpoint buffer is owned by the USB controller hardware
- bit 0 : owner for Endpoint 0 IN buffer 0
- bit 1 : owner for Endpoint 1 IN buffer 0
- bit 2 : owner for Endpoint 2 IN buffer 0
- bit 3 : owner for Endpoint 3 IN buffer 0
- bit 4 : owner for Endpoint 4 IN buffer 0

- bit 5 : owner for Endpoint 5 IN buffer 0
- bit 6 : owner for Endpoint 6 IN buffer 0
- bit 7 : owner for Endpoint 7 IN buffer 0

USB_IN_OWNER.buf1_owner

Field	Bits	Sys Reset	Access	Description
buf1_owner	31:16	00000000h	R/W	Owner for IN Buffer 1 for Endpoints

Bits 0 to 7 indicate the owner of the corresponding Endpoint IN buffer 1 as follows:

- 0 - Endpoint buffer is owned by application software
- 1 - Endpoint buffer is owned by the USB controller hardware
- bit 0 : owner for Endpoint 0 IN buffer 1
- bit 1 : owner for Endpoint 1 IN buffer 1
- bit 2 : owner for Endpoint 2 IN buffer 1
- bit 3 : owner for Endpoint 3 IN buffer 1
- bit 4 : owner for Endpoint 4 IN buffer 1
- bit 5 : owner for Endpoint 5 IN buffer 1
- bit 6 : owner for Endpoint 6 IN buffer 1
- bit 7 : owner for Endpoint 7 IN buffer 1

7.5.4.9 USB_OUT_OWNER

USB_OUT_OWNER.buf0_owner

Field	Bits	Sys Reset	Access	Description
buf0_owner	15:0	00000000h	R/W	Owner for OUT Buffer 0 for Endpoints

Bits 0 to 7 indicate the owner of the corresponding Endpoint OUT buffer 0 as follows:

- 0 - Endpoint buffer is owned by application software
- 1 - Endpoint buffer is owned by the USB controller hardware
- bit 0 : owner for Endpoint 0 OUT buffer 0
- bit 1 : owner for Endpoint 1 OUT buffer 0
- bit 2 : owner for Endpoint 2 OUT buffer 0
- bit 3 : owner for Endpoint 3 OUT buffer 0
- bit 4 : owner for Endpoint 4 OUT buffer 0

- bit 5 : owner for Endpoint 5 OUT buffer 0
- bit 6 : owner for Endpoint 6 OUT buffer 0
- bit 7 : owner for Endpoint 7 OUT buffer 0

USB_OUT_OWNER.buf1_owner

Field	Bits	Sys Reset	Access	Description
buf1_owner	31:16	00000000h	R/W	Owner for OUT Buffer 1 for Endpoints

Bits 0 to 7 indicate the owner of the corresponding Endpoint OUT buffer 1 as follows:

- 0 - Endpoint buffer is owned by application software
- 1 - Endpoint buffer is owned by the USB controller hardware
- bit 0 : owner for Endpoint 0 OUT buffer 1
- bit 1 : owner for Endpoint 1 OUT buffer 1
- bit 2 : owner for Endpoint 2 OUT buffer 1
- bit 3 : owner for Endpoint 3 OUT buffer 1
- bit 4 : owner for Endpoint 4 OUT buffer 1
- bit 5 : owner for Endpoint 5 OUT buffer 1
- bit 6 : owner for Endpoint 6 OUT buffer 1
- bit 7 : owner for Endpoint 7 OUT buffer 1

7.5.4.10 USB_IN_INT

USB_IN_INT.inbav

Field	Bits	Sys Reset	Access	Description
inbav	7:0	00000000b	W1C	Endpoint Buffer Available Interrupt Flags

These interrupt flags are set by the USB controller after it has successfully transferred an IN packet to the USB host and has received an ACK handshake in reply. This indicates that the endpoint buffer is now available to be written by software.

- bit 0: flag for Endpoint 0 buffer
- bit 1: flag for Endpoint 1 buffer
- bit 2: flag for Endpoint 2 buffer
- bit 3: flag for Endpoint 3 buffer
- bit 4: flag for Endpoint 4 buffer
- bit 5: flag for Endpoint 5 buffer

- bit 6: flag for Endpoint 6 buffer
- bit 7: flag for Endpoint 7 buffer

Write 1 to clear.

7.5.4.11 USB_OUT_INT

USB_OUT_INT.outdav

Field	Bits	Sys Reset	Access	Description
outdav	7:0	00000000b	W1C	Endpoint Data Available Interrupt Flags

These interrupt flags are set by the USB controller when it has successfully received an OUT packet from the host and has loaded it into the designated endpoint buffer. This indicates that the packet data is available to be read by software.

- bit 0: flag for Endpoint 0
- bit 1: flag for Endpoint 1
- bit 2: flag for Endpoint 2
- bit 3: flag for Endpoint 3
- bit 4: flag for Endpoint 4
- bit 5: flag for Endpoint 5
- bit 6: flag for Endpoint 6
- bit 7: flag for Endpoint 7

Write 1 to clear.

7.5.4.12 USB_NAK_INT

USB_NAK_INT.nak

Field	Bits	Sys Reset	Access	Description
nak	7:0	00000000b	W1C	Endpoint NAK Interrupt Flags

These int flags are set by the USB controller after it sends a NAK handshake to the host in response to an IN request. This indicates that the endpoint buffer has no data available to be transmitted to the USB host.

- bit 0: flag for Endpoint 0
- bit 1: flag for Endpoint 1
- bit 2: flag for Endpoint 2

- bit 3: flag for Endpoint 3
- bit 4: flag for Endpoint 4
- bit 5: flag for Endpoint 5
- bit 6: flag for Endpoint 6
- bit 7: flag for Endpoint 7

Write 1 to clear.

7.5.4.13 USB_DMA_ERR_INT

USB_DMA_ERR_INT.dma_err

Field	Bits	Sys Reset	Access	Description
dma_err	7:0	00000000b	W1C	Endpoint DMA Error Interrupt Flags

These int flags are set by the USB controller when a USB DMA error occurs, meaning that the USB controller was unable to transfer data to or from the corresponding endpoint over the AHB bus.

- bit 0: flag for Endpoint 0
- bit 1: flag for Endpoint 1
- bit 2: flag for Endpoint 2
- bit 3: flag for Endpoint 3
- bit 4: flag for Endpoint 4
- bit 5: flag for Endpoint 5
- bit 6: flag for Endpoint 6
- bit 7: flag for Endpoint 7

Write 1 to clear.

7.5.4.14 USB_BUF_OVR_INT

USB_BUF_OVR_INT.buf_ovr

Field	Bits	Sys Reset	Access	Description
buf_ovr	7:0	00000000b	W1C	Endpoint Buffer Overflow Interrupt Flags

These int flags are set by the USB controller when a data packet to or from the USB host is larger than the size of the corresponding endpoint buffer in memory.

- bit 0: flag for Endpoint 0

- bit 1: flag for Endpoint 1
- bit 2: flag for Endpoint 2
- bit 3: flag for Endpoint 3
- bit 4: flag for Endpoint 4
- bit 5: flag for Endpoint 5
- bit 6: flag for Endpoint 6
- bit 7: flag for Endpoint 7

Write 1 to clear.

7.5.4.15 USB_SETUP0

USB_SETUP0.byte0

Field	Bits	Sys Reset	Access	Description
byte0	7:0	00h	R/O	SETUP Packet Byte 0

Byte 0 of the last SETUP packet received by the USB controller.

USB_SETUP0.byte1

Field	Bits	Sys Reset	Access	Description
byte1	15:8	00h	R/O	SETUP Packet Byte 1

Byte 1 of the last SETUP packet received by the USB controller.

USB_SETUP0.byte2

Field	Bits	Sys Reset	Access	Description
byte2	23:16	00h	R/O	SETUP Packet Byte 2

Byte 2 of the last SETUP packet received by the USB controller.

USB_SETUP0.byte3

Field	Bits	Sys Reset	Access	Description
byte3	31:24	00h	R/O	SETUP Packet Byte 3

Byte 3 of the last SETUP packet received by the USB controller.

7.5.4.16 USB_SETUP1

USB_SETUP1.byte0

Field	Bits	Sys Reset	Access	Description
byte0	7:0	00h	R/O	SETUP Packet Byte 4

Byte 4 of the last SETUP packet received by the USB controller.

USB_SETUP1.byte1

Field	Bits	Sys Reset	Access	Description
byte1	15:8	00h	R/O	SETUP Packet Byte 5

Byte 5 of the last SETUP packet received by the USB controller.

USB_SETUP1.byte2

Field	Bits	Sys Reset	Access	Description
byte2	23:16	00h	R/O	SETUP Packet Byte 6

Byte 6 of the last SETUP packet received by the USB controller.

USB_SETUP1.byte3

Field	Bits	Sys Reset	Access	Description
byte3	31:24	00h	R/O	SETUP Packet Byte 7

Byte 7 of the last SETUP packet received by the USB controller.

7.5.4.17 USB_EP

USB_EP.ep_dir

Field	Bits	Sys Reset	Access	Description
ep_dir	1:0	11b	R/W	Endpoint Direction

For Endpoint 0 - Read-only; always set to 11b (CONTROL pipe accepting IN, OUT, and STATUS packets) for Endpoint 0.

For other endpoints (1 through 7) -

- 00b: Endpoint is disabled.
- 01b: Endpoint is configured for OUT transfers.
- 10b: Endpoint is configured for IN transfers.
- 11b: Reserved (only EP0 can be set to CONTROL mode).

USB_EP.ep_buf2

Field	Bits	Sys Reset	Access	Description
ep_buf2	3	0	R/W	Endpoint Double Buffered Enable

- 0: Endpoint is single buffered.
- 1: Endpoint is double buffered.

USB_EP.ep_int_en

Field	Bits	Sys Reset	Access	Description
ep_int_en	4	0	R/W	Endpoint Transfer Complete Interrupt Enable

1: Enable generation of interrupts for this endpoint upon completion of an IN or OUT data transfer.

USB_EP.ep_nak_en

Field	Bits	Sys Reset	Access	Description
ep_nak_en	5	0	R/W	Endpoint NAK Interrupt Enable

1: Enable generation of interrupts for this endpoint when a NAK handshake is returned to the host.

USB_EP.ep_dt

Field	Bits	Sys Reset	Access	Description
ep_dt	6	0	R/W	Endpoint Data Toggle Clear

Write to 1 to reset the data toggle field to DATA0 and reset the current buffer to buffer 0. In addition, the IN and OUT Buffer Owner bits for this endpoint will be cleared as well. Writes to 0 are ignored.

Cleared to 0 by hardware after the data clear / reset process has been completed.

USB_EP.ep_stall

Field	Bits	Sys Reset	Access	Description
ep_stall	8	0	R/W	Endpoint Stall

- 0: Endpoint is not stalled.
- 1: Endpoint is stalled.

Upon receiving a SETUP packet, this bit is automatically cleared to 0.

USB_EP.ep_st_stall

Field	Bits	Sys Reset	Access	Description
ep_st_stall	9	0	R/W	Endpoint Stall Status Stage of Control Transfer

- 0: Do not send STALL.
- 1: Send STALL to host for status stage.

Upon receiving a SETUP packet, this bit is automatically cleared to 0.

USB_EP.ep_st_ack

Field	Bits	Sys Reset	Access	Description
ep_st_ack	10	0	R/W	Endpoint Acknowledge Status Stage of Control Transfer

- 0: Do not send ACK.
- 1: Send ACK to host for status stage.

Upon receiving a SETUP packet, this bit is automatically cleared to 0.

7.6 SPI XIP

7.6.1 Overview

The **MAX32620** provides a dedicated SPI master interface which allows the CPU to transparently execute instructions stored in an external SPI flash device. This interface for external memory, known as SPI Execute In Place (SPIX), can also be used to access large amounts of static data that would otherwise reside in internal flash memory. Instructions fetched via the SPIX interface will be cached and executed in the same manner as instructions fetched from the internal program flash memory.

In order for the external SPI flash memory to be accessed in this manner, the SPIX master must be configured by firmware with the proper protocol, mode, and timing settings specific to the SPI flash memory device being used. Once the SPIX has been configured correctly, the external SPI flash memory will be mapped to a dedicated area in the standard code memory space, and instruction or data fetches from this area will be handled automatically by the SPIX module; no additional handling is required by the application firmware.

The SPI master provided by the SPIX module is a single-purpose master interface optimized to quickly translate instruction fetches (over the I-Code bus) or data fetches (over the D-Code bus) into high-speed data read sequences from an external SPI flash memory device. It is important to note that this SPI master cannot be used to program or configure external SPI flash memory devices: if the application needs to access the external SPI flash memory to perform other functions (e.g., erase, write, read/write configuration settings, set/clear write protect modes), it must do so using one of the general-purpose SPI master interfaces".

The SPIX module can support up to three multiple external SPI flash memory devices, each with its own slave select signal. However, all external memory devices accessed via the SPIX interface are mapped to the same code memory region, so only one external memory device may be used at a time. Additionally, switching between one external memory device and another (i.e., changing the active slave select signal) must be performed manually by firmware.

Features provided by the SPIX module include the following:

- Transparent access to code/data stored in external SPI flash memory devices.
- Up to 16MB of external memory supported per slave device (switching between slave devices, or other device-specific paging mechanisms, must be handled manually by firmware).
- Dedicated SPI master interface (supporting SPI Mode 0 and Mode 3) with up to three slave selects.
- Flexible interface with single I/O, dual I/O, and quad I/O modes available.
- Configurable command sequence allows multiple SPI memory flash device families to be supported.
- Interface widths for command, address/mode, and data read portions of the SPI memory read sequence can be set independently; for example, the SPIX can send read command and address can be sent using a single SDIO line and then switch to quad I/O mode for reading back the flash memory data.
- Mode clock feature allows a configurable number of 'dummy clocks' from 0 to 15 to be sent in between the address and data portions of the read sequence; this period may optionally be used to transmit device-specific configuration information to the SPI flash memory device if needed.

7.6.2 SPIX Pin Configuration

All signal pins used by the SPIX interface are multiplexed with GPIO as shown in the table below. Before the SPIX interface can be used, the proper settings must be made in the I/O Manager to request connections between the needed GPIO pins and the SPIX pin functions.

SPIX Pin Mapping

Logic Signal	Port and Pin	Setting to Enable using I/O Manager
SCLK	P1.0	Write IOMAN_SPIX_REQ.core_io_req to 1
SS[0]	P1.3	Write IOMAN_SPIX_REQ.ss0_io_req to 1
SS[1]	P3.6	Write IOMAN_SPIX_REQ.ss1_io_req to 1
SS[2]	P3.7	Write IOMAN_SPIX_REQ.ss2_io_req to 1
SDIO[0]	P1.1	Write IOMAN_SPIX_REQ.core_io_req to 1
SDIO[1]	P1.2	Write IOMAN_SPIX_REQ.core_io_req to 1
SDIO[2]	P1.4	Write IOMAN_SPIX_REQ.quad_io_req to 1
SDIO[3]	P1.5	Write IOMAN_SPIX_REQ.quad_io_req to 1

7.6.3 Device Requirements for Use with SPIX

In order for an external SPI flash memory device to be accessed via the SPIX interface and mapped into memory, it must meet the following interface requirements.

- SPI slave supporting Mode 0 or Mode 3 SPI clock formatting.
- Byte addressable memory, 24-bit memory address.
- Accessible using a read data sequence consisting of an 8-bit command code (transmitted by SPIX master), a 24-bit byte address (transmitted by SPIX master), an optional mode clock phase (0 to 15 'dummy clocks'), and a read data phase (SPIX master reads data transmitted by SPI flash memory slave).
- Single I/O, dual I/O, or quad I/O access modes supported.
- Any number of memory data bytes can be returned by a single read command (slave continues to transmit memory data bytes until the sequence is ended by deassertion of slave select).

7.6.4 SPIX Memory

7.6.4.1 SPIX Memory Mapping

The currently selected SPI flash memory device (accessed via the SPIX module) will be mapped into code and data memory starting at byte address 1000_0000h - beyond the end of the internal flash program memory space. Access to the SPI flash memory using this memory mapped region is always read-only: the SPIX interface cannot be used to program or erase locations in the external SPI flash memory device.

The SPIX module does not provide any mechanism for specifying the size of external SPI flash memory devices. The maximum supported size of memory that can be accessed via the SPIX interface is 16MB. Access to a given location in SPIX mapped memory is translated to a 24-bit, device-local address by removing the base

memory offset of 1000_0000h; thus, an access to location 1000_0100h in ARM memory space will be translated to a read from the external flash memory address location 000100h.

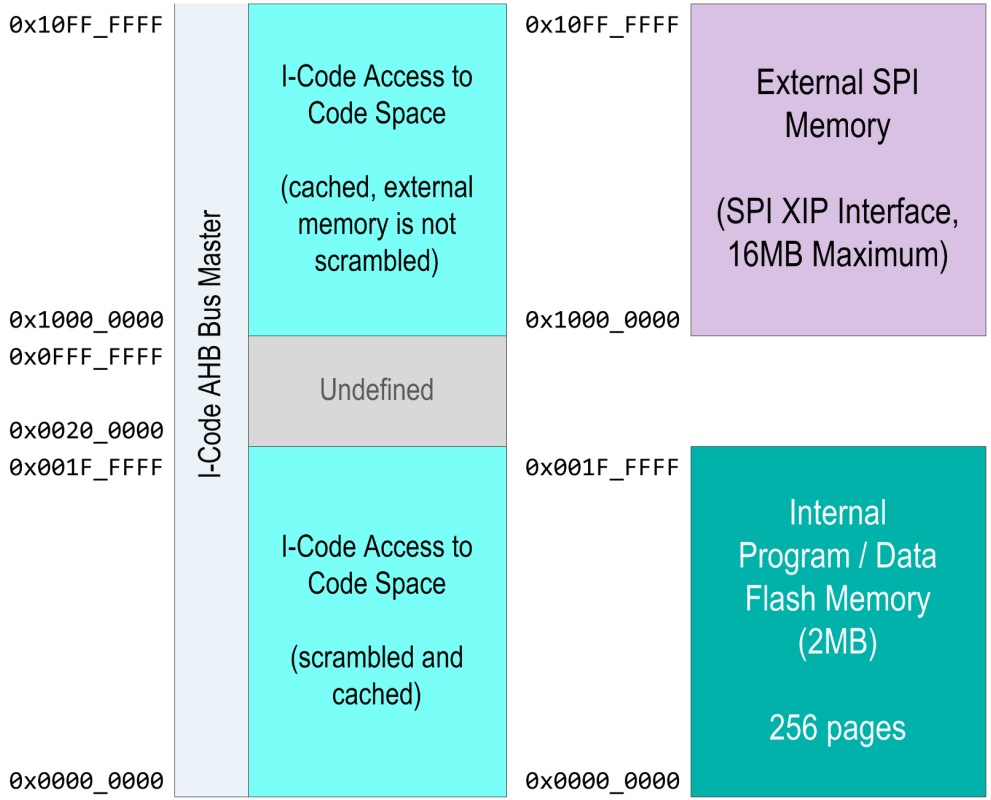

Figure 7.18: SPIX External Memory Mapping in Code Space

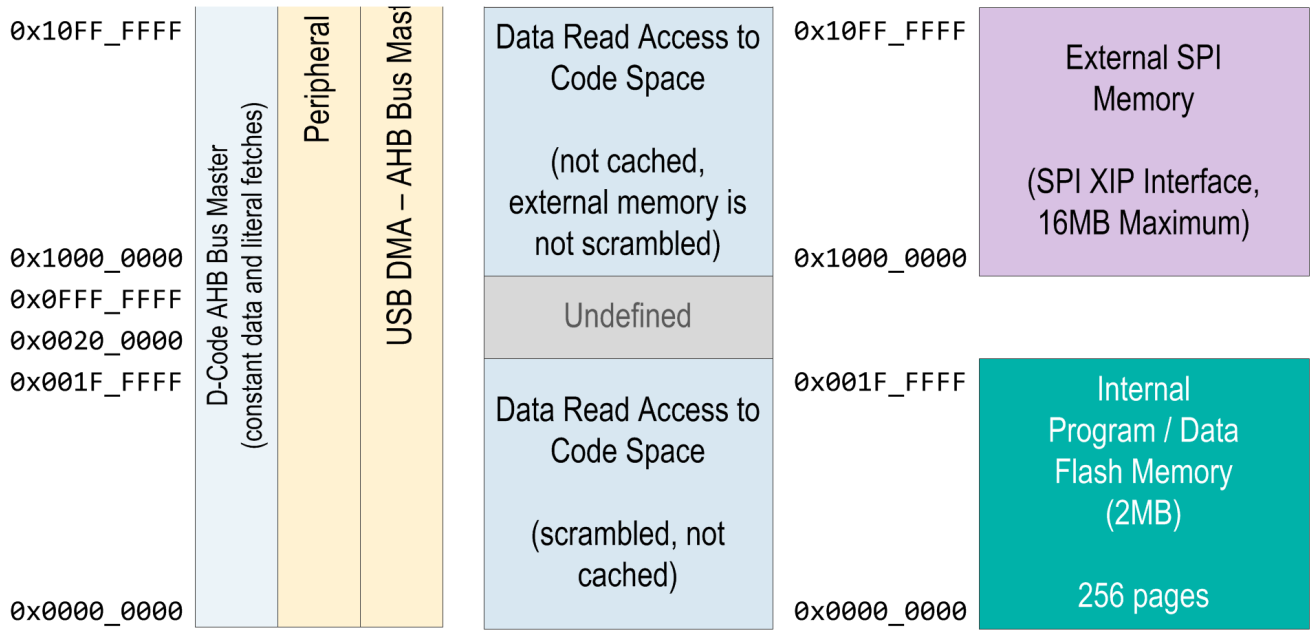

Figure 7.19: SPIX External Memory Mapping in Data Space

7.6.4.2 SPIX External Memory Caching and Scrambling

Unlike the contents of internal flash memory, the scrambling/descrambling mechanism for flash memory contents and location is not used when accessing SPI external memory devices via the SPIX interface. This means that all contents of the SPI external memory are stored as unencrypted, unscrambled data.

As with data stored in any other memory location, it is possible for the application to encrypt data stored in the external SPI flash memory using AES or another algorithm. However, this must be implemented in firmware by the application developer. Only external flash memory data can be protected in this manner: application code stored in external memory must be unencrypted in order to be directly readable by the SPIX interface.

Instruction code stored in an external flash memory device and fetched via the SPIX interface is cached in the same manner as instruction code fetched from the internal program flash memory. If the SPIX module is reconfigured to switch between external SPI flash memory devices, or if the contents of the external SPI flash memory are modified, then the instruction cache must be flushed in order to ensure that future instruction fetches from the SPIX mapped memory space do not return invalid data.

7.6.4.3 SPIX Memory Access

The SPIX master interface reads data from the SPI external flash memory device using the read sequence defined in the following sections. Certain phases of this sequence may be optional under some circumstances for a given device; for example, some devices may support skipping the command phase after the first read access has been performed, while other devices may not require a mode clock phase or may allow special device-specific configuration data or commands to be sent during the mode clock phase.

At a minimum, the address phase and data read phase are required in order to read data from the external memory device. When reading code or data, the SPIX interface will read from the SPI external flash memory in 128-bit blocks (i.e., 16 bytes, or four 32-bit doublewords). For the instruction fetches, this allows the resulting code data to be cached. Data memory fetches are not cached in the same manner as instruction fetches, but a 128-bit data buffer is used to reduce the number of external SPI memory fetches required.

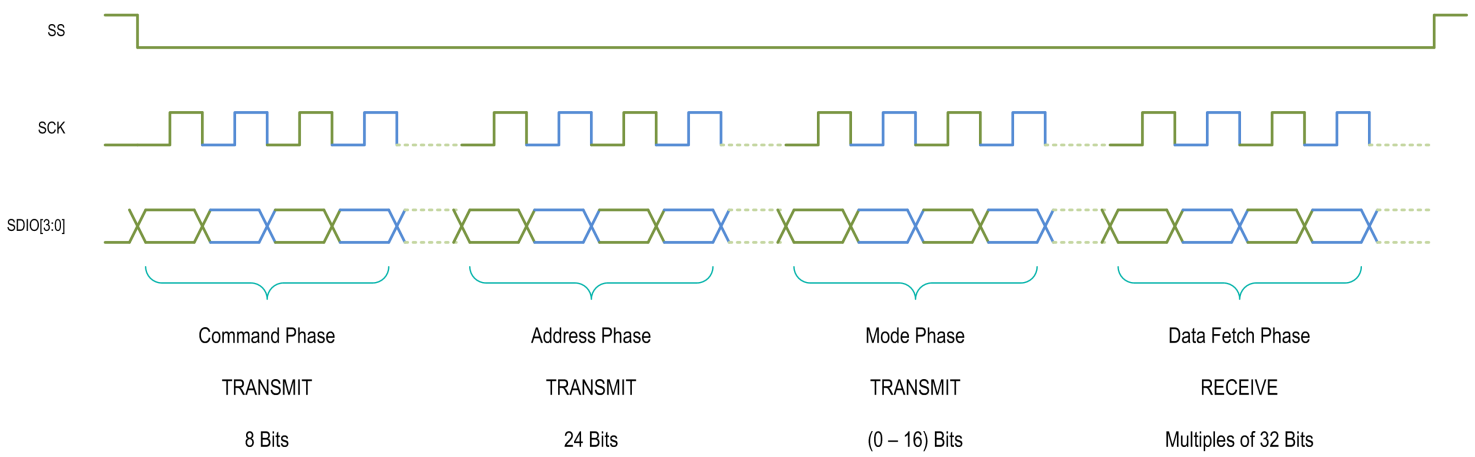

Figure 7.20: SPIX Memory Read Transaction

7.6.4.4 Configuring SPIX Memory Access

Typically, SPIX configuration parameters are set once following system reset, corresponding to the particular SPI flash memory device being used, and there is no need to modify registers in the SPIX module after initial configuration. The exception comes if more than one external SPI flash memory device will be used, in which case the active slave select setting must be modified to switch from one external SPI flash memory device to another.

7.6.5 SPIX Clock Selection and Clock Gating

The SPIX module in the **MAX32620** operates from a module clock which is derived from the main system clock source. This clock source is configured using the register CLKMAN_SYS_CLK_CTRL_2_SPIX. By default, the SPIX module clock is disabled; the spix_clk_scale field in this register is used both to enable the module clock and to select the divide down rate (if any) used when deriving the module clock from the system clock source.

