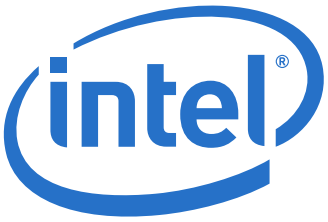
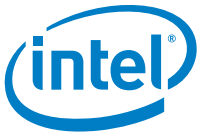


Intel® Quark™ microcontroller D1000

User Guide

April 2016

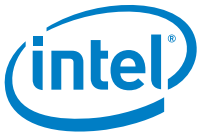
Document Number: 332912-004



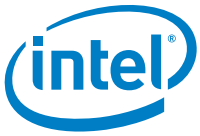
Revision History

Date	Revision	Description
November 2015	003	Revised Table 182. I^2C *Interrupt Mask Register
October 2015	002	Updated the following figures: • Figure 4, Chapter 3 • Figure 11, Chapter 6 • Figure 15, Chapter 9 • Figure 16, Chapter 9 • Figure 17, Chapter 9 • Figure 18, Chapter 9 • Figure 19, Chapter 9 • Figure 20, Chapter 9 • Figure 21, Chapter 9 • Figure 36, Chapter 9 • Figure 38, Chapter 9 • Figure 39, Chapter 9 • Figure 40, Chapter 9 • Figure 41, Chapter 9 • Figure 42, Chapter 9
September 2015	001	Initial release

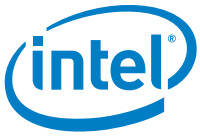
Contents

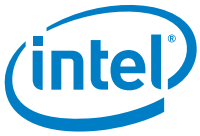


Figures



Tables

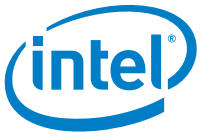




1.0 Introduction

The Intel® Quark™ microcontroller D1000 (also called MCU) is an extremely low power microcontroller at the top of its class in computational performance. It is optimized for long battery life applications such as wearable sensors and RFID tags. Its many features and benefits include:

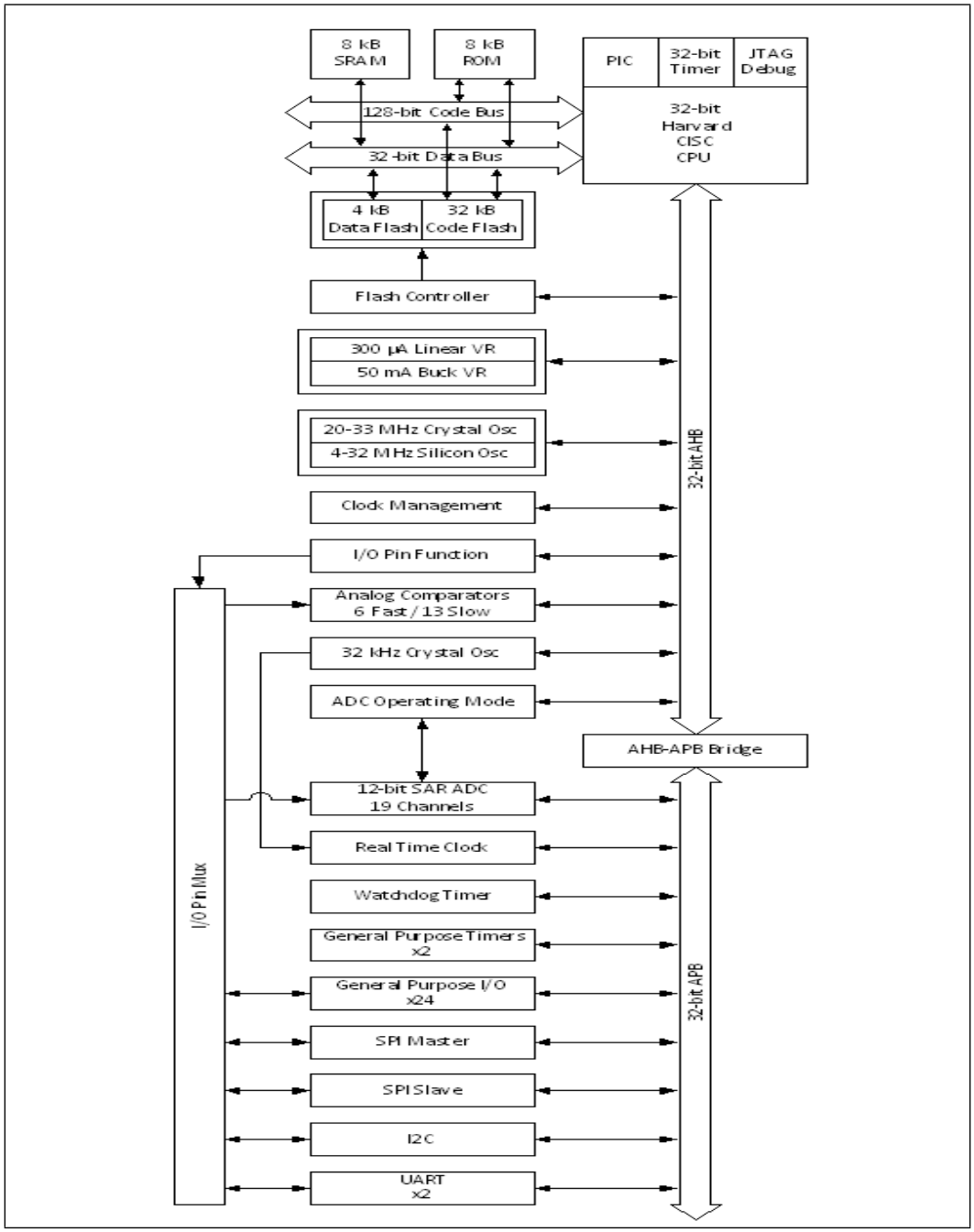
- Powerful 32-bit Harvard CISC CPU with single cycle barrel shifter, two cycle multiplier, multi-cycle divider, integrated 32-bit timer, programmable interrupt controller, and JTAG debugger.
- 128-bit wide 32 kB code flash and 8 kB ROM with built in helper functions for CRC, AES, flash programming, self-calibration, and power management.
- 32-bit wide 8 kB SRAM and 4 kB data flash for system configuration and general purpose user non-volatile memory.
- 300 µA low quiescent current linear VR for low power sleep regimes and 50 mA buck VR for active regimes and off chip devices.
- 20-33 MHz crystal oscillator for low jitter precision frequency and 4-32 MHz silicon oscillator for reduced power.
- 32 kHz low power crystal oscillator and real-time clock for precise time keeping and wake up even during deep sleep regimes.
- Programmable analog, serial interface or GPIO functions on 24 pins.
- 19-channel 12-bit 2.4 MSps SAR ADC with 32-entry arbitrary channel scan table.
- 6 high-speed and 13 low-power comparators for wake up, envelope detection, and demodulation functions.
- Versatile fine grained clock management including branch gating and automatic fast frequency changes.
- 32-bit watchdog timer with interrupt on first then reset on second expiration.
- Two 32-bit general purpose timers with independent clock frequencies.
- 24 general purpose I/O with edge detection and interrupt capability.
- Master and slave 16 Mbps serial peripheral interfaces supporting Motorola* SPI, Texas Instruments* SSP, and National Semiconductor* Microwire* formats, supporting up 4 slave devices from the master SPI port.
- I^2C master/slave interface supporting both 100 kbps standard and 400 kbps fast modes.
- Two UARTs with hardware handshaking.
- 24 ESD-protected versatile digital I/O buffers with high current drive and programmable direction, slew rate, and pull-up control.
- 19 ESD-protected analog inputs sharing the same pins as digital I/O for fast wake up from digital or analog input signals.
- Sleep regimes down to 1.5 µA.

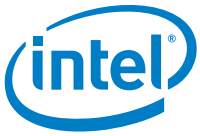


- Wake up in as little as 2.0 µs.
- Active regimes down to 320 µA.
- Operating power supply range 1.62-3.63 V.

See Figure 1 for details of the Intel® Quark™ microcontroller D1000 block diagram.

Figure 1. Intel® Quark™ microcontroller D1000 Block Diagram





1.1 Reference/Related Documents

Table 1. Reference Documents

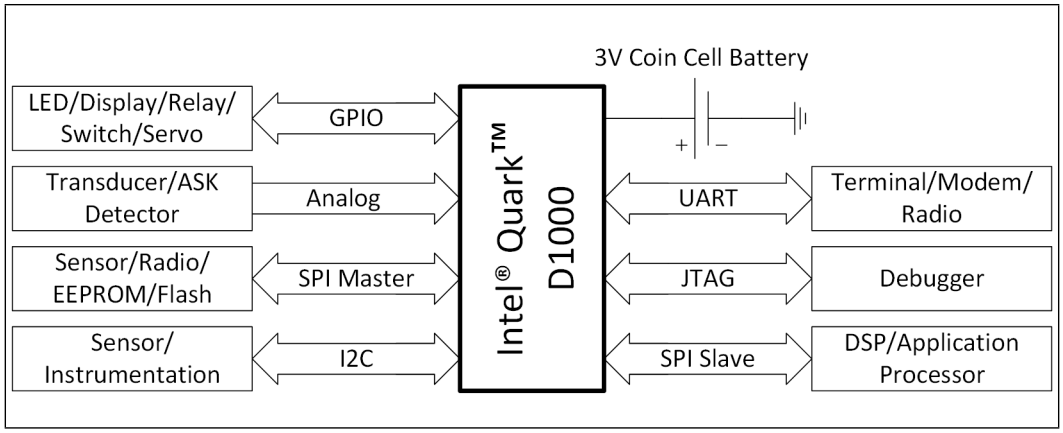
Document	Document Number
Intel® Quark™ microcontroller D1000 Datasheet	332910
Intel® Quark™ microcontroller D1000 Programmers Reference Manual	332913

§ §

2.0 004Applications and Usage

The Intel® Quark™ microcontroller D1000 supports a wide range of applications. Its comparators and ADC make it easy to connect to transducers and RF front ends. Its serial peripheral interfaces make it easy to connect with popular sensors, radios, memories, DSPs and application processors. Its high drive current and slew rate controlled general purpose digital I/O with integrated pull-ups permit direct connection to LEDs, relays, switches, H-bridges and servos. Its high performance CPU can support compute intensive security and signal processing tasks. It can operate from a single 3.3 V battery with a discharge curve from 3.6 V down to 2.0 V and has comprehensive power management for extended battery life. Optionally, the D1000's integrated voltage regulator can be disabled and it can operate from a single regulated 1.8 V ± 10% power supply. An example application demonstrating the use of the MCU's versatile interfaces is shown in Figure 2. Unlike many other microcontrollers in this class, the Intel® Quark™ microcontroller D1000 supports all features and operating frequencies over the full input voltage range.

Figure 2. An example application demonstrating the use of Intel® Quark™ microcontroller D1000's versatile interfaces.

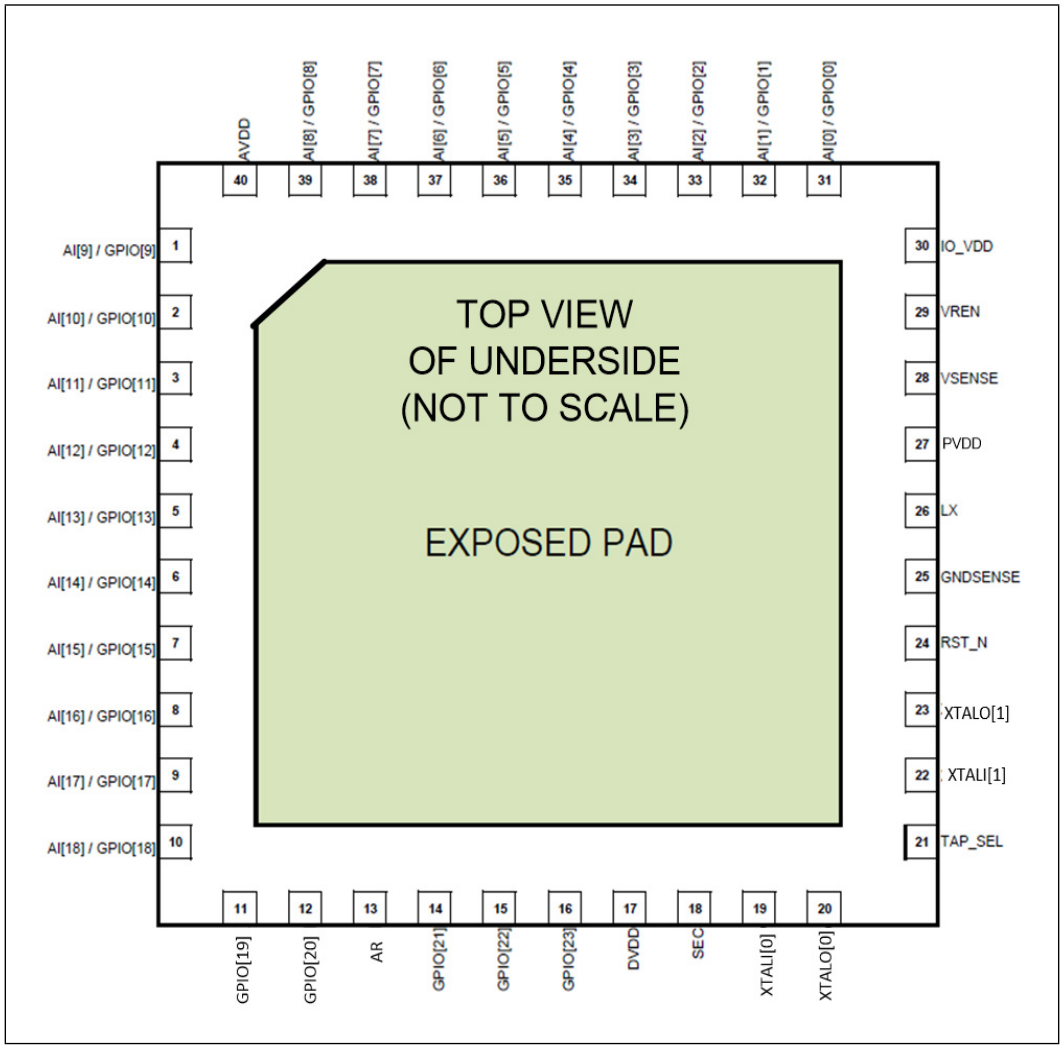


§ §

3.0 External Interfaces

The Intel® Quark™ microcontroller D1000 is packaged in a 6x6 mm 40-pin QFN. The package pin-out is shown in Figure 3.

Figure 3. Intel® Quark™ D1000 Package Pin-out



I/O pins are multiplexed to support a wide range of applications. The MCU has 24 general purpose digital I/O (GPIO) pins. Each pin is software configurable as analog input, digital input, push-pull output, open drain output, open source output, or bidirectional. When operating as digital inputs, they can be level sensitive or edge detecting with the ability to generate interrupts. Integrated pull-ups can be enabled on each pin and when operating as outputs, fast or slow slew rate can be selected.

Nineteen of the GPIO pins also have analog capabilities. They are connected to both a comparator and an ADC channel. Analog functions are available even when a pin is configured as digital. This is a very useful wake-up feature, which will be discussed in more detail in the sections that follow. When a pin is configured as purely analog, digital functions are disabled. Table 2 lists the functions available on each pin. Table 3 provides a detailed description of each pin function. All pins have built-in ESD protection.

Table 2. Intel® Quark™ D1000 Pin Function by Function Code (Sheet 1 of 2)

Package Pin	Pin Function Code			
	#0	#1	#2	#3
1	AI[9]	GPIO[9]	SLV_M2SD	AI[9]
2	AI[10]	GPIO[10]	SLV_S2MD	AI[10]
3	AI[11]	GPIO[11]	SLV_M2SS	AI[11]
4	AI[12]	GPIO[12]	TXD[1]	AI[12]
5	AI[13]	GPIO[13]	RXD[1]	AI[13]
6	AI[14]	GPIO[14]	RTS[1]	AI[14]
7	AI[15]	GPIO[15]	CTS[1]	AI[15]
8	AI[16]	GPIO[16]	MST_M2SC	AI[16]
9	AI[17]	GPIO[17]	MST_M2SD	AI[17]
10	AI[18]	GPIO[18]	MST_S2MD	AI[18]
11	-	GPIO[19]	TDO	-
12	TXD[0]	GPIO[20]	TRST_N	-
13	AR	AR	AR	AR
14	RXD[0]	GPIO[21]	TCK	-
15	RTS[0]	GPIO[22]	TMS	-
16	CTS[0]	GPIO[23]	TDI	-
17	DVDD	DVDD	DVDD	DVDD
18	SEC	SEC	SEC	SEC
19	XTALI[0]	XTALI[0]	XTALI[0]	XTALI[0]
20	XTALO[0]	XTALO[0]	XTALO[0]	XTALO[0]
21	TAP_SEL	TAP_SEL	TAP_SEL	TAP_SEL
22	XTALI[1]	XTALI[1]	XTALI[1]	XTALI[1]
23	XTALO[1]	XTALO[1]	XTALO[1]	XTALO[1]
24	RST_N	RST_N	RST_N	RST_N
25	GNDSENSE	GNDSENSE	GNDSENSE	GNDSENSE
26	LX	LX	LX	LX
27	PVDD	PVDD	PVDD	PVDD
28	VSENSE	VSENSE	VSENSE	VSENSE

Table 2. Intel® Quark™ D1000 Pin Function by Function Code (Sheet 2 of 2)

Package Pin	Pin Function Code			
	#0	#1	#2	#3
29	VREN	VREN	VREN	VREN
30	IO_VDD	IO_VDD	IO_VDD	IO_VDD
31	AI[0]	GPIO[0]	MST_M2SS[0]	AI[0]
32	AI[1]	GPIO[1]	MST_M2SS[1]	AI[1]
33	AI[2]	GPIO[2]	MST_M2SS[2]	AI[2]
34	AI[3]	GPIO[3]	MST_M2SS[3]	AI[3]
35	AI[4]	GPIO[4]	-	AI[4]
36	AI[5]	GPIO[5]	MST_S2M_SS	AI[5]
37	AI[6]	GPIO[6]	SCL	AI[6]
38	AI[7]	GPIO[7]	SDA	AI[7]
39	AI[8]	GPIO[8]	SLV_M2SC	AI[8]
40	AVDD	AVDD	AVDD	AVDD
Pad	VSS	VSS	VSS	VSS

Table 3. Detailed Pin Function Description (Sheet 1 of 2)

Interface	Pin Name	Package Pin	Type	Description
Power	VSS	Pad	Ground	QFN package ground plane
	DVDD	17	Supply	1.35-1.8 V regulated core power supply[1]
	PVDD	27	Supply	2.0-3.6 V unregulated VR power supply[2,3]
	AVDD	40	Supply	1.6-3.6 V analog power supply[3]
	IO_VDD	30	Supply	1.6-3.6 V I/O power supply[3]
	VSENSE	28	Analog input	Core power voltage sense
	GNDSENSE	25	Analog input	Core power ground sense
	LX	26	Supply	Core voltage regulator output[4]
	VREN	29	Analog input	Voltage regulator enable: PVDD = enable VSS = disable
Clocking	XTALI[0]	19	Logic input	System crystal or external oscillator input
	XTALO[0]	20	Logic output	System crystal
	XTALI[1]	22	Logic input	Real-time clock crystal or external oscillator input
	XTALO[1]	23	Logic output	Real-time clock crystal
Reset	RST_N	24	Analog input	Low true reset w/hysteresis. Tie to AVDD for internal power-on reset. <0.8V = reset >1.1V = not reset
	SEC	18	Logic input	Security

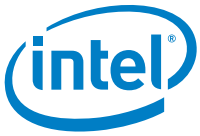


Table 3. Detailed Pin Function Description (Sheet 2 of 2)

Interface	Pin Name	Package Pin	Type	Description
GPIO	GPIO[23:0]	1 - 12, 14 - 16, 31 - 39	Logic I/O	General purpose I/O
I^2C	SCL	37	Logic I/O	Open drain clock
	SDA	38	Logic I/O	Open drain data
UART	TXD[1:0]	4	Logic output	Transmit data
	RXD[1:0]	5	Logic input	Receive data
	RTS[1:0]	6	Logic output	Request to send
	CTS[1:0]	7	Logic input	Clear to send
Slave SPI	SLV_M2SC	39	Logic input	Clock
	SLV_M2SD	1	Logic input	Receive data (MOSI)
	SLV_M2SS	3	Logic input	Slave select
	SLV_S2MD	2	Logic output	Transmit data (MISO)
Master SPI	MST_M2SC	8	Logic output	Clock
	MST_M2SD	9	Logic output	Transmit data (MOSI)
	MST_M2SS[3:0]	31 - 34	Logic output	Slave select (for up to 4 SPI device groups)
	MST_S2MD	10	Logic input	Receive data (MISO)
	MST_S2MSS	36	Logic input	Slave select from other master(s)
Analog	AI[18:0]	1 - 10, 31 - 39	Analog input	Comparator and ADC inputs
	AR	13	Analog input	Comparator reference
JTAG	TRST_N	12	Logic input	TAP controller reset
	TDI	16	Logic input	TAP data input
	TMS	15	Logic input	TAP mode select
	TCK	14	Logic input	TAP clock
	TDO	11	Logic output	TAP data output
	TAP_SEL	21	Logic input	TAP select 0 = Debug TAP

1. Digital I/O operation is degraded below 1.62 V and timing is not guaranteed.
2. Voltage range of PVDD is 1.62-3.63 V when the integrated VR is disabled.
3. AVDD, IOVDD and PVDD may be AC isolated, but must be supplied from a common power source.
4. A 47 µH and 4.7 µF external LC filter is required between LX and DVDD when the integrated VR is enabled.

Nineteen of the GPIO pins have a choice of two digital functions: (1) GPIO and (2) serial peripheral interface (Pin #35 has no serial function). The GPIO function provides the CPU direct access to the pin via memory mapped registers. The serial peripheral interface functions include I^2C, UART, Master SPI and Slave SPI. All five serial peripheral interfaces can be used simultaneously.

Five of the GPIO pins have no analog capabilities, but offer a standard 5-pin JTAG interface for software debugging. JTAG pins are disabled by default. However, they will be enabled if flash is found to be erased or corrupted, if the system configuration stored in flash enables them, or if the user application enables them. The ability to disable JTAG prevents exposure of any secrets the user may store in flash. The SEC pin can be used to securely force enablement of JTAG if the application developer is accidentally locked out or if returned units need failure analysis or reprogramming.

The bootstrap procedure checks the SEC pin and erases flash (if configured to do so), then enables JTAG if a floating or logic high condition is found. SEC, along with a comparator, can be used for tamper detection as well.

After power-on or reset, pin functions default to code #3, where all GPIO pins are in a high impedance state and only analog functions are enabled. Software can select GPIO or serial communication pin functions by changing a single bit of the function code to zero. This avoids glitches that could result if both bits were to change at once.

Comparators have rail-to-rail common mode range. By default, an internal 0.95 V reference is selected as the threshold voltage. This is a good value for detecting logic transitions on digital signals. Optionally, the AR pin can be used to provide one or more analog comparators with a threshold voltage reference of the user's choosing. This could be any voltage from 0 to the analog supply voltage (AVDD).

The XTALI and XTALO pins can be used to connect crystals to the system and real time clock oscillators. Alternately, external clocks can be connected to the XTALI inputs. See the Intel® Quark™ D1000 microcontroller Datasheet for recommended voltage levels on these pins.

The TAP_SEL pin selects between the boundary scan and debug TAP controllers. This pin should be pulled low for normal operation.

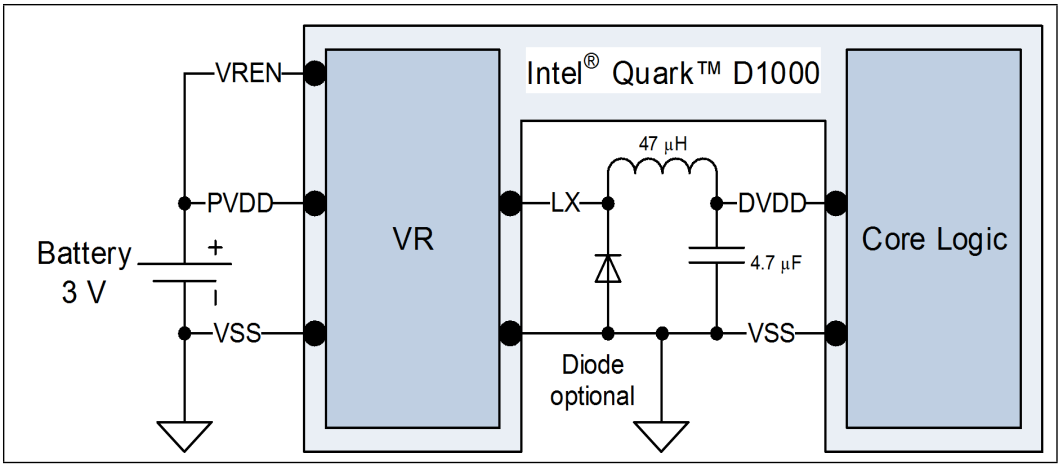
The RST_N pin can be connected to an external reset source such as a push button switch or brownout detector. A detailed discussion of the reset procedure is given in a later section.

AVDD, PVDD and IO_VDD pins must be connected to a common power source such as a 3V battery. However, in order to prevent noise coupling from the I/O (IO_VDD) and voltage regulator (PVDD) supplies to the analog (AVDD) supply, ferrite beads or other high frequency blocking filters should be used to AC isolate these pins.

If the internal voltage regulator (VR) is to be used, VREN, LX, DVDD and VSS must be connected as shown in Figure 4. GNDSENSE and VSENSE pins (not shown) must be star connected to the 4.7 µF DVDD decoupling capacitor's VSS and DVDD pins respectively. Current must not flow through GNDSENSE and VSENSE connections.

The capacitor should be ceramic and the inductor should have a DC current rating of at least 75 mA. For highest efficiency, DC resistance should be low and the core material should have low losses in the 1 MHz frequency range. Also, an optional Schottky diode should be used. The diode should have low forward voltage and low reverse current (reverse current through the diode adds to the leakage current in standby and retention power regimes). With careful component selection, the internal VR will provide peak efficiency approaching 95% and better than 90% efficiency over the 5-50 mA load current range.

Figure 4. Schematic Diagram Showing the Connection of the Internal Voltage Regulator to Required External Components



If the internal VR will not be used, PVDD must still be connected to the common power source, but VREN must be tied to ground and a regulated 1.8V supply connected to DVDD. The other VR pins (LX, VSENSE, and GNDSENSE) may be left floating.

§ §

4.0 Acronyms

Throughout this document when describing registers the acronyms listed in Table 4 are used to describe the access type. When a register or bit field has both read and write access types, those types will be separated by a "/". For example: "R/W" means that the register bit field permits both read and write access.

Table 4. Acronyms Used for Register Bit Field Functional Type Definition

Tag	Name	Description
R	Read	Field might be read without restrictions
W	Write	Field might be written without restrictions
RO	Read-Only	Only read operation makes sense for this field. Writing to the bit(s) is ignored
RAZ	Read-As-Zero	Read-only field returning zeros in all bits
RAO	Read-As-One	Read-only field returning ones in all bits
WO	Write-Only	Only write operation makes sense for this field. Reading of the bit(s) always returns zeroes
W1TP	Write 1 To Pulse	Writing of 1(s) activates 1 clock cycle assertion of internal signal(s) leading to appropriate actions in the device. The pulse does not change the value read from the bit, at least directly. Writing of 0(s) has no effect.
W1TS	Write 1 To Set	Writing of 1(s) sets the corresponding bit(s). Writing of 0(s) has no effect. Clearing of the bit(s) is performed by HW in response to certain specific event(s) (refer to the bit field description for details)
W1TC	Write 1 To Clear	Writing of 1(s) clears the corresponding bit(s). Writing of 0(s) has no effect. Setting of the bit(s) is performed by HW in response to certain specific event(s) (refer to the bit field description for details)

§ §

5.0 Memory Map

Figure 5 illustrates the MCU memory map. The 4 GB physical address range is divided into the following ordered memory:

- lower 2 GB of weakly ordered memory
- upper 2 GB of strongly ordered memory

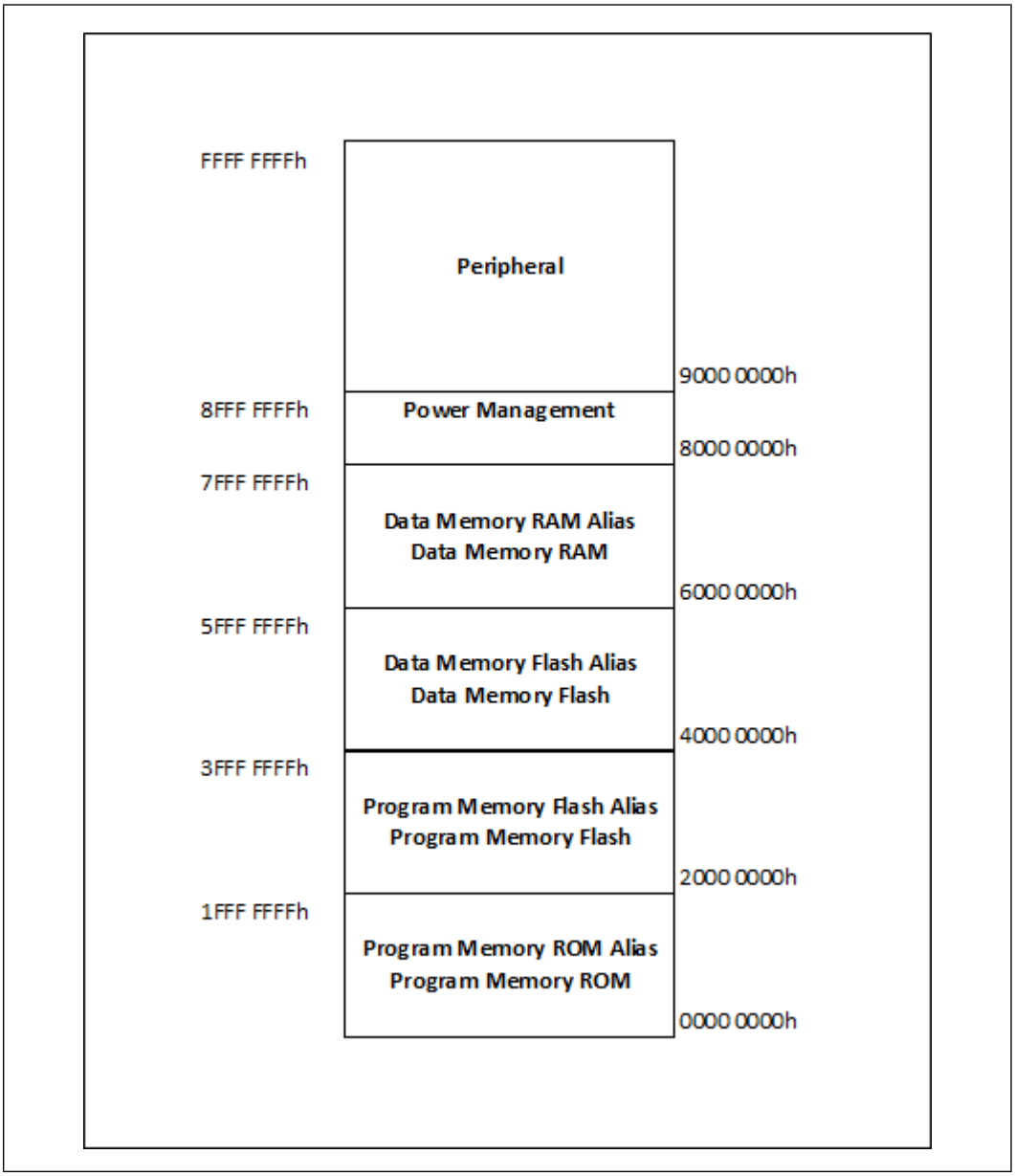
5.1 Lower 2 GB of Weakly Ordered Memory

Accesses to weakly ordered memory may be reordered or retried when there is an interrupt. ROM, Flash memory, and RAM, which do not have side effects when accessed are mapped in the lower 2GB of weakly ordered memory. Following a system RESET, the CPU begins execution from address 0x0000 0000 in the Program Memory ROM.

5.2 Upper 2 GB of Strongly Ordered Memory

Accesses to strongly ordered memory are never reordered or retried. Interrupts are held off until accesses to strongly ordered memory are complete. Hardware peripherals with FIFOs or other registers that have side effects when accessed are mapped in the upper 2 GB of strongly ordered memory.

Figure 5. MCU Memory Map



5.3 Program Memory

MCU Program Memory consists of two regions:

- ROM Bootstrap and Utilities region, which occupies the address range 0x0000 0000 to 0x1FFF FFFF
- Flash User Program region, which occupies the address range 0x2000 0000 to 0x3FFF FFFF

The Program Memory can only be read. Any attempt to write to this address range will result in a machine check exception. Figure 6 illustrates the program memory map.