SPIX Module Clock Control

spix_clk_scale	SPIX Module Clock Frequency
0	SPIX Module Clock is disabled
1	System Clock Source / 1
2	System Clock Source / 2
3	System Clock Source / 4
4	System Clock Source / 8
5	System Clock Source / 16
6	System Clock Source / 32
7	System Clock Source / 64
8	System Clock Source / 128
9	System Clock Source / 256
other	Reserved

In order to optimize power consumption, the SPIX module uses a dynamic clock gating scheme which automatically turns off the local module clock whenever the SPIX is inactive. Normally, this dynamic mode (default setting) should remain set. If there is any reason to modify the module clock gating mode, this can be done by setting CLKMAN_CLK_GATE_CTRL0.spix_clk_gater as shown in the table below.

SPIX Module Clock Gating Control

owm_clk_gater	Setting
0	**Clock forced off** - SPIX clock is disabled

owm_clk_gater	Setting
1	**Dynamic Clock Gating Enabled** - SPIX clock is active only when the SPIX module is in use (default)
2, 3	**Clock forced on** - SPIX clock is enabled at all times

7.6.5.1 SPIX Protocol Format and Timing

The SPI clock timing and format modes for the SPIX module are set in the same manner as the identical mode settings used by the general-purpose SPI master modules, with the following exceptions:

- SPI clock format must be set to either Mode 0 or Mode 3.
- Alternate timing settings for SCK High and SCK Low will be used (if enabled) only during the read data phase.

7.6.6 SPIX Configuration

Configuration Settings for Command Phase

The first phase of each memory read sequence is normally the command phase as shown below. In this phase, the SPIX master sends an 8-bit command code (specified using the register field SPIX_FETCH_CTRL.cmd_value) to the SPI flash memory device to begin the read sequence. The format and meaning of this command code are device-specific.

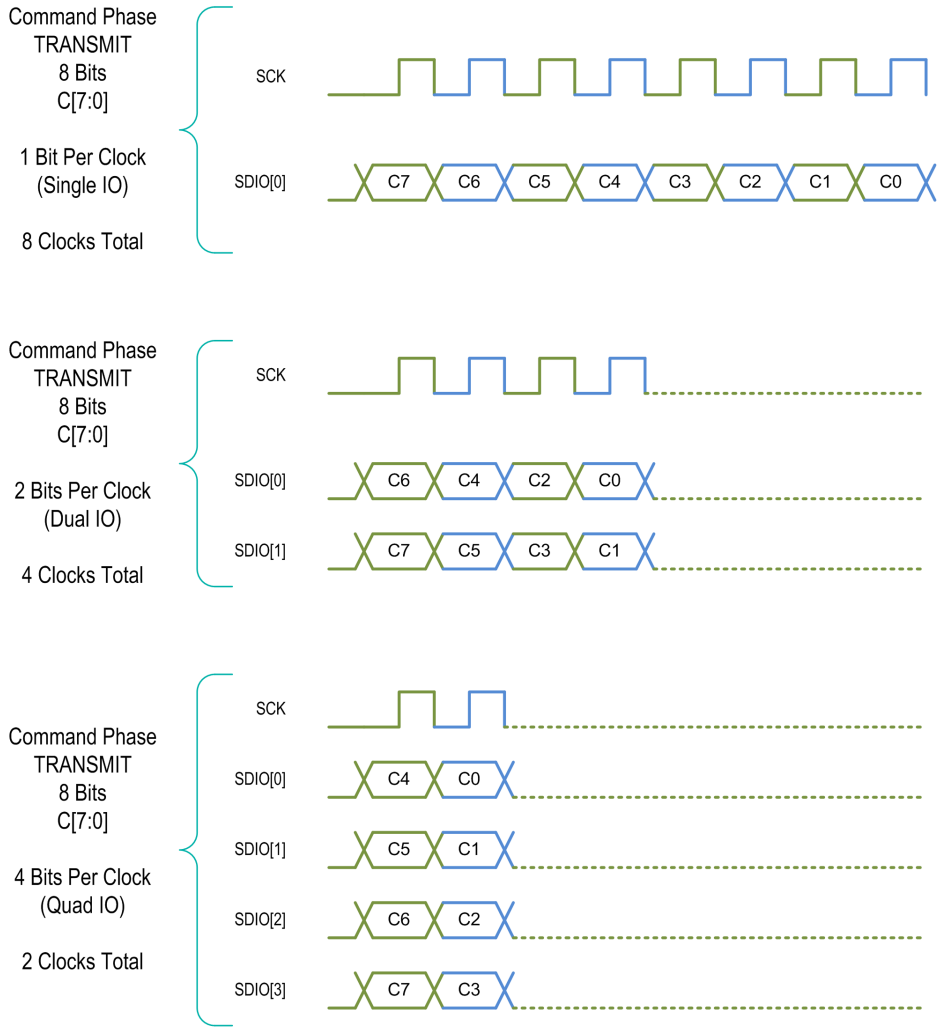

Figure 7.21: SPIX Memory Read - Command Phase

The 8-bit command code may be sent in single-wire, dual I/O, or quad I/O mode. This is determined by the setting of the command width field SPIX_FETCH_CTRL.cmd_width. This may be the same as or different than the I/O widths used for the address/mode phase and the data read phase: I/O widths for each phase are specified independently.

Some SPI flash memory devices allow the command phase to be skipped (reusing the previously transmitted command code) after the command code has been set the first time. To enable this mode for the SPIX interface, set the SPIX_MODE_CTRL.no_cmd_mode bit to 1. After this bit has been set to 1, the command code will be transmitted only once (on the next SPIX read sequence that takes place) and will be skipped after that. Rewriting the bit to 1 repeats the sequence; when the bit is cleared to 0, the command phase is included in every read sequence.

Note For no_cmd_mode to work correctly, the field must be configured in concert with the control fields in SPIX_MODE_DATA. If SPIX_MODE_DATA stipulates enable no_cmd_mode, the SPI flash will not expect a command on the next slave select assertion; if SPIX_MODE_DATA does not stipulate enable no_cmd_mode, the SPI flash will expect a command on the next slave select assertion. The master's behavior here is defined in the no_cmd_mode field. Further information is provided below in the *Configuration Settings for Mode Clock Phase* section.

Configuration Settings for Address Phase

The address phase normally follows the command phase in a read sequence, except when the command phase is skipped as detailed above; in this case, the address phase will be the first phase in the read sequence.

In the address phase, the SPIX master sends a 24-bit byte address (MSB first) to the SPI flash memory device corresponding to the code/data location that was accessed in the memory mapped SPIX region. The address may be sent using single-I/O, dual-I/O, or quad-I/O mode depending on the setting of the SPIX_FETCH_CTRL.addr_width field. This field also controls the width setting for any data transmitted during the mode clock phase.

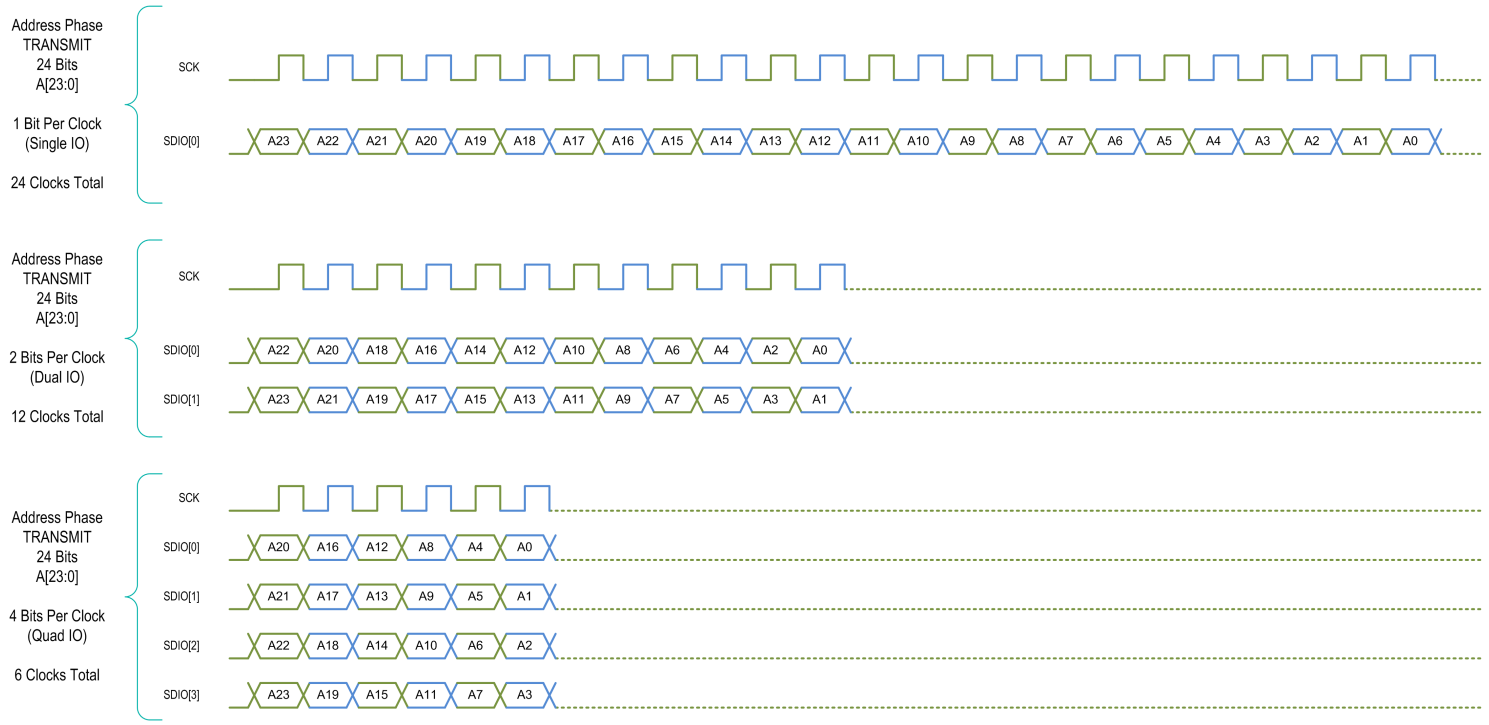

Figure 7.22: SPIX Memory Read - Address Phase

Configuration Settings for Mode Clock Phase

The mode clock phase is optional; if it is used, then it contains a number of mode clocks (also known as 'dummy clocks') from 1 to 15, specified by setting the field SPIX_MODE_CTRL.mode_clocks. If this field is set to 0 (the default), no mode clock phase will be used.

The I/O width of the mode clock phase is determined by the width setting of the address phase. By default, no data is transmitted during the mode clock phase. For many devices, the mode clock phase is simply a time padding mechanism to allow the SPI flash memory time to begin accessing the memory contents before the SPIX master begins clocking the data bytes out.

Some devices allow special configuration data or mode settings to be made by the SPI master during this phase. In order for this to occur, the data (contents of which

are device-specific) must be transmitted along with the mode clocks. In order to enable sending of this optional data, the proper data bits (up to 16 are supported) must be written to SPIX_MODE_DATA.mode_data_bits, while for each bit of the mode_data_bits field that needs to actually be transmitted, a 1 bit must be written to the corresponding position in SPIX_MODE_DATA.mode_data_oe.

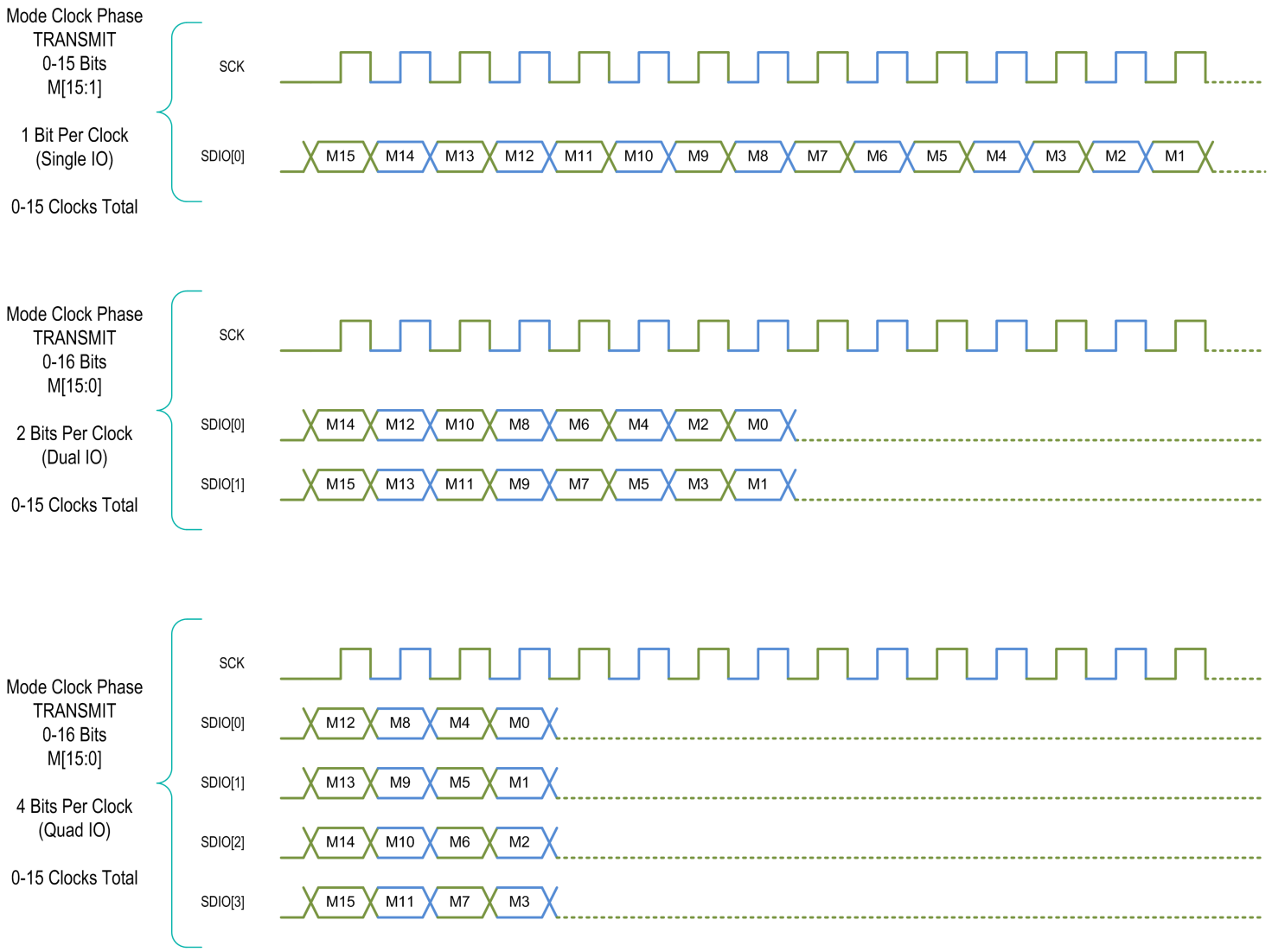

Figure 7.23: SPIX Memory Read - Mode Clock Phase

Note that as with all values transmitted over the SPIX interface, the MSB of the mode data will be sent first. This means that in order to transmit N bits during the mode clock phase, the highest N bits of SPIX_MODE_DATA.mode_data_bits must be set to the bit values to be transmitted (in order starting with the MSB), and the highest N bits of SPIX_MODE_DATA.mode_data_oe must be set to 1.

For example, in order to transmit the value 55h (MSB first) during the mode clock phase (which will require a minimum of eight mode clocks in single-wire, four mode clocks in dual I/O mode, and two mode clocks in quad I/O mode), the SPIX_MODE_DATA fields must be set as follows:

- mode_data_bits: 5500h
- mode_data_oe: FF00h

Configuration Settings for Read Data Phase

In the final phase of the SPI flash memory read sequence, the SPIX switches from transmit to receive mode, and the SPI flash memory begins clocking out the data read from the specified location one byte at a time. The width of this transfer is specified by SPIX_FETCH_CTRL.data_width. After 16 bytes have been read (for the 128-bit wide data buffer or cache line buffer fill), the SPIX master will end the SPI transaction by deasserting the slave select to the SPI flash memory.

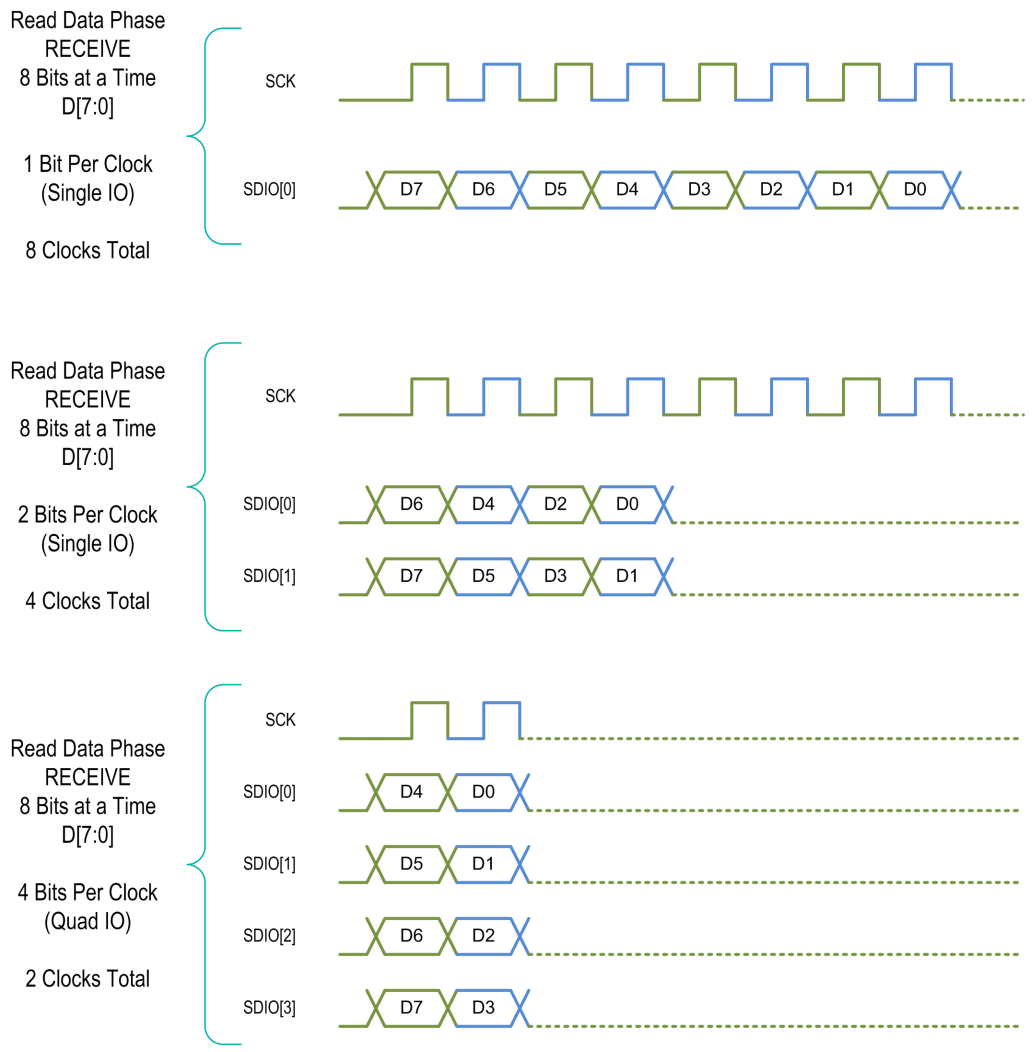

Figure 7.24: SPIX Memory Read - Read Data Phase

7.6.7 Registers (SPIX)

Address	Register	Access	Description	Reset By
0x40004000	SPIX_MASTER_CFG	R/W	SPIX Master Configuration	Sys
0x40004004	SPIX_FETCH_CTRL	R/W	SPIX Fetch Control	Sys
0x40004008	SPIX_MODE_CTRL	R/W	SPIX Mode Control	Sys
0x4000400C	SPIX_MODE_DATA	R/W	SPIX Mode Data	Sys

7.6.7.1 SPIX_MASTER_CFG

SPIX_MASTER_CFG.spi_mode

Field	Bits	Sys Reset	Access	Description
spi_mode	1:0	00b	R/W	SPIX Mode

Defines SPI clock polarity and phase; valid settings are 0, 3.

- 00b: SCK is active high, data is sampled on clock rising edge.
- 11b: SCK is active low, data is sampled on clock rising edge.

SPIX_MASTER_CFG.ss_act_lo

Field	Bits	Sys Reset	Access	Description
ss_act_lo	2	0	R/W	SPIX Slave Select Polarity

Slave Selects are active low when set.

- 0: Enabled slave select (SS) is active high.
- 1: Enabled slave select (SS) is active low.

SPIX_MASTER_CFG.alt_timing_en

Field	Bits	Sys Reset	Access	Description
alt_timing_en	3	0	R/W	Alternate Timing Mode Enable

Enables use of alternate timing for SCK generation, when the fetch protocol dictates its use.

- 0: Alternate timing is disabled.
- 1: Alternate timing will be enabled automatically when needed.

SPIX_MASTER_CFG.slave_sel

Field	Bits	Sys Reset	Access	Description
slave_sel	6:4	000b	R/W	SPIX Slave Select

Identifies which SPI slave to fetch from. Three slave selects are supported on this implementation.

- 000b: SS0
- 001b: SS1
- 010b: SS2
- else: reserved

SPIX_MASTER_CFG.sck_hi_clk

Field	Bits	Sys Reset	Access	Description
sck_hi_clk	11:8	0000b	R/W	SCK High Clocks

Number of SPIX module clocks SCK will be in the active state (determined by clock polarity setting). If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the SPIX module clock.

Note The SPIX module clock is derived (and optionally divided down from) the current system clock source, with the divide ratio selected by CLKMAN_SYS_CLK_CTRL_2.

SPIX_MASTER_CFG.sck_lo_clk

Field	Bits	Sys Reset	Access	Description
sck_lo_clk	15:12	0000b	R/W	SCK Low Clocks

Number of SPIX module clocks SCK will be in the inactive state (determined by clock polarity setting). If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the SPIX module clock.

Note The SPIX module clock is derived (and optionally divided down from) the current system clock source, with the divide ratio selected by CLKMAN_SYS_CLK_CTRL_2.

SPIX_MASTER_CFG.act_delay

Field	Bits	Sys Reset	Access	Description
act_delay	17:16	00b	R/W	SS Active Timing

Controls the delay from automatic assertion of slave select (SS) to the beginning of the SCK clock pulses as well as the delay from the end of the clock pulses until the automatic deassertion of SS.

- 00b: 0 SPIX module clocks
- 01b: 2 SPIX module clocks
- 10b: 4 SPIX module clocks
- 11b: 8 SPIX module clocks

SPIX_MASTER_CFG.inact_delay

Field	Bits	Sys Reset	Access	Description
inact_delay	19:18	00b	R/W	SS Inactive Timing

Controls the delay between deassertion of SS at the end of a transaction and reassertion of SS to begin the next transaction, for back-to-back SPI transactions.

- 00b: 0 SPIX module clocks
- 01b: 2 SPIX module clocks
- 10b: 4 SPIX module clocks
- 11b: 8 SPIX module clocks

SPIX_MASTER_CFG.alt_sck_hi_clk

Field	Bits	Sys Reset	Access	Description
alt_sck_hi_clk	23:20	0000b	R/W	Alt SCK High Clocks

Number of SPIX module clocks SCK will be in the active state (determined by clock polarity setting), when alternate timing generation is enabled. If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the SPIX module clock.

Note The SPIX module clock is derived (and optionally divided down from) the current system clock source, with the divide ratio selected by CLKMAN_SYS_CLK_CTRL_2.

SPIX_MASTER_CFG.alt_sck_lo_clk

Field	Bits	Sys Reset	Access	Description
alt_sck_lo_clk	27:24	0000b	R/W	Alt SCK Low Clocks

Number of SPIX module clocks SCK will be in the inactive state (determined by clock polarity setting), when alternate timing generation is enabled. If this is set to 0, "Fast Mode" is enabled, in which SCK is a gated version of the SPIX module clock.

Note The SPIX module clock is derived (and optionally divided down from) the current system clock source, with the divide ratio selected by CLKMAN_SYS_CLK_CTRL_2.

7.6.7.2 SPIX_FETCH_CTRL

SPIX_FETCH_CTRL.cmd_value

Field	Bits	Sys Reset	Access	Description
cmd_value	7:0	00h	R/W	Command Value

Command value to send to target to initiate fetch.

SPIX_FETCH_CTRL.cmd_width

Field	Bits	Sys Reset	Access	Description
cmd_width	9:8	00b	R/W	Command Width

Number of data I/O used to send command.

- 00b: 1
- 01b: 2
- 10b: 4

SPIX_FETCH_CTRL.addr_width

Field	Bits	Sys Reset	Access	Description
addr_width	11:10	00b	R/W	Address Width

Number of data I/O used to send address.

- 00b: 1
- 01b: 2
- 10b: 4

SPIX_FETCH_CTRL.data_width

Field	Bits	Sys Reset	Access	Description
data_width	13:12	00b	R/W	Data Width

Number of data I/O used to receive data.

- 00b: 1
- 01b: 2
- 10b: 4

7.6.7.3 SPIX_MODE_CTRL

SPIX_MODE_CTRL.mode_clocks

Field	Bits	Sys Reset	Access	Description
mode_clocks	3:0	0000b	R/W	Mode Clocks

Number of SPI clocks needed during mode phase of fetch.

SPIX_MODE_CTRL.no_cmd_mode

Field	Bits	Sys Reset	Access	Description
no_cmd_mode	8	0	R/W	No Command Mode

Read command sent only once after this bit is set.

7.6.7.4 SPIX_MODE_DATA

SPIX_MODE_DATA.mode_data_bits

Field	Bits	Sys Reset	Access	Description
mode_data_bits	15:0	0	R/W	Mode Data

Data to send with Mode clocks. Width of mode transmission is defined by SPIX_FETCH_CTRL.addr_width.

SPIX_MODE_DATA.mode_data_oe

Field	Bits	Sys Reset	Access	Description
mode_data_oe	31:16	0	R/W	Mode Output Enable

Output Enable state for each corresponding data bit in Mode Data.