Figure 6. Program Memory ROM and Program Memory Flash Memory Map

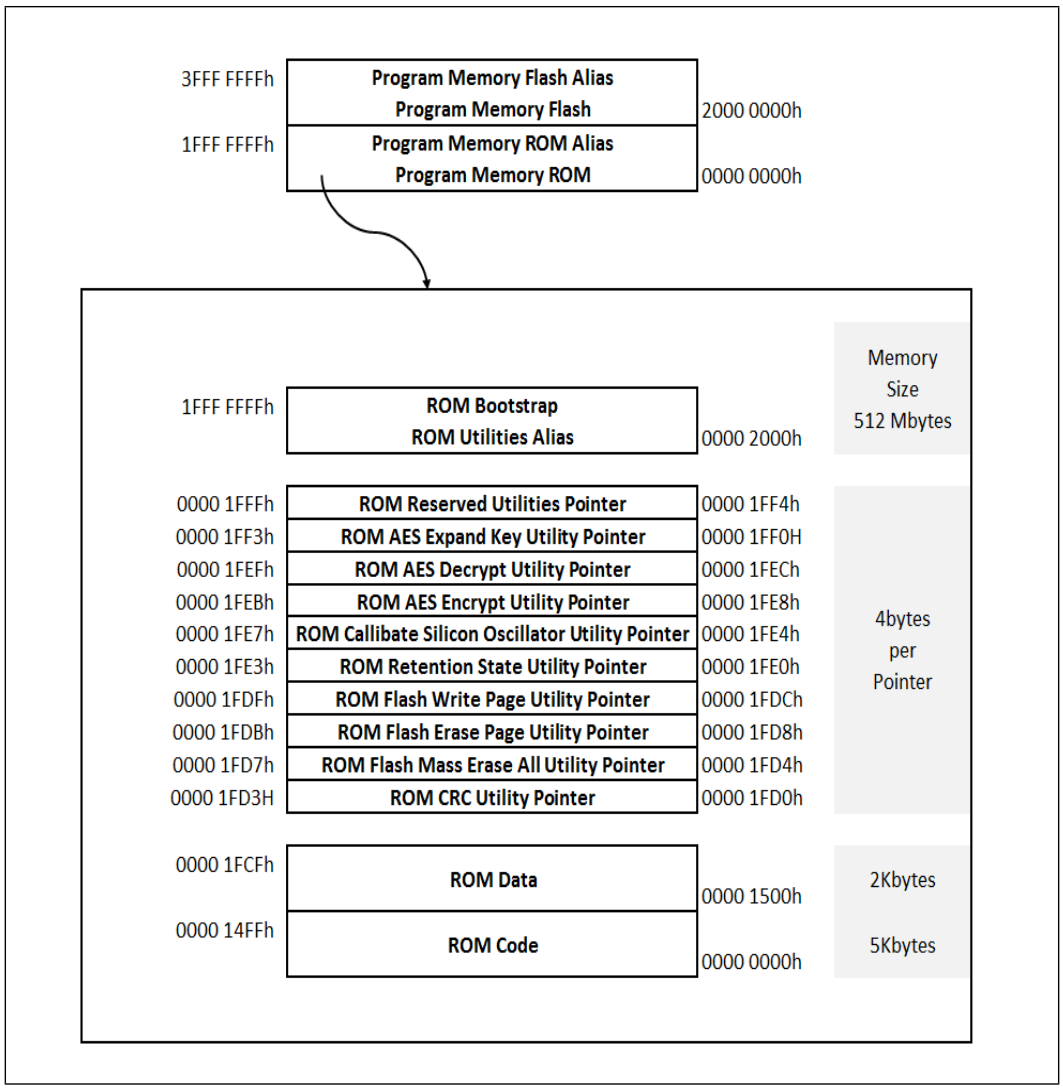
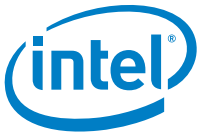


Table 5. ROM Size and Flash User Program Size

Region	Size	Aliased Throughout	Example
ROM	8 kB	512 MB Address Range	The 13-bit ROM address equals the 32-bit physical address modulo 8 kB.
Flash User Program	32 kB	512 MB Address Range	The 15-bit Flash User Program address equals the 32-bit physical address modulo 32 kB.



5.3.1 ROM Code

The ROM Code region that occupies address range 0x0000 0000 to 0x0000 14FF contains ROM bootstrap and utilities code. After a reset, the CPU will start executing from address 0x0000 0000.

5.3.2 ROM Data

The ROM Data region contains ROM bootstrap and utilities data while occupying address range 0x0000 1500 to 0x0000 1FCF.

5.3.3 ROM Utility Pointers

ROM Utility Pointers, each 4-byte wide, occupy the address range 0x0000 1FD0 to 0x0000 1FF3. Each pointer contains the target address of its corresponding utility function, which is located in the ROM Code region. The functions of each of these utilities are described below.

5.3.3.1 ROM CRC Utility Pointer

The function at this target address performs a CCITT-CRC16 algorithm on the specified data.

*unsigned short r_crc16 (unsigned char *p_msg, unsigned short msg_size)*

Input	*p_msg*	A pointer to the data to CRC
	msg_size	The length of the data to CRC in bytes
Output	*16-bit CRC*	The 16-bit CRC for the data

5.3.3.2 Flash Mass Erase All Utility

The function at this target address erases all code and data pages in the flash (including the configuration sections) and then jumps to the startup. This effectively "bricks" the device.

void r_erase_all_flash (void)

5.3.3.3 Flash Erase Page Utility

The function at this target address erases 1 page from the flash. The page can be either a 2 kB data page or 4 kB instruction page.

*int r_erase_flash_data_page (const unsigned int *p_start_addr)*

Input	*p_start_addr*	The starting address of the page to erase. This address **MUST** be page aligned!
Output	*0*	Returns 0 on success
	-1	ERR_BAD_PAGE_ADDR
	-2	ERR_PAGE_LOCKED

5.3.3.4 Flash Write Page Utility

This function writes 1 page in the flash. The function will erase the page before writing, but it will not save any data on the page before erasure (except the User Configuration and Global Configuration information in the upper page of data flash). It is the user's responsibility to copy the page into their buffer, modify the contents, and then provide the modified data to this function. The page can be either a 2 kB data page or 4 kB instruction page. The length of the buffer is automatically determined by the address of the page to which the data is written. If CRC checksums are turned on for this block of flash, the appropriate checksum will automatically be updated in the User Configuration area.

*int r_write_flash_data_page (unsigned int *p_start_addr, unsigned int *p_buffer)*

Input	*p_start_addr*	The starting address of the page to write to. This address MUST be page aligned!
	p_buffer	The buffer containing the data to be written
Output	*0*	Returns 0 on success
	-1	ERR_BAD_PAGE_ADDR
	-2	ERR_PAGE_LOCKED

Note: The data in the buffer MAY be destroyed during the flash process!

5.3.3.5 ROM Retention State Utility Pointer

The function at this target address enters retention state and then exits once the previously programmed wake event occurs (see Section 6.2.4 below for a detailed description of retention state).

void r_retention_mode (void)

5.3.3.6 Calibrate Silicon Oscillator Utility

The function at this target address calibrates the frequency of the silicon oscillator. It requires a 32,768 Hz crystal connected between XTALI[1] and XTALO[1].

int r_calibrate_osc_freq (unsigned int new_freq)

Input	*new_freq*	The new frequency in Hz. Supported values are 2,660,000-32,000,000.
Output	*actual_freq*	The actual frequency in Hz (success)
	3	The 32 kHz crystal oscillator did not lock
	4	The real time clock did not increment
	5	The value of input parameter *new_freq* is invalid

Note: In case of error, the frequency of the silicon oscillator will remain unchanged.

5.3.3.7 ROM AES Encrypt Utility Pointer

The function at this target address encrypts a 128-bit block of data with the specified round keys. The time required to encrypt the block is 2478 clocks (approximately 77.4 microseconds at 32 MHz).

*void r_aesenc (unsigned char *p_msg, unsigned char *p_key_sched, aes_round_taes_rounds)*

Input	*p_msg*	A pointer to the data to encrypt
	p_key_sched	A pointer to the key schedule (r_expand_key)
	aes_rounds	The number of rounds to perform (aes_round_t). The only supported value currently is 10 for AES-128.

5.3.3.8 ROM AES Decrypt Utility Pointer

The function at this target address decrypts a 128-bit block of data with the specified round keys. The time required to decrypt the block is 3156 clocks (Approximately 98.6 microseconds at 32 MHz).

*void r_aesdec (unsigned char *p_msg, unsigned char *p_key_sched, aes_round_taes_rounds)*

Input	*p_msg*	A pointer to the data to decrypt
	p_key_sched	A pointer to the key schedule (r_expand_key)
	aes_rounds	The number of rounds to perform (aes_round_t). The only supported value currently is 10 for AES-128.

5.3.3.9 ROM AES Expand Key Utility Pointer

The function at this target address uses the provided key to create the round keys to use with encryption and decryption. The time required to compute the round key is 1087 clocks (Approximately 34 microseconds at 32 MHz).

*void r_expandkey (unsigned char *key, unsigned char *expkey, aes_round_taes_rounds)*

Input	*key*	A pointer to the key to expand
	expkey	A pointer to the key schedule. This is the resultant of this operation.
	aes_rounds	The number of rounds to perform (aes_round_t). The only supported value currently is 10 for AES-128.

5.3.4 Flash User Program

Flash User Program Memory occupies address range 0x2000 0000 to 0x3FFF FFFF. User application code is stored here.

5.4 Data Memory

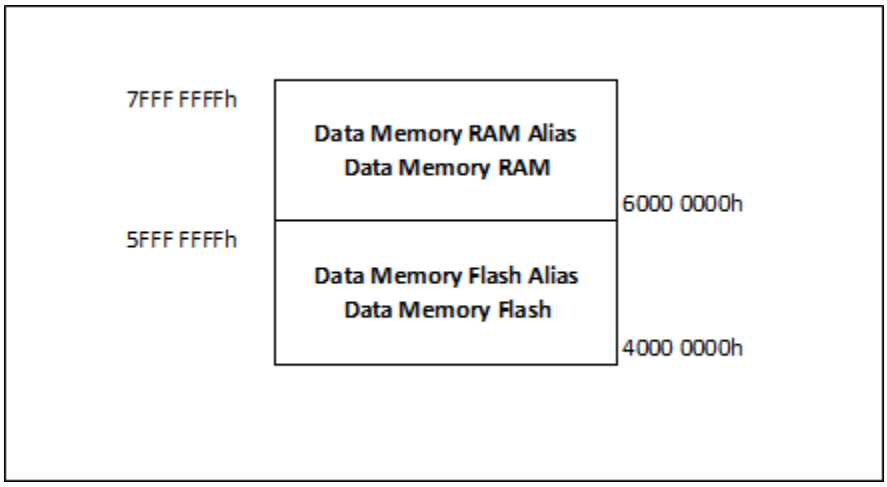
MCU Data Memory consists of two regions:

- Flash Data Memory region occupying 0x4000 0000 to 0x5FFF FFFF

- RAM Data Memory region occupying address range 0x60000000 to 0x7FFF FFFF

Data memory can be accessed only via operand fetch or debug controller. Any attempt to fetch instructions from this address range will result on a machine check exception. Any attempt to write to Flash Data Memory will result in a machine check exception. Figure 7 illustrates the Data Memory map.

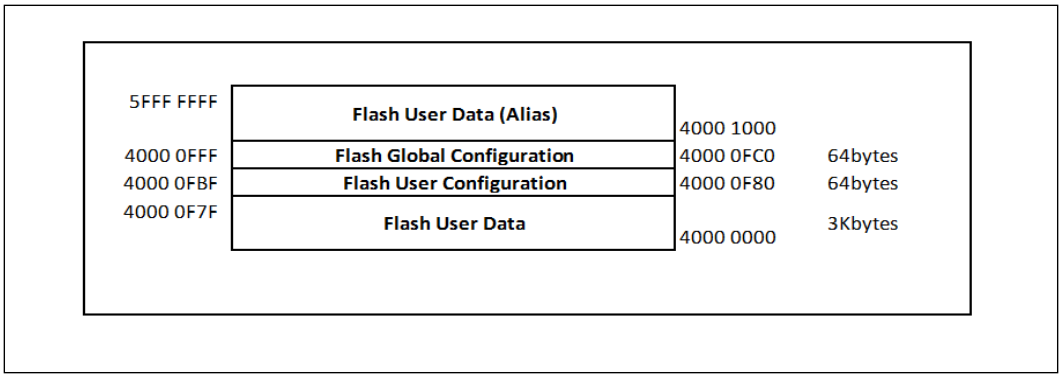
Figure 7. Data Memory Map



5.4.1 Flash Data Memory

A 4 kB Flash Data Memory occupies address range 0x4000 0000 to 0x4000 0FFF. It is aliased throughout the 512 MB address range (i.e. the 12-bit Flash Data Memory address equals the 32-bit physical address modulo 4 kB). Figure 8 illustrates the Flash Data Memory map.

Figure 8. Flash User Data Memory Map



5.4.1.1 Flash User Configuration

The MCU allocates 64 bytes of Flash Data Memory for Flash User Configuration data. This data provides the user with a way to configure optional features of the bootstrap flow as well as enabling optional locks and CRCs for the various flash pages. A detailed description of the user configuration region is provided in Section 6.1.2 below.

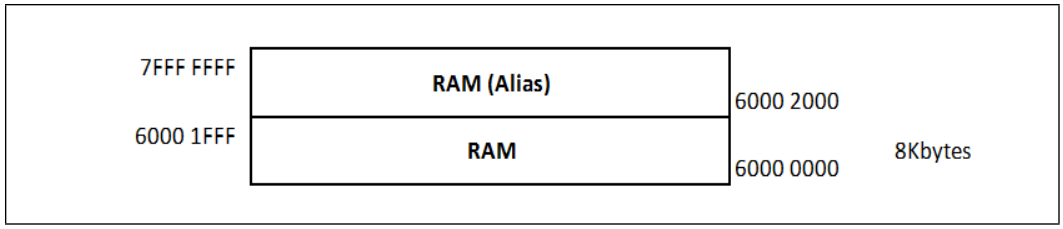
5.4.1.2 Flash Global Configuration

The MCU allocates 64 bytes of Flash Data Memory for Flash Global Configuration data. This is a non-volatile storage area for factory specific configuration information. Currently, only the oscillator trim code is stored here. The flash utilities provided in ROM will not allow the user to overwrite this information. However, it is possible to overwrite this information by directly accessing the flash controller. The flash controller is password protected to prevent accidental flash corruption. A detailed description of the global configuration region is provided in Section 6.1.3 below.

5.4.2 RAM Data Memory

An 8 kB RAM Data Memory occupies address range 0x6000 0000 to 0x6000 1FFF. It is aliased throughout the 512 MB address range (i.e. the 13-bit RAM Data Memory address equals the 32-bit physical address modulo 8 kB). Figure 9 illustrates the RAM Data Memory map.

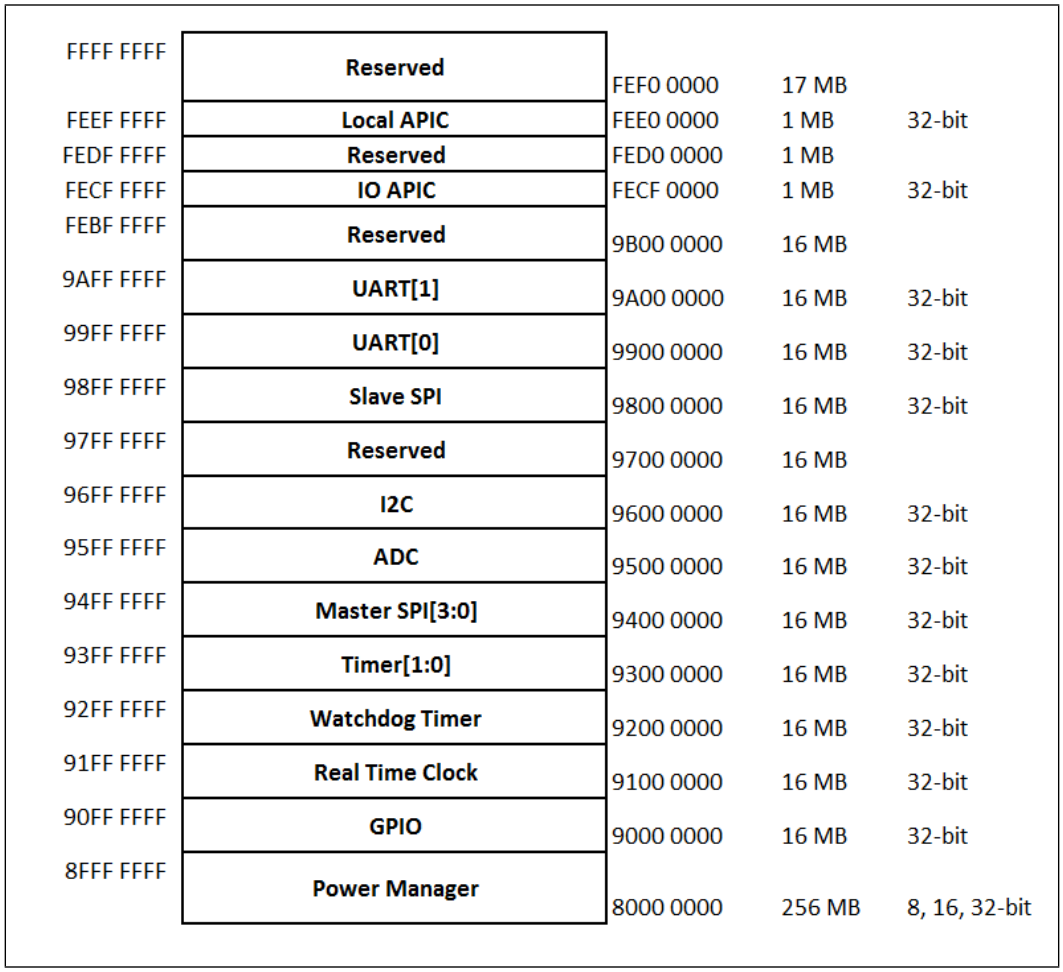
Figure 9. RAM Data Memory Map



5.5 Memory Mapped I/O

The upper 2 GB of the physical address range consists of registers for controlling the MCU hardware functions. The registers can be accessed only via operand fetch or debug controller. Any attempt to fetch instructions from this address range will result in a machine check exception. Figure 10 illustrates the Memory Mapped I/O.

Figure 10. Peripheral Memory Map



5.5.1 Power Management Registers

Power management registers can be accesses in 8-bit, 16-bit or 32-bit sizes, unless stated otherwise.

Table 6. Power Management Registers (Sheet 1 of 2)

Register	Start (hex)	End (hex)
Clock Enable Register	8000 0000	8000 0003
Clock Select Register	8000 0004	8000 0007
Clock AHB Register	8000 0008	8000 000B
APB Reset Register	8000 000C	8000 000F
33 MHz Oscillator Configuration Register 0	8000 0010	8000 0013
33 MHz Oscillator Configuration Register 1	8000 0014	8000 0017
32 kHz Oscillator Configuration Register	8000 0018	8000 001B

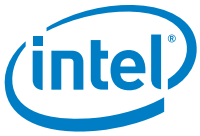


Table 6. Power Management Registers (Sheet 2 of 2)

Register	Start (hex)	End (hex)
Voltage Regulator Control/Status Register	8000 001C	8000 001F
Flash Controller Wait States Register	8000 0040	8000 0043
Flash Controller Erase/Write Control Register	8000 0044	8000 0047
Flash Controller Status Register	8000 0048	8000 004B
Flash Controller Instruction Write Address Register	8000 004C	8000 004F
Flash Controller Instruction Write Data 0 Register	8000 0050	8000 0053
Flash Controller Instruction Write Data 1 Register	8000 0054	8000 0057
Flash Controller Data Write Address Register	8000 0060	8000 0063
Flash Controller Data Write Data Register	8000 0064	8000 0067
I/O Pin Pull-up Enable Register	8000 0080	8000 0083
I/O Pin[7:0] Function Control Register	8000 0084	8000 0087
I/O Pin[15:8] Function Control Register	8000 0088	8000 008B
I/O Pin[23:16] Function Control Register	8000 008C	8000 008F
I/O Pin Slew Rate Control Register	8000 0090	8000 0093
Comparator Interrupt Enable Register	8000 00C0	8000 00C3
Comparator Reference Register	8000 00C4	8000 00C7
Comparator Polarity Register	8000 00C8	8000 00CB
Comparator Power Register	8000 00CC	8000 00CF
Comparator Interrupt Status Register	8000 00D0	8000 00D3
ADC Operating Mode Register (OM)	8000 00E0	8000 00E3

5.5.2 Peripheral Registers

Peripheral registers can only be accessed in 32-bit size and alignment.

Table 7. GPIO Registers (Sheet 1 of 2)

Register	Start (hex)	End (hex)
GPIO Data Register	9000 0000	9000 0003
GPIO Data Direction Register	9000 0004	9000 0007
GPIO Interrupt Enable Register	9000 0030	9000 0033
GPIO Interrupt Mask Register	9000 0034	9000 0037
GPIO Interrupt Type Register	9000 0038	9000 003B
GPIO Interrupt Polarity Register	9000 003C	9000 003F
GPIO Interrupt Status Register	9000 0040	9000 0043
GPIO Raw Interrupt Status Register	9000 0044	9000 0047
GPIO End of Interrupt Register	9000 004C	9000 004F
GPIO External Pin State Register	9000 0050	9000 0053

Table 7. GPIO Registers (Sheet 2 of 2)

Register	Start (hex)	End (hex)
GPIO Interrupt Synchronization Register	9000 0060	9000 0063
GPIO ID Code Register	9000 0064	9000 0067
GPIO Version Code Register	9000 006C	9000 006F
GPIO Configuration Register 2	9000 0070	9000 0073
GPIO Configuration Register 1	9000 0074	9000 0077

Table 8. Real Time Clock Registers

Register	Start (hex)	End (hex)
RTC Current Counter Value Register	9100 0000	9100 0003
RTC Counter Match Register	9100 0004	9100 0007
RTC Counter Load Register	9100 0008	9100 000B
RTC Counter Control Register	9100 000C	9100 000F
RTC Interrupt Status Register	9100 0010	9100 0013
RTC Interrupt Raw Status Register	9100 0014	9100 0017
RTC End of Interrupt Status Register	9100 0018	9100 001B
RTC Component Version Register	9100 001C	9100 001F

Table 9. Watchdog Timer Registers

Register	Start (hex)	End (hex)
WDT Control Register	9200 0000	9200 0003
WDT Timeout Range Register	9200 0004	9200 0007
WDT Current Counter Value Register	9200 0008	9200 000B
WDT Counter Restart Register	9200 000C	9200 000F
WDT Interrupt Status Register	9200 0010	9200 0013
WDT End of Interrupt Register	9200 0014	9200 0017
WDT Component Parameters Register 5	9200 00E4	9200 00E7
WDT Component Parameters Register 4	9200 00E8	9200 00EB
WDT Component Parameters Register 3	9200 00EC	9200 00EF
WDT Component Parameters Register 2	9200 00F0	9200 00F3
WDT Component Parameters Register 1	9200 00F4	9200 00F7
WDT Version Code Register	9200 00F8	9200 00FB
WDT Component Type Register	9200 00FC	9200 00FF

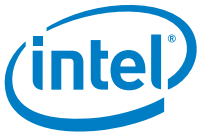


Table 10. General Purpose Timers Registers

Register	Timer [0]		Timer [1]	
	Start (hex)	End (hex)	Start (hex)	End (hex)
Timer Load Count Register	9300 0000	9300 0003	9300 0014	9300 0017
Timer Current Value Register	9300 0004	9300 0007	9300 0018	9300 001B
Timer Control Register	9300 0008	9300 000B	9300 001C	9300 001F
Timer End-of-Interrupt Register	9300 000C	9300 000F	9300 0020	9300 0023
Timer Interrupt Status Register	9300 0010	9300 0013	9300 0024	9300 0027
Timers Interrupt Status Register	9300 00A0	9300 00A3	9300 00A0	9300 00A3
Timer End-of-Interrupt Register	9300 00A4	9300 00A7	9300 00A4	9300 00A7
Timers Raw Interrupt Status Register	9300 00A8	9300 00AC	9300 00A8	9300 00AC
Timers Version Code Register	9300 00AC	9300 00AF	9300 00AC	9300 00AF

Table 11. Master SPI Registers (Sheet 1 of 2)

Register	Start (hex)	End (hex)
SPI Master Control Register 0	9400 0000	9400 0003
SPI Master Control Register 1	9400 0004	9400 0007
SPI Mater SSI Enable Register	9400 0008	9400 000B
SPI Master Microwire Control Register	9400 000C	9400 000F
SPI Master Slave Enable Register	9400 0010	9400 0013
SPI Master Baud Rate Select	9400 0014	9400 0017
SPI Master Transmit FIFO Threshold Level	9400 0018	9400 001B
SPI Master Receive FIFO Threshold Level	9400 001C	9400 001F
SPI Master Transmit FIFO Level Register	9400 0020	9400 0023
SPI Master Receive FIFO Level Register	9400 0024	9400 0027
SPI Master Status Register	9400 0028	9400 002B
SPI Master Interrupt Mask Register	9400 002C	9400 002F
SPI Master Interrupt Status Register	9400 0030	9400 0033
SPI Master Raw Interrupt Status Register	9400 0034	9400 0037
SPI Master Transmit FIFO Overflow Interrupt Clear Register	9400 0038	9400 003B
SPI Master Receive FIFO Overflow Interrupt Clear Register	9400 003C	9400 003F
SPI Master Receive FIFO Underflow Interrupt Clear Register	9400 0040	9400 0043
SPI Master Multi-Master Interrupt Clear Register	9400 0044	9400 0047
SPI Master Interrupt Clear Register	9400 0048	9400 004B

Table 11. Master SPI Registers (Sheet 2 of 2)

Register	Start (hex)	End (hex)
SPI Master Identification Register	9400 0058	9400 005B
SPI Master Component Version Register	9400 005C	9400 005F
SPI Master Data Register	9400 0060	9400 00EF

Table 12. ADC Registers

Register	Start (hex)	End (hex)
ADC Channel Sequence Table Register (CS)	9500 0000	9500 001F
ADC Command Register (Command)	9500 0020	9500 0023
ADC Interrupt Status Register (IS)	9500 0024	9500 0027
ADC Interrupt Enable Register (IE)	9500 0028	9500 002B
ADC Sample Register (Sample)	9500 002C	9500 002F
ADC Calibration Data Register (CD)	9500 0030	9500 0033
ADC FIFO Count Register (Count)	9500 0034	9500 0037

Table 13. I^2C Registers (Sheet 1 of 2)

Register	Start (hex)	End (hex)
I^2C Control Register	9600 0000	9600 0003
I^2C Target Address Register	9600 0004	9600 0007
I^2C Slave Address Register	9600 0008	9600 000B
I^2C Rx/Tx Data Buffer and Command Register	9600 0010	9600 0013
I^2C Standard Speed Clock SCL High Count Register	9600 0014	9600 0017
I^2C Standard Speed Clock SCL Low Count Register	9600 0018	9600 001B
I^2C Fast Speed Clock SCL High Count Register	9600 001C	9600 001F
I^2C Fast Speed Clock SCL Low Count Register	9600 0020	9600 0023
I^2C Interrupt Status Register	9600 002C	9600 002F
I^2C Interrupt Mask Register	9600 0030	9600 0033
I^2C Raw Interrupt Status Register	9600 0034	9600 0037
I^2C Receive FIFO Threshold Register	9600 0038	9600 003B
I^2C Transmit FIFO Threshold Register	9600 003C	9600 003F
I^2C Clear Combined and Individual Interrupt Register	9600 0040	9600 0043
I^2C Clear RX_UNDER Interrupt Register	9600 0044	9600 0047
I^2C Clear RX_OVER Interrupt Register	9600 0048	9600 004B
I^2C Clear TX_OVER Interrupt Register	9600 004C	9600 004F
I^2C Clear RD_REQ Interrupt Register	9600 0050	9600 0053
I^2C Clear TX_ABORT Interrupt Register	9600 0054	9600 0057

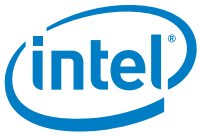


Table 13. I²C Registers (Sheet 2 of 2)

Register	Start (hex)	End (hex)
I²C Clear RX_DONE Interrupt Register	9600 0058	9600 005B
I²C Clear ACTIVITY Interrupt Register	9600 005C	9600 005F
I²C Clear STOP_DET Interrupt Register	9600 0060	9600 0063
I²C Clear START_DET Interrupt Register	9600 0064	9600 0067
I²C Clear GEN_CALL Interrupt Register	9600 0068	9600 006B
I²C Enable Register	9600 006C	9600 006F
I²C Status Register	9600 0070	9600 0073
I²C Transmit FIFO Level Register	9600 0074	9600 0077
I²C Receive FIFO Level Register	9600 0078	9600 007B
I²C SDA Hold Time Length Register	9600 007C	9600 007F
I²C SDA Setup Register	9600 0094	9600 0097
I²C Transmit Abort Source Register	9600 0080	9600 0083
I²C ACK General Call Register	9600 0098	9600 009B
I²C Enable Status Register	9600 009C	9600 009F
I²C Component Parameter Register 1	9600 00F4	9600 00F7
I²C Component Version	9600 00F8	9600 00FB
I²C Component Type Register	9600 00FC	9600 00FF

Table 14. Slave SPI Registers (Sheet 1 of 2)

Register	Start (hex)	End (hex)
SPI Slave Control Register 0	98000000	9800 0003
SPI Slave Enable Register	98000008	9800 000B
SPI Slave Microwire Control Register	9800000C	9800 000F
SPI Slave Transmit FIFO Threshold Level	9800 0018	9800 001B
SPI Slave Receive FIFO Threshold Level	9800 001C	9800 001F
SPI Slave Transmit FIFO Level Register	9800 0020	9800 0023
SPI Slave Receive FIFO Level Register	9800 0024	9800 0027
SPI Slave Status Register	9800 0028	9800 002B
SPI Slave Interrupt Mask Register	9800 002C	9800 002F
SPI Slave Interrupt Status Register	9800 0030	9800 0033
SPI Slave Raw Interrupt Status Register	9800 0034	9800 0037
SPI Slave Transmit FIFO Overflow Interrupt Clear Register	9800 0038	9800 003B
SPI Slave Receive FIFO Overflow Interrupt Clear Register	9800 003C	9800 003F
SPI Slave Receive FIFO Underflow Interrupt Clear Register	9800 0040	9800 0043
SPI Slave Interrupt Clear Register	9800 0048	9800 004B

Table 14. Slave SPI Registers (Sheet 2 of 2)

Register	Start (hex)	End (hex)
SPI Slave Identification Register	9800 0058	9800 005B
SPI Slave Component Version Register	9800 005C	9800 005F
SPI Slave Data Register	9800 0060	9800 00EF

Table 15. UART Registers

Register	Timer [0]		Timer [1]	
	Start (hex)	End (hex)	Start (hex)	End (hex)
UART Receive Buffer Register	9900 0000	9900 0003	9A00 0000	9A00 0003
UART Transmit Holding Register	9900 0000	9900 0003	9A00 0000	9A00 0003
UART Divisor Latch Low Register	9900 0000	9900 0003	9A00 0000	9A00 0003
UART Divisor Latch High Register	9900 0004	9900 0007	9A00 0004	9A00 0007
UART Interrupt Enable Register	9900 0004	9900 0007	9A00 0004	9A00 0007
UART Interrupt Identity Register	9900 0008	9900 000B	9A00 0008	9A00 000B
UART FIFO Control Register	9900 0008	9900 000B	9A00 0008	9A00 000B
UART Line Control Register	9900 000C	9900 000F	9A00 000C	9A00 000F
UART Modem Control Register	9900 0010	9900 0013	9A00 0010	9A00 0013
UART Line Status Register	9900 0014	9900 0017	9A00 0014	9A00 0017
UART Modem Status Register	9900 0018	9900 001B	9A00 0018	9A00 001B
UART Scratch Pad Register	9900 001C	9900 001F	9A00 001C	9A00 001F
UART Shadow Receive Buffer Register.	9900 0030	9900 006F	9A00 0030	9A00 006F
UART Shadow Transmit Holding Register	9900 0030	9900 006F	9A00 0030	9A00 006F
UART Status Register	9900 007C	9900 007F	9A00 007C	9A00 007F
UART Transmit FIFO Level	9900 0080	9900 0083	9A00 0080	9A00 0083
UART Receive FIFO Level	9900 0084	9900 0087	9A00 0084	9A00 0087
UART Software Reset Register	9900 0088	9900 008B	9A00 0088	9A00 008B
UART Shadow Request to Send	9900 008C	9900 008F	9A00 008C	9A00 008F
UART Shadow Break Control Register	9900 0090	9900 0093	9A00 0090	9A00 0093
UART Shadow Break Control Register	9900 0094	9900 0097	9A00 0094	9A00 0097
UART Shadow Receive Threshold	9900 0098	9900 009B	9A00 0098	9A00 009B
UART Shadow Break Control Register	9900 009C	9900 009F	9A00 009C	9A00 009F
UART Component Parameter Register	9900 00F4	9900 00F7	9A00 00F4	9A00 00F7
UART Component Version	9900 00F8	9900 00FB	9A00 00F8	9A00 00FB
UART Component Type Register	9900 00FC	9900 00FF	9A00 00FC	9A00 00FF

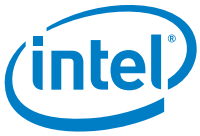


Table 16. Local APIC Registers

Register	Start (hex)	End (hex)
Task Priority Register	FEE0 0080	FEE0 0083
Process Priority Register	FEE0 00A0	FEE0 00A3
End-of-Interrupt Register	FEE0 00B0	FEE0 00B3
Spurious Interrupt Vector Register	FEE0 00F0	FEE0 00F3
In-Service Register	FEE0 0110	FEE0 0113
Interrupt Request Register	FEE0 0210	FEE0 0213

Table 17. I/O APIC Registers

Register	Start (hex)	End (hex)
IOAPIC Register Select	FEC0 0000	FEC0 0003
IOAPIC Register Window	FEC0 0010	FEC0 0013

Table 18. APIC Timer Registers

Register	Start (hex)	End (hex)
Local Vector Table Timer Register	FEE0 0320	FEE0 0323
Timer Initial Count Register	FEE0 0380	FEE0 0383
Timer Current Count Register	FEE0 0390	FEE0 0393

5.6 Memory Errors

There are certain illegal memory transactions that will result in a machine check exception. These are:

1. Attempting to fetch an instruction from an address beyond ICCM address range.
2. Attempting to write data to an address below DCCM RAM address range.