8 Analog to Digital Converter

8.1 Analog to Digital Converter Overview

The **MAX32620** 32620 features an onboard 10-bit analog-to-digital converter (ADC) with a 10-channel analog input multiplexer as shown in the ADC Subsystem Diagram below. This multiplexer selects an input channel from four external input lines and four internal power supply monitors.

An integrated bandgap and buffer can provide the reference voltage; alternately, the buffer can be disabled and an external reference can be provided at the V_{REF} pad. The **MAX32620** 10-bit ADC supports the following features:

- 8MHz clock rate
- Fixed sample rate control (1024 clock cycles)
- Programmable out of range (limit) detection
- Operation in LP3:RUN mode or low-power LP2:PMU mode with PMU control

8.2 Analog to Digital Converter Architecture

The ADC on the **MAX32620** is a sigma-delta analog-to-digital converter with a 10:1 input multiplexer, high impedance input buffer, and integrated reference generator. The input multiplexer supports four external inputs (AIN0-3) and four internal power supply monitors (V_{DDB}, V_{DD18}, V_{DD12}, and V_{RTC}). The optional input buffer is high input impedance and can be used to minimize the capacitive loading of high impedance external inputs.

The ADC is a first order sigma-delta converter. It operates off an internally generated 8MHz clock and requires 1024 clock cycles for a conversion. The ADC reference voltage can be based on an internal bandgap reference or an external pin.

ADC offset and gain errors are factory trimmed, and the trim settings are stored in the internal flash info block memory. These values are automatically transferred to the ADC controller upon power-up. Gain error is trimmed such that the total errors of the ADC, bandgap, and reference buffer are calibrated out. A value of 1.200V should always be used as "Full Scale", regardless of the voltage at V_{REF} pin.

Due to its low-power design, the reference buffer cannot drive a resistive external load, nor can it drive significant external capacitance without affecting settling time. If V_{REF} is to be output and used externally, it must be externally buffered. Do not place an external bypass capacitor on this pin.

ADC sampling capacitors are switched at the 8MHz clock rate; this creates a small dynamic current and settling time requirement at the AIN pins. This behavior also establishes an upper limit to the source impedance of the signal. A source impedance of up to 10kOhm can directly drive the ADC. For high-impedance paths (e.g., resistor divided supply monitor) an internal buffer is provided. Normally, this buffer is powered down and the input mux is connected directly to the ADC. Alternately, any of the muxed inputs can pass through the buffer. This buffer is chopped to minimize offsets and low frequency noise. A PMOS-only input structure with a charge-pumped supply allows rail-rail inputs without crossover distortion. However, note that this unity-gain buffer is limited by its output swing, and therefore the practical signal limits are 50mV to (V_{DD}-50mV) before distortion may occur. For signals that must go to ground, the buffer should be bypassed.

The ADC supports an input gain of 1x (unity) or 2x. For signals less than half the voltage reference, the 2x gain can provide additional resolution. For unity gain, set

ADC_CTRL.adc_scale to 1 and ADC_CTRL.adc_refscl to 1. For 2x gain, set ADC_CTRL.adc_scale to 0 and ADC_CTRL.adc_refscl to 1.

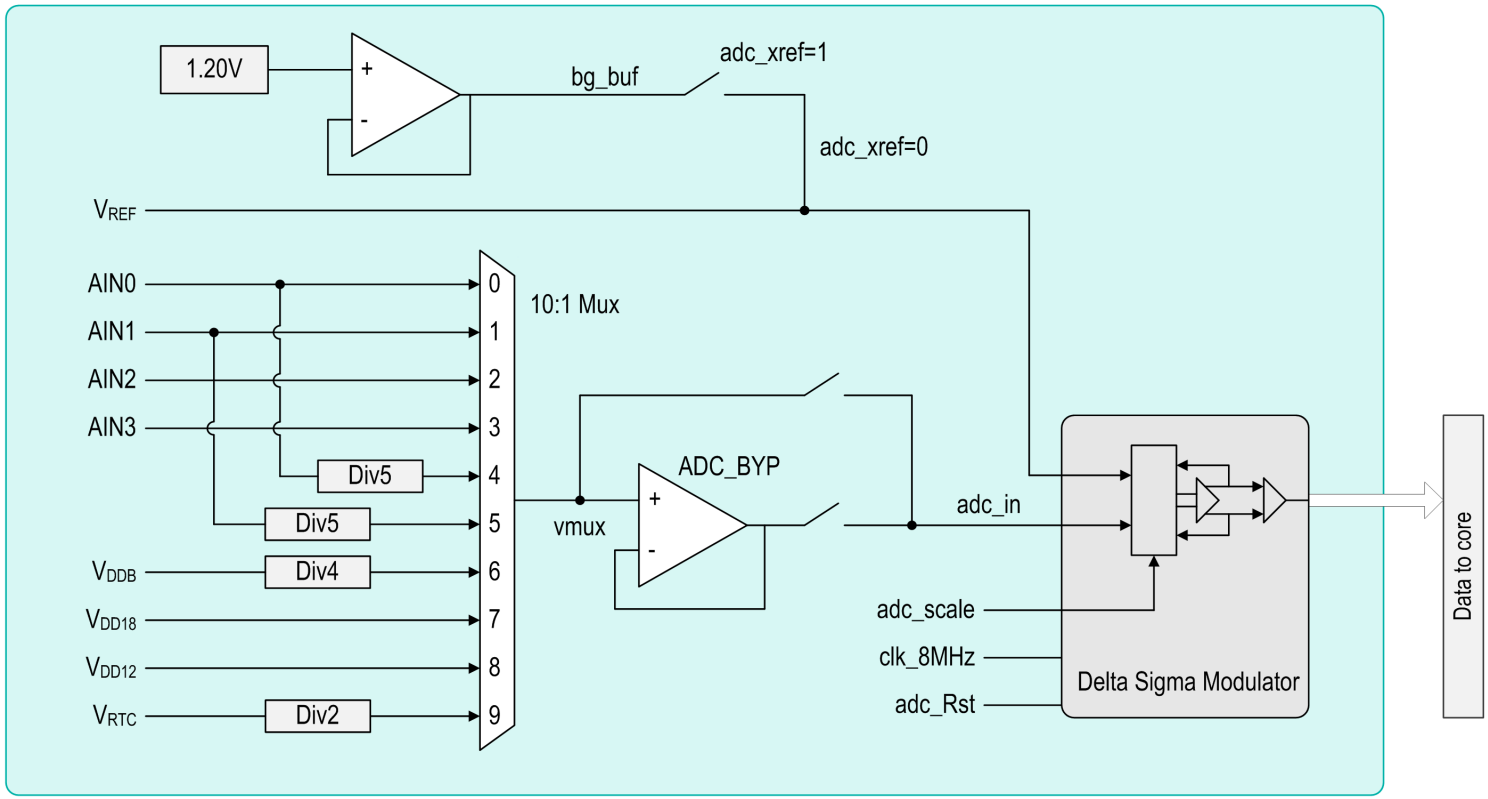

Figure 8.1: ADC Subsystem Diagram

8.3 Analog to Digital Converter Operation

8.3.1 Control Interface

The control registers for the ADC are used by application firmware to setup, begin, and retrieve the results of ADC conversion operations.

The ADC control interface enables:

- Configuration of the multiplexer, buffer, ADC, and reference.
- Power control and generation of interrupt when the charge pump power up delay is complete.
- ADC serial data stream processing and trim calibration.
- ADC conversion start and conversion done interrupt generation.
- ADC data limits and interrupt generation.

After an ADC conversion is completed, the data is available and accessible via the ADC_DATA register. The 10-bit data can be MSB or LSB justified in the 16-bit register field. The ADC data value is a function of the input voltage, the reference voltage, and optional data scaling.

8.3.2 Using an External Reference

An external reference voltage, typically 1.23V, can be driven to the VREF pin. For this configuration, set ADC_CTRL.adc_xref to 1 (before setting ADC_CTRL.adc_refbuf_pu to 1) to prevent contention on the VREF pin.

The ADC gain error trim registers are used to correct for internal bandgap variation and reference buffer offset when using the internal ADC reference. When using an external reference, all four gain error trim registers should be set to the centered value of 0x200, as follows.

- Set TRIM_REG11_ADC_TRIM0.adctrim_x0r0 to 0x200.
- Set TRIM_REG11_ADC_TRIM0.adctrim_x1r0 to 0x200.
- Set TRIM_REG12_ADC_TRIM1.adctrim_x0r1 to 0x200.
- Set TRIM_REG12_ADC_TRIM1.adctrim_x0r1 to 0x200.

Note The offset error trim, TRIM_REG12_ADC_TRIM1.adctrim_dc, must remain the same whether an internal or external reference is being used. The value of this field should not be modified by application firmware.

8.4 Analog to Digital Converter Configuration

Note The ADC input buffer (ADC_CTRL.buf_pu) should be enabled if the output impedance of the measured source is >10kohm.

8.4.1 Power-Up Sequence

The following describes the process necessary to apply power to the ADC and the AFE and enable operation. Before the ADC can be utilized, its clock must be enabled and power must be applied. Note that the ADC's charge pump requires a 10μs delay from initial power-up before it is in steady state and available for use. The interface detects when the ADC circuits are enabled and will generate an interrupt afe_pwr_up_dly; once the programmable delay has passed, the ADC is ready for use. See ADC_CTRL and ADC_INTR register documentation for further details.

- Enable power to AFE - PWRMAN_PWR_RST_CTRL.afe_powered = 1
- Enable dynamic clock gating to ADC - CLKMAN_CLK_GATE_CTRL2.adc_clk_gater = 01b
- Enable ADC clock - CLKMAN_CLK_CTRL.adc_clock_enable = 1
- Enable ADC clock interface clock - ADC_CTRL.adc_clk_en = 1
- Clear AFE ready interrupt - ADC_INTR.adc_ref_ready_if = 1
- Enable AFE ready interrupt - ADC_INTR.adc_ref_ready_ie = 1
- If using an external reference, set ADC_CTRL.adc_xref to 1
- Enable Power - ADC_CTRL[4:1]=0xF
  - ADC_CTRL.adc_pu = 1
  - ADC_CTRL.buf_pu = 0 (default); set to 1 to enable ADC input buffer
  - ADC_CTRL.adc_refbuf_pu = 1
  - ADC_CTRL.adc_chgpump_pu = 1
  - *NOTE:* ADC_CTRL.buf_bypass != ADC_CTRL.buf_pu
- Wait for interrupt, AFE ready once seen - ADC_CTRL.afe_pwr_up_dly
- Disable AFE ready interrupt - ADC_INTR.adc_ref_ready_ie = 0

8.4.2 Conversion Process

The ADC is ready for data conversion following the power-up sequence. The following delineates the steps necessary to use the ADC for data conversion.

- Configure the ADC for:
  - Channel selection - ADC_CTRL.adc_chsel
  - Data or reference scaling - ADC_CTRL.adc_scale, ADC_CTRL.adc_refscl
  - Buffer enable/disable - ADC_CTRL.buf_pu
  - Data alignment - ADC_CTRL.adc_dataalign
  - Limits: enable, channel, Hi/Lo limits - See ADC Data Limits below
- Clear the ADC interrupt done flag - ADC_INTR.adc_done_if = 1
- Enable the ADC interrupt done - ADC_INTR.adc_done_ie = 1
- Start the ADC - ADC_CTRL.cpu_adc_start = 1
- Wait for ADC done interrupt
- Read the result from ADC_DATA register (Note: repeat *Channel Selection* step to sample additional channels)

8.4.3 Peripheral Clock Configuration

There are two steps needed to enable the 8MHz frequency clock that is needed by the ADC. First, the ADC clock must be enabled in the clock manager by setting CLKMAN_CLK_CTRL.adc_clock_enable = 1. Second, the ADC clock must be enabled at the ADC module level by setting ADC_CTRL.adc_clk_en = 1.

8.4.4 Firmware Control of the ADC Sample Rate

The ADC requires 1024 clocks at 8MHz for a single conversion. The theoretical maximum sample rate is 7.812kHz. If a periodic sample rate is desired, a firmware solution with a timer resource must be utilized.

8.4.5 Power-Down Sequence

The following steps are needed when powering down the ADC.

- Set ADC_CTRL.adc_pu = 0
- Set ADC_CTRL.buf_pu = 0
- Set ADC_CTRL.adc_refbuf_pu = 0
- Set ADC_CTRL.adc_chgpump_pu = 0
- *NOTE:* ADC_CTRL.buf_bypass != ADC_CTRL.buf_pu
- Disable ADC clock interface clock - ADC_CTRL.adc_clk_en = 0
- Disable Dynamic clock gating to ADC - CLKMAN_CLK_GATE_CTRL2.adc_clk_gater = 00b
- Disable power to AFE - PWRMAN_PWR_RST_CTRL.afe_powered = 0

8.4.6 ADC Data Limits

To simplify and minimize power consumption during power supply monitoring, the ADC supports programmable data limits illustrated below. The ADC output is compared to a limit (ChxHiLimit) when a selected channel (ChxSel) is sampled. If the detector is enabled (ChxHiLimitEn) and the over-limit interrupt is enabled, an interrupt will be generated. This same capability is provided for the under-range limit detector as illustrated below.

The combination of these two detectors enables up to four channels to be sampled and compared to high and low limits. An interrupt can be generated on out-of-bounds conditions. No data transfer conditions are required. The PMU can control the ADC channel selection and conversion control enabling power supply monitoring in LP2:PMU and only wake the processor if an error condition occurs.

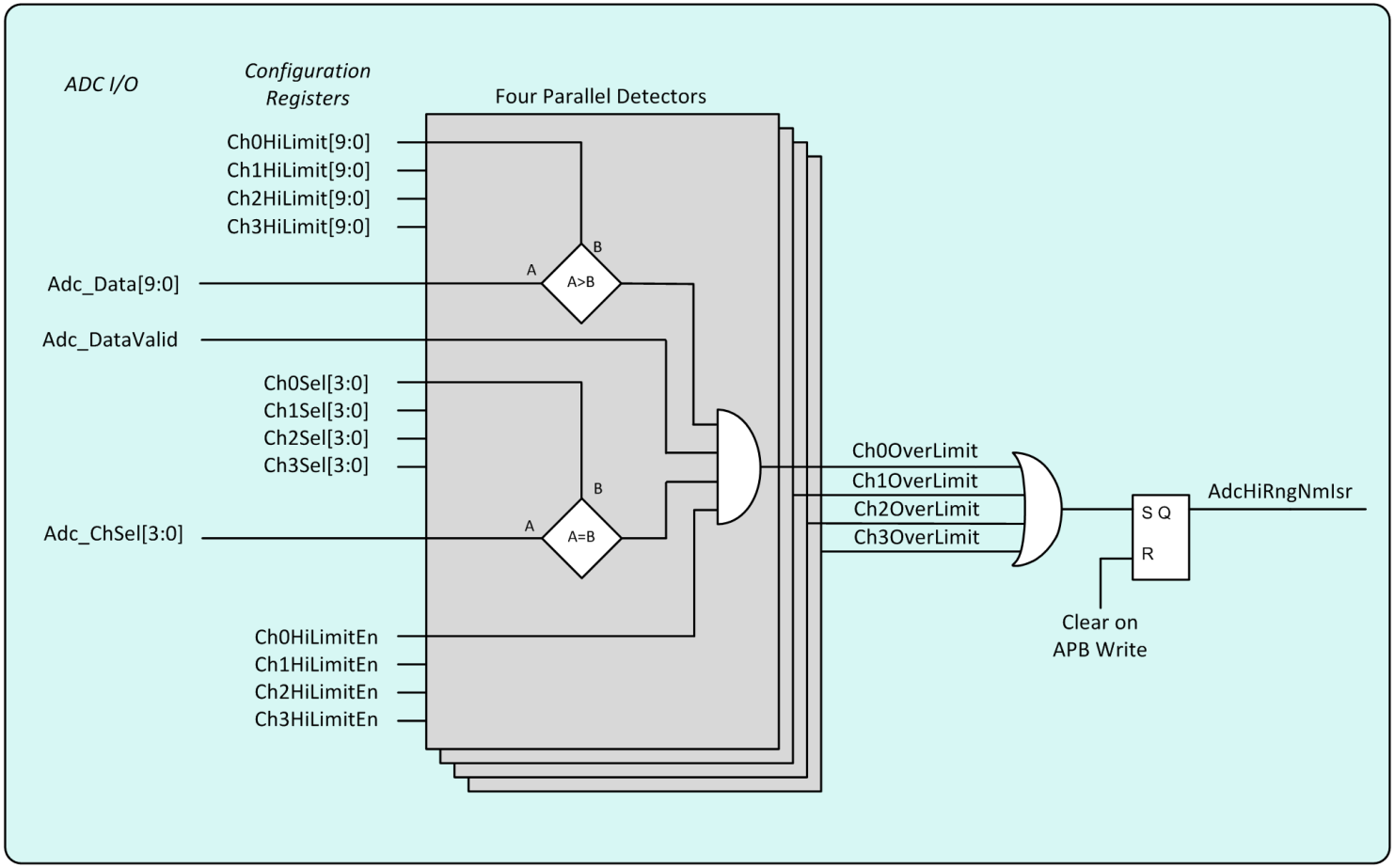

Figure 8.2: Data Over-Range Limit Detector

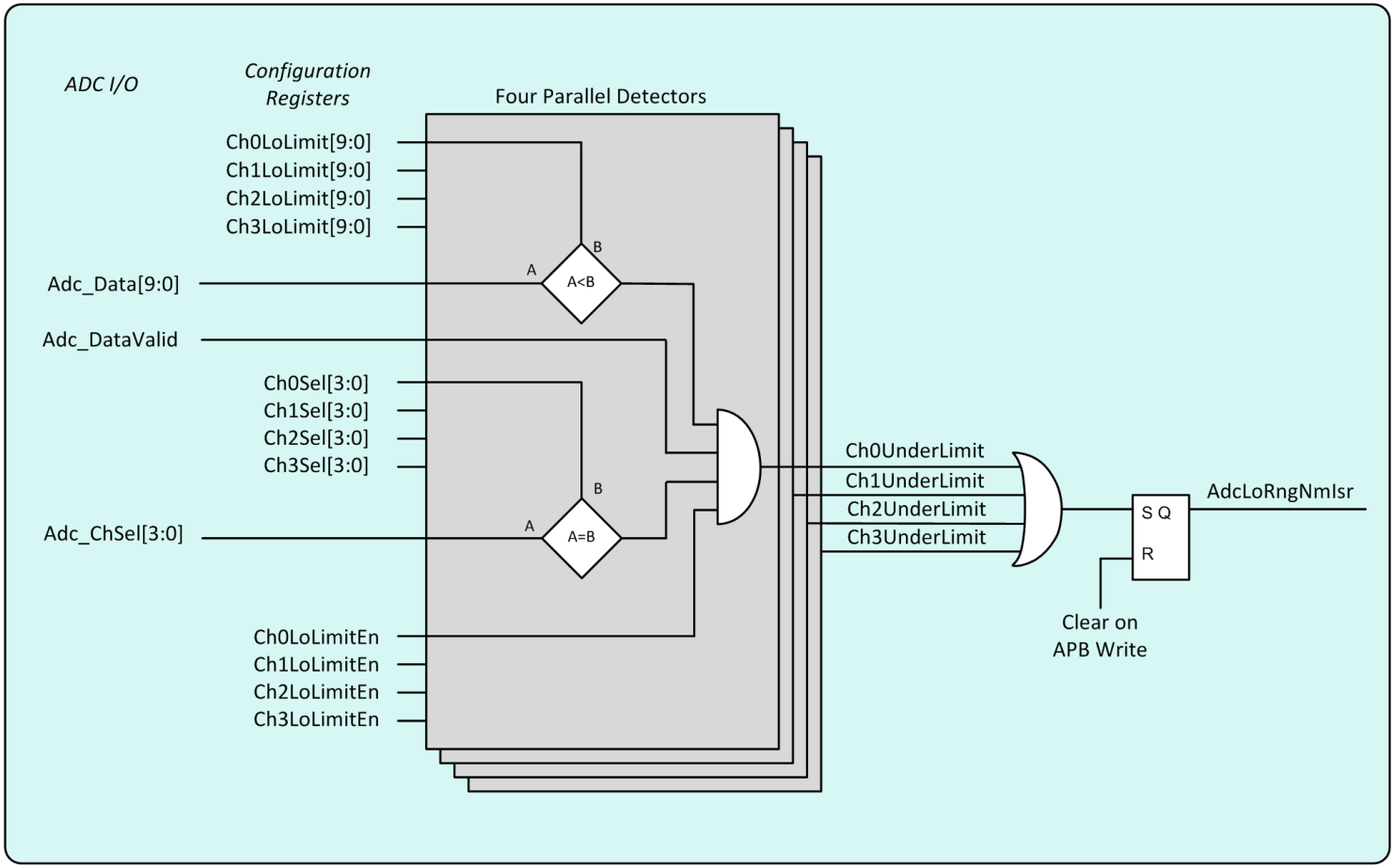

Figure 8.3: Data Under-Range Limit Detector

8.4.7 Data Value Equations

The following equations care used to calculate the ADC data values for the applicable analog channels.

Channels 0 - 3

$$\text{AdcData}[9:0] = round\left\{\frac{\frac{AIN}{2^{adc\_scale}}}{\frac{V_{REF}}{2}} \times (2^{10}-1)\right\}$$

Note For this configuration, the following settings must be used: adc_scale == (0 or 1), adc_refscl == 1.

Channels 4 - 5

$$\text{AdcData}[9:0] = round\left\{\frac{AIN}{5 \times V_{REF}} \times (2^{10}-1)\right\}$$

Note For this configuration, the following settings must be used: adc_scale == 1, adc_refscl == 1.

Channel 6

$$\text{AdcData}[9:0] = round\left\{\frac{V_{DDB}}{4 \times V_{REF}} \times (2^{10}-1)\right\}$$

Note For this configuration, the following settings must be used: adc_scale == 1, adc_refscl == 1.

Channel 7

$$\text{AdcData}[9:0] = round\left\{\frac{V_{DD18}}{2 \times V_{REF}} \times (2^{10}-1)\right\}$$

Note For this configuration, the following settings must be used: adc_scale == 1, adc_refscl == 0.

Channel 8

$$\text{AdcData}[9:0] = round\left\{\frac{V_{DD12}}{2 \times V_{REF}} \times (2^{10}-1)\right\}$$

Note For this configuration, the following settings must be used: adc_scale == 1, adc_refscl == 0.

Channel 9

$$\text{AdcData}[9:0] = round\left\{\frac{V_{RTC}}{2 \times V_{REF}} \times (2^{10} - 1)\right\}$$

Note For this configuration, the following settings must be used: adc_scale == 1, adc_refscl == 1.

8.5 Registers (ADC)

Address	Register	Access	Description	Reset By
0x4001F000	ADC_CTRL	R/W	ADC Control	Sys
0x4001F004	ADC_STATUS	R/O	ADC Status	Sys
0x4001F008	ADC_DATA	R/O	ADC Output Data	Sys
0x4001F00C	ADC_INTR	***	ADC Interrupt Control Register	Sys
0x4001F010	ADC_LIMIT0	R/W	ADC Limit 0	Sys
0x4001F014	ADC_LIMIT1	R/W	ADC Limit 1	Sys
0x4001F018	ADC_LIMIT2	R/W	ADC Limit 2	Sys
0x4001F01C	ADC_LIMIT3	R/W	ADC Limit 3	Sys
0x4001F020	ADC_AFE_CTRL	R/W	AFE Control Register	Sys
0x4001F024	ADC_RO_CAL0	R/W	RO Trim Calibration Register 0	Sys
0x4001F028	ADC_RO_CAL1	R/W	RO Trim Calibration Register 1	Sys
0x4001F02C	ADC_RO_CAL2	R/W	RO Trim Calibration Register 2	Sys

8.5.1 ADC_CTRL

ADC_CTRL.cpu_adc_start

Field	Bits	Sys Reset	Access	Description
cpu_adc_start	0	0	R/W	Start ADC Conversion

Write to 1 to start ADC conversion.

Hardware automatically clears this bit to 0 after the ADC conversion has completed.

ADC_CTRL.adc_pu

Field	Bits	Sys Reset	Access	Description
adc_pu	1	0	R/W	ADC Power Up

- 0: ADC is powered off (default).
- 1: ADC is powered on.

ADC_CTRL.buf_pu

Field	Bits	Sys Reset	Access	Description
buf_pu	2	0	R/W	ADC Input Buffer Power Up

- 0: ADC input buffer is powered off (default).
- 1: ADC input buffer is powered on.

ADC_CTRL.adc_refbuf_pu

Field	Bits	Sys Reset	Access	Description
adc_refbuf_pu	3	0	R/W	ADC Reference Buffer Power Up

- 0: ADC reference buffer is powered off (default).
- 1: ADC reference buffer is powered on.

ADC_CTRL.adc_chgpump_pu

Field	Bits	Sys Reset	Access	Description
adc_chgpump_pu	4	0	R/W	ADC Charge Pump Power Up

- 0: ADC charge pump is powered off (default).
- 1: ADC charge pump is powered on.

ADC_CTRL.buf_bypass

Field	Bits	Sys Reset	Access	Description
buf_bypass	7	0	R/W	Bypass Input Buffer

- 0: Use input buffer (default)
- 1: Bypass input buffer stage.

ADC_CTRL.adc_refscl

Field	Bits	Sys Reset	Access	Description
adc_refscl	8	0	R/W	ADC Reference Scale

- 0: Normal operation (default)
- 1: Scale ADC reference down by 1/2

ADC_CTRL.adc_scale

Field	Bits	Sys Reset	Access	Description
adc_scale	9	0	R/W	ADC Scale

- 0: Normal operation (default)
- 1: Scale ADC data down by 1/2

ADC_CTRL.adc_clk_en

Field	Bits	Sys Reset	Access	Description
adc_clk_en	11	0	R/W	ADC Clock Enable

- 0: ADC clock disabled
- 1: ADC clock enabled

ADC_CTRL.adc_chsel

Field	Bits	Sys Reset	Access	Description
adc_chsel	15:12	0	R/W	ADC Channel Select

- 0: AIN0
- 1: AIN1
- 2: AIN2
- 3: AIN3
- 4: AIN0 / 5
- 5: AIN1 / 5
- 6: VDD33 / 4
- 7: VDD18
- 8: VDD12
- 9: VRTC / 2

Note The ADC input scaling (adc_scale) mode may be set to 0 or 1 when channels 0-3 (AIN0, AIN1, AIN2, AIN3) are selected. For any other channel, adc_scale must be set to 1.
The ADC reference scaling (adc_ref_scale) mode must be set to 0 when channels 7 (VDD18) or 8 (VDD12) are selected. For any other channel, adc_ref_scale must be set to 1.

ADC_CTRL.adc_xref

Field	Bits	Sys Reset	Access	Description
adc_xref	16	0	R/W	Enable Use of ADC External Reference

- 0: Use internal reference for ADC
- 1: Use external reference for ADC

ADC_CTRL.adc_dataalign

Field	Bits	Sys Reset	Access	Description
adc_dataalign	17	0	R/W	ADC Data Alignment Select

- 0: Data is LSB justified in 16-bit output word (high 6 bits are padded with zeroes)
- 1: Data is MSB justified in 16-bit output word (low 6 bits are padded with zeroes)

ADC_CTRL.afe_pwr_up_dly

Field	Bits	Sys Reset	Access	Description
afe_pwr_up_dly	31:24	A0h	R/W	Delay from ADC Powerup Until ADC Ready Asserted

Delay from ADC powerup until the ADC Ready interrupt flag is set. Given in microseconds.

8.5.2 ADC_STATUS

ADC_STATUS.adc_active

Field	Bits	Sys Reset	Access	Description
adc_active	0	0	R/O	ADC Conversion In Progress

- 0: ADC is idle.
- 1: ADC conversion is currently in progress.

ADC_STATUS.ro_cal_atomic_active

Field	Bits	Sys Reset	Access	Description
ro_cal_atomic_active	1	0	R/O	RO Frequency Calibration Active (If Atomic)

- 0: No atomic RO frequency calibration is active.
- 1: RO frequency calibration is active; only applies if the calibration was started in atomic mode (not manual).

ADC_STATUS.afe_pwr_up_active

Field	Bits	Sys Reset	Access	Description
afe_pwr_up_active	2	0	R/O	AFE Power Up Delay Active

- 0: AFE is not in power up delay.
- 1: AFE is currently in the power up delay state.

ADC_STATUS.adc_overflow

Field	Bits	Sys Reset	Access	Description
adc_overflow	3	0	R/O	ADC Overflow

- 0: The last output from the ADC did not overflow.
- 1: The last converted output from the ADC was an overflow.

8.5.3 ADC_DATA

ADC_DATA.adc_data

Field	Bits	Sys Reset	Access	Description
adc_data	15:0	0	R/O	ADC Converted Sample Data Output

If adc_dataalign=0 (LSB justified) this field contains the ADC output data as follows: output[15:10]=000000b, output[9:0]=(ADC output sample)

If adc_dataalign=1 (MSB justified) this field contains the ADC output data as follows: output[15:6]=(ADC output sample), output[5:0]=000000b

8.5.4 ADC_INTR

ADC_INTR.adc_done_ie

Field	Bits	Sys Reset	Access	Description
adc_done_ie	0	0	R/W	ADC Done Interrupt Enable

- 0: Interrupt source is disabled.
- 1: Enables assertion of ADC interrupt when ADC Done Interrupt Flag is set.

ADC_INTR.adc_ref_ready_ie

Field	Bits	Sys Reset	Access	Description
adc_ref_ready_ie	1	0	R/W	ADC Reference Ready Interrupt Enable

- 0: Interrupt source is disabled.
- 1: Enables assertion of ADC interrupt when ADC Reference Ready Interrupt Flag is set.

ADC_INTR.adc_hi_limit_ie

Field	Bits	Sys Reset	Access	Description
adc_hi_limit_ie	2	0	R/W	ADC Hi Limit Monitor Interrupt Enable

- 0: Interrupt source is disabled.
- 1: Enables assertion of ADC interrupt when ADC Hi Limit Monitor Interrupt Flag is set.

ADC_INTR.adc_lo_limit_ie

Field	Bits	Sys Reset	Access	Description
adc_lo_limit_ie	3	0	R/W	ADC Lo Limit Monitor Interrupt Enable

- 0: Interrupt source is disabled.
- 1: Enables assertion of ADC interrupt when ADC Lo Limit Monitor Interrupt Flag is set.

ADC_INTR.adc_overflow_ie

Field	Bits	Sys Reset	Access	Description
adc_overflow_ie	4	0	R/W	ADC Overflow Interrupt Enable

- 0: Interrupt source is disabled.
- 1: Enables assertion of ADC interrupt when ADC Overflow Interrupt Flag is set.

ADC_INTR.ro_cal_done_ie

Field	Bits	Sys Reset	Access	Description
ro_cal_done_ie	5	0	R/W	RO Cal Done Interrupt Enable

- 0: Interrupt source is disabled.
- 1: Enables assertion of ADC interrupt when RO Cal Done Interrupt Flag is set.

ADC_INTR.adc_done_if

Field	Bits	Sys Reset	Access	Description
adc_done_if	16	0	W1C	ADC Done Interrupt Flag

Set by hardware when the associated interrupt event occurs.

Write 1 to clear.

ADC_INTR.adc_ref_ready_if

Field	Bits	Sys Reset	Access	Description
adc_ref_ready_if	17	0	W1C	ADC Reference Ready Interrupt Flag

Set by hardware when the associated interrupt event occurs.

Write 1 to clear.

ADC_INTR.adc_hi_limit_if

Field	Bits	Sys Reset	Access	Description
adc_hi_limit_if	18	0	W1C	ADC Hi Limit Monitor Interrupt Flag

Set by hardware when the associated interrupt event occurs.

Write 1 to clear.

ADC_INTR.adc_lo_limit_if

Field	Bits	Sys Reset	Access	Description
adc_lo_limit_if	19	0	W1C	ADC Lo Limit Monitor Interrupt Flag

Set by hardware when the associated interrupt event occurs.

Write 1 to clear.

ADC_INTR.adc_overflow_if

Field	Bits	Sys Reset	Access	Description
adc_overflow_if	20	0	W1C	ADC Overflow Interrupt Flag

Set by hardware when the associated interrupt event occurs.

Write 1 to clear.

ADC_INTR.ro_cal_done_if

Field	Bits	Sys Reset	Access	Description
ro_cal_done_if	21	0	W1C	RO Cal Done Interrupt Flag

Set by hardware when the associated interrupt event occurs.

Write 1 to clear.

ADC_INTR.adc_int_pending

Field	Bits	Sys Reset	Access	Description
adc_int_pending	22	0	R/O	ADC Interrupt Pending Status

- 0: No ADC interrupt is pending to the CPU.
- 1: At least one ADC interrupt is pending and enabled.

8.5.5 ADC_LIMIT0

ADC_LIMIT0.ch_lo_limit

Field	Bits	Sys Reset	Access	Description
ch_lo_limit	9:0	000h	R/W	Low Limit Threshold

Low limit threshold level for ADC(channel_sel) $<$ lo_limit_th.

ADC_LIMIT0.ch_hi_limit

Field	Bits	Sys Reset	Access	Description
ch_hi_limit	21:12	3FFh	R/W	High Limit Threshold

High limit threshold level for ADC(channel_sel) $>$ hi_limit_th.

ADC_LIMIT0.ch_sel

Field	Bits	Sys Reset	Access	Description
ch_sel	27:24	0	R/W	ADC Channel Select

Compared against currently selected ADC channel to determine if a limit threshold comparision should be made using the ADC sample output.

ADC_LIMIT0.ch_lo_limit_en

Field	Bits	Sys Reset	Access	Description
ch_lo_limit_en	28	0	R/W	Low Limit Monitoring Enable

- 0: This limit is not enabled to trigger an interrupt.
- 1: The low limit comparison for this channel can trigger the ADC Lo Limit Monitor interrupt.

ADC_LIMIT0.ch_hi_limit_en

Field	Bits	Sys Reset	Access	Description
ch_hi_limit_en	29	0	R/W	High Limit Monitoring Enable

- 0: This limit is not enabled to trigger an interrupt.
- 1: The high limit comparison for this channel can trigger the ADC Hi Limit Monitor interrupt.

8.5.6 ADC_LIMIT1

ADC_LIMIT1.ch_lo_limit

Field	Bits	Sys Reset	Access	Description
ch_lo_limit	9:0	000h	R/W	Low Limit Threshold

Low limit threshold level for ADC(channel_sel) $<$ lo_limit_th.

ADC_LIMIT1.ch_hi_limit

Field	Bits	Sys Reset	Access	Description
ch_hi_limit	21:12	3FFh	R/W	High Limit Threshold

High limit threshold level for ADC(channel_sel) $>$ hi_limit_th.

ADC_LIMIT1.ch_sel

Field	Bits	Sys Reset	Access	Description
ch_sel	27:24	1h	R/W	ADC Channel Select

Compared against currently selected ADC channel to determine if a limit threshold comparision should be made using the ADC sample output.

ADC_LIMIT1.ch_lo_limit_en

Field	Bits	Sys Reset	Access	Description
ch_lo_limit_en	28	0	R/W	Low Limit Monitoring Enable

- 0: This limit is not enabled to trigger an interrupt.
- 1: The low limit comparison for this channel can trigger the ADC Lo Limit Monitor interrupt.

ADC_LIMIT1.ch_hi_limit_en

Field	Bits	Sys Reset	Access	Description
ch_hi_limit_en	29	0	R/W	High Limit Monitoring Enable

- 0: This limit is not enabled to trigger an interrupt.
- 1: The high limit comparison for this channel can trigger the ADC Hi Limit Monitor interrupt.

8.5.7 ADC_LIMIT2

ADC_LIMIT2.ch_lo_limit

Field	Bits	Sys Reset	Access	Description
ch_lo_limit	9:0	000h	R/W	Low Limit Threshold

Low limit threshold level for ADC(channel_sel) $<$ lo_limit_th.

ADC_LIMIT2.ch_hi_limit

Field	Bits	Sys Reset	Access	Description
ch_hi_limit	21:12	3FFh	R/W	High Limit Threshold

High limit threshold level for ADC(channel_sel) > hi_limit_th.

ADC_LIMIT2.ch_sel

Field	Bits	Sys Reset	Access	Description
ch_sel	27:24	2h	R/W	ADC Channel Select

Compared against currently selected ADC channel to determine if a limit threshold comparision should be made using the ADC sample output.

ADC_LIMIT2.ch_lo_limit_en

Field	Bits	Sys Reset	Access	Description
ch_lo_limit_en	28	0	R/W	Low Limit Monitoring Enable

- 0: This limit is not enabled to trigger an interrupt.
- 1: The low limit comparison for this channel can trigger the ADC Lo Limit Monitor interrupt.

ADC_LIMIT2.ch_hi_limit_en

Field	Bits	Sys Reset	Access	Description
ch_hi_limit_en	29	0	R/W	High Limit Monitoring Enable

- 0: This limit is not enabled to trigger an interrupt.
- 1: The high limit comparison for this channel can trigger the ADC Hi Limit Monitor interrupt.

8.5.8 ADC_LIMIT3

ADC_LIMIT3.ch_lo_limit

Field	Bits	Sys Reset	Access	Description
ch_lo_limit	9:0	000h	R/W	Low Limit Threshold

Low limit threshold level for ADC(channel_sel) $<$ lo_limit_th.

ADC_LIMIT3.ch_hi_limit

Field	Bits	Sys Reset	Access	Description
ch_hi_limit	21:12	3FFh	R/W	High Limit Threshold

High limit threshold level for ADC(channel_sel) $>$ hi_limit_th.

ADC_LIMIT3.ch_sel

Field	Bits	Sys Reset	Access	Description
ch_sel	27:24	3h	R/W	ADC Channel Select

Compared against currently selected ADC channel to determine if a limit threshold comparision should be made using the ADC sample output.

ADC_LIMIT3.ch_lo_limit_en

Field	Bits	Sys Reset	Access	Description
ch_lo_limit_en	28	0	R/W	Low Limit Monitoring Enable

- 0: This limit is not enabled to trigger an interrupt.
- 1: The low limit comparison for this channel can trigger the ADC Lo Limit Monitor interrupt.

ADC_LIMIT3.ch_hi_limit_en

Field	Bits	Sys Reset	Access	Description
ch_hi_limit_en	29	0	R/W	High Limit Monitoring Enable

- 0: This limit is not enabled to trigger an interrupt.
- 1: The high limit comparison for this channel can trigger the ADC Hi Limit Monitor interrupt.

8.5.9 ADC_AFE_CTRL

ADC_AFE_CTRL.tmon_intbias_en

Field	Bits	Sys Reset	Access	Description
tmon_intbias_en	8	0	R/W	Enable internal temperature measurement bias generator

- 0: Disabled
- 1: Enabled

ADC_AFE_CTRL.tmon_extbias_en

Field	Bits	Sys Reset	Access	Description
tmon_extbias_en	9	0	R/W	Enable external temperature measurement bias generator

- 0: Disabled
- 1: Enabled

8.5.10 ADC_RO_CAL0

ADC_RO_CAL0.ro_cal_en

Field	Bits	Sys Reset	Access	Description
ro_cal_en	0	0	R/W	RO Calibration Enable

Selects trim value that will be used for the high-speed system relaxation oscillator.

- 0: Factory trim settings (stored in flash info block) will be used.
- 1: The output of the RO calibration loop will be used.

ADC_RO_CAL0.ro_cal_run

Field	Bits	Sys Reset	Access	Description
ro_cal_run	1	0	R/W	RO Calibration Run

Write to 1 to start system RO calibration with manual timing.

Firmware must manually clear this bit to 0 to end the calibration loop.

ADC_RO_CAL0.ro_cal_load

Field	Bits	Sys Reset	Access	Description
ro_cal_load	2	0	R/W	RO Calibration Load Initial Value

Writing this bit to 1 causes the RO Trim Initial Value (trm_init) to be loaded as the starting point for the calibration loop.

Hardware will automatically clear this bit to 0 once the load operation has been completed.

ADC_RO_CAL0.ro_cal_atomic

Field	Bits	Sys Reset	Access	Description
ro_cal_atomic	4	0	R/W	RO Calibration Run Atomic

Writing this bit to 1 starts the RO calibration in Atomic mode. In this mode, the calibration loop runs for a number of milliseconds given in the Auto Cal Time Delay field (auto_cal_done_cnt) and then halts automatically.

Hardware will automatically clear this bit to 0 once the RO calibration has completed.

ADC_RO_CAL0.dummy

Field	Bits	Sys Reset	Access	Description
dummy	7:5	0	R/W	Dummy Write Field

Writing to this field has no effect. This field always reads 0.

ADC_RO_CAL0.trm_mu

Field	Bits	Sys Reset	Access	Description
trm_mu	19:8	0	R/W	RO Trim Adaptation Gain

Adaptation gain for the loop. Higher values increase filtering and convergence time. Lower values decrease filtering and decrease convergence time but may lead to the output dithering too much.

ADC_RO_CAL0.ro_trm

Field	Bits	Sys Reset	Access	Description
ro_trm	31:23	0	R/W	RO Trim Calibration Result

Result of RO trim calibration loop, used as the high-speed system relaxation oscillator trim when (ro_cal_en == 1).

8.5.11 ADC_RO_CAL1

ADC_RO_CAL1.trm_init

Field	Bits	Sys Reset	Access	Description
trm_init	8:0	0	R/W	RO Trim Initial Value

This value is loaded into the calibration loop when ro_cal_load is written to 1.

ADC_RO_CAL1.trm_min

Field	Bits	Sys Reset	Access	Description
trm_min	18:10	0	R/W	RO Trim Maximum Adaptive Limit

Upper limit for the automatically calibrated RO trim value.

ADC_RO_CAL1.trm_max

Field	Bits	Sys Reset	Access	Description
trm_max	28:20	0	R/W	RO Trim Minimum Adaptive Limit

Lower limit for the automatically calibrated RO trim value.

8.5.12 ADC_RO_CAL2

ADC_RO_CAL2.auto_cal_done_cnt

Field	Bits	Sys Reset	Access	Description
auto_cal_done_cnt	7:0	100	R/W	Auto Cal Time Delay for Atomic Calibration (in milliseconds)

Time delay that will be used when the RO calibration is started in Atomic mode. Given in milliseconds.

9 Pulse Train Engine

9.1 Pulse Train (PT) Engine Overview

The **MAX32620** includes 16 independent pulse train engines, which can be used to generate square wave or repeating patterns (up to 32 bits in length) on GPIO pins. Each GPIO pin can be driven by the output of one of the 16 pulse train engines as shown in the table below.

Pulse Train Drive Options for GPIO

GPIO Pin(s)	Can Be Driven By PT
P0.0, P2.0, P4.0	PT0
P0.1, P2.1, P4.1	PT1
P0.2, P2.2, P4.2	PT2
P0.3, P2.3, P4.3	PT3
P0.4, P2.4, P4.4	PT4
P0.5, P2.5, P4.5	PT5
P0.6, P2.6, P4.6	PT6
P0.7, P2.7, P4.7	PT7
P1.0, P3.0, P5.0	PT8
P1.1, P3.1, P5.1	PT9
P1.2, P3.2, P5.2	PT10
P1.3, P3.3, P5.3	PT11
P1.4, P3.4, P5.4	PT12
P1.5, P3.5, P5.5	PT13
P1.6, P3.6, P5.6	PT14
P1.7, P3.7, P5.7	PT15
P6.0	PT0

Each pulse train generator can be set to output either a square wave or a repeating pattern from 2 bits to 32 bits in length. The frequency of each enabled pulse train generator is also set separately, with the frequency of the Pulse Train module clock used as a common basis.

Any single pulse train generator or any desired group of pulse train generators can be restarted at the beginning of their patterns and synchronized with one another enabling each synchronized pulse train generator to start simultaneously.

9.2 Output Mode Selection

Two modes of operation are supported: Pulse Train and Square Wave. Both modes use a rate counter that defines the number of Pulse Train module clock cycles that occur before the output state is changed. In both modes, any desired single pulse train or combination of pulse train engines can be synchronized or enabled/disabled simultaneously as needed.

9.3 Pulse Train Module Clock Rate Configuration

All of the pulse train engines in the overall Pulse Train Module (PT) use a single common module clock as a basis for timing generation. This clock is the Pulse Train Module Clock and is generated from the current system clock source as selected by the CLKMAN_SYS_CLK_CTRL_7_PT register. The Pulse Train Module Clock is disabled by default; before any of the pulse train engines can be used, this clock must be enabled and selected to run from either the undivided system clock source or a clock frequency based on a divide down of the system clock source, as shown in the table below.

Pulse Train Module Clock Selection Table

pulse_train_clk_scale	Pulse Train Module Clock Rate
0	Module clock is disabled
1	(System Clock Source / 1)
2	(System Clock Source / 2)
3	(System Clock Source / 4)
4	(System Clock Source / 8)
5	(System Clock Source / 16)
6	(System Clock Source / 32)
7	(System Clock Source / 64)
8	(System Clock Source / 128)
9	(System Clock Source / 256)

9.4 Pulse Train Module Global Controls

In the Pulse Train Module, registers beginning with PTG contain global controls or status flags for all 16 pulse train instances in the Pulse Train Module. Registers specific to a given pulse train instance have names beginning with PT(n), where (n) is the pulse train instance number. For example, all registers beginning with PT0 are specific to pulse train 0.

9.4.1 Enabling and Disabling Pulse Train Instances

The global control register PTG_ENABLE allows all 16 pulse train engines (or a single pulse train engine, or any combination of pulse train engines) to be enabled or disabled with a single register write. Each of the 16 single-bit fields in this register corresponds to one of the 16 pulse train engines. Writing any of these bits to 1 will enable the corresponding pulse train engine, while writing the bit to 0 will disable the pulse train engine.

Note The default register settings (following reset) for any particular pulse train engine will cause that pulse train to be effectively disabled even if its enable bit in PTG_ENABLE is set to 1. Before enabling a pulse train globally, its registers must be loaded with the desired output pattern, pattern length/mode, and pattern frequency settings.

Once a pulse train has been enabled, it will continue running indefinitely until one of the following events occurs:

- The pulse train is disabled by setting its enable bit in PTG_ENABLE to 0.
- The pulse train is disabled and reset by setting its resync bit in PTG_RESYNC to 0.
- The pulse train is running in loop count mode (PTn_LOOP was set to a nonzero value before the pulse train was enabled) and the pattern has been output the number of times specified by the loop count. Note that this only applies in pulse train mode; the loop count has no effect in square wave output mode.

9.4.2 Interrupt Controls for All Pulse Train Instances

The Pulse Train Module is capable of generating an interrupt to the NVIC when any specified pulse train engine stops running (when it goes from the enabled to the disabled state). The PTG_INTFL register contains a Pulse Train Stopped interrupt flag for each of the 16 pulse train instances. Whether or not any given interrupt flag in the PTG_INTFL register actually triggers an interrupt from the Pulse Train Module to the NVIC is determined by the interrupt enable/disable controls in the PTG_INTEN register.

9.4.3 Synchronizing Pulse Train Instances

The PTG_RESYNC register is used to reset a pulse train engine or engines which are already enabled. It can also be used to reset and synchronize multiple pulse train engines so their waveforms are output in sync with one another. To reset and synchronize one or more pulse train engines, write the corresponding bits in PTG_RESYNC to 1. The application should then wait until all bits in PTG_RESYNC are cleared back to zero by the hardware; this indicates that the reset/sync process has completed.

After the reset/sync completes, hardware will clear the enable bits in PTG_ENABLE for all pulse train engines that were reset and synchronized. At this point, simply

write those same enable bits back to 1, and the pulse trains will restart operation (at the beginning of their output patterns) with all output waveforms synchronized.

9.5 Pulse Train Engine Modes

The sections below describe the available operating modes for the pulse train engines and which configuration registers are used to specify the operating modes and other settings.

Note	The Pulse Train Engine configuration registers may be modified at any time by the user. If a given pulse train engine is enabled when these registers are modified, the output state is immediately affected and might result in undesired or unintended effects on the output.

9.5.1 Output Rate Control

Whenever a pulse train engine is operating, the time that it takes to transition from one output state to the next is determined by the PTn_RATE_LENGTH.rate_control field. By default, this field is set to 0, which is an invalid setting that will result in the pulse train remaining disabled even if the pulse train's enable bit in PTG_ENABLE has been set to 1.

Setting this field to a nonzero value will allow the pulse train to operate.

9.5.2 Pulse Train Mode

In Pulse Train mode, the pulse train engine output is generated from a programmable bit pattern set by the application. This pattern may be between 2 and 32 bits in length.

The time between output transitions is determined by PTn_RATE_LENGTH.rate_control and is specified as a number of Pulse Train Module clock cycles. Effectively, this means that the output bit frequency of the given pulse train will be set as follows (for rate_control $>$ 0)

$$f_{out} = \frac{PulseTrainModuleClock}{rate\_control} bps$$

The length of the pattern is set by writing PTn_RATE_LENGTH.mode to the appropriate value. For a length of 32 bits (the maximum), this field must be written to zero; for other length values (from 2 to 31), the field should be set to the desired length in bits. Note that the value '1' is not used for this field in pulse train mode; setting PTn_RATE_LENGTH.mode to 1 selects the square wave output operating mode instead of the pulse train output mode.

The pattern itself is contained in the PTn_TRAIN register. This pattern will be output LSB first. For lengths less than 32 bits, only the low N bits of PTn_TRAIN will be used when generating the pattern, where N is the bit length as selected by the PTn_RATE_LENGTH.mode field. Once all N bits of the pattern have been output, the pattern starts again with the LSB.

9.5.3 Pulse Train Loop Count

By default, an enabled pulse train engine running in pulse train mode will repeat the specified output pattern continuously until the pulse train is disabled manually.

It is also possible to set a given pulse train to repeat its pulse train output pattern a specified number of times, with the pulse train engine being automatically disabled after the pattern has been output for the final time.

To select this operating mode, a loop count value must be written to the PTn_LOOP register for a given pulse train before that pulse train is enabled. The loop count value only applies when the pulse train engine is operating in pulse train output mode; it has no effect in square wave output mode.

The value written to PTn_LOOP will then determine how many times the specified output pattern will be enabled. With each iteration of the output pattern, the loop counter for that pulse train decrements by 1. When the loop counter reaches zero, the enable bit for the pulse train engine will be automatically cleared, disabling the pulse train.

9.5.4 Square Wave Mode

The other option for pulse train output generation is square wave output mode. This mode is selected by setting the PTn_RATE_LENGTH.mode field for a given pulse train engine to 1.

In square wave output mode, the PTn_TRAIN and PTn_LOOP registers are not used. Instead, the pulse train engine simply outputs a continuous square wave beginning with an output of 0 when the pulse train is first enabled.

The time between output transitions is determined by (PTn_RATE_LENGTH.rate_control - 1) and is specified as a number of Pulse Train Module clock cycles. Effectively, this means that the output bit frequency of the given pulse train (in square wave mode) will be set as follows (for rate_control $>$ 1)

$$f_{out} = \frac{PulseTrainModuleClock}{(rate\_control - 1)} bps$$

Note that since the square wave mode does not involve a fixed pattern, the loop count value does not apply in this mode.