§ §

6.0 Operating Modes

Describe reset bootstrap, flash configuration space, ROM utilities, power regimes, VR and clock network. Refer to Intel® Quark™ microcontroller D1000 Datasheet for more details on the operating modes.

6.1 Reset and Bootstrap

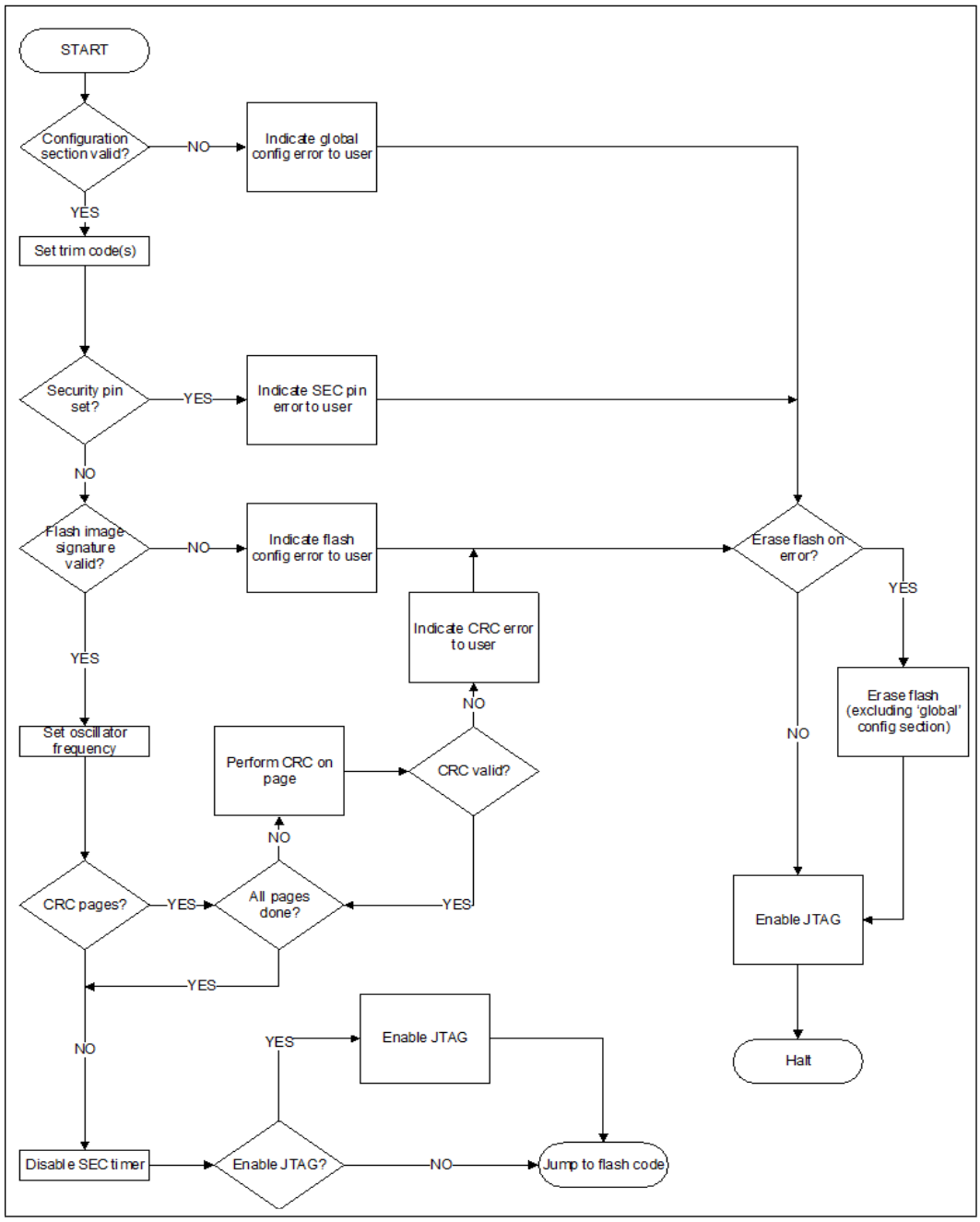
Intel® Quark™ microcontroller D1000 has a built in power-on reset. In addition, the RST_N pin provides a user asserted low true reset. RST_N is an analog input with hysteresis that can be strapped to AVDD, connected to an RC delay pulled-up to DVDD to extend reset duration, or driven by some external device such as a brownout detector. If RST_N is less than 0.8 V, reset is active. If RST_N is greater than 1.1 V, reset is inactive. The actual threshold is somewhere in between with no less than 3.4mV hysteresis.

6.1.1 Functional Description

While reset is active, all GPIO pins are in the high impedance state. A low to high transition on reset causes the MCU to reboot from integrated ROM. A configuration table is provided in the upper data page of flash memory to control the bootstrap process. Figure 11 shows a flow chart of the bootstrap procedure. The bootstrap procedure begins by checking for a valid configuration section in flash. This is indicated by a predefined signature stored at a fixed location. If the signature matches the predefined value, oscillator trim data and various configuration options contained within the configuration section are processed. Any other signature value results in the processor halting with JTAG enabled and flash erased if so configured.

Oscillator trim data stored in the configuration section is applied to the silicon oscillator. This data is computed at the factory and can be recomputed by the user to calibrate to non-standard frequencies and operating conditions. Other configuration options include the presence of a valid program in flash and optional CRC checks. If there is a valid program in flash and the required CRC checks pass, the bootstrap program jumps to it. If there is no valid program in flash or a CRC check fails, the bootstrap program enables JTAG and halts with flash erased if so configured.

Figure 11. Bootstrap flowchart showing security checks and JTAG enabling



One final configuration option is how to proceed in case of a bootstrap failure. For secure applications, the bootstrap procedure can be configured to erase flash just prior to JTAG enabling. For unsecure applications, JTAG is enabled without erasing flash allowing the contents of flash to be examined for failure analysis. If the bootstrap procedure ends in processor halt, error information will be stored at addresses 0x6000 1000 and 0x6000 1004. The format of error information is shown in Table 19 below.

Table 19. Bootstrap Error Codes

Error Type	Error Code in Address 0x6000 1000	Extended Information in Address 0x6000 1004
ROM_CRC_ERROR	0x0000 0001	Starting address of page with error
ROM_INV_GLOBAL_SIG	0x0000 0002	-
ROM_INV_FLASH_SIG	0x0000 0003	-
ROM_SEC_SET	0x0000 0004	-

6.1.2 User Configuration Data

Intel® Quark™ microcontroller D1000 provides the user with a way to configure optional features of the bootstrap flow as well as enabling optional locks and CRCs for the various flash pages.

Table 20. User Configuration Data Stored in Upper Page of Data Flash (Sheet 1 of 2)

Address (hex)	Bits	Description
4000 0F80	15:0	Configuration Section Valid – a value of 0xAB8D indicates that the contents of this configuration section are valid. Any other value results in bootstrap terminating with JTAG enabled.
	17:16	Bootstrap Clock Frequency – this field configures the clock frequency at which the bootstrap procedure is executed. It is encoded thusly: 0 = 32 MHz 1 = 16 MHz 2 = 8 MHz 3 = 4 MHz
	31:18	Reserved
4000 0F84	7:0	Flash Instruction Page Lock – these bits control whether or not an instruction flash page is locked from erasing and writing using the built-in ROM utilities. Bit 0 corresponds to the 4 kB page beginning at address 0x2000 0000, bit 1 to the page beginning at address 0x2000 1000, and so on up to bit 7 which corresponds to the page beginning at address 0x2000 7000. When a bit is '0', the corresponding page is locked from erasing and writing.
	9:8	Flash Data Page Lock – these bits control whether or not a data flash page is locked from erasing and writing using the built-in ROM utilities. Bit 8 corresponds to the 2 kB page beginning at address 0x40000000 and bit 9 to the page beginning at address 0x4000 0800. When a bit is '0', the corresponding page is locked from erasing and writing.
	15:10	Reserved
	23:16	Erase Flash on Bootstrap Error – this field informs the bootstrap procedure how to proceed if there is an error. Any value other than 0xFF results in flash erasure prior to enabling JTAG. A value of 0xFF results in JTAG being enabled without flash erasure. Note that the global configuration section is never erased.
	31:24	Enable JTAG – this field informs the bootstrap procedure whether or not to enable JTAG prior to executing to the user program. A value of 0xFF results in JTAG being enabled just prior to executing the user program. Any value other than 0xFF results in executing the user program with JTAG disabled. Note that the user program can enable JTAG at any point after bootstrap.

Table 20. User Configuration Data Stored in Upper Page of Data Flash (Sheet 2 of 2)

Address (hex)	Bits	Description
4000 0F88	7:0	Flash Instruction Page CRC Disable – these bits control whether or not an instruction flash page is validated using CRC before the user program is executed. Bit 0 corresponds to the 4 kB page beginning at address 0x2000 0000, bit 1 to the page beginning at address 0x2000 1000, and so on up to bit 7 which corresponds to the page beginning at address 0x2000 7000. When a bit is '1', the corresponding page is not validated.
	9:8	Flash Data Page CRC Disable – these bits control whether or not a data flash page is validated using CRC before the user program is executed. Bit 8 corresponds to the 2 kB page beginning at address 0x4000 0000 and bit 9 to the page beginning at address 0x4000 0800. When a bit is '1', the corresponding page is not validated.
	31:10	Reserved
4000 0F8C	15:0	Flash Instruction Page 0 CRC – when the CRC validation for page 0 is enabled, the calculated CRC must equal this value. A mismatch results in bootstrap terminating with JTAG enabled.
	31:16	Flash Instruction Page 1 CRC – when the CRC validation for page 1 is enabled, the calculated CRC must equal this value. A mismatch results in bootstrap terminating with JTAG enabled.
4000 0F90	15:0	Flash Instruction Page 2 CRC – when the CRC validation for page 0 is enabled, the calculated CRC must equal this value. A mismatch results in bootstrap terminating with JTAG enabled.
	31:16	Flash Instruction Page 3 CRC – when the CRC validation for page 1 is enabled, the calculated CRC must equal this value. A mismatch results in bootstrap terminating with JTAG enabled.
4000 0F94	15:0	Flash Instruction Page 4 CRC – when the CRC validation for page 0 is enabled, the calculated CRC must equal this value. A mismatch results in bootstrap terminating with JTAG enabled.
	31:16	Flash Instruction Page 5 CRC – when the CRC validation for page 1 is enabled, the calculated CRC must equal this value. A mismatch results in bootstrap terminating with JTAG enabled.
4000 0F98	15:0	Flash Instruction Page 6 CRC – when the CRC validation for page 0 is enabled, the calculated CRC must equal this value. A mismatch results in bootstrap terminating with JTAG enabled.
	31:16	Flash Instruction Page 7 CRC – when the CRC validation for page 1 is enabled, the calculated CRC must equal this value. A mismatch results in bootstrap terminating with JTAG enabled.
4000 0F9C	15:0	Flash Data Page 0 CRC – when the CRC validation for page 0 is enabled, the calculated CRC must equal this value. A mismatch results in bootstrap terminating with JTAG enabled.
	31:16	Flash Data Page 1 CRC – when the CRC validation for page 1 is enabled, the calculated CRC must equal this value. A mismatch results in bootstrap terminating with JTAG enabled.
4000 0FA0 – 4000 0FBC	31:0	Reserved

6.1.3 Global Configuration

Intel® Quark™ microcontroller D1000 provides a non-volatile storage area for factory specific configuration information. Currently, only the oscillator trim code is stored here.

The flash utilities provided in ROM will not allow the user to overwrite this information.

However, it is possible to overwrite this information by directly accessing the flash controller. The flash controller is password protected to prevent accidental flash corruption.

Table 21. Global Configuration Data Stored in Upper Page of Data Flash

Address (hex)	Bits	Description
40000FC0	7:0	Version – this field informs the bootstrap procedure which version of configuration section is in use. Since Intel® Quark™ microcontroller D1000 is the first generation in a series, this field is not currently used.
	15:8	Reserved
	31:16	Configuration Section Valid – a value of 0x5CAB indicates that the contents of this configuration section are valid. Any other value results in bootstrap terminating with JTAG enabled.
40000FC4	9:0	Silicon Oscillator Trim Code – this value trims the silicon oscillator to the desired frequency. The frequency is a monotonic function of this value. The oscillator is trimmed before configuring the bootstrap frequency.
	31:10	Reserved
40000FC8 – 40000FFC	31:0	Reserved

6.1.4 APB Reset Register

The APB Reset Register is aligned on 4-byte boundaries and can be accessed as bytes, words, or double words. It is a readable and writable register memory mapped at address 0x8000 000C to 0x8000 000F. It is used to reset APB peripherals. The bits are self-clearing.

Table 22. APB Reset Register

Mnemonic	Access	Bits	Description	Reset
APB	R/W	0	APB Reset: when written '1', all APB peripherals are reset including all of their programming registers. Writing '0' has no effect. Reset is active when read '0', inactive when read '1'.	0
SPIM	R/W	1	SPI Master Reset: when written '1', the SPI master is reset, but programming registers are unaffected. Writing '0' has no effect. Reset is active when read '0', inactive when read '1'.	0
SPIS	R/W	2	SPI Slave Reset: when written '1', the SPI slave is reset, but programming registers are unaffected. Writing '0' has no effect. Reset is active when read '0', inactive when read '1'.	0
I^2C	R/W	3	I^2C Reset: when written '1', the I^2C peripheral is reset, but programming registers are unaffected. Writing '0' has no effect. Reset is active when read '0', inactive when read '1'.	0
RTC	R/W	4	Real Time Clock Reset: when written '1', the real time clock is reset, but programming registers are unaffected. Writing '0' has no effect. Reset is active when read '0', inactive when read '1'.	0
TMR0	R/W	5	Timer[0] Reset: when written '1', the timer[0] peripheral is reset, but programming registers are unaffected. Writing '0' has no effect. Reset is active when read '0', inactive when read '1'.	0
TMR1	R/W	6	Timer[1] Reset: when written '1', the timer[1] peripheral is reset, but programming registers are unaffected. Writing '0' has no effect. Reset is active when read '0', inactive when read '1'.	0

6.2 Fine-Grained Power Management

The MCU features fine-grained power management. Power consumption can be fine-tuned to the user application by powering down unneeded modules, shutting off unneeded clocks and adjusting the frequencies of others. However, in order to facilitate discussion of features and characterization we will broadly define six discrete power states:

- Active – In this state, the CPU is executing. All pipeline, memory, and AHB clocks are running. The core voltage is at 1.8 V and the buck regulator is selected.
- Halt – In this state, the CPU is halted. All pipeline, memory, and AHB clocks are stopped. The core voltage is at 1.8 V and the buck regulator is selected. Wake-up comes from any unmasked interrupt.
- ADC Sleep – In this state, all clocks except ADC and optionally the 32 kHz oscillator are stopped. The core voltage is set to 1.8 V and the buck regulator is selected. Wake-up comes from the ADC.
- Standby – In this state, all clocks except optionally the 32 kHz oscillator are stopped. The core voltage is set to 1.8 V and the linear regulator is selected. Digital I/O may be enabled, but should not be switching. SEC must be de-asserted (strapped to VSS) so as not to cause wake up. Wake-up comes from the real-time clock or a comparator.
- Retention – In this state, all clocks except optionally the 32 kHz oscillator are stopped. The core voltage is set to 1.35 V and the linear regulator is selected. Digital I/O may be enabled, but should not be switching. SEC must be de-asserted (strapped to VSS) so as not to cause wake-up. Wake-up comes from the real-time clock or a comparator.