9.6 Registers (PT)

Address	Register	Access	Description	Reset By
0x40011000	PTG_ENABLE	R/W	Global Enable/Disable Controls for All Pulse Trains	Sys
0x40011004	PTG_RESYNC	R/W	Global Resync (All Pulse Trains) Control	Sys
0x40011008	PTG_INTFL	R/W	Pulse Train Interrupt Flags	Sys
0x4001100C	PTG_INTEN	R/W	Pulse Train Interrupt Enable/Disable	Sys
0x40011010	PT0_RATE_LENGTH	R/W	Pulse Train 0 Configuration	Sys
0x40011014	PT0_TRAIN	R/W	Pulse Train 0 Output Pattern	Sys
0x40011018	PT0_LOOP	R/W	Pulse Train 0 Loop Count	Sys
0x4001101C	PT1_RATE_LENGTH	R/W	Pulse Train 1 Configuration	Sys
0x40011020	PT1_TRAIN	R/W	Pulse Train 1 Output Pattern	Sys
0x40011024	PT1_LOOP	R/W	Pulse Train 1 Loop Count	Sys
0x40011028	PT2_RATE_LENGTH	R/W	Pulse Train 2 Configuration	Sys
0x4001102C	PT2_TRAIN	R/W	Pulse Train 2 Output Pattern	Sys
0x40011030	PT2_LOOP	R/W	Pulse Train 2 Loop Count	Sys
0x40011034	PT3_RATE_LENGTH	R/W	Pulse Train 3 Configuration	Sys
0x40011038	PT3_TRAIN	R/W	Pulse Train 3 Output Pattern	Sys
0x4001103C	PT3_LOOP	R/W	Pulse Train 3 Loop Count	Sys
0x40011040	PT4_RATE_LENGTH	R/W	Pulse Train 4 Configuration	Sys
0x40011044	PT4_TRAIN	R/W	Pulse Train 4 Output Pattern	Sys
0x40011048	PT4_LOOP	R/W	Pulse Train 4 Loop Count	Sys
0x4001104C	PT5_RATE_LENGTH	R/W	Pulse Train 5 Configuration	Sys
0x40011050	PT5_TRAIN	R/W	Pulse Train 5 Output Pattern	Sys
0x40011054	PT5_LOOP	R/W	Pulse Train 5 Loop Count	Sys

Address	Register	Access	Description	Reset By
0x40011058	PT6_RATE_LENGTH	R/W	Pulse Train 6 Configuration	Sys
0x4001105C	PT6_TRAIN	R/W	Pulse Train 6 Output Pattern	Sys
0x40011060	PT6_LOOP	R/W	Pulse Train 6 Loop Count	Sys
0x40011064	PT7_RATE_LENGTH	R/W	Pulse Train 7 Configuration	Sys
0x40011068	PT7_TRAIN	R/W	Pulse Train 7 Output Pattern	Sys
0x4001106C	PT7_LOOP	R/W	Pulse Train 7 Loop Count	Sys
0x40011070	PT8_RATE_LENGTH	R/W	Pulse Train 8 Configuration	Sys
0x40011074	PT8_TRAIN	R/W	Pulse Train 8 Output Pattern	Sys
0x40011078	PT8_LOOP	R/W	Pulse Train 8 Loop Count	Sys
0x4001107C	PT9_RATE_LENGTH	R/W	Pulse Train 9 Configuration	Sys
0x40011080	PT9_TRAIN	R/W	Pulse Train 9 Output Pattern	Sys
0x40011084	PT9_LOOP	R/W	Pulse Train 9 Loop Count	Sys
0x40011088	PT10_RATE_LENGTH	R/W	Pulse Train 10 Configuration	Sys
0x4001108C	PT10_TRAIN	R/W	Pulse Train 10 Output Pattern	Sys
0x40011090	PT10_LOOP	R/W	Pulse Train 10 Loop Count	Sys
0x40011094	PT11_RATE_LENGTH	R/W	Pulse Train 11 Configuration	Sys
0x40011098	PT11_TRAIN	R/W	Pulse Train 11 Output Pattern	Sys
0x4001109C	PT11_LOOP	R/W	Pulse Train 11 Loop Count	Sys
0x400110A0	PT12_RATE_LENGTH	R/W	Pulse Train 12 Configuration	Sys
0x400110A4	PT12_TRAIN	R/W	Pulse Train 12 Output Pattern	Sys
0x400110A8	PT12_LOOP	R/W	Pulse Train 12 Loop Count	Sys
0x400110AC	PT13_RATE_LENGTH	R/W	Pulse Train 13 Configuration	Sys
0x400110B0	PT13_TRAIN	R/W	Pulse Train 13 Output Pattern	Sys
0x400110B4	PT13_LOOP	R/W	Pulse Train 13 Loop Count	Sys
0x400110B8	PT14_RATE_LENGTH	R/W	Pulse Train 14 Configuration	Sys

Address	Register	Access	Description	Reset By
0x400110BC	PT14_TRAIN	R/W	Pulse Train 14 Output Pattern	Sys
0x400110C0	PT14_LOOP	R/W	Pulse Train 14 Loop Count	Sys
0x400110C4	PT15_RATE_LENGTH	R/W	Pulse Train 15 Configuration	Sys
0x400110C8	PT15_TRAIN	R/W	Pulse Train 15 Output Pattern	Sys
0x400110CC	PT15_LOOP	R/W	Pulse Train 15 Loop Count	Sys

9.6.1 PTG_ENABLE

PTG_ENABLE.[pt0, pt1, pt2, pt3, pt4, pt5, pt6, pt7]

Field	Bits	Sys Reset	Access	Description
pt0	0	0	R/W	Enable/Disable control for PT0
pt1	1	0	R/W	Enable/Disable control for PT1
pt2	2	0	R/W	Enable/Disable control for PT2
pt3	3	0	R/W	Enable/Disable control for PT3
pt4	4	0	R/W	Enable/Disable control for PT4
pt5	5	0	R/W	Enable/Disable control for PT5
pt6	6	0	R/W	Enable/Disable control for PT6
pt7	7	0	R/W	Enable/Disable control for PT7

Setting this bit to 1 enables the corresponding pulse train. This bit is automatically cleared to zero when the corresponding resync control bit is set, or when the end of the loop count for the pulse train is reached.

PTG_ENABLE.[pt8, pt9, pt10, pt11, pt12, pt13, pt14, pt15]

Field	Bits	Sys Reset	Access	Description
pt8	8	0	R/W	Enable/Disable control for PT8
pt9	9	0	R/W	Enable/Disable control for PT9
pt10	10	0	R/W	Enable/Disable control for PT10
pt11	11	0	R/W	Enable/Disable control for PT11
pt12	12	0	R/W	Enable/Disable control for PT12
pt13	13	0	R/W	Enable/Disable control for PT13
pt14	14	0	R/W	Enable/Disable control for PT14
pt15	15	0	R/W	Enable/Disable control for PT15

Setting this bit to 1 enables the corresponding pulse train. This bit is automatically cleared to zero when the corresponding resync control bit is set, or when the end of the loop count for the pulse train is reached.

9.6.2 PTG_RESYNC

PTG_RESYNC.[pt0, pt1, pt2, pt3, pt4, pt5, pt6, pt7]

Field	Bits	Sys Reset	Access	Description
pt0	0	0	R/W	Resync control for PT0
pt1	1	0	R/W	Resync control for PT1
pt2	2	0	R/W	Resync control for PT2
pt3	3	0	R/W	Resync control for PT3
pt4	4	0	R/W	Resync control for PT4
pt5	5	0	R/W	Resync control for PT5
pt6	6	0	R/W	Resync control for PT6
pt7	7	0	R/W	Resync control for PT7

Setting this bit to 1 resets the rate counter and pulse train pointer for this pulse train, clears the corresponding enable bit for this pulse train, and (when in square wave mode) resets the output state for the pulse train to 0.

The pulse train will remain reset until re-enabled using the corresponding enable bit. The resync bit will automatically (on the next PT module clock edge) clear back to 0 after being set; application firmware should verify that the auto-clear has taken place before re-enabling the pulse train.

PTG_RESYNC.[pt8, pt9, pt10, pt11, pt12, pt13, pt14, pt15]

Field	Bits	Sys Reset	Access	Description
pt8	8	0	R/W	Resync control for PT8
pt9	9	0	R/W	Resync control for PT9
pt10	10	0	R/W	Resync control for PT10
pt11	11	0	R/W	Resync control for PT11
pt12	12	0	R/W	Resync control for PT12
pt13	13	0	R/W	Resync control for PT13
pt14	14	0	R/W	Resync control for PT14
pt15	15	0	R/W	Resync control for PT15

Setting this bit to 1 resets the rate counter and pulse train pointer for this pulse train, clears the corresponding enable bit for this pulse train, and (when in square wave mode) resets the output state for the pulse train to 0.

The pulse train will remain reset until re-enabled using the corresponding enable bit. The resync bit will automatically (on the next PT module clock edge) clear back to 0 after being set; application firmware should verify that the auto-clear has taken place before re-enabling the pulse train.

9.6.3 PTG_INTFL

PTG_INTFL.[pt0, pt1, pt2, pt3, pt4, pt5, pt6, pt7]

Field	Bits	Sys Reset	Access	Description
pt0	0	0	R/W	Pulse Train 0 Stopped Interrupt Flag
pt1	1	0	R/W	Pulse Train 1 Stopped Interrupt Flag
pt2	2	0	R/W	Pulse Train 2 Stopped Interrupt Flag
pt3	3	0	R/W	Pulse Train 3 Stopped Interrupt Flag
pt4	4	0	R/W	Pulse Train 4 Stopped Interrupt Flag
pt5	5	0	R/W	Pulse Train 5 Stopped Interrupt Flag
pt6	6	0	R/W	Pulse Train 6 Stopped Interrupt Flag
pt7	7	0	R/W	Pulse Train 7 Stopped Interrupt Flag

This bit is set to 1 when the corresponding pulse train stops running.

PTG_INTFL.[pt8, pt9, pt10, pt11, pt12, pt13, pt14, pt15]

Field	Bits	Sys Reset	Access	Description
pt8	8	0	R/W	Pulse Train 8 Stopped Interrupt Flag
pt9	9	0	R/W	Pulse Train 9 Stopped Interrupt Flag
pt10	10	0	R/W	Pulse Train 10 Stopped Interrupt Flag
pt11	11	0	R/W	Pulse Train 11 Stopped Interrupt Flag
pt12	12	0	R/W	Pulse Train 12 Stopped Interrupt Flag
pt13	13	0	R/W	Pulse Train 13 Stopped Interrupt Flag
pt14	14	0	R/W	Pulse Train 14 Stopped Interrupt Flag
pt15	15	0	R/W	Pulse Train 15 Stopped Interrupt Flag

This bit is set to 1 when the corresponding pulse train stops running.

9.6.4 PTG_INTEN

PTG_INTEN.[pt0, pt1, pt2, pt3, pt4, pt5, pt6, pt7]

Field	Bits	Sys Reset	Access	Description
pt0	0	0	R/W	Pulse Train 0 Stopped Interrupt Enable/Disable
pt1	1	0	R/W	Pulse Train 1 Stopped Interrupt Enable/Disable
pt2	2	0	R/W	Pulse Train 2 Stopped Interrupt Enable/Disable
pt3	3	0	R/W	Pulse Train 3 Stopped Interrupt Enable/Disable
pt4	4	0	R/W	Pulse Train 4 Stopped Interrupt Enable/Disable
pt5	5	0	R/W	Pulse Train 5 Stopped Interrupt Enable/Disable
pt6	6	0	R/W	Pulse Train 6 Stopped Interrupt Enable/Disable
pt7	7	0	R/W	Pulse Train 7 Stopped Interrupt Enable/Disable

PTG_INTEN.[pt8, pt9, pt10, pt11, pt12, pt13, pt14, pt15]

Field	Bits	Sys Reset	Access	Description
pt8	8	0	R/W	Pulse Train 8 Stopped Interrupt Enable/Disable
pt9	9	0	R/W	Pulse Train 9 Stopped Interrupt Enable/Disable
pt10	10	0	R/W	Pulse Train 10 Stopped Interrupt Enable/Disable
pt11	11	0	R/W	Pulse Train 11 Stopped Interrupt Enable/Disable
pt12	12	0	R/W	Pulse Train 12 Stopped Interrupt Enable/Disable
pt13	13	0	R/W	Pulse Train 13 Stopped Interrupt Enable/Disable
pt14	14	0	R/W	Pulse Train 14 Stopped Interrupt Enable/Disable
pt15	15	0	R/W	Pulse Train 15 Stopped Interrupt Enable/Disable

9.6.5 PTn_RATE_LENGTH

PTn_RATE_LENGTH.rate_control

Field	Bits	Sys Reset	Access	Description
rate_control	26:0	26'b0	R/W	Pulse Train Enable/Rate Control

Defines rate at which the pulse train or squarewave output changes state. If this field is zero, the PT instance is disabled.

For square wave mode, this field affects the output toggling rate as follows:

- 0: Halted; output does not change state.
- 1: Reserved; this setting should not be used.
- 2 or higher: Output changes state at a rate defined by (Pulse Train Module Clock / (rate_control - 1)) bps. For example, if this field is set to 2, the output will toggle every PT module clock. If the field is set to 3, the output will toggle every other module clock, and so on.

For pulse train mode, this field affects the output toggling rate as follows:

- 0: Halted; output does not change state.
- 1 or higher: Pulse train output advances to the next bit pattern output state at a rate defined by (Pulse Train Module Clock / (rate_control)) bps. For example, if this field is set to 1, the train pattern will be output at a rate of 1 bit per module clock. If the field is set to 2, the train pattern will be advanced every other module clock, and so on.

PTn_RATE_LENGTH.mode

Field	Bits	Sys Reset	Access	Description
mode	31:27	00001b	R/W	Pulse Train Output Mode/Train Length

Sets either square wave mode or pulse train mode; for pulse train mode, defines pulse train length.

- 00000b: Pulse train, 32 bits
- 00001b: Square Wave Mode (PTx_TRAIN not used)
- 00010b: Pulse train, 2 bits
- 00011b: Pulse train, 3 bits
- ...
- 11111b: Pulse train, 31 bits

9.6.6 PTn_TRAIN

PT0_TRAIN

Sys Reset	Access	Description
00000000h	R/W	Pulse Train Output Pattern

In square wave mode, this register has no effect. In pulse train mode, this register contains the repeating pattern that will be shifted out as the pulse train output stream (starting with LSB)

9.6.7 PTn_LOOP

PT0_LOOP

Sys Reset	Access	Description
0000h	R/W	Pulse Train Loop Count

If set to a nonzero value, this loop counter determines the number of times (once enabled) the pulse train pattern will be output before the pulse train is automatically disabled. This register has no effect in square wave mode. Setting this register to zero (default value) indicates that the pulse train pattern will be output continuously until the pulse train is manually disabled.

Only the low 16 bits of this register are used.

10 Timer/Counters

10.1 Overview

The **MAX32620** supports six 32-bit reloadable timers that can be used for timing, event counting, and generation of pulse-width modulated (PWM) signals. The clock timer input (for all modes) is the ARM Cortex-M4 System Clock. Featured is a programmable prescaler with prescale values from 1 to 4096 from the system clock. Additionally, each of the six timers is able to be split into two 16-bit timers for a possible total of 12 timers in the system.

In 32-bit mode, the following modes of operation are supported:

- **One-Shot Mode**: Timer counts to terminal count value then halts.
- **Continuous Mode**: Timer counts to terminal count value then resets to 0 and repeats.
- **Counter Mode**: counts input edges on the timer I/O pin.
- **PWM Mode**: Generates PWM output waveform based on programmable frequency and duty cycle.
- **Capture Mode**: Captures a snapshot of the current timer count value based on an input edge of the timer I/O pin.
- **Compare Mode**: Toggles output on timer I/O pin when the timer count exceeds the programmable terminal count.
- **Gated Mode**: Timer counts only when the input on the timer I/O pin is asserted.
- **Measurement Mode**: Timer is enabled and begins counting when the timer I/O input is asserted; the timer count value is captured when the timer I/O input is deasserted.

Note Input pin signal frequency is limited to a maximum of 1/4 the currently selected System Clock frequency.

In 16-bit mode, the following modes of operation are supported:

- **One-Shot Mode**: Timer counts to terminal count value then halts.
- **Continuous Mode**: Timer counts to terminal count value then resets to 0 and repeats.

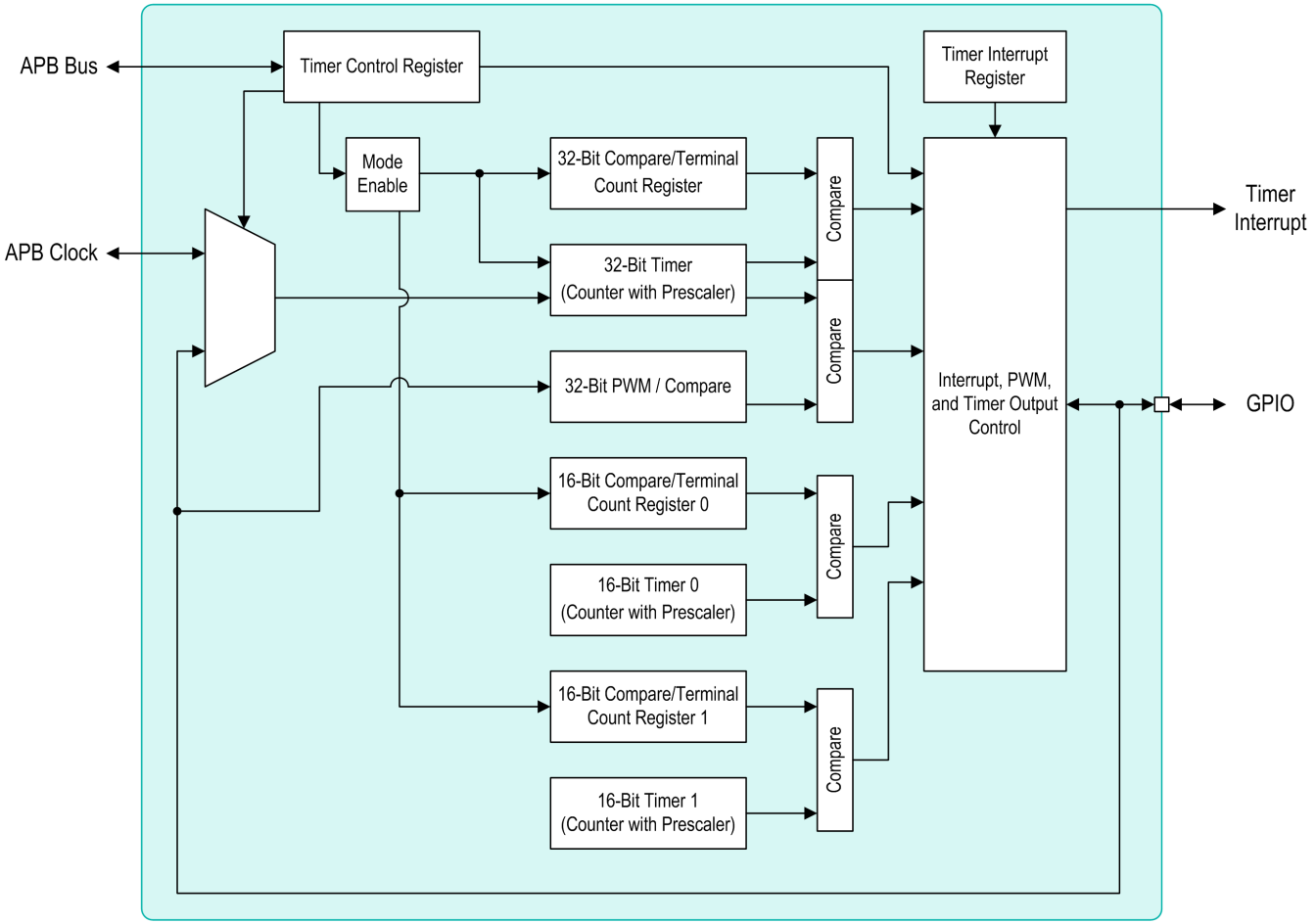

Figure 10.1: Timer Block Diagram

10.2 Timer Port and Pin Configurations

The table below contains the available pin configurations for the six timers (TMR0-5):

NOTE: See Pin Configurations, Packages, and Special Function Multiplexing for a detailed mapping of **MAX32620** multiplexed function locations and priority distinction. Timer modules interface with the GPIO pins, which claim the lowest functional priority (see GPIO Pins and Peripheral Mode Functions for further information).

GPIO Function	Port and Pin
TMR0	P0.0, P0.6, P1.4, P2.2, P3.0, P3.6, P4.4, P5.2, P6.0
TMR1	P0.1, P0.7, P1.5, P2.3, P3.1, P3.7, P4.5, P5.3
TMR2	P0.2, P1.0, P1.6, P2.4, P3.2, P4.0, P4.6, P5.4
TMR3	P0.3, P1.1, P1.7, P2.5, P3.3, P4.1, P4.7, P5.5
TMR4	P0.4, P1.2, P2.0, P2.6, P3.4, P4.2, P5.0, P5.6
TMR5	P0.5, P1.3, P2.1, P2.7, P3.5, P4.3, P5.1, P5.7

10.3 32-bit Mode Timer Operation

The timers are 32-bit up-counter timers. Use TMRn_CTRL.mode to configure the operating control register for the timer mode. Additional mode information is found below.

Minimum time-out delay is set by:

- Loading the value 0x0000_0001 into the Timer Compare register, TMRn_TERM_CNT32
- Setting the prescale value to 1 in the TMRn_CTRL.prescale field

Maximum time-out delay is set by:

- Loading the value 0x0000_0000 into the Timer Compare register, TMRn_TERM_CNT32
- Setting the prescale value to 4096 in the TMRn_CTRL.prescale

If the timer reaches 0xFFFF_FFFF, the timer rolls over to 0x0000_0000 and continues counting.

The current count value, TMRn_COUNT32, in the timers can be read while the timer is counting and enabled. This read action does not affect the timer's operation.

As a rule, the timer output is toggled every time the counter is reloaded.

10.3.1 One-Shot Mode

In One-Shot mode, the timer counts from the start count value stored in the Timer register up to the 32-bit Compare value stored in the Timer Compare register. Upon reaching the Compare value, the timer generates an interrupt and the count value in the Timer register is reset to 0x0000_0001. Then, the timer is automatically disabled and stops counting. Also, if the Timer Output function is enabled in the GPIO, the Timer output pin will change state for one clock cycle and then return to the polarity value (the TMRn_CTRL.polarity bit in the Timer Control register).

The steps for configuring a 32-bit timer for One-Shot mode and initiating the count are as follows:

1. Write the following in the TMRn_CTRL register:
   - Disable the timer, TMRn_CTRL.enable0 = "0"
   - Select 32-bit timer mode, TMRn_CTRL.tmr2x16 = "0"
   - Configure the timer for One-Shot Mode, TMRn_CTRL.mode = "000"
   - Set the prescale value, TMRn_CTRL.prescale
2. If using the Timer Output function, set the initial output level (High or Low) via TMRn_CTRL.polarity field
3. Write the starting count value, TMRn_COUNT32
4. Write the Compare/Terminal count value, TMRn_TERM_CNT32
5. If desired, enable the timer interrupt, TMRn_INTEN.timer0
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function
7. Enable timer and start counting, TMRn_CTRL.enable0 = "1"

In One-Shot mode, the Scaled System Clock always provides the timer input. The timer period is given by the following equation:

$$\text{One Shot Timeout Period}(s) = \frac{(Reload - StartValue)}{SystemClockFrequency(Hz)} \times Prescale$$

10.3.2 Continuous Mode

In Continuous Mode, the timer counts up to the 32-bit Compare value stored in the Timer Compare register, TMRn_TERM_CNT32. The timer input is the Scaled System Clock. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001, and counting resumes. Additionally, if the Timer Output function is enabled in the GPIO, the Timer Output pin changes state from Low-to-High or from High-to-Low upon reaching Timer Compare value.

The steps for configuring a timer for Continuous Mode and initiating the count are as follows:

1. Write the following in the TMRn_CTRL register:
   - Disable the timer, TMRn_CTRL.enable0 = "0"
   - Select 32-bit timer mode, TMRn_CTRL.tmr2x16 = "0"
   - Configure the timer for Continuous Mode, TMRn_CTRL.mode = "001"
   - Set the prescale value, TMRn_CTRL.prescale
2. If using the Timer Output function, set the initial output level (High or Low) via TMRn_CTRL.polarity field
3. Write to the Timer Count register to set the starting count value TMRn_COUNT32
   - This only affects the first pass in Continuous Mode
   - After the first Timer Compare in Continuous Mode, counting always begins at the reset value of 0x0000_0001
4. Write the Compare Count value, TMRn_TERM_CNT32
5. If desired, enable the timer interrupt, TMRn_INTEN.timer0
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function

The timer period is given by the following equation:

$$\text{Continuous Timeout Period}(s) = \frac{Reload}{SystemClockFrequency(Hz)} \times Prescale$$

Note If an initial starting value other than 0x0000_0001 is loaded into the Timer register, the One-Shot Mode equation must be used to determine the first time-out period.

10.3.3 Counter Mode

In Counter Mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the GPIO port pin Timer Input function. The TMRn_CTRL.polarity bit in the Timer Control register selects whether the count occurs on the rising edge (polarity = "0") or the falling edge (polarity = "1") of the Timer Input signal.

For the timer input, the output mode for the GPIO pin must be set to mode 4.

In Counter Mode, the prescaler is disabled. Any value assigned to TMRn_CTRL.prescale in Counter Mode will have no effect. *The input frequency of the Timer Input signal must not exceed one-fourth the currently selected System Clock frequency.* Upon reaching the Compare value stored in the Timer Terminal Count register, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001 and counting resumes. Also, if the Timer Output function is enabled in the GPIO, the Timer Output pin changes state from Low-to-High or from High-to-Low at Timer Compare.

The steps for configuring a timer for Counter Mode and initiating the count are as follows:

1. Write the following in the TMRn_CTRL register:
   - Disable the timer, TMRn_CTRL.enable0 = "0"
   - Select 32-bit timer mode, TMRn_CTRL.tmr2x16 = "0"
   - Configure the timer for Counter Mode, TMRn_CTRL.mode = "010"
   - Select either the rising edge or falling edge of the Timer Input signal for the count, TMRn_CTRL.polarity (**NOTE**: This also sets the initial logic level (High or Low) for the Timer Output function; however, the Timer Output function *does not* have to be enabled via GPIO)
2. Write to the Timer register to set the starting count value, TMRn_COUNT32
   - This only affects the first pass in Counter Mode
   - After the first Timer Compare in Counter Mode, counting always begins at the reset value of 0x0000_0001
   - In Counter Mode in general, the Timer register should be written with the value 0x0000_0001
3. Write the Compare Count value, TMRn_TERM_CNT32
4. If desired, enable the timer interrupt, TMRn_INTEN.timer0
5. Configure the associated GPIO port pin for the Timer Input function, and set the output mode for the GPIO pin to mode 4.
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See GPIO Pins and Peripheral Mode Functions for further information.)
7. Enable timer and start counting, TMRn_CTRL.enable0 = "1"

In Counter Mode, the number of Timer Input transitions since the timer start is given by the following equation:

$$\text{Counter Mode Timer Input Transition} = CurrentCountValue - StartValue$$

10.3.4 PWM Mode

In PWM Mode, the timer outputs a Pulse-Width Modulated (PWM) output signal through a GPIO port pin. The timer first counts up to the 32-bit PWM compare value stored in the TMRn_PWM_CAP32 register. When the timer count value matches the PWM value, the Timer Output toggles. The timer continues counting until it reaches the Compare value stored in the TMRn_TERM_CNT32. Upon reaching the TMRn_TERM_CNT32 value, the Timer Output signal toggles again, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001, and counting resumes.