6.2.1 Halt State

Halt state is entered whenever a halt instruction is executed. Exit from halt state occurs when an unmasked interrupt is received. After servicing the interrupt, execution continues with the instruction following the halt. This is typically used with interrupt driven programs that have an idle main loop. Resumption of active state is within 3-4 clocks of an interrupt event.

6.2.2 ADC Sleep State

ADC sleep state is entered by stopping all clocks except ADC clock and optionally the 32 kHz oscillator midstream. Exit from ADC sleep state occurs when the ADC completes a conversion operation. Interrupt service is optional since masking the ADC interrupt does not prevent wake-up. This is typically used in programs that can sleep (i.e. all peripherals are stopped) and then periodically wake-up to process an ensemble of analog samples. Resumption of active state is typically within 2 µs of completion of the ADC conversion command (see Section 9.3.2.3 for a list of ADC conversion commands).

When transitioning into ADC sleep state, care must be taken to ensure that the conversion command starts but does not complete before you disable the AHB clock. The step-by-step procedure is given below:

- Globally disable interrupts using “CLI” assembly language instruction
- Initiate the conversion command.
- Read back the conversion command (this ensures that the ADC command finite state machine acknowledges the command before you disable the AHB clock).
- Disable AHB clock.
- Globally re-enable interrupts using “STI” assembly language instruction

6.2.3 Standby State

Standby state is entered by stopping all clocks except optionally 32 kHz oscillator midstream. Exit from standby state occurs when the real-time clock reaches terminal count or a comparator detects a reference voltage threshold crossing. Resumption of active state is typically within 2-5 µs of a threshold crossing depending on edge polarity and whether a high-speed or low-power comparator was used. Like ADC sleep, care must be taken to ensure that the real-time clock or comparator event does not occur prior to disabling the AHB clock.

6.2.4 Retention State

Retention state is entered by lowering the core voltage to 1.35 V and stopping all clocks except optionally the 32 kHz oscillator midstream. Exit from retention state occurs when the real-time clock reaches terminal count or a comparator detects a reference voltage threshold crossing. Resumption of active state is typically within 800 µs of a threshold crossing (the relatively long wake-up time is needed in order to raise the core voltage to 1.8 V and ensure that the voltage regulator is in regulation). Like ADC sleep and standby, care must be taken to ensure that the real-time clock or comparator event does not occur prior to disabling the AHB clock. Since entering retention state requires a very specific sequence of actions by software, there are many opportunities to get it wrong. Therefore, a helper function is provided in ROM to simplify retention state entry.

6.3 Voltage Regulators (VR)

The MCU features two integrated VRs:

- A 300 µA low quiescent current linear VR for low power sleep regimes.
- A 50 mA buck VR for active regimes and powering external devices.

The application program must choose the optimal regulator for the operating regime. The application program can also adjust the core voltage from the default 1.8 V active state down to the 1.35 V retention state. Lowering core voltage reduces leakage current, but care must be taken not to exceed the maximum clock frequency for that voltage. Refer to the Intel® Quark™ microcontroller D1000 Datasheet for information on power consumption at various frequencies. A helper function has been provided in ROM to safely enter the 1.35 V retention state. Also, the VR control/status register is password protected to prevent accidental misprogramming.

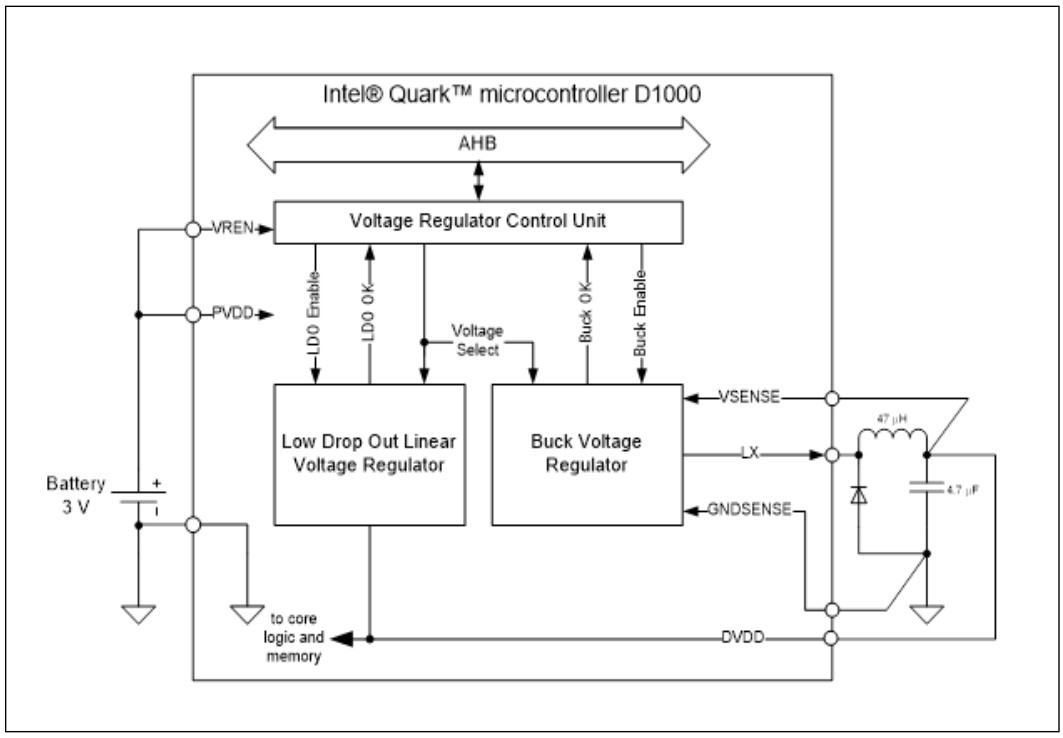
The buck regulator is capable of supplying much more current than the MCU can consume. The excess current can be used to power external devices. The amount of current available to power external devices depends on the maximum current that the application running on the MCU will consume. Refer to the Intel® Quark™ microcontroller D1000 Datasheet for information on power consumption under various operating conditions.

6.3.1 Functional Description

The buck VR is enabled at power-up and after reset. The buck VR must be used in most active processing modes. In standby and retention modes, the 80 µA nominal quiescent current of the buck regulator is much higher than the 2.0 µA and 1.5 µA nominal standby and retention currents. Therefore, a low drop-out linear VR with 0.2 µA nominal quiescent current is provided for these low power modes. However, it is designed for a maximum load current of only 300 µA. Therefore, when the linear VR is selected, the system clock frequency should be set to 125 kHz or less.

Figure 12 shows a schematic diagram of the VRs as they are integrated into the MCU. A memory mapped control/status register in the Voltage Regulator Control Unit is provided for software control of the VRs. Software can change the core voltage or switch between buck and linear VRs using this register. The register is password protected to prevent accidental misprogramming.

Figure 12. Integrated Voltage Regulators



6.3.2 Voltage Regulator Control/Status Register

The Voltage Regulator Control/Status Register is a readable and writable register memory mapped at address 0x8000 001C to 0x8000 001F. It must be accessed as a double word. This register is used to control the VR and read its status.

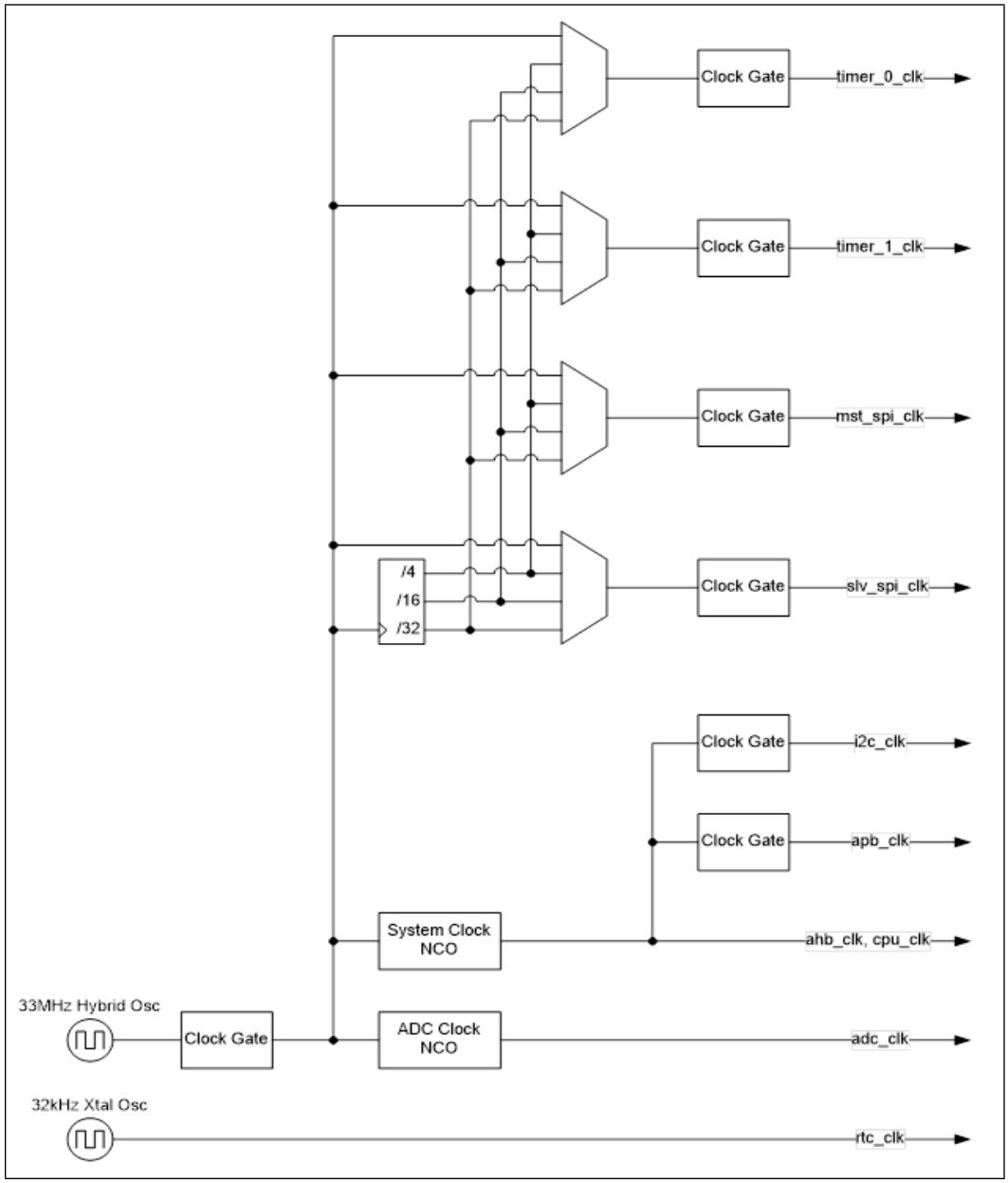
Table 23. Voltage Regulator Control/Status Register

Mnemonic	Access	Bits	Description	Reset
VS	R/W	4:0	Voltage Select – voltage select encoded thusly: 11 = 1.35 V 12 = 1.40 V 13 = 1.50 V 14 = 1.60 V 15 = 1.70 V 16 = 1.80 V	0x10
VSS	R/W	5	Voltage Select Strobe – when '1', the voltage select code on VS is loaded into the voltage regulator. When '0', the voltage regulator holds the last loaded value. WARNING: this signal must only be set to '1' when the switching regulator is selected (VRS == '0'). Also, at least 2 µs must elapse between setting VRS to '0' and setting VSS to '1', and at least 500 ns must elapse between changing VS and setting VSS to '1'. After setting VSS to '1', VRS, VS and VSS must remain stable for at least 1 µs. Finally, VS must not change while VSS is '1' and at least 500 ns must elapse between setting VSS to '0' and changing VRS to '1'.	0x0
VRS	R/W	6	Voltage Regulator Select – when '1', the low current linear regulator is selected. When '0', the high current switching regulator is selected. WARNING: VRS must be set to '0' and remain at '0' from at least 2 µs prior to setting VSS to '1' until at least 500 ns after VSS is set to '0'.	0x0
ROKM	R/W	7	Voltage is in Regulation Mask – when '0', an out of regulation condition will result in system failure due to loss of I/O buffer function. When '1', I/O buffers are masked off such that loss of regulation does not result in system failure. ROK is unaffected by this bit. This bit should be set to '1' during firmware initialization.	0x0
ROK	R	8	Voltage is in Regulation – when '1', this bit indicates that the switching regulator is selected and that the voltage is in regulation. When '0', this bit indicates that the linear regulator is selected or that the voltage is out of regulation.	X
-	-	15:9	Reserved	X
PC	W	23:16	Pass Code – when written along with bits [7:0], a value of 0xC4 will enable the write operation. Any other value inhibits the write operation.	X

6.4 Clock Management

Intel® Quark™ microcontroller D1000 provides the user with fine-grained clock management. A schematic diagram of the clock network is shown in Figure 13.

Figure 13. Schematic diagram of clock network showing branch NCOs, dividers, and gates



6.4.1 Functional Description

All clocks except the real-time clock are derived from the same oscillator. They are pseudo-synchronous meaning that frequencies may differ, but edges align. While signals transferring between clock domains do not require resynchronization, a request-acknowledge handshake is used to ensure that signals are sampled.

The 33 MHz hybrid oscillator has a frequency range from 3.3-33 MHz and a shutdown mode for power savings. It provides the system clock source for the processor, AMBA subsystem, and ADC. It is a hybrid silicon-crystal design providing a flexible clocking solution, which allows the user application program to trade off power consumption, settling time, jitter and frequency accuracy. Features include:

- Glitch-free switching from crystal to silicon oscillator modes and vice-versa.
- Low leakage shutdown mode with glitch free power on.
- Fast start up and reduced power consumption in silicon oscillator mode.
- Fast switching 4, 8, 16 and 32 MHz frequency octaves in silicon oscillator mode.
- Low jitter high precision frequency in crystal oscillator mode.
- 20-33 MHz crystals supported.
- External clock input using crystal pin.

In silicon oscillator mode, power is reduced at the expense of increased jitter and frequency error. The trim range for or the various frequency octaves is ±50% and the PVT variation of untrimmed frequency is around ±35%. Once trimmed, the PVT frequency variation is less than ±2% in the trimmed octave and less than ±4% in untrimmed octaves. The Intel® Quark™ microcontroller D1000 is factory trimmed at 32 MHz. Crystals used with this oscillator must have maximum ESR of 50 ohms at 20 MHz and 80 ohms at 33 MHz. Suitable crystals include the ABM8G series by Abracon (Abracon Corp., 2011).

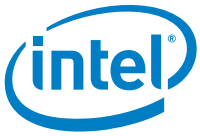
The 32 kHz crystal oscillator provides the clock source for the real-time clock. It also has a shutdown mode for power savings and external clock input using crystal pins. Suitable crystals include the ABS06 series by Abracon (Abracon Corp., 2011).

The system clock numerically controlled oscillator (NCO) provides the clock source for the CPU, AHB, APB, watchdog timer, UART and I^2C. In order to support applications requiring time synchronization with a remote time reference, the NCO provides fine frequency adjustments of ±3.125%. The NCO supports the programmable divide ratios listed in Table 24 (negative frequencies indicate phase reversal). The default phase increment following reset is 0 (NCO bypass). Phase increments that are powers of two generate a jitter free output. Other phase increments result in clock jitter of no more than 3.125% of the crystal oscillator period. Changing the NCO phase increment affects a change in the NCO output frequency beginning with the next output clock edge. A change in phase increment can be immediate (on the next clock edge after writing) or in response to an event (i.e. an interrupt).

Table 24. System Clock NCO Phase Increments and Corresponding Divide Ratios

Φ Inc.	Div. Ratio	Φ Inc.	Div. Ratio	Φ Inc.	Div. Ratio	Φ Inc.	Div. Ratio
1	0.03125	2	0.06250	3	0.09375	4	0.12500
5	0.15625	6	0.18750	7	0.21875	8	0.25000
9	0.28125	10	0.31250	11	0.34375	12	0.37500
13	0.40625	14	0.43750	15	0.46875	16	0.50000
17	-0.46875	18	-0.43750	19	-0.40625	20	-0.37500
21	-0.34375	22	-0.31250	23	-0.28125	24	-0.25000
25	-0.21875	26	-0.18750	27	-0.15625	28	-0.12500
29	-0.09375	30	-0.06250	31	-0.03125	0	1.00000

The ADC NCO generates integer frequency ratios of the input clock from 1:256 to 128:256 in 1:256 steps or undivided pass through. It too has a disable mode for power savings. An 8-bit phase accumulator permits fine frequency adjustments of



±0.390625%. This NCO supports the divide ratios listed in Table 25 (negative frequencies indicate phase reversal). The default phase increment following reset is 1. At a crystal frequency of 32 MHz, the ADC clock frequency programming resolution would be 32 MHz / 256 = 125 kHz. A lower frequency crystal results in proportionally finer frequency resolution. When the ADC is disabled, the ADC clock is also disabled.