When TMRn_CTRL.polarity = "0": Timer Output signal begins as a Low ("0") and then transitions to a High ("1") when the timer value matches the PWM TMRn_PWM_CAP32 value. The Timer Output signal returns to a Low ("0") after the timer reaches the Compare value, TMRn_TERM_CNT32, which is reset to 0x0000_0001. When TMRn_CTRL.polarity = "1": Timer Output signal begins as a High ("1") and then transitions to a Low ("0") when the timer value matches the PWM TMRn_PWM_CAP32 Compare/Match value. The Timer Output signal returns to a High ("1") after the timer reaches the Compare value, TMRn_TERM_CNT32, which is reset to 0x0000_0001.

The steps for configuring a timer for PWM mode and initiating the PWM operation are as follows:

1. Write the following in the TMRn_CTRL register:
   - Disable the timer, TMRn_CTRL.enable0 = "0"
   - Select 32-bit timer mode, TMRn_CTRL.tmr2x16 = "0"
   - Configure the timer for PWM Mode, TMRn_CTRL.mode = "011"
   - Set the prescale value, TMRn_CTRL.prescale
   - Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output function, TMRn_CTRL.polarity.
2. Write to the Timer register, TMRn_COUNT32, to set the starting count value (typically 0x0000_0001)
   - This only affects the first pass in PWM mode
   - This value is only used on the initial PWM signal pass when the PWM starts
   - After the first timer reset in PWM mode, counting always begins at the reset value of 0x0000_0001
3. Write to the PWM Compare register, TMRn_PWM_CAP32, to set the desired match value
4. Write to the Timer Compare/Match register, TMRn_TERM_CNT32, to set the Compare value
   - **NOTE**: The Compare value must be greater than the PWM value
5. If desired, enable the timer interrupt TMRn_INTEN.timer0
6. Configure the associated GPIO port pin for the Timer Output function. (See GPIO Pins and Peripheral Mode Functions for further information.)
7. Enable timer and start counting, TMRn_CTRL.enable0 = "1"

The PWM period is given by the following equation:

$$\text{PWM Period}(s) = \frac{ReloadValue}{SystemClockFrequency(Hz)} \times Prescale$$

Starting count values other than 0x0000_0001

If an initial starting value other than 0x0000_0001 is loaded into the Timer register, the One-Shot Mode equation must be used to determine the first PWM time-out period.

If TMRn_CTRL.polarity is set to "0", the ratio of the PWM output High time to the total period is given by:

$$\text{PWM Output High Time Ratio}(\%) = \frac{(ReloadValue - PWMValue)}{ReloadValue} \times 100$$

If TMRn_CTRL.polarity is set to "1", the ratio of the PWM output High time to the total period is given by:

$$\text{PWM Output High Time Ratio}(\%) = \frac{PWMValue}{ReloadValue} \times 100$$

10.3.5 Capture Mode

In Capture Mode, the current timer count value is recorded when the desired external Timer Input transition occurs. The Capture Mode count value is stored in the Timer PWM register. The TMRn_CTRL.polarity bit in the Timer Control register determines if the Capture occurs on a rising edge (TMRn_CTRL.polarity = "0") or a falling edge (TMRn_CTRL.polarity = "1") of the Timer Input signal.

For the timer input, the output mode for the GPIO pin must be set to mode 4.

When the Capture event occurs, an interrupt is generated, and the timer continues counting. The timer continues counting up to the 32-bit compare value stored in the Timer Compare/Match register, TMRn_TERM_CNT32. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer count register TMRn_COUNT32 is reset to 0x0000_0001, and counting resumes.

The steps for configuring a timer for Capture Mode and initiating the count are as follows:

1. Write the following in the TMRn_CTRL register:
   - Disable the timer, TMRn_CTRL.enable0 = "0"
   - Select 32-bit timer mode, TMRn_CTRL.tmr2x16 = "0"
   - Configure the timer for Capture Mode, TMRn_CTRL.mode = "100"
   - Set the prescale value, TMRn_CTRL.prescale
   - Set the Capture edge (rising or falling) for the Timer Input, TMRn_CTRL.polarity
2. Write to the Timer Count register, TMRn_COUNT32, to set the starting count value, typically 0x0000_0001
3. Write to the Timer Compare register, TMRn_TERM_CNT32, to set the Compare value
4. If desired, enable the timer interrupt, TMRn_INTEN.timer0
5. Configure the associated GPIO port pin for the Timer Input function, and set the output mode for the GPIO pin to mode 4.
6. Enable timer and start counting, TMRn_CTRL.enable0 = "1"

In Capture mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time}(s) = \frac{(CaptureValue - StartValue)}{SystemClockFrequency(Hz)} \times Prescale$$

10.3.6 Compare Mode

In Compare Mode, the timer counts up to the compare value stored in the Timer Compare register. Upon reaching the compare value, the timer generates an interrupt and counting continues (the timer value is not reset to 0x0000_0001). Also, if the Timer Output function is enabled (in the GPIO), the Timer Output pin changes state (from Low-to-High or from High-to-Low) upon a Compare Match. When the Timer reaches 0xFFFF_FFFF, the timer rolls over to 0x0000_0000 and continues counting.

The steps for configuring a timer for Compare mode and initiating the count are as follows:

1. Write the following to the TMRn_CTRL register:
   - Disable the timer, TMRn_CTRL.enable0 = "0"
   - Select 32-bit timer mode, TMRn_CTRL.tmr2x16 = "0"
   - Configure the timer for Compare Mode, TMRn_CTRL.mode = "101"
   - Set the prescale value, TMRn_CTRL.prescale
2. If using the Timer Output function, set the initial logic level (High or Low), TMRn_CTRL.polarity
3. Write to the Timer Count register, TMRn_COUNT32, to set the starting count value
4. Write to the Timer Compare register, TMRn_TERM_CNT32, to set the Compare value
5. If desired, enable the timer interrupt, TMRn_INTEN.timer0
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See GPIO Pins and Peripheral Mode Functions for further information.)
7. Enable timer and start counting, TMRn_CTRL.enable0 = "1"

In Compare mode, the currently selected System Clock always provides the timer input. The Compare time is given by the following equation:

$$\text{Compare Mode Time}(s) = \frac{(CompareValue - StartValue)}{SystemClockFrequency(Hz)} \times Prescale$$

10.3.7 Gated Mode

In Gated Mode, the timer counts only when the Timer Input signal is in its active state (asserted), as determined by the TMRn_CTRL.polarity bit in the Timer Control register. When the Timer Input signal is asserted, counting begins. A timer interrupt is generated when the Timer Input signal is deasserted or a Timer Compare occurs. The timer counts up to the 32-bit Compare value stored in the Timer Compare register, TMRn_TERM_CNT32. The timer input is the currently selected System Clock. When reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001, and counting

resumes (assuming the Timer Input signal is still asserted). Also, if the Timer Output function is enabled (in the GPIO), the Timer Output pin changes state (from Low-to-High or from High-to-Low) at Timer Compare.

For the timer input, the output mode for the GPIO pin must be set to mode 4.

The steps for configuring a timer for Gated Mode and initiating the count are as follows:

1. Write the following to the TMRn_CTRL register:
   - Disable the timer, TMRn_CTRL.enable0 = "0"
   - Select 32-bit timer mode, TMRn_CTRL.tmr2x16 = "0"
   - Configure the timer for Gated Mode, TMRn_CTRL.mode = "110"
   - Set the prescale value, TMRn_CTRL.prescale
2. Write to the Timer Count register, TMRn_COUNT32, to set the starting count value
   - This only affects the first pass in Gated Mode
   - Counting always begins at 0x0000_0001 after the first timer reset
3. Write to the Timer Compare register, TMRn_TERM_CNT32, to set the Compare value
4. If desired, enable the timer interrupt, TMRn_INTEN.timer0
5. Configure the associated GPIO port pin for the Timer Input function, and set the output mode for the GPIO pin to mode 4.
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See GPIO Pins and Peripheral Mode Functions for further information.)
7. Enable the timer, TMRn_CTRL.enable0 = "1"
8. Assert the Timer Input signal to initiate the counting
   - High if TMRn_CTRL.polarity = "0"
   - Low if TMRn_CTRL.polarity = "1"

10.3.8 Measurement Mode

In Measurement Mode, the timer begins counting after the first desired external Timer Input transition occurs. The selected transition (rising edge or falling edge) is set by the TMRn_CTRL.polarity bit in the Timer Control Register. The timer input is the currently selected System Clock. Every subsequent transition, after the first transition, captures the current count value. The Captured timer value is written to the Timer PWM register, TMRn_PWM_CAP32. When the Capture event occurs, an interrupt is generated, the count value in the Timer register is reset to 0x0000_0001, and counting resumes. If no Capture event occurs, the timer counts up to the 32-bit Compare value stored in the Timer Compare register, TMRn_TERM_CNT32. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001, and counting resumes.

For the timer input, the output mode for the GPIO pin must be set to mode 4.

The steps for configuring a timer for Measurement mode and initiating the count are as follows:

1. Write the following to the TMRn_CTRL register:
   - Disable the timer, TMRn_CTRL.enable0 = "0"
   - Select 32-bit timer mode, TMRn_CTRL.tmr2x16 = "0"
   - Configure the timer for Measurement mode, TMRn_CTRL.mode = "111"
   - Set the prescale value, TMRn_CTRL.prescale
   - Set the Timer Input Capture edge
     - Rising edge: TMRn_CTRL.polarity = "0"
     - Falling edge: TMRn_CTRL.polarity = "1"
2. Write to the Timer register to set the starting count value, typically 0x0000_0001
3. Write to the Timer Compare register, TMRn_COUNT32, to set the Compare value
4. If desired, enable the timer interrupt, TMRn_INTEN.timer0
5. Configure the associated GPIO port pin for the Timer Input function, and set the output mode for the GPIO pin to mode 4.
6. Enable the timer, TMRn_CTRL.enable0 = "1"
7. Counting begins after the first desired transition of the Timer Input signal
   - No interrupt is generated by this first edge.

In Measurement mode, the elapsed time from timer start to the Capture event can be calculated using the following equation:

$$\text{Measure Elapsed Time}(s) = \frac{(MeasurementValue - StartValue) \times Prescale}{SystemClockFrequency(Hz)}$$

10.4 16-bit Mode Timer Operation

Each of the six 32-bit timers on the **MAX32620** can be split into 2 x 16-bit timers, for a total of up to 12 16-bit timers. Configuration and operation of 16-bit mode is very similar to the 32-bit modes of operation, but when configured as two 16-bit timers, only One-Shot and Continuous Modes of operation are supported.

Minimum time-out delay for each 16-bit timer is set by:

- Loading the value 0x0001 into the Timer Compare register, either the TMRn_TERM_CNT16_0 or TMRn_TERM_CNT16_1
- Setting the prescale value to 1 in the TMRn_CTRL.prescale field

Maximum time-out delay is set by:

- Loading the value 0x0000 into the Timer Compare register, TMRn_TERM_CNT16_0 or TMRn_TERM_CNT16_1
- Setting the prescale value to 4096 in the TMRn_CTRL.prescale field

- If the Timer reaches 0xFFFF, the timer rolls over to 0x0000 and continues counting

The current count value, [TMRn_COUNT16_0.value}(field_TMRn_COUNT16_0_value) or TMRn_COUNT16_1.value, in the timers can be read while the timer is counting and enabled. This action does not affect the timer's operation.

Timer output does not change in 16-bit mode. If a GPIO pin is configured to act as a timer output for a timer instance which has been split into two 16-bit timers, the timer output will be fixed and will always equal the output polarity value set in TMRn_CTRL.

10.4.1 One-Shot Mode

In One-Shot mode, the timer counts from the start count value stored in the Timer register up to the 16-bit Compare value stored in the Timer Compare register. Upon reaching the Compare value, the timer generates an interrupt, and the count value in the Timer register is reset to 0x0001; the timer is then automatically disabled and stops counting.

The steps for configuring a 16-bit timer for One-Shot mode and initiating the count are:

1. Write the following to the TMRn_CTRL register:
   - Disable the timer, TMRn_CTRL.enable0 or TMRn_CTRL.enable1 = "0"
   - Select the dual 16-bit timer mode, TMRn_CTRL.tmr2x16 = "1"
   - Configure the timer for One-Shot Mode, TMRn_CTRL.mode = "000"
   - Set the prescale value, TMRn_CTRL.prescale
2. Write the starting count value, TMRn_COUNT16_0.value or TMRn_COUNT16_1.value, depending on which one of the 16-bit timers is being used
3. Write the Compare/Terminal count value,TMRn_TERM_CNT16_0.term_count or TMRn_TERM_CNT16_1.term_count
4. If desired, enable the timer interrupt, TMRn_INTEN.timer0 or TMRn_INTEN.timer1
5. Enable timer and start counting, TMRn_CTRL.enable0 or TMRn_CTRL.enable1 = "1"

In One-Shot mode, the System Clock always provides the timer input. The timer period is given by the following equation:

$$\text{One Shot Timeout Period}(s) = \frac{(Reload - StartValue)}{SystemClockFrequency(Hz)} \times Prescale$$

10.4.2 Continuous Mode

In Continuous Mode, the timer counts up to the 16-bit Compare value stored in the Timer Compare register, TMRn_TERM_CNT16_0 or TMRn_TERM_CNT16_1. The timer input is the currently selected system clock. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0x0001, and counting resumes.

The steps for configuring a timer for Continuous Mode and initiating the count are as follows:

1. Write the following to the TMRn_CTRL register:
   - Disable the timer, TMRn_CTRL.enable0 or TMRn_CTRL.enable1 = "0"
   - Select 16-bit timer mode, TMRn_CTRL.tmr2x16 = "1"
   - Configure the timer for Continuous Mode, TMRn_CTRL.mode = "001"
   - Set the prescale value, TMRn_CTRL.prescale
2. Write to the Timer register to set the starting count value, TMRn_COUNT16_0.value or TMRn_COUNT16_1.value, depending on which one of the 16-bit timers is being used
   - This only affects the first, starting, operation
   - After the first Timer Compare in Continuous Mode, counting always begins at the reset value of 0x0001
3. Write the Compare Count value, TMRn_TERM_CNT16_0.term_count or TMRn_TERM_CNT16_1.term_count
4. If desired, enable the timer interrupt, TMRn_INTEN.timer0 or TMRn_INTEN.timer1
5. Enable timer and start counting, TMRn_CTRL.enable0 or TMRn_CTRL.enable1 = "1"

In Continuous Mode, the currently selected System Clock always provides the timer input. The timer period is given by the following equation:

$$\text{Continuous Timeout Period}(s) = \frac{ReloadValue}{SystemClockFrequency(Hz)} \times Prescale$$

If an initial starting value other than 0x0001 is loaded into the Timer register, the One-Shot Mode equation must be used to determine the first time-out period.

10.5 Registers (TMR)

Address	Register	Access	Description	Reset By
0x4000B000	TMR0_CTRL	R/W	Timer 0 Control Register	Sys
0x4000B004	TMR0_COUNT32	R/W	Timer 0 [32 bit] Current Count Value	Sys
0x4000B008	TMR0_TERM_CNT32	R/W	Timer 0 [32 bit] Terminal Count Setting	Sys
0x4000B00C	TMR0_PWM_CAP32	R/W	Timer 0 [32 bit] PWM Compare Setting or Capture/Measure Value	Sys
0x4000B010	TMR0_COUNT16_0	R/W	Timer 0 [16 bit] Current Count Value, 16-bit Timer 0	Sys
0x4000B014	TMR0_TERM_CNT16_0	R/W	Timer 0 [16 bit] Terminal Count Setting, 16-bit Timer 0	Sys
0x4000B018	TMR0_COUNT16_1	R/W	Timer 0 [16 bit] Current Count Value, 16-bit Timer 1	Sys
0x4000B01C	TMR0_TERM_CNT16_1	R/W	Timer 0 [16 bit] Terminal Count Setting, 16-bit Timer 1	Sys
0x4000B020	TMR0_INTFL	W1C	Timer 0 Interrupt Flags	Sys
0x4000B024	TMR0_INTEN	R/W	Timer 0 Interrupt Enable/Disable Settings	Sys
0x4000C000	TMR1_CTRL	R/W	Timer 1 Control Register	Sys
0x4000C004	TMR1_COUNT32	R/W	Timer 1 [32 bit] Current Count Value	Sys
0x4000C008	TMR1_TERM_CNT32	R/W	Timer 1 [32 bit] Terminal Count Setting	Sys
0x4000C00C	TMR1_PWM_CAP32	R/W	Timer 1 [32 bit] PWM Compare Setting or Capture/Measure Value	Sys
0x4000C010	TMR1_COUNT16_0	R/W	Timer 1 [16 bit] Current Count Value, 16-bit Timer 0	Sys
0x4000C014	TMR1_TERM_CNT16_0	R/W	Timer 1 [16 bit] Terminal Count Setting, 16-bit Timer 0	Sys
0x4000C018	TMR1_COUNT16_1	R/W	Timer 1 [16 bit] Current Count Value, 16-bit Timer 1	Sys
0x4000C01C	TMR1_TERM_CNT16_1	R/W	Timer 1 [16 bit] Terminal Count Setting, 16-bit Timer 1	Sys
0x4000C020	TMR1_INTFL	W1C	Timer 1 Interrupt Flags	Sys
0x4000C024	TMR1_INTEN	R/W	Timer 1 Interrupt Enable/Disable Settings	Sys
0x4000D000	TMR2_CTRL	R/W	Timer 2 Control Register	Sys
0x4000D004	TMR2_COUNT32	R/W	Timer 2 [32 bit] Current Count Value	Sys

Address	Register	Access	Description	Reset By
0x4000D008	TMR2_TERM_CNT32	R/W	Timer 2 [32 bit] Terminal Count Setting	Sys
0x4000D00C	TMR2_PWM_CAP32	R/W	Timer 2 [32 bit] PWM Compare Setting or Capture/Measure Value	Sys
0x4000D010	TMR2_COUNT16_0	R/W	Timer 2 [16 bit] Current Count Value, 16-bit Timer 0	Sys
0x4000D014	TMR2_TERM_CNT16_0	R/W	Timer 2 [16 bit] Terminal Count Setting, 16-bit Timer 0	Sys
0x4000D018	TMR2_COUNT16_1	R/W	Timer 2 [16 bit] Current Count Value, 16-bit Timer 1	Sys
0x4000D01C	TMR2_TERM_CNT16_1	R/W	Timer 2 [16 bit] Terminal Count Setting, 16-bit Timer 1	Sys
0x4000D020	TMR2_INTFL	W1C	Timer 2 Interrupt Flags	Sys
0x4000D024	TMR2_INTEN	R/W	Timer 2 Interrupt Enable/Disable Settings	Sys
0x4000E000	TMR3_CTRL	R/W	Timer 3 Control Register	Sys
0x4000E004	TMR3_COUNT32	R/W	Timer 3 [32 bit] Current Count Value	Sys
0x4000E008	TMR3_TERM_CNT32	R/W	Timer 3 [32 bit] Terminal Count Setting	Sys
0x4000E00C	TMR3_PWM_CAP32	R/W	Timer 3 [32 bit] PWM Compare Setting or Capture/Measure Value	Sys
0x4000E010	TMR3_COUNT16_0	R/W	Timer 3 [16 bit] Current Count Value, 16-bit Timer 0	Sys
0x4000E014	TMR3_TERM_CNT16_0	R/W	Timer 3 [16 bit] Terminal Count Setting, 16-bit Timer 0	Sys
0x4000E018	TMR3_COUNT16_1	R/W	Timer 3 [16 bit] Current Count Value, 16-bit Timer 1	Sys
0x4000E01C	TMR3_TERM_CNT16_1	R/W	Timer 3 [16 bit] Terminal Count Setting, 16-bit Timer 1	Sys
0x4000E020	TMR3_INTFL	W1C	Timer 3 Interrupt Flags	Sys
0x4000E024	TMR3_INTEN	R/W	Timer 3 Interrupt Enable/Disable Settings	Sys
0x4000F000	TMR4_CTRL	R/W	Timer 4 Control Register	Sys
0x4000F004	TMR4_COUNT32	R/W	Timer 4 [32 bit] Current Count Value	Sys
0x4000F008	TMR4_TERM_CNT32	R/W	Timer 4 [32 bit] Terminal Count Setting	Sys
0x4000F00C	TMR4_PWM_CAP32	R/W	Timer 4 [32 bit] PWM Compare Setting or Capture/Measure Value	Sys
0x4000F010	TMR4_COUNT16_0	R/W	Timer 4 [16 bit] Current Count Value, 16-bit Timer 0	Sys
0x4000F014	TMR4_TERM_CNT16_0	R/W	Timer 4 [16 bit] Terminal Count Setting, 16-bit Timer 0	Sys
0x4000F018	TMR4_COUNT16_1	R/W	Timer 4 [16 bit] Current Count Value, 16-bit Timer 1	Sys

Address	Register	Access	Description	Reset By
0x4000F01C	TMR4_TERM_CNT16_1	R/W	Timer 4 [16 bit] Terminal Count Setting, 16-bit Timer 1	Sys
0x4000F020	TMR4_INTFL	W1C	Timer 4 Interrupt Flags	Sys
0x4000F024	TMR4_INTEN	R/W	Timer 4 Interrupt Enable/Disable Settings	Sys
0x40010000	TMR5_CTRL	R/W	Timer 5 Control Register	Sys
0x40010004	TMR5_COUNT32	R/W	Timer 5 [32 bit] Current Count Value	Sys
0x40010008	TMR5_TERM_CNT32	R/W	Timer 5 [32 bit] Terminal Count Setting	Sys
0x4001000C	TMR5_PWM_CAP32	R/W	Timer 5 [32 bit] PWM Compare Setting or Capture/Measure Value	Sys
0x40010010	TMR5_COUNT16_0	R/W	Timer 5 [16 bit] Current Count Value, 16-bit Timer 0	Sys
0x40010014	TMR5_TERM_CNT16_0	R/W	Timer 5 [16 bit] Terminal Count Setting, 16-bit Timer 0	Sys
0x40010018	TMR5_COUNT16_1	R/W	Timer 5 [16 bit] Current Count Value, 16-bit Timer 1	Sys
0x4001001C	TMR5_TERM_CNT16_1	R/W	Timer 5 [16 bit] Terminal Count Setting, 16-bit Timer 1	Sys
0x40010020	TMR5_INTFL	W1C	Timer 5 Interrupt Flags	Sys
0x40010024	TMR5_INTEN	R/W	Timer 5 Interrupt Enable/Disable Settings	Sys

10.5.1 TMRn_CTRL

TMRn_CTRL.mode

Field	Bits	Sys Reset	Access	Description
mode	2:0	000b	R/W	Operating Modes for 32-bit/16-bit Timers

For single 32-bit timer mode (tmr2x16=0):

- 000b: One Shot Mode
- 001b: Continuous Mode
- 010b: Counter Mode
- 011b: PWM Mode
- 100b: Capture Mode
- 101b: Compare Mode
- 110b: Gated Mode
- 111b: Measurement Mode

For dual 16-bit timer mode (tmr2x16=1):

Bit 0 controls operating mode for 16-bit timer 0:

- 0: One Shot Mode
- 1: Continuous Mode

Bit 1 controls operating mode for 16-bit timer 1:

- 0: One Shot Mode
- 1: Continuous Mode

Bit 2 is not used in dual 16-bit timer mode.

TMRn_CTRL.tmr2x16

Field	Bits	Sys Reset	Access	Description
tmr2x16	3	0	R/W	Dual 16-bit Timer Mode

This bit determines whether this timer instance operates as a single 32-bit timer or dual 16-bit timers.

- 0: Single 32-bit timer
- 1: Dual 16-bit timers

TMRn_CTRL.prescale

Field	Bits	Sys Reset	Access	Description
prescale	7:4	0000b	R/W	Timer Clock Prescale Setting

Determines what divide down value (if any) is used when generating the timer clock from the current system clock. This setting is used by both the 32-bit timer and by one or both of the 16-bit timers (when enabled). Disabling the 32-bit timer or both 16-bit timers automatically resets this field to 0 (div by 1).

- 0: Divide by 1
- 1: Divide by 2
- 2: Divide by 4
- 3: Divide by 8
- 4: Divide by 16
- 5: Divide by 32
- 6: Divide by 64
- 7: Divide by 128
- 8: Divide by 256
- 9: Divide by 512
- 10: Divide by 1024
- 11: Divide by 2048
- 12: Divide by 4096
- Other: Undefined

TMRn_CTRL.polarity

Field	Bits	Sys Reset	Access	Description
polarity	8	0	R/W	Timer I/O Polarity

Polarity control for pad I/O when the 32-bit timer is used in Counter Mode, PWM Mode, Capture Mode, Compare Mode, Gated Mode, or Measurement Mode.,

- 0: Normal polarity
- 1: Inverted polarity

TMRn_CTRL.enable0

Field	Bits	Sys Reset	Access	Description
enable0	12	0	R/W	Enable 32-bit timer / 16-bit timer 0

For single 32-bit timer mode (tmr2x16=0):

- 0: 32-bit timer is disabled
- 1: 32-bit timer is enabled

For dual 16-bit timer mode (tmr2x16=1):

- 0: 16-bit timer 0 is disabled
- 1: 16-bit timer 0 is enabled

TMRn_CTRL.enable1

Field	Bits	Sys Reset	Access	Description
enable1	13	0	R/W	Enable 16-bit timer 1

(Not used for single 32-bit timer mode)

For dual 16-bit timer mode (tmr2x16=1):

- 0: 16-bit timer 1 is disabled
- 1: 16-bit timer 1 is enabled

10.5.2 TMRn_COUNT32

TMR0_COUNT32

Sys Reset	Access	Description
0000 0000 0000 0000 0000 0000 0000 0000b	R/W	Timer [32 bit] Current Count Value

In 32-bit timer mode (tmr2x16=0), this register holds the current count value for the 32-bit timer.

In 16-bit timer mode (tmr2x16=1), this register cannot be accessed and all reads from this location will return zero.

10.5.3 TMRn_TERM_CNT32

TMR0_TERM_CNT32

Sys Reset	Access	Description
0000 0000 0000 0000 0000 0000 0000 0000b	R/W	Timer [32 bit] Terminal Count Setting

In 32-bit timer mode (tmr2x16=0), this register holds the terminal count value for the 32-bit timer.

In 16-bit timer mode (tmr2x16=1), this register cannot be accessed and all reads from this location will return zero.

10.5.4 TMRn_PWM_CAP32

TMR0_PWM_CAP32

Sys Reset	Access	Description
0000 0000 0000 0000 0000 0000 0000 0000b	R/W	Timer [32 bit] PWM Compare Setting or Capture/Measure Value

In 32-bit timer mode (tmr2x16=0), this register holds the PWM compare setting or the capture/measure value (depending on the operating mode) for the 32-bit timer.

In 16-bit timer mode (tmr2x16=1), this register cannot be accessed and all reads from this location will return zero.

10.5.5 TMRn_COUNT16_0

TMRn_COUNT16_0.value

Field	Bits	Sys Reset	Access	Description
value	15:0	0000h	R/W	Count Value

In 16-bit timer mode (tmr2x16=1), this register holds the current count value for 16-bit timer 0.

In 32-bit timer mode (tmr2x16=0), this register cannot be accessed and all reads from this location will return zero.

10.5.6 TMRn_TERM_CNT16_0

TMRn_TERM_CNT16_0.term_count

Field	Bits	Sys Reset	Access	Description
term_count	15:0	0000h	R/W	Terminal Count Setting

In 16-bit timer mode (tmr2x16=1), this register holds the terminal count setting for 16-bit timer 0.

In 32-bit timer mode (tmr2x16=0), this register cannot be accessed and all reads from this location will return zero.

10.5.7 TMRn_COUNT16_1

TMRn_COUNT16_1.value

Field	Bits	Sys Reset	Access	Description
value	15:0	0000h	R/W	Count Value

In 16-bit timer mode (tmr2x16=1), this register holds the current count value for 16-bit timer 1.

In 32-bit timer mode (tmr2x16=0), this register cannot be accessed and all reads from this location will return zero.

10.5.8 TMRn_TERM_CNT16_1

TMRn_TERM_CNT16_1.term_count

Field	Bits	Sys Reset	Access	Description
term_count	15:0	0000h	R/W	Terminal Count Setting

In 16-bit timer mode (tmr2x16=1), this register holds the terminal count setting for 16-bit timer 1.

In 32-bit timer mode (tmr2x16=0), this register cannot be accessed and all reads from this location will return zero.

10.5.9 TMRn_INTFL

TMRn_INTFL.timer0

Field	Bits	Sys Reset	Access	Description
timer0	0	0	W1C	Interrupt Flag for 32-bit Timer / 16-bit Timer 0

Hardware sets this flag to 1 when the timer reaches the terminal count or a capture value is obtained (32-bit mode only) or when the input pad is deasserted in Gated mode (32-bit mode only).

Write 1 to clear this flag to 0.

TMRn_INTFL.timer1

Field	Bits	Sys Reset	Access	Description
timer1	1	0	W1C	Interrupt Flag for 16-bit Timer 1

Hardware sets this flag to 1 when the timer reaches the terminal count.

Write 1 to clear this flag to 0.

10.5.10 TMRn_INTEN

TMRn_INTEN.timer0

Field	Bits	Sys Reset	Access	Description
timer0	0	0	R/W	Interrupt Enable for 32-bit Timer / 16-bit Timer 0

Enable/disable setting to allow a timer interrupt to be triggered when the corresponding interrupt flag is set.

0: Interrupt disabled (no interrupt will be triggered) 1: Interrupt enabled (interrupt may trigger normally)

TMRn_INTEN.timer1

Field	Bits	Sys Reset	Access	Description
timer1	1	0	R/W	Interrupt Enable for 16-bit Timer 1

Enable/disable setting to allow a timer interrupt to be triggered when the corresponding interrupt flag is set.

0: Interrupt disabled (no interrupt will be triggered) 1: Interrupt enabled (interrupt may trigger normally)

11 Real Time Clock (RTC)

11.1 Real Time Clock Overview

The Real Time Clock (RTC) on the **MAX32620** is designed to operate largely independent of the other digital and analog functions on the device. The RTC has its own clock that runs off a 4kHz clock derived from the external crystal 32kHz oscillator output.

The RTC always runs from the V_{RTC} dedicated backup supply, which is intended to remain on even during the lowest power state LP0:STOP.

The RTC consumes very little power and can operate from its dedicated 4kHz clock base. It is, therefore, possible for the RTC to continue operating while the rest of the **MAX32620** is in the lowest power saving mode, LP0:STOP. The RTC can be used to wake the device automatically from LP0:STOP or LP1:STANDBY after a preprogrammed time interval (using the time of day alarm function). It can also be used to maintain a stable time value that continues to keep counting time properly even when the rest of the system has been powered down or power has been removed entirely. As long as the backup supply V_{RTC} is present, the Real Time Clock can continue operating normally.

11.1.1 Real Time Clock Features

The Real Time Clock on the **MAX32620** includes the following features:

- Dedicated low-frequency, low-power 4kHz clock source (derived from 32kHz external crystal oscillator).
- Continued operation when main portion of system is powered off.
- 32-bit RTC timer.
- Integrated prescaler determines interval between RTC timer ticks: from 244 microseconds (fastest RTC tick rate) to eight seconds (slowest RTC tick rate).
- Automatic synchronization of timers, configuration registers, and status/interrupt flags between high-frequency and 4kHz clock domains.
- Two separate 32-bit timer compare registers allow a wakeup and/or interrupt event to occur when the RTC timer reaches a predetermined alarm value.
- Integrated prescaler compare mask allows a wakeup and/or interrupt event to occur at a regular sub-RTC-tick interval (subsecond/interval alarm).
- 'Snooze' function allows the **MAX32620** to wake up and go back to sleep at periodic intervals (by automatically incrementing the compare value of the time-of-day alarm by a preset value each wake/sleep cycle) with no CPU read/modify of the time-of-day alarm compare register needed.

11.2 RTC Resets

The standard System Reset and Power-on Reset (POR) events do not halt the operation of the RTC nor clear its register contents. Once the RTC has been configured properly and the timer is running, the RTC will continue normal operation during System Reset and/or POR conditions.

Since the RTC has its own backup power supply (V_{RTC}), it will only reset in the event that power fails on that supply. In this case, the RTC will be reset and all associated RTC registers will be cleared to the default state. This is also known as the Battery On Reset (BOR) event.

The RSTN power sequencer reset pin does not affect the operation of the RTC and does not clear any RTCTMR or RTCCFG register contents.

Simply resetting the RTC oscillator (e.g., setting osc_bypass to 1) does not reset the RTC as a whole nor clear RTC register contents.

11.3 RTC Interrupts

The RTC module reports interrupts to the CPU core using the following interrupt vector channels.

Interrupt Number	Vector	Interrupt Source(s)
3	0x13	RTC Alarm (Time-of-Day) Compare 0 (match between COMP0 and RTC timer)
4	0x14	RTC Alarm (Time-of-Day) Compare 1 (match between COMP1 and RTC timer)
5	0x15	RTC Interval Alarm (masked PRESCALE matches zero)
6	0x16	RTC Timer Overflow Interrupt, RTC Trim Adjust Interrupt

Although the 4kHz clock domain interrupt flag can be set while the main system is powered off (e.g., if a time of day alarm is matched), main power must be present in order for the system to wake up and service the interrupt. The CPU cannot operate from the V_{RTC} backup supply without the main power supplies (V_{DD18} and V_{DD12}) active.

In order for the values of interrupt flags to be synchronized between the low-power RTC block and the CPU-accessible registers in the digital core, the clock source for the synchronizer block must be enabled by setting CLKMAN_SYS_CLK_CTRL_1_SYNC.sync_clk_scale to a non-zero value. By default, this scaled clock is disabled.

An RTC flag may be used to generate a CPU interrupt if the corresponding RTCTMR_INTEN bits are set. Additionally, an RTC flag may be used to wake up from LP0:STOP or LP1:STANDBY if the corresponding PWRSEQ_MSK_FLAGS are configured.

sync_clk_scale	Synchronizer Clock Setting
0	CLK is Disabled (Default)
1	CLK = (System Clock Source / 1)
2	CLK = (System Clock Source / 2)
3	CLK = (System Clock Source / 4)
4	CLK = (System Clock Source / 8)
5	CLK = (System Clock Source / 16)
6	CLK = (System Clock Source / 32)
7	CLK = (System Clock Source / 64)
8	CLK = (System Clock Source / 128)

sync_clk_scale	Synchronizer Clock Setting
9	CLK = (System Clock Source / 256)
other	CLK = (System Clock Source / 1)

Note This clock is disabled by default following reset.

11.4 RTC Configuration

In order for firmware to configure the RTC for a desired application and start the operation of the RTC, the following operations must be performed:

- The 32kHz crystal oscillator (which provides the time base for the RTC module) must be enabled (PWR_REG_0[12:11]).
- *OPTIONAL*: For the CPU to be interrupted by RTC flags, the clock for the RTC synchronizers (which translate signals between the 4kHz and high-frequency clock domains) must be started. This step is not necessary for RTC timer functionality or for the device to wake up from LP0:STOP or LP1:STANDBY.
- The RTC prescaler must be configured to determine the time period represented by the LSB of the RTC timer.
- *OPTIONAL*: If the RTC digital trim function will be used, the prescaler must be within 0x3 and 0xC.
- The RTC can now be enabled by setting the enable bit to 1.

Set the RTC timer to a zero starting value

- Write RTCTMR_TIMER to 00000000h to zero the RTC timer count value.
- The pending status bit will change to 1 to indicate that a clock domain synchronization is pending as indicated by the rtc_enable_active bit. The pending bit will change back to 0 once the synchronization has completed.

Note It is not necessary to wait for the pending bit to go low before writing to other RTC registers. This pending bit is most useful to signal when it is permitted to go to LP0:STOP or LP1:STANDBY.

If the RTC Time of Day Alarm will be used, set the appropriate compare register(s)

- Write RTCTMR_COMP0 and/or RTCTMR_COMP1 to the desired compare value(s).
- The pending status bit will change to 1 to indicate that clock domain synchronization is pending; the bit will change back to 0 once the synchronization has completed.

Set the prescaler to the desired setting

- Write the prescale field to the desired clock rate prescale setting.

- The pending status bit will change to 1 to indicate that clock domain synchronization is pending; the bit will change back to 0 once the synchronization has completed.

Determining the Proper RTC Timer Prescale Value

The RTC timer register is always a fixed 32 bits in length; the input to the RTC block is always a fixed 4kHz frequency derived from the output of the 32kHz external crystal oscillator. However, by using the prescaler, it is possible to change the rate at which the RTC timer increments, effectively determining the unit of time assigned to the RTC timer LSB.

As shown in the table below, the assigned prescale value determines the number of 4kHz clock ticks that are needed in order to increment the main RTC timer register by one.

PRESCALE	Prescale Reload	4kHz ticks in LSB	Min Timer Value (s)	Max Timer Value (s)	Max Timer Value in Days	Max Timer Value in Years
0h	000h	1	0.00024	1048576	12	0.0
1h	001h	2	0.00049	2097152	24	0.1
2h	003h	4	0.00098	4194304	49	0.1
3h	007h	8	0.00195	8388608	97	0.3
4h	00Fh	16	0.00391	16777216	194	0.5
5h	01Fh	32	0.00781	33554432	388	1.1
6h	03Fh	64	0.01563	67108864	777	2.2
7h	07Fh	128	0.03125	134217728	1553	4.4
8h	0FFh	256	0.06250	268435456	3107	8.7
9h	1FFh	512	0.12500	536870912	6214	17.5
Ah	3FFh	1024	0.25000	1073741824	12428	34.9
Bh	7FFh	2048	0.50000	2147483648	24855	69.8
Ch	FFFh	4096	1.00000	4294967296	49710	139.6

For each prescale setting, the Prescale Reload indicates the reload value that will be loaded to the RTC's internal 12-bit prescaler counter when the RTC timer is

first started or when the prescaler counter reaches zero. On each 4kHz tick, the prescaler counter will either be reset with the reload value (if the prescaler counter equals zero) or it will be decremented by one (if the prescaler counter is nonzero). Each time the prescaler counter 'rolls under' at zero and is reloaded, the main RTC timer register RTCTMR_TIMER will be incremented by one.

The (PRESCALE = 0h) setting represents the minimum prescale reload value, which means that the RTC timer will be incremented each 4kHz clock period. This is also reflected in the value in the '4kHz ticks in LSB' column above. Setting (PRESCALE = 1h) will then give the RTC timer a resolution of (2/4kHz) and a tick frequency of 2048Hz, etc.

Setting (PRESCALE = 0h) gives the RTC timer its fastest possible frequency (4096Hz), causing it to reach its maximum value of FFFFFFFFh and roll over in 1048576 seconds (2^{32} / 4096), or approximately 12 days. Setting the maximum prescale value of (PRESCALE = Ch) results in the slowest possible RTC timer frequency of 1Hz, causing it to reach its maximum value of FFFFFFFFh and roll over in 2^{32} seconds, or about 139.6 years.

The prescaler range is intentionally wide to allow for a variety of potential applications. If the intent of the RTC for a particular application is to measure time across a long device lifetime, then a long period makes the most sense. If the RTC will instead be used to measure events and time intervals of shorter duration, then a shorter period (and a smaller prescaler value) will reduce the maximum time value the RTC can contain but will increase the effective resolution of the RTC timer to allow shorter intervals of time to be measured (or to allow RTC timer-based wakeup operations to occur more precisely).

Starting the RTC Timer

Once the RTC timer has been properly configured, it can be started by setting the enable bit to 1. This will cause the RTC timer to begin counting based on the selected prescaler rate.

11.5 Registers (RTCTMR)

Address	Register	Access	Description	Reset By
0x40000A00	RTCTMR_CTRL	***	RTC Timer Control	POR
0x40000A04	RTCTMR_TIMER	R/W	RTC Timer Count Value	Sys; POR
0x40000A08	RTCTMR_COMP0	R/W	RTC Time of Day Alarm 0 Compare Register	Sys; POR
0x40000A0C	RTCTMR_COMP1	R/W	RTC Time of Day Alarm 1 Compare Register	Sys; POR
0x40000A10	RTCTMR_FLAGS	***	CPU Interrupt and RTC Domain Flags	Sys POR
0x40000A14	RTCTMR_SNZ_VAL	R/W	RTC Timer Alarm Snooze Value	Sys
0x40000A18	RTCTMR_INTEN	R/W	Interrupt Enable Controls	Sys
0x40000A1C	RTCTMR_PRESCALE	R/W	RTC Timer Prescale Setting	Sys
0x40000A24	RTCTMR_PRESCALE_MASK	R/W	RTC Timer Prescale Compare Mask	Sys
0x40000A28	RTCTMR_TRIM_CTRL	***	RTC Timer Trim Controls	Sys
0x40000A2C	RTCTMR_TRIM_VALUE	R/W	RTC Timer Trim Adjustment Interval	Sys

11.5.1 RTCTMR_CTRL

RTCTMR_CTRL.enable

Field	Bits	Sys Reset	Alt Reset	Access	Description
enable	0	X	VRTC_POR:0	R/W	RTC Timer Enable

- 0: RTC Timer increment is disabled
- 1: RTC Timer increments normally (running)

RTCTMR_CTRL.clear

Field	Bits	Sys Reset	Access	Description
clear	1	n/a	W/O	RTC Timer Clear Bit

Write to 1 to clear the RTC timer.

Always reads 0.

RTCTMR_CTRL.pending

Field	Bits	Sys Reset	Access	Description
pending	2	n/a	R/O	RTC Transaction Pending

- 0: No transactions are pending
- 1: One or more RTC transactions are pending as indicated by the associated _active flags in this register.

RTCTMR_CTRL.use_async_flags

Field	Bits	Sys Reset	Alt Reset	Access	Description
use_async_flags	3	X	VRTC_POR:0	R/W	Use Async RTC Flags

- 0: (default) Use flags synchronized to the core clock. This is redundant because they will get synchronized again by the digital core.
- 1: Connect flags synchronized to the RTC clock directly to the core, since they will be re-synchronized by the core. (This should have been the default setting but currently is not.)

RTCTMR_CTRL.aggressive_rst

Field	Bits	Sys Reset	Alt Reset	Access	Description
aggressive_rst	4	X	VRTC_POR:0	R/W	Use Aggressive Reset Mode

- 0: (default) When resetting all of the RTC flags using the async_clear_flags bit, the reset to the flags will be released on the next edge of the 4kHz RTC clock. In other words, the system clock is used to assert the reset to the flags but the release of the reset is synchronized to the 4kHz RTC clock.
- 1: When resetting all of the RTC flags using async_clear_flags, the reset to the flags will be released without regards to the 4KHz RTC clock. In other words, the system clock is used to issue a completely unsynchronized reset pulse to the flags.

The purpose of this bit, aggressive_rst, is to allow a user to have control of the release of the reset to the flags when using async_clear_flags. This is to accommodate two schools of thought, 1) never release reset on a clock edge and 2) don't care. User 1 may feel that the release of the reset at the same time that a clock edge occurs could cause an unstable condition and thus would not want to set this bit. User 2, who does not feel that this situation would ever be an issue and needs the reset to be released quickly so that the flags will be operational for the next 4Khz RTC clock edge, would want to set this bit.

RTCTMR_CTRL.auto_update_disable

Field	Bits	Sys Reset	Alt Reset	Access	Description
auto_update_disable	5	X	VRTC_POR:0	R/W	Auto Update Disable

When hardware detects a difference between data in the 4Khz domain and the system clock domain an asynchronous request is made to the core to automatically synchronize RTC data between the two clock domains.

This bit can be used to prevent this asynchronous request. This will save some power.

- 0: Normal operation. RTC_TIMER and RTC_FLAGS[4:0] are automatically updated.
- 1: Disable auto updates - RTC_TIMER and RTC_FLAGS[4:0] are NOT automatically updated.

Setting this bit to 1 requires the user to ensure that RTC_TIMER has been updated before using its contents.

RTCTMR_CTRL.snooze_enable

Field	Bits	Sys Reset	Alt Reset	Access	Description
snooze_enable	6	X	VRTC_POR:0	R/W	Snooze Enable

- 0: Snooze feature disabled.
- 1: Snooze feature enabled.

RTCTMR_CTRL.rtc_enable_active

Field	Bits	Sys Reset	Access	Description
rtc_enable_active	16	n/a	R/O	tmr_en_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.osc_goto_low_active

Field	Bits	Sys Reset	Access	Description
osc_goto_low_active	17	n/a	R/O	osc_goto_low_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.osc_frce_sm_en_active

Field	Bits	Sys Reset	Access	Description
osc_frce_sm_en_active	18	n/a	R/O	osc_frce_sm_en_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.osc_frce_st_active

Field	Bits	Sys Reset	Access	Description
osc_frce_st_active	19	n/a	R/O	osc_frce_st_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.rtc_set_active

Field	Bits	Sys Reset	Access	Description
rtc_set_active	20	n/a	R/O	timer_set_active

This sync transaction with the 4kHz clock domain occurs when a new timer value is written to the RTCTMR_TIMER register.

Reads 1 when the associated transaction is pending.

RTCTMR_CTRL.rtc_clr_active

Field	Bits	Sys Reset	Access	Description
rtc_clr_active	21	n/a	R/O	timer_clr_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.rollover_clr_active

Field	Bits	Sys Reset	Access	Description
rollover_clr_active	22	n/a	R/O	rollover_clr_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.prescale_cmpr0_active

Field	Bits	Sys Reset	Access	Description
prescale_cmpr0_active	23	n/a	R/O	prescale_cmpr0_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.prescale_update_active

Field	Bits	Sys Reset	Access	Description
prescale_update_active	24	n/a	R/O	prescale_update_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.cmpr1_clr_active

Field	Bits	Sys Reset	Access	Description
cmpr1_clr_active	25	n/a	R/O	cmpr1_clr_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.cmpr0_clr_active

Field	Bits	Sys Reset	Access	Description
cmpr0_clr_active	26	n/a	R/O	cmpr0_clr_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.trim_enable_active

Field	Bits	Sys Reset	Access	Description
trim_enable_active	27	n/a	R/O	trim_enable_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.trim_slower_active

Field	Bits	Sys Reset	Access	Description
trim_slower_active	28	n/a	R/O	trim_slower_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.trim_clr_active

Field	Bits	Sys Reset	Access	Description
trim_clr_active	29	n/a	R/O	trim_clr_active

Reads 1 when the associated transaction is pending

RTCTMR_CTRL.active_trans_0

Field	Bits	Sys Reset	Access	Description
active_trans_0	30	n/a	R/O	active_trans_0

Reads 1 when the associated transaction is pending

11.5.2 RTCTMR_TIMER

RTCTMR_TIMER

Sys Reset	Alt Reset	Access	Description
XXXX_XXXXh	VRTC_POR:0000_0000h	R/W	RTC Timer Count Value

This timer value is synchronized with TIMER_A, which is the RTC timer counter in the 4kHZ clock domain.

11.5.3 RTCTMR_COMP0

RTCTMR_COMP0

Sys Reset	Alt Reset	Access	Description
XXXX_XXXXh	VRTC_POR:FFFF_FFFFh	R/W	RTC Time of Day Alarm 0 Compare Register

This value is compared against the RTC timer value in the 4kHz clock domain. When a match occurs, the comp0_flag_a is set.

11.5.4 RTCTMR_COMP1

RTCTMR_COMP1

Sys Reset	Alt Reset	Access	Description
XXXX_XXXXh	VRTC_POR:FFFF_FFFFh	R/W	RTC Time of Day Alarm 1 Compare Register

This value is compared against the RTC timer value in the 4kHz clock domain. When a match occurs, the comp1_flag_a is set.

11.5.5 RTCTMR_FLAGS

RTCTMR_FLAGS.comp0

Field	Bits	Sys Reset	Alt Reset	Access	Description
comp0	0	X	VRTC_POR:0	W1C	RTC Compare 0 Interrupt Flag

When the 4kHz domain version of this flag (comp0_flag_a) transitions from 0 to 1, this flag will be automatically set to 1.

Write 1 to clear this flag.

RTCTMR_FLAGS.comp1

Field	Bits	Sys Reset	Alt Reset	Access	Description
comp1	1	X	VRTC_POR:0	W1C	RTC Compare 1 Interrupt Flag

When the 4kHz domain version of this flag (comp1_flag_a) transitions from 0 to 1, this flag will be automatically set to 1.

Write 1 to clear this flag. This flag will also be cleared automatically when the snooze function is triggered (by writing a 1 to RTCTMR_FLAGS.snooze when RTCTMR_CTRL.snooze_enable == 1).

RTCTMR_FLAGS.prescale_comp

Field	Bits	Sys Reset	Alt Reset	Access	Description
prescale_comp	2	X	VRTC_POR:0	W1C	RTC Prescale Compare Int Flag

Write 1 to clear.

Set to 1 by hardware when a match occurs between the prescale counter and zero (bits compared determined by the prescale compare mask register).

RTCTMR_FLAGS.overflow

Field	Bits	Sys Reset	Alt Reset	Access	Description
overflow	3	X	VRTC_POR:0	W1C	RTC Overflow Interrupt Flag

Write 1 to clear.

Set to 1 by hardware when the RTC Timer overflows.

RTCTMR_FLAGS.trim

Field	Bits	Sys Reset	Alt Reset	Access	Description
trim	4	X	VRTC_POR:0	W1C	RTC Trim Interrupt Flag

Write 1 to clear.

Set to 1 by hardware when a trim adjustment event occurs.

RTCTMR_FLAGS.snooze

Field	Bits	Sys Reset	Access	Description
snooze	5	n/a	W/O	Activate RTC Snooze Alarm

If the snooze function has been enabled (by setting RTCTMR_CTRL.snooze_enable to 1) for RTC timer count comparator 1, then writing a 1 to this bit will trigger the following sequence of events:

1. RTCTMR_COMP1 is set to: (RTCTMR_COMP1 + RTCTMR_SNZ_VAL.value)
2. RTCTMR_FLAGS.comp1 is automatically cleared to zero, triggering a one-shot operation which also clears the same flag in the 4kHz domain.

This bit always reads 0.

RTCTMR_FLAGS.comp0_flag_a

Field	Bits	Sys Reset	Alt Reset	Access	Description
comp0_flag_a	8	X	VRTC_POR:0	R/O	RTC Compare 0 4kHz Flag

This is the asynchronous (to the high-speed RTC clock domain) version of this comparison flag. It is set to 1 when the 4kHz version of the RTC timer count matches the RTC timer count comparator 0 value, which is set in RTCTMR_COMP0.

When firmware clears the RTC Compare 0 Interrupt Flag (by writing a 1 to it), this will automatically trigger a one-shot operation which will cause this flag to be cleared in the 4kHz RTC clock domain on the next falling edge of the 4kHz RTC clock.

RTCTMR_FLAGS.comp1_flag_a

Field	Bits	Sys Reset	Alt Reset	Access	Description
comp1_flag_a	9	X	VRTC_POR:0	R/O	RTC Compare 1 4kHz Flag

This is the asynchronous (to the high-speed RTC clock domain) version of this comparison flag. It is set to 1 when the 4kHz version of the RTC timer count matches the RTC timer count comparator 1 value, which is set in RTCTMR_COMP1.

When firmware clears the RTC Compare 1 Interrupt Flag (by writing a 1 to it), this will automatically trigger a one-shot operation which will cause this flag to be cleared in the 4kHz RTC clock domain on the next falling edge of the 4kHz RTC clock.

RTCTMR_FLAGS.prescl_flag_a

Field	Bits	Sys Reset	Alt Reset	Access	Description
prescl_flag_a	10	X	VRTC_POR:0	R/O	RTC Prescale Compare 4kHz Flag

Original event detection flag from 4kHz domain.

RTCTMR_FLAGS.overflow_flag_a

Field	Bits	Sys Reset	Alt Reset	Access	Description
overflow_flag_a	11	X	VRTC_POR:0	R/O	RTC Overflow 4kHz Flag

Original event detection flag from 4kHz domain.

RTCTMR_FLAGS.trim_flag_a

Field	Bits	Sys Reset	Alt Reset	Access	Description
trim_flag_a	12	X	VRTC_POR:0	R/O	RTC Trim Event 4kHz Flag

Original event detection flag from 4kHz domain.