Table 25. ADC Clock NCO Phase Increments and Corresponding Divide Ratios (Sheet 1 of 2)

Φ Inc.	Div. Ratio	Φ Inc.	Div. Ratio	Φ Inc.	Div. Ratio	Φ Inc.	Div. Ratio
1	0.00390625	2	0.00781250	3	0.01171875	4	0.01562500
5	0.01953125	6	0.02343750	7	0.02734375	8	0.03125000
9	0.03515625	10	0.03906250	11	0.04296875	12	0.04687500
13	0.05078125	14	0.05468750	15	0.05859375	16	0.06250000
17	0.06640625	18	0.07031250	19	0.07421875	20	0.07812500
21	0.08203125	22	0.08593750	23	0.08984375	24	0.09375000
25	0.09765625	26	0.10156250	27	0.10546875	28	0.10937500
29	0.11328125	30	0.11718750	31	0.12109375	32	0.12500000
33	0.12890625	34	0.13281250	35	0.13671875	36	0.14062500
37	0.14453125	38	0.14843750	39	0.15234375	40	0.15625000
41	0.16015625	42	0.16406250	43	0.16796875	44	0.17187500
45	0.17578125	46	0.17968750	47	0.18359375	48	0.18750000
49	0.19140625	50	0.19531250	51	0.19921875	52	0.20312500
53	0.20703125	54	0.21093750	55	0.21484375	56	0.21875000
57	0.22265625	58	0.22656250	59	0.23046875	60	0.23437500
61	0.23828125	62	0.24218750	63	0.24609375	64	0.25000000
65	0.25390625	66	0.25781250	67	0.26171875	68	0.26562500
69	0.26953125	70	0.27343750	71	0.27734375	72	0.28125000
73	0.28515625	74	0.28906250	75	0.29296875	76	0.29687500
77	0.30078125	78	0.30468750	79	0.30859375	80	0.31250000
81	0.31640625	82	0.32031250	83	0.32421875	84	0.32812500
85	0.33203125	86	0.33593750	87	0.33984375	88	0.34375000
89	0.34765625	90	0.35156250	91	0.35546875	92	0.35937500
93	0.36328125	94	0.36718750	95	0.37109375	96	0.37500000
97	0.37890625	98	0.38281250	99	0.38671875	100	0.39062500
101	0.39453125	102	0.39843750	103	0.40234375	104	0.40625000
105	0.41015625	106	0.41406250	107	0.41796875	108	0.42187500
109	0.42578125	110	0.42968750	111	0.43359375	112	0.43750000
113	0.44140625	114	0.44531250	115	0.44921875	116	0.45312500
117	0.45703125	118	0.46093750	119	0.46484375	120	0.46875000
121	0.47265625	122	0.47656250	123	0.48046875	124	0.48437500
125	0.48828125	126	0.49218750	127	0.49609375	128	0.50000000
129	-0.49609375	130	-0.49218750	131	-0.48828125	132	-0.48437500
133	-0.48046875	134	-0.47656250	135	-0.47265625	136	-0.46875000
137	-0.46484375	138	-0.46093750	139	-0.45703125	140	-0.45312500
141	-0.44921875	142	-0.44531250	143	-0.44140625	144	-0.43750000

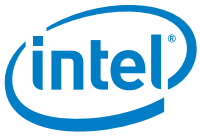
Table 25. ADC Clock NCO Phase Increments and Corresponding Divide Ratios (Sheet 2 of 2)

Φ Inc.	Div. Ratio	Φ Inc.	Div. Ratio	Φ Inc.	Div. Ratio	Φ Inc.	Div. Ratio
145	-0.43359375	146	-0.42968750	147	-0.42578125	148	-0.42187500
149	-0.41796875	150	-0.41406250	151	-0.41015625	152	-0.40625000
153	-0.40234375	154	-0.39843750	155	-0.39453125	156	-0.39062500
157	-0.38671875	158	-0.38281250	159	-0.37890625	160	-0.37500000
161	-0.37109375	162	-0.36718750	163	-0.36328125	164	-0.35937500
165	-0.35546875	166	-0.35156250	167	-0.34765625	168	-0.34375000
169	-0.33984375	170	-0.33593750	171	-0.33203125	172	-0.32812500
173	-0.32421875	174	-0.32031250	175	-0.31640625	176	-0.31250000
177	-0.30859375	178	-0.30468750	179	-0.30078125	180	-0.29687500
181	-0.29296875	182	-0.28906250	183	-0.28515625	184	-0.28125000
185	-0.27734375	186	-0.27343750	187	-0.26953125	188	-0.26562500
189	-0.26171875	190	-0.25781250	191	-0.25390625	192	-0.25000000
193	-0.24609375	194	-0.24218750	195	-0.23828125	196	-0.23437500
197	-0.23046875	198	-0.22656250	199	-0.22265625	200	-0.21875000
201	-0.21484375	202	-0.21093750	203	-0.20703125	204	-0.20312500
205	-0.19921875	206	-0.19531250	207	-0.19140625	208	-0.18750000
209	-0.18359375	210	-0.17968750	211	-0.17578125	212	-0.17187500
213	-0.16796875	214	-0.16406250	215	-0.16015625	216	-0.15625000
217	-0.15234375	218	-0.14843750	219	-0.14453125	220	-0.14062500
221	-0.13671875	222	-0.13281250	223	-0.12890625	224	-0.12500000
225	-0.12109375	226	-0.11718750	227	-0.11328125	228	-0.10937500
229	-0.10546875	230	-0.10156250	231	-0.09765625	232	-0.09375000
233	-0.08984375	234	-0.08593750	235	-0.08203125	236	-0.07812500
237	-0.07421875	238	-0.07031250	239	-0.06640625	240	-0.06250000
241	-0.05859375	242	-0.05468750	243	-0.05078125	244	-0.04687500
245	-0.04296875	246	-0.03906250	247	-0.03515625	248	-0.03125000
249	-0.02734375	250	-0.02343750	251	-0.01953125	252	-0.01562500
253	-0.01171875	254	-0.00781250	255	-0.00390625	0	1.00000000

The APB, I^2C, CPU and AHB clocks (*apb_clk, i2c_clk, cpu_clk* and *ahb_clk* respectively), are at the same frequency. However, I^2C and APB clock are gated. Four of the five peripheral clocks (*timer_1_clk, timer_0_clk, slv_spi_clk,* and *mst_spi_clk*) allow software to select between 1:1, 1:4, 1:16, or 1:32 integer frequency ratios of the AHB clock or gated off entirely. This allows the CPU clock frequency to be adjusted up or down depending on workload without affecting timers or SPI transactions. The clock mux and gate are glitch free.

The following list provides important restrictions to note with respect to clock frequencies:

- *apb_clk* must be running for APB peripherals to generate interrupts.



- $f_{MST\_SPI\_CLK} \leq f_{APB\_CLK}$
- $f_{SLV\_SPI\_CLK} \leq f_{APB\_CLK}$
- $f_{TIMER\_N\_CLK} \leq f_{APB\_CLK}$

6.4.2 Registers

Clock management registers are aligned on 4-byte boundaries and can be accessed as bytes, words, or double words. Bit definitions are given in the following sections.

6.4.2.1 Clock Enable Register

The Clock Enable Register is a readable and writable register memory mapped at address 0x8000 0000 to 0x8000 0003. This register is used to enable and disable various clocks.

Table 26. Clock Enable Register

Mnemonic	Access	Bits	Description	Reset
APB	R/W	0	APB Clock: when '1', the clock is enabled. When '0', the clock is stopped. WARNING: GPIO, UART, and watchdog timer stall when APB clock is stopped. Proper function of I^2C and SPI modules has not been verified. APB timers continue to run, but will not generate an interrupt when APB clock is stopped. The real time clock continues functions normally even when APB clock is stopped.	0x0
SPIM	R/W	1	SPI Master Clock: when '1', the clock is enabled. When '0', the clock is stopped.	0x0
SPIS	R/W	2	SPI Slave Clock: when '1', the clock is enabled. When '0', the clock is stopped.	0x0
I^2C	R/W	3	I2C Clock: when '1', the clock is enabled. When '0', the clock is stopped.	0x0
TMR0	R/W	4	Timer[0] Clock: when '1', the clock is enabled. When '0', the clock is stopped.	0x0
TMR1	R/W	5	Timer[1] Clock: when '1', the clock is enabled. When '0', the clock is stopped.	0x0
WDT	R/W	6	Watchdog Timer: when '1', the clock is enabled. When '0', the clock is stopped.	0x0
NCOD	R/W	7	System NCO Disable: when '0', the system NCO is enabled. When '1', the system NCO is stopped.	0x0
OSCPD	R/W	8	Hybrid Oscillator Power Down: when '0', the oscillator is enabled. When '1', the oscillator is shutdown.	0x0
RTCPD	R/W	9	Real Time Clock Oscillator Power Down: when '0', the oscillator is enabled. When '1', the oscillator is shutdown.	0x1
MSPI	R	10	Master SPI Idle: when '1', indicates that the master SPI is idle and APB and SPI clocks can be disabled. When '0', APB and SPI clocks must remain enabled for proper functioning.	0x0
SSPI	R	11	Slave SPI Idle: when '1', indicates that the slave SPI is idle and APB and SPI clocks can be stopped. When '0', APB and SPI clocks must remain enabled for proper functioning.	0x0
SEC	R/W	12	SEC Enable: when '1', the SEC timer is enabled. When '0', the SEC timer is disabled.	0x1
-	-	14:13	Reserved	X
WS	R/W	15	Watchdog Speedup: when '1', the watchdog counter restart value is 255 regardless of the selected timeout period (used to speed up test times). When '0', watchdog timer functions normally.	0x0

6.4.2.2 Clock Select Register

The Clock Select Register is a readable and writable register memory mapped at address 0x8000 0004 to 0x8000 0007. This register is used to select the clock frequency for the various peripheral clocks.

Table 27. Clock Select Register

Mnemonic	Access	Bits	Description	Reset
SPIS	R/W	1:0	SPI Slave Clock: integer frequency ratio of hybrid oscillator frequency encoded thusly: 0 = 1:1 1 = 1:4 2 = 1:16 3 = 1:32 WARNING: Due to SPI module design limitations, $f_{SLV\_SPI\_CLK} \leq f_{APB\_CLK}$	0x3
-	-	3:2	Reserved	X
SPIM	R/W	5:4	SPI Master Clock: integer frequency ratio of hybrid oscillator frequency encoded thusly: 0 = 1:1 1 = 1:4 2 = 1:16 3 = 1:32 WARNING: Due to SPI module design limitations, $f_{MST\_SPI\_CLK} \leq f_{APB\_CLK}$	0x3
-	-	11:6	Reserved	X
TMR0	R/W	13:12	Timer[0] Clock: integer frequency ratio of hybrid oscillator frequency encoded thusly: 0 = 1:1 1 = 1:4 2 = 1:16 3 = 1:32 WARNING: Due to timer module design limitations, $f_{TIMER\_0\_CLK} \leq f_{APB\_CLK}$	0x3
-	-	15:14	Reserved	X
TMR1	R/W	17:16	Timer[1] Clock: integer frequency ratio of hybrid oscillator frequency encoded thusly: 0 = 1:1 1 = 1:4 2 = 1:16 3 = 1:32 WARNING: Due to timer module design limitations, $f_{TIMER\_1\_CLK} \leq f_{APB\_CLK}$	0x3
-	-	31:18	Reserved	X

6.4.2.3 Clock AHB Register

The Clock AHB Register is a readable and writable register memory mapped at address 0x8000 0008 to 0x8000 000B. This register is used to configure the AHB clock frequency.

Table 28. Clock AHB Register

Mnemonic	Access	Bits	Description	Reset
IMPI	R/W	4:0	Immediate NCO Phase Increment: writing to this bit field initiates an immediate change in the NCO phase increment. Reading this bit field returns the current state. It is encoded thusly: $PI = \begin{cases} 32\frac{ahbFreq}{oscFreq}, & ahbFreq \leq \frac{oscFreq}{2} \\ 0, & ahbFreq = oscFreq \end{cases}$ WARNING: the following restrictions should be considered when programming IMPI: · $f_{MST\_SPI\_CLK} \leq f_{APB\_CLK}$ · $f_{SLV\_SPI\_CLK} \leq f_{APB\_CLK}$ · $f_{TIMER\_N\_CLK} \leq f_{APB\_CLK}$	0x0
-	-	5	Reserved	X
IMFS		7:6	Immediate Oscillator Frequency Select: writing to this bit field initiates an immediate change in the silicon oscillator frequency. Reading this bit field returns the current state. It is encoded thusly: 0 = 32 MHz 1 = 16 MHz 2 = 8 MHz 3 = 4 MHz WARNING: changing oscillator frequency affects the frequency on all clocks except the real time clock.	0x3
EVPI	R/W	12:8	Event NCO Phase Increment: writing to this bit field loads the phase increment to switch to following the next IRQ. Reading this bit field returns the last value written. It is encoded thusly: $PI = \begin{cases} 32\frac{ahbFreq}{oscFreq}, & ahbFreq \leq \frac{oscFreq}{2} \\ 0, & ahbFreq = oscFreq \end{cases}$ WARNING: the following restrictions should be considered when programming EVPI: · $f_{MST\_SPI\_CLK} \leq f_{APB\_CLK}$ · $f_{SLV\_SPI\_CLK} \leq f_{APB\_CLK}$ · $f_{TIMER\_N\_CLK} \leq f_{APB\_CLK}$	0x0
-	-	13	Reserved	X
EVFS	R/W	15:14	Event Oscillator Frequency Select: writing to this bit field loads the silicon oscillator frequency to switch to following the next IRQ. Reading this bit field returns the last value written. It is encoded thusly: 0 = 32 MHz 1 = 16 MHz 2 = 8 MHz 3 = 4 MHz WARNING: changing oscillator frequency affects the frequency on all clocks except the real time clock.	0x3

6.4.2.4 33 MHz Oscillator Configuration Register 0

The 33 MHz Oscillator Configuration Register 0 is a readable and writable register memory mapped at address 0x8000 0010 to 0x8000 0013. This register is used to configure oscillator test parameters.

Table 29. 33 MHz Oscillator Configuration Register 0

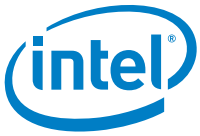
Mnemonic	Access	Bits	Description	Reset
VL	R/W	0	Voltage Level – when '1', bias currents are set for 1.35V supply level. When '0', bias currents are set for 1.8V supply voltage.	0x0
TO	R/W	1	Trim Offset – when '1', a small frequency offset is introduced.When '0', no offset is introduced. This bit should only be set to '1' while trimming. Introducing a small frequency offset during trimming ensures that the trimmed oscillator will have ±2% accuracy over the full process-voltage-temperature range. If best accuracy at ambient temperature is desired, set this bit to '0' while trimming. This bit should always be set to '0' during normal operation.	0x0
-	-	31:2	Reserved – must be 0.	0x0

6.4.2.5 33 MHz Oscillator Configuration Register 1

The 33 MHz Oscillator Configuration Register 1 is a readable and writable register memory mapped at address 0x8000 0014 to 0x8000 0017. This register is used to configure oscillator operating parameters.

Table 30. 33 MHz Oscillator Configuration Register 1

Mnemonic	Access	Bits	Description	Reset
EXO	R/W	0	Enable Crystal Oscillator – when '1', crystal oscillator is enabled.	0x0
ESO	R/W	1	Enable Silicon Oscillator – when '1', silicon oscillator is enabled.	0x1
-	-	2	Reserved	X
MS	R/W	3	Mode Select – when '1', the crystal oscillator output is selected. When '0', the silicon oscillator output is selected.	0x0
BX	R/W	4	Bypass Crystal Oscillator – when '1', an external oscillator can be connected to the XTALI[0] input port. When '0', a crystal can be connected between XTALI[0] and XTALO[0]. This bit is only relevant when EXO == 1.	0x0
ST	R/W	7:5	Startup Trim – trims startup characteristics of silicon oscillator.	0x0
-	-	9:8	Reserved	X
IBT	R/W	12:10	Current Bias Trim – trims current bias in silicon oscillator.	0x0
TCT	R/W	14:13	Temperature Compensation Trim – trims temperature compensation.	0x0
-	-	15	Reserved	X
XOFT	R/W	19:16	Crystal Oscillator Frequency Trim – trims crystal oscillator frequency.	0x0
SOFT	R/W	29:20	Silicon Oscillator Frequency Trim – trims silicon oscillator frequency	0x0
SOL	R	30	Silicon Oscillator Locked – when read '1', the silicon oscillator is stable. When read '0', the silicon oscillator is unstable.	X
XOL	R	31	Crystal Oscillator Locked – when read '1', the crystal oscillator is stable. When read '0', the crystal oscillator is unstable.	X



6.4.2.6 32 kHz Oscillator Configuration Register

The 32 kHz Oscillator Configuration Register is a readable and writable register memory mapped at address 0x8000 0018 to 0x8000 001B. This register is used to configure oscillator parameters.