RTCTMR_FLAGS.async_clr_flags

Field	Bits	Sys Reset	Access	Description
async_clr_flags	31	0	W/O	Asynchronous RTC Flag Clear

Writing a 1 to this bit triggers a one-shot operation which clears the RTC Compare 0/1 flags and the RTC Prescale Compare flag both in this register and in the 4kHz clock domain.

This bit always reads 0.

11.5.6 RTCTMR_SNZ_VAL

RTCTMR_SNZ_VAL.value

Field	Bits	Sys Reset	Access	Description
value	9:0	0 (RTC POR only)	R/W	Snooze Value

11.5.7 RTCTMR_INTEN

RTCTMR_INTEN.comp0

Field	Bits	Sys Reset	Access	Description
comp0	0	0	R/W	RTC Time of Day Alarm (Compare 0) Interrupt Enable

- 0: Disabled.
- 1: The RTC Time of Day Alarm 0 Interrupt is enabled.

RTCTMR_INTEN.comp1

Field	Bits	Sys Reset	Access	Description
comp1	1	0	R/W	RTC Time of Day Alarm (Compare 1) Interrupt Enable

- 0: Disabled.
- 1: The RTC Time of Day Alarm 1 Interrupt is enabled.

RTCTMR_INTEN.prescale_comp

Field	Bits	Sys Reset	Access	Description
prescale_comp	2	0	R/W	RTC Prescale Compare Int Enable

- 0: Disabled.
- 1: The RTC Prescale Compare (Interval) Interrupt is enabled.

RTCTMR_INTEN.overflow

Field	Bits	Sys Reset	Access	Description
overflow	3	0	R/W	RTC Overflow Interrupt Enable

- 0: Disabled.
- 1: The RTC Overflow Interrupt is enabled.

RTCTMR_INTEN.trim

Field	Bits	Sys Reset	Access	Description
trim	4	0	R/W	RTC Trim Adjust Event Interrupt Enable

- 0: Disabled.
- 1: The RTC Trim Adjust Event Interrupt is enabled.

11.5.8 RTCTMR_PRESCALE

RTCTMR_PRESCALE.prescale

Field	Bits	Sys Reset	Access	Description
prescale	3:0	0000b (RTC POR only)	R/W	RTC Timer Prescale Setting

This field selects the initial value that is reloaded into the RTC timer prescale counter each time the prescale counter reaches 0 (or when the RTC timer is first initialized). The larger the initial prescale value, the slower the overall RTC timer will run.

- 0000b: 000h (RTC timer runs at 4kHz)

- 0001b: 001h (2kHz)
- 0010b: 003h (1kHz)
- 0011b: 007h (512Hz)
- 0100b: 00Fh (256Hz)
- 0101b: 01Fh (128Hz)
- 0110b: 03Fh (64Hz)
- 0111b: 07Fh (32Hz)
- 1000b: 0FFh (16Hz)
- 1001b: 1FFh (8Hz)
- 1010b: 3FFh (4Hz)
- 1011b: 7FFh (2Hz)
- 1100b: FFFh (1Hz)

11.5.9 RTCTMR_PRESCALE_MASK

RTCTMR_PRESCALE_MASK.prescale_mask

Field	Bits	Sys Reset	Access	Description
prescale_mask	3:0	0000b (RTC POR only)	R/W	RTC Timer Prescale Compare Mask

This mask field determines which bits of the prescale counter will be checked against a zero value; 1 bits cause a check to occur.

- 0000b: 0000h - mask setting
- 0001b: 0001h
- 0010b: 0003h
- 0011b: 0007h
- 0100b: 000Fh
- 0101b: 001Fh
- 0110b: 003Fh
- 0111b: 007Fh
- 1000b: 00FFh
- 1001b: 01FFh
- 1010b: 03FFh
- 1011b: 07FFh
- 1100b: 0FFFh

11.5.10 RTCTMR_TRIM_CTRL

RTCTMR_TRIM_CTRL.trim_enable_r

Field	Bits	Sys Reset	Access	Description
trim_enable_r	0	0 (RTC POR only)	R/W	Enable RTL Trim of RTC Timer

- 0: Trim disabled
- 1: Trim enabled

RTCTMR_TRIM_CTRL.trim_faster_ovr_r

Field	Bits	Sys Reset	Access	Description
trim_faster_ovr_r	1	0 (RTC POR only)	R/W	Force RTC Trim to Faster

- 0: Disabled; trim direction controlled by high bit of trim value.
- 1: Force trim to always occur in faster direction.

RTCTMR_TRIM_CTRL.trim_slower_r

Field	Bits	Sys Reset	Access	Description
trim_slower_r	2	0 (RTC POR only)	R/O	RTC Trim Direction Status

Indicates current mode of RTC trim adjustments.

- 0: Trim adjustments will cause RTC to count faster.
- 1: Trim adjustments will cause RTC to count slower.

11.5.11 RTCTMR_TRIM_VALUE

RTCTMR_TRIM_VALUE.trim_value

Field	Bits	Sys Reset	Access	Description
trim_value	17:0	0x3D0	R/W	Trim PPM Value

Only bits 17:8 are writeable; bits 7:0 always read 0.

RTC_TRIM_VALUE[17:0] = 1,000,000/PPM (PPM is the target correction). Minimum PPM adjustment is 4; maximum PPM adjustment is 125.

RTCTMR_TRIM_VALUE.trim_slower_control

Field	Bits	Sys Reset	Access	Description
trim_slower_control	18	0 (RTC POR only)	R/W	Trim Direction

If TRIM_CTRL.trim_faster_ovr_r == 0:

- 0: Trim faster - trim adjustments cause RTC to increment faster
- 1: Trim slower - trim adjustments cause RTC to increment slower If TRIM_CTRL.trim_faster_ovr_r == 1, then this bit has no effect and trim adjustments will always cause the RTC to increment faster.

11.6 Registers (RTCCFG)

Address	Register	Access	Description	Reset By
0x40000A70	RTCCFG_NANO_CNTR	R/O	Nano Oscillator Counter Read Register	Sys
0x40000A74	RTCCFG_CLK_CTRL	R/W	RTC Clock Control Settings	Sys
0x40000A7C	RTCCFG_OSC_CTRL	***	RTC Oscillator Control	Sys

11.6.1 RTCCFG_NANO_CNTR

RTCCFG_NANO_CNTR.nanoring_counter

Field	Bits	Sys Reset	Access	Description
nanoring_counter	15:0	s	R/O	Nano Oscillator Counter

Returns a value (asynchronous read) of a counter clocked by the nano oscillator output.

11.6.2 RTCCFG_CLK_CTRL

RTCCFG_CLK_CTRL.osc1_en

Field	Bits	Sys Reset	Access	Description
osc1_en	0		R/W	osc1_en

RTCCFG_CLK_CTRL.osc2_en

Field	Bits	Sys Reset	Access	Description
osc2_en	1		R/W	osc2_en

1:Enable clk_osc1_gated that goes to the dsensor. The source of this clock is clk_nano.

RTCCFG_CLK_CTRL.nano_en

Field	Bits	Sys Reset	Access	Description
nano_en	2		R/W	nano_en

11.6.3 RTCCFG_OSC_CTRL

RTCCFG_OSC_CTRL.osc_bypass

Field	Bits	Sys Reset	Access	Description
osc_bypass	0	0 (RTC POR)	R/W	osc_bypass

- 0: No effect (default)
- 1: The RTC oscillator is held in reset.

RTCCFG_OSC_CTRL.osc_disable_r

Field	Bits	Sys Reset	Access	Description
osc_disable_r	1	0 (RTC POR)	R/W	osc_disable_r

- 0: No effect (default)
- 1: If (osc_disable_sel == 1), the RTC oscillator is held in reset.

RTCCFG_OSC_CTRL.osc_disable_sel

Field	Bits	Sys Reset	Access	Description
osc_disable_sel	2	0 (RTC POR)	R/W	osc_disable_sel

- 0: The reset state of the RTC oscillator will be controlled by the power sequencer (default).
- 1: The reset state of the RTC oscillator will be controlled by the osc_disable_r bit.

RTCCFG_OSC_CTRL.osc_disable_o

Field	Bits	Sys Reset	Access	Description
osc_disable_o	3	s	R/O	osc_disable_o

Read-only indicator:

- 0: RTC oscillator is not held in reset.
- 1: RTC oscillator is held in reset.

12 Trust Protection Unit (TPU)

The TPU on the **MAX32620** provides support for cryptographic data security and other features useful in countering external attacks against the device. Although the AES engine is covered in a separate section, it is considered to be part of the TPU as well.

Features included in the TPU:

- AES cryptographic engine supporting symmetric encrypt/decrypt operations for 128-bit, 192-bit, and 256-bit key lengths.
- Dedicated 128-bit secure key storage memory that is automatically erased when a tamper response is generated.
- Modular arithmetic accelerator engine (MAA) supporting large-number calculations which can be used in cryptographic algorithms such as RSA. (**MAX32621** only)
- A pseudo-random seed generator, providing initial entropy data which can be used to seed a cryptographically-strong pseudo-random number generation algorithm. (**MAX32621** only)

For more details on the MAA and pseudo-random seed generator, refer to the **MAX32621** User Guide Supplement.

12.1 Registers (TPU_TSR)

Address	Register	Access	Description	Reset By
0x40007C10	TPU_TSR_SKS0	R/W	TPU Secure Key Storage Register 0 (Cleared on Tamper Detect)	Sys
0x40007C14	TPU_TSR_SKS1	R/W	TPU Secure Key Storage Register 1 (Cleared on Tamper Detect)	Sys
0x40007C18	TPU_TSR_SKS2	R/W	TPU Secure Key Storage Register 2 (Cleared on Tamper Detect)	Sys
0x40007C1C	TPU_TSR_SKS3	R/W	TPU Secure Key Storage Register 3 (Cleared on Tamper Detect)	Sys

12.1.1 TPU_TSR_SKS0

TPU_TSR_SKS0

Sys Reset	Access	Description
.	R/W	TPU Secure Key Storage Register 0 (Cleared on Tamper Detect)

12.1.2 TPU_TSR_SKS1

TPU_TSR_SKS1

Sys Reset	Access	Description
.	R/W	TPU Secure Key Storage Register 1 (Cleared on Tamper Detect)

12.1.3 TPU_TSR_SKS2

TPU_TSR_SKS2

Sys Reset	Access	Description
.	R/W	TPU Secure Key Storage Register 2 (Cleared on Tamper Detect)

12.1.4 TPU_TSR_SKS3

TPU_TSR_SKS3

Sys Reset	Access	Description
.	R/W	TPU Secure Key Storage Register 3 (Cleared on Tamper Detect)

12.2 Registers (AES)

Address	Register	Access	Description	Reset By
0x40007400	AES_CTRL	***	AES Control and Status	Sys
0x40007408	AES_ERASE_ALL	W/O	Write to Trigger AES Memory Erase	Sys
0x40102000	AES_MEM_INP0	R/W	AES Input 0 (least significant 32 bits)	Sys
0x40102004	AES_MEM_INP1	R/W	AES Input 1	Sys
0x40102008	AES_MEM_INP2	R/W	AES Input 2	Sys
0x4010200C	AES_MEM_INP3	R/W	AES Input 3 (most significant 32 bits)	Sys
0x40102010	AES_MEM_KEY0	R/W	AES Symmetric Key 0 (least significant 32 bits)	Sys
0x40102014	AES_MEM_KEY1	R/W	AES Symmetric Key 1	Sys
0x40102018	AES_MEM_KEY2	R/W	AES Symmetric Key 2	Sys
0x4010201C	AES_MEM_KEY3	R/W	AES Symmetric Key 3	Sys
0x40102020	AES_MEM_KEY4	R/W	AES Symmetric Key 4	Sys
0x40102024	AES_MEM_KEY5	R/W	AES Symmetric Key 5	Sys
0x40102028	AES_MEM_KEY6	R/W	AES Symmetric Key 6	Sys
0x4010202C	AES_MEM_KEY7	R/W	AES Symmetric Key 7 (most significant 32 bits)	Sys
0x40102030	AES_MEM_OUT0	R/W	AES Output 0 (least significant 32 bits)	Sys
0x40102034	AES_MEM_OUT1	R/W	AES Output 1	Sys
0x40102038	AES_MEM_OUT2	R/W	AES Output 2	Sys
0x4010203C	AES_MEM_OUT3	R/W	AES Output 3 (most significant 32 bits)	Sys
0x40102040	AES_MEM_EXPKEY0	R/W	AES Expanded Key Data 0	Sys
0x40102044	AES_MEM_EXPKEY1	R/W	AES Expanded Key Data 1	Sys
0x40102048	AES_MEM_EXPKEY2	R/W	AES Expanded Key Data 2	Sys
0x4010204C	AES_MEM_EXPKEY3	R/W	AES Expanded Key Data 3	Sys

Address	Register	Access	Description	Reset By
0x40102050	AES_MEM_EXPKEY4	R/W	AES Expanded Key Data 4	Sys
0x40102054	AES_MEM_EXPKEY5	R/W	AES Expanded Key Data 5	Sys
0x40102058	AES_MEM_EXPKEY6	R/W	AES Expanded Key Data 6	Sys
0x4010205C	AES_MEM_EXPKEY7	R/W	AES Expanded Key Data 7	Sys

12.2.1 AES_CTRL

AES_CTRL.start

Field	Bits	Sys Reset	Access	Description
start	0	0	R/W	AES Start/Busy

Writing this bit to 1 initiates an AES operation.

This bit is cleared from 1 to 0 by hardware when an AES operation has completed.

AES_CTRL.crypt_mode

Field	Bits	Sys Reset	Access	Description
crypt_mode	1	0	R/W	AES Encrypt/Decrypt Mode

This field can only be written when the AES engine is idle (AES Busy == 0).

- 0: Perform AES encryption operation
- 1: Perform AES decryption operation

AES_CTRL.exp_key_mode

Field	Bits	Sys Reset	Access	Description
exp_key_mode	2	0	R/W	AES Expanded Key Mode

This field can only be written when the AES engine is idle (AES Busy == 0).

- 0: Calculate expanded key / round key as needed for this operation.
- 1: Use the expanded key / round key data that was calculated for the previous operation. Only valid until key or enc/dec mode changes.

AES_CTRL.key_size

Field	Bits	Sys Reset	Access	Description
key_size	4:3	00b	R/W	AES Key Size Select

This field can only be written when the AES engine is idle (AES Busy == 0). Size of key to use for AES operation.

- 0: 128-bit key size
- 1: 192-bit key size
- 2: 256-bit key size
- 3: Reserved

AES_CTRL.inten

Field	Bits	Sys Reset	Access	Description
inten	5	0	R/W	AES Interrupt Enable

- 0: Disable interrupts from the AES
- 1: Allow an interrupt to be triggered by bit 6.

AES_CTRL.intfl

Field	Bits	Sys Reset	Access	Description
intfl	6	0	W1C	AES Interrupt Flag

Set by hardware; write 1 to clear.

If AES Interrupt Enable is set, this bit will trigger an AES Interrupt when set to 1.

12.2.2 AES_ERASE_ALL

AES_ERASE_ALL

Sys Reset	Access	Description
0	W/O	Write to Trigger AES Memory Erase

A write to this location triggers an erase of all AES memory locations.

12.2.3 AES_MEM_INP0

AES_MEM_INP0

Sys Reset	Access	Description
0000_0000h	R/W	AES Input 0 (least significant 32 bits)

12.2.4 AES_MEM_INP1

AES_MEM_INP1

Sys Reset	Access	Description
0000_0000h	R/W	AES Input 1

12.2.5 AES_MEM_INP2

AES_MEM_INP2

Sys Reset	Access	Description
0000_0000h	R/W	AES Input 2

12.2.6 AES_MEM_INP3

AES_MEM_INP3

Sys Reset	Access	Description
0000_0000h	R/W	AES Input 3 (most significant 32 bits)

12.2.7 AES_MEM_KEY0

AES_MEM_KEY0

Sys Reset	Access	Description
0000_0000h	R/W	AES Symmetric Key 0 (least significant 32 bits)

For all key sizes, this doubleword contains the least significant 32 bits of the key value.

12.2.8 AES_MEM_KEY1

AES_MEM_KEY1

Sys Reset	Access	Description
0000_0000h	R/W	AES Symmetric Key 1

For all key sizes, this doubleword contains bits 63..32 of the key value.

12.2.9 AES_MEM_KEY2

AES_MEM_KEY2

Sys Reset	Access	Description
0000_0000h	R/W	AES Symmetric Key 2

For all key sizes, this doubleword contains bits 95..64 of the key value.

12.2.10 AES_MEM_KEY3

AES_MEM_KEY3

Sys Reset	Access	Description
0000_0000h	R/W	AES Symmetric Key 3

For all key sizes, this doubleword contains bits 127..96 of the key value. For the 128-bit key size, these are the 32 most significant bits of the key.

12.2.11 AES_MEM_KEY4

AES_MEM_KEY4

Sys Reset	Access	Description
0000_0000h	R/W	AES Symmetric Key 4

For the 192-bit and 256-bit key sizes, this doubleword contains bits 159..132 of the key value. For the 128-bit key size, this doubleword is not used.

12.2.12 AES_MEM_KEY5

AES_MEM_KEY5

Sys Reset	Access	Description
0000_0000h	R/W	AES Symmetric Key 5

For the 192-bit and 256-bit key sizes, this doubleword contains bits 191..160 of the key value. For the 192-bit key size, these are the 32 most significant bits of the key. For the 128-bit key size, this doubleword is not used.

12.2.13 AES_MEM_KEY6

AES_MEM_KEY6

Sys Reset	Access	Description
0000_0000h	R/W	AES Symmetric Key 6

For the 256-bit key size, this doubleword contains bits 223..192 of the key value. For the 128-bit key size and the 192-bit key size, this doubleword is not used.

12.2.14 AES_MEM_KEY7

AES_MEM_KEY7

Sys Reset	Access	Description
0000_0000h	R/W	AES Symmetric Key 7 (most significant 32 bits)

For the 256-bit key size, this doubleword contains bits 255..224 of the key value, which are the 32 most significant bits of the key. For the 128-bit key size and the 192-bit key size, this doubleword is not used.

12.2.15 AES_MEM_OUT0

AES_MEM_OUT0

Sys Reset	Access	Description
0000_0000h	R/W	AES Output 0 (least significant 32 bits)

12.2.16 AES_MEM_OUT1

AES_MEM_OUT1

Sys Reset	Access	Description
0000_0000h	R/W	AES Output 1

12.2.17 AES_MEM_OUT2

AES_MEM_OUT2

Sys Reset	Access	Description
0000_0000h	R/W	AES Output 2

12.2.18 AES_MEM_OUT3

AES_MEM_OUT3

Sys Reset	Access	Description
0000_0000h	R/W	AES Output 3 (most significant 32 bits)

12.2.19 AES_MEM_EXPKEY0

AES_MEM_EXPKEY0

Sys Reset	Access	Description
0000_0000h	R/W	AES Expanded Key Data 0

12.2.20 AES_MEM_EXPKEY1

AES_MEM_EXPKEY1

Sys Reset	Access	Description
0000_0000h	R/W	AES Expanded Key Data 1

12.2.21 AES_MEM_EXPKEY2

AES_MEM_EXPKEY2

Sys Reset	Access	Description
0000_0000h	R/W	AES Expanded Key Data 2

12.2.22 AES_MEM_EXPKEY3

AES_MEM_EXPKEY3

Sys Reset	Access	Description
0000_0000h	R/W	AES Expanded Key Data 3

12.2.23 AES_MEM_EXPKEY4

AES_MEM_EXPKEY4

Sys Reset	Access	Description
0000_0000h	R/W	AES Expanded Key Data 4

12.2.24 AES_MEM_EXPKEY5

AES_MEM_EXPKEY5

Sys Reset	Access	Description
0000_0000h	R/W	AES Expanded Key Data 5

12.2.25 AES_MEM_EXPKEY6

AES_MEM_EXPKEY6

Sys Reset	Access	Description
0000_0000h	R/W	AES Expanded Key Data 6

12.2.26 AES_MEM_EXPKEY7

AES_MEM_EXPKEY7

Sys Reset	Access	Description
0000_0000h	R/W	AES Expanded Key Data 7

13 CRC16 and CRC32 Hardware Accelerator

13.1 Overview

The Cyclic Redundancy Check (CRC) hardware peripheral, incorporated into the **MAX32620**, provides a fast method for calculating 16-bit and 32-bit CRC calculations on 32-bit data values. The CRC peripheral is accessible from the Cortex-M4 processor or via the Peripheral Management Unit (PMU). Both the CRC-16-CCITT and CRC-32 calculations complete in a single clock cycle and use the following polynomials in the calculations.

CRC-16-CCITT:

$$\text{CRC16} = X^{16} + X^{12} + X^5 + 1$$

CRC-32:

$$\text{CRC32} = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$$

13.2 CRC Clock Gating

The CRC hardware peripheral uses the System Core Clock to operate and all CRC calculations complete in a single system clock cycle. The peripheral clock and clock gate are enabled by default for the CRC peripheral. The register field CLKMAN_CLK_GATE_CTRL0.crc_clk_gater is used to control the peripheral clock and clock gating for the CRC module. The table below shows the modes supported for the CRC peripheral clock gating.

crc_clk_gater (xxb)	Clock State
00b	Peripheral clock disabled. Lowest power mode when peripheral not in use.
01b	*Default:* Peripheral clock enabled with Dynamic Clock Gating. Clock is active only when using the peripheral.
1xb	Peripheral clock is enabled and active at all times. Highest performance and highest current option.

13.3 CRC Operation

The CRC hardware peripheral calculates a 16-bit or 32-bit CRC value based on a 32-bit data value combined with an initial seed value (default is 0xFFFF for 16-bit CRC and 0xFFFFFFFF for 32-bit CRC calculations). For data values that are not 32-bits wide, the upper bytes must be padded with zeroes to the next 32-bit boundary. Data may be written to CRC_DATA_VALUE16 or CRC_DATA_VALUE32 in byte, word, or double word format by either the Cortex-M4 processor or PMU. A user-specified initial seed value can be used by writing to the appropriate register (CRC_SEED16 or CRC_SEED32) along with setting the appropriate reseed

enable bit (CRC_RESEED.crc16 or CRC_RESEED.crc32).

Note For write accesses less than a double word (32-bits) in length, the hardware only initiates a CRC calculation update when the MSB is written to. This means, if byte writes are done, only a write to B3 starts a CRC calculation on (B3, B2, B1, B0); if word writes are done, a CRC calculation begins with the write to W1 on (W1, W0).

13.4 CRC-16-CCITT Example Calculation

1. If an initial seed value other than 0xFFFF is desired, write the initial seed value to CRC_SEED16.
2. Write CRC_RESEED.crc16 to 1 to apply the new initial seed value.
3. Write the first data value to CRC_DATA_VALUE16. This data value may be 8-bits, 16-bits, or 32-bits in width. If the value is less than 32 bits wide, the upper bits must be set to zero.
4. Repeat step 3 until all data that needs to be included in the CRC calculation has been written.
5. Read from CRC_DATA_VALUE16 to obtain the final CRC-16 calculation result.

13.5 CRC-32 Example Calculation

1. If an initial seed value other than 0xFFFF_FFFF is desired, write the initial seed value to CRC_SEED32.
2. Write CRC_RESEED.crc32 to 1 to apply the new initial seed value.
3. Write the first data value to CRC_DATA_VALUE32. This data value may be 8-bits, 16-bits, or 32-bits in width. If the value is less than 32 bits wide, the upper bits must be set to zero.
4. Repeat step 3 until all data that needs to be included in the CRC calculation has been written.
5. Read from CRC_DATA_VALUE32 to obtain the final CRC-32 calculation result.

13.6 Registers (CRC)

Address	Register	Access	Description	Reset By
0x40006000	CRC_RESEED	W1C	CRC-16/CRC-32 Reseed Controls	Sys
0x40006004	CRC_SEED16	R/W	Reseed Value for CRC-16 Calculations	Sys
0x40006008	CRC_SEED32	R/W	Reseed Value for CRC-32 Calculations	Sys
0x40101000	CRC_DATA_VALUE16	R/W	Write Next CRC-16 Data Value / Read CRC-16 Result Value	Sys
0x40101800	CRC_DATA_VALUE32	R/W	Write Next CRC-32 Data Value / Read CRC-32 Result Value	Sys

13.6.1 CRC_RESEED

CRC_RESEED.crc16

Field	Bits	Sys Reset	Access	Description
crc16	0	0	W1C	Reseed CRC-16 Generator

Write to 1 to reseed the CRC-16 generator.

Cleared to 0 by hardware when reseed operation has completed.

CRC_RESEED.crc32

Field	Bits	Sys Reset	Access	Description
crc32	1	0	W1C	Reseed CRC-32 Generator

Write to 1 to reseed the CRC-32 generator.

Cleared to 0 by hardware when reseed operation has completed.

13.6.2 CRC_SEED16

CRC_SEED16

Sys Reset	Access	Description
00000000h	R/W	Reseed Value for CRC-16 Calculations

This register contains the value that will be used when a CRC16 reseed operation is triggered.

13.6.3 CRC_SEED32

CRC_SEED32

Sys Reset	Access	Description
00000000h	R/W	Reseed Value for CRC-32 Calculations

This register contains the value that will be used when a CRC32 reseed operation is triggered.

13.6.4 CRC_DATA_VALUE16

CRC_DATA_VALUE16

Sys Reset	Access	Description
00000000h	R/W	Write Next CRC-16 Data Value / Read CRC-16 Result Value

Writing to this register loads the next value into the CRC engine to be processed for the CRC-16 running calculation.

Reading from this register returns the current CRC-16 calculation result.

13.6.5 CRC_DATA_VALUE32

CRC_DATA_VALUE32

Sys Reset	Access	Description
00000000h	R/W	Write Next CRC-32 Data Value / Read CRC-32 Result Value

Writing to this register loads the next value into the CRC engine to be processed for the CRC-32 running calculation.

Reading from this register returns the current CRC-32 calculation result.

14 Trademarks and Service Marks

The following registered trademarks and registered service marks are referenced in the text of this document:

- 1-Wire and iButton are registered trademarks of Maxim Integrated Products, Inc.
- ARM is a registered trademark and registered service mark and Cortex is a registered trademark of ARM Limited.
- The Bluetooth word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by Maxim is under license.
- IEEE is a registered service mark of the Institute of Electrical and Electronics Engineers, Inc.