Table 31. 32 kHz Oscillator Configuration Register

Mnemonic	Access	Bits	Description	Reset
-	-	0	Reserved	X
BX	R/W	1	Bypass Crystal Oscillator – when '1', an external oscillator can be connected to the XTALI[1] input port. When '0', a crystal can be connected between XTALI[1] and XTALO[1].	0x0
-	-	7:2	Reserved	X
-	-	13:8	Reserved – must be 0.	0x0
IBT	R/W	15:14	Current Bias Trim – trims current bias in silicon oscillator. 0 = Normal bias 1 = 50% increase 2 = 25% increase 3 = 75% increase	0x0
-	-	23:16	Reserved – must be 0.	0x0
XOL	R	24	Crystal Oscillator Locked – when read '1', the crystal oscillator is stable. When read '0', the crystal oscillator is unstable.	X
-	-	31:25	Reserved	X

6.5 I/O Pin Function

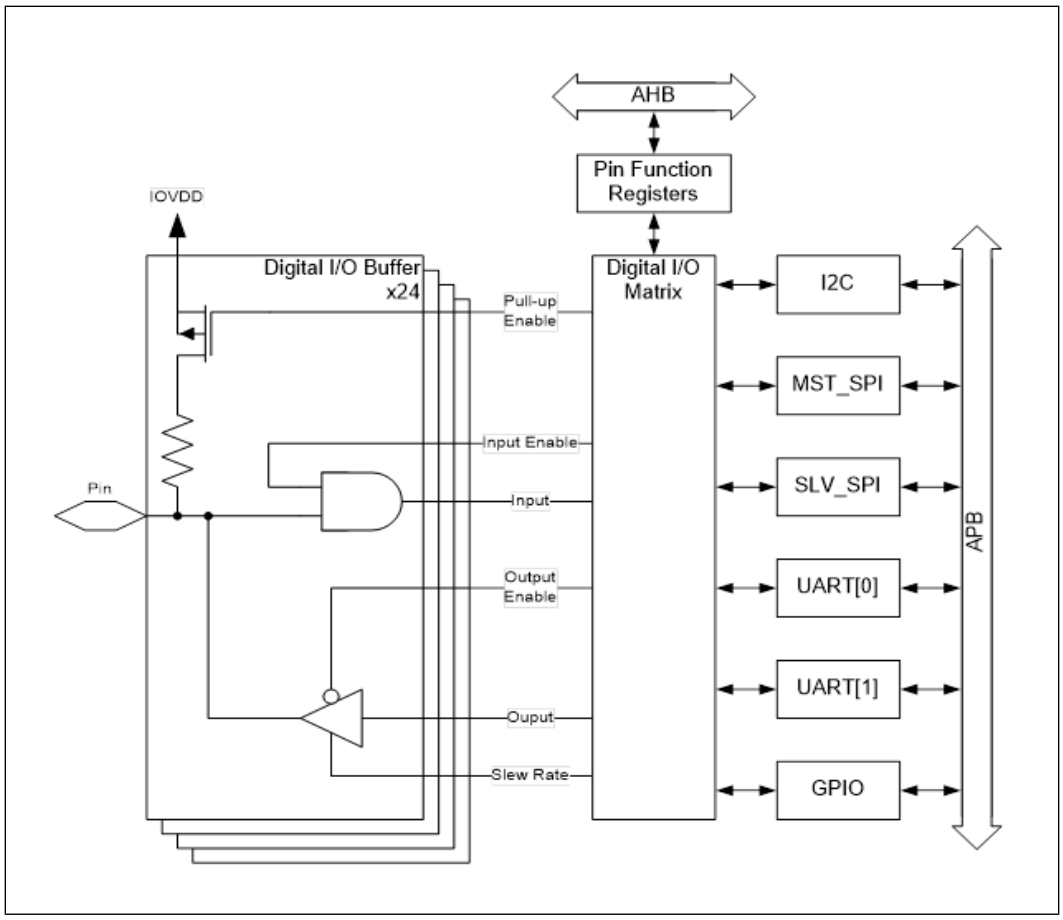
The MCU features 24 multi-function I/O pins. Each pin can be configured as a serial interface signal or a general purpose I/O (GPIO). In addition to digital functions, 19 pins are connected to both an analog comparators and an ADC channel. When a pin is configured for a digital function, analog functions are still possible. When pin is configured for analog functions, digital functions are disabled on that pin and the digital I/O buffer attached to it is in the high impedance state.

6.5.1 Functional Description

Figure 14 shows the architecture of digital pin multiplexing. A digital I/O matrix connects a pin to a serial peripheral or GPIO via an interconnect matrix. The connections are programmed by the CPU using the pin function registers. When an analog function is selected for a pin, the interconnect matrix disables the I/O buffer entirely and sets any digital input associated with that pin to the deasserted state.

Regardless of the programmed pin function, analog functions are always possible on 19 of the 24 pins. This is a very useful wake-up feature. Digital functions require a running clock to detect a change in signal state. However, analog comparators do not need a clock. Therefore, they can be used to wake the MCU up from standby or retention at the start of an incoming transaction. Since wake-up from standby requires only a few microseconds, it can be used to detect a start bit on UART RXD, start signal on I^2C, or SPI slave select. As long as the bit rate is not too high, the peripheral will process the transaction properly.

Figure 14. Architecture of Digital I/O Multiplexing (I/O buffer level shifters not shown)



6.5.2 Registers

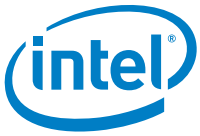
I/O pin function registers are aligned on 4-byte boundaries and can be accessed as bytes, words, or double words. Bit definitions are given in the following sections.

6.5.2.1 I/O Pin Pull-up Enable Register

The I/O Pin Pull-up Enable Register is a readable and writable register memory mapped at address 0x8000 0080 to 0x8000 0083. This register is used to enable I/O pin pull-ups.

Table 32. I/O Pin Pull-up Enable Register

Mnemonic	Access	Bits	Description	Reset
PUE	R/W	23:0	Pin Pull-up Enable: when '1', the Pull-up is enabled.	0x0
-	-	30:24	Reserved	X
SPUE	R/W	31	SEC Pull-up Enable – when '1', the SEC pin Pull-up is enabled.	0x0



6.5.2.2 I/O Pin[7:0] Function Control Register

The I/O Pin[7:0] Function Control Register is a readable and writable register memory mapped at address 0x8000 0084 to 0x8000 0087. This register is used to configure the function of I/O pins 0-7.

Table 33. I/O Pin[7:0] Function Control Register

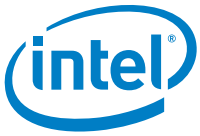
Mnemonic	Access	Bits	Description	Reset
PF0	R/W	1:0	I/O pin 0 Function: I/O pin 0 function encoded thusly: 0 = AI[0] 1 = GPIO[0] 2 = MST_M2SS[0] 3= AI[0]	0x3
-	-	3:2	Reserved	X
PF1	R/W	5:4	I/O pin 1 Function: I/O pin 1 function encoded thusly: 0 = AI[1] 1 = GPIO[1] 2 = MST_M2SS[1] 3= AI[1]	0x3
-	-	7:6	Reserved	X
PF2	R/W	9:8	I/O pin 2 Function: I/O pin 2 function encoded thusly: 0 = AI[2] 1 = GPIO[2] 2 = MST_M2SS[2] 3 = AI[2]	0x3
-	-	11:10	Reserved	X
PF3	R/W	13:12	I/O pin 3 Function: I/O pin 3 function encoded thusly: 0 = AI[3] 1 = GPIO[3] 2 = MST_M2SS[3] 3 = AI[3]	0x3
-	-	15:14	Reserved	X
PF4	R/W	17:16	I/O pin 4 Function: I/O pin 4 function encoded thusly: 0 = AI[4] 1 = GPIO[4] 2 = Disabled 3 = AI[4]	0x3
-	-	19:18	Reserved	X
PF5	R/W	21:20	I/O pin 5 Function: I/O pin 5 function encoded thusly: 0 = AI[5] 1 = GPIO[5] 2 = MST_S2M_SS 3 = AI[5]	0x3
-	-	23:22	Reserved	X
PF6	R/W	25:24	I/O pin 6 Function: I/O pin 6 function encoded thusly: 0 = AI[6] 1 = GPIO[6] 2 = SCL 3 = AI[6]	0x3
-	-	27:26	Reserved	X
PF7	R/W	29:28	I/O pin 7 Function: I/O pin 7 function encoded thusly: 0 = AI[7] 1 = GPIO[7] 2 = SDA 3 = AI[7]	0x3
-	-	31:30	Reserved	X

6.5.2.3 I/O Pin[15:8] Function Control Register

The I/O Pin[15:8] Function Control Register is a readable and writable register memory mapped at address 0x8000 0088 to 0x8000 008B. This register is used to configure the function of I/O pins 8-15.

Table 34. I/O Pin[15:8] Function Control Register

Mnemonic	Access	Bits	Description	Reset
PF8	R/W	1:0	I/O pin 8 Function: I/O pin 8 function encoded thusly: 0 = AI[8] 1 = GPIO[8] 2 = SLV_M2SC 3 = AI[8]	0x3
-	-	3:2	Reserved	X
PF9	R/W	5:4	I/O pin 9 Function: I/O pin 9 function encoded thusly: 0 = AI[9] 1 = GPIO[9] 2 = SLV_M2SD 3 = AI[9]	0x3
-	-	7:6	Reserved	X
PF10	R/W	9:8	I/O pin 10 Function: I/O pin 10 function encoded thusly: 0 = AI[10] 1 = GPIO[10] 2 = SLV_S2MD 3 = AI[10]	0x3
-	-	11:10	Reserved	X
PF11	R/W	13:12	I/O pin 11 Function: I/O pin 11 function encoded thusly: 0 = AI[11] 1 = GPIO[11] 2 = SLV_M2SS 3 = AI[11]	0x3
-	-	15:14	Reserved	X
PF12	R/W	17:16	I/O pin 12 Function: I/O pin 12 function encoded thusly: 0 = AI[12] 1 = GPIO[12] 2 = TXD[1] 3 = AI[12]	0x3
-	-	19:18	Reserved	X
PF13	R/W	21:20	I/O pin 13 Function: I/O pin 13 function encoded thusly: 0 = AI[13] 1 = GPIO[13] 2 = RXD[1] 3 = AI[13]	0x3
-	-	23:22	Reserved	X
PF14	R/W	25:24	I/O pin 14 Function: I/O pin 14 function encoded thusly: 0 = AI[14] 1 = GPIO[14] 2 = RTS[1] 3 = AI[14]	0x3
-	-	27:26	Reserved	X



PF15	R/W	29:28	I/O pin 15 Function: I/O pin 15 function encoded thusly: 0 = AI[15] 1 = GPIO[15] 2 = CTS[1] 3 = AI[15]	0x3
-	-	31:30	Reserved	X

6.5.2.4 I/O Pin[23:16] Function Control Register

The I/O Pin[23:16] Function Control Register is a readable and writable register memory mapped at address 0x8000 008C to 0x8000 008F. This register is used to configure the function of I/O pins 16-23.

Table 35. I/O Pin[23:16] Function Control Register

Mnemonic	Access	Bits	Description	Reset
PF16	R/W	1:0	I/O pin 16 Function: I/O pin 16 function encoded thusly: 0 = AI[16] 1 = GPIO[16] 2 = MST_M2SC 3 = AI[16]	0x3
-	-	3:2	Reserved	X
PF17	R/W	5:4	I/O pin 17 Function: I/O pin 17 function encoded thusly: 0 = AI[17] 1 = GPIO[17] 2 = MST_M2SD 3 = AI[17]	0x3
-	-	7:6	Reserved	X
PF18	R/W	9:8	I/O pin 18 Function: I/O pin 18 function encoded thusly: 0 = AI[18] 1 = GPIO[18] 2 = MST_S2MD 3 = AI[18]	0x3
-	-	11:10	Reserved	X
PF19	R/W	13:12	I/O pin 19 Function: I/O pin 19 function encoded thusly: 0 = AI[19] 1 = GPIO[19] 2 = JTAG TDO 3 = Disabled	0x3
-	-	15:14	Reserved	X
PF20	R/W	17:16	I/O pin 20 Function: I/O pin 20 function encoded thusly: 0 = TXD[0] 1 = GPIO[20] 2 = JTAG TRST_N 3 = Disabled	0x3
-	-	19:18	Reserved	X
PF21	R/W	21:20	I/O pin 21 Function: I/O pin 21 function encoded thusly: 0 = RXD[0] 1 = GPIO[21] 2 = JTAG TCK 3 = Disabled	0x3
-	-	23:22	Reserved	X

PF22	R/W	25:24	I/O pin 22 Function: I/O pin 22 function encoded thusly: 0 = RTS[0] 1 = GPIO[22] 2 = JTAG TMS 3 = Disabled	0x3
-	-	27:26	Reserved	X
PF23	R/W	29:28	I/O pin 23 Function: I/O pin 23 function encoded thusly: 0 = CTS[0] 1 = GPIO[23] 2 = JTAG TDI 3 = Disabled	0x3
-	-	31:30	Reserved	X

6.5.2.5 I/O Pin Slew Rate Control Register

The I/O Pin Slew Rate Control Register is a readable and writable register memory mapped at address 0x8000 0090 to 0x8000 0093. This register can be used to configure I/O pin slew rate.

Table 36. Pin Slew Rate Control Register

Mnemonic	Access	Bits	Description	Reset
PSR	R/W	23:0	I/O Pin Slew Rate: when '1', the output slew rate is fast. When '0', the output slew rate is slow.	0x0
-	-	31:24	Reserved	X

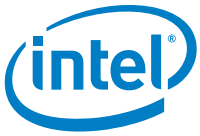
§ §

7.0 Central Processing Unit (CPU)

The MCU features a 32-bit Harvard CISC CPU. Other features and benefits include:

- Single-issue, in-order 5 stage pipeline.
- Backward taken, forward not taken branch prediction.
- 32-bit data memory width.
- 128-bit instruction memory width.
- Single cycle barrel shifter.
- Two cycle multiplier.
- Multi-cycle hardware divider.
- JTAG debugging.
- Integrated
- PIC.
- Integrated 32-bit timer.

For more details on the CPU and it's advanced programmable interrupt controller (APIC), refer to the Intel® Quark™ microcontroller D1000 Programmer's Reference Manual.



§ §

8.0 JTAG Debugger

The MCU features an integrated JTAG debug controller with IEEE 1149.1 compatible behavior. Other features and benefits include:

- Start/stop run control
- Single step
- Instruction and data breakpoint registers
- R/W peripheral, memory and core register content
- Stream data transfer capability

8.1 Functional Description

The MCU has a standard 5-pin IEEE 1149.1 compatible JTAG interface to the TAP controller with the exception that immediately following reset the JTAG pins are disabled. JTAG is enabled in the following cases:

- Flash is found out to be erased or corrupted
- Enable by system configuration stored in flash
- Enable by user application
- SEC pin is high or floating

The SEC pin can be used to securely force JTAG to be enabled if the application developer is reprogramming, accidentally locked out or returned units need failure analysis. If the SEC pin is found to be high or floating following a reset, the CPU will check the User Configuration and erase the flash if the CPU is configured to do so before enabling JTAG.

The TAP_SEL pin must be strapped to a logic low in order for the debug TAP controller to be accessible. The debug TAP controller supports GDB software debug as well as flash memory programming. Its IDCODE register value is 0x053A 1013.

8.2 TAP Registers

This section describes the JTAG accessible registers in the debug TAP controller.

Note: The following registers are not memory mapped and are not accessible by the CPU or any software running on it.

8.2.1 Instruction Register (IR)

The TAP Controller features a 4-bit Instruction Register (IR) with IEEE 1149.1 compatible behavior.

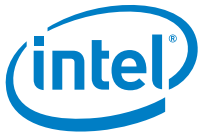


Table 37. Instructions

Name	Code	DR Used
IDCODE	0x2	DR_DEVICE_ID
BYPASS	0xF	DR_BYPASS
CMD	0x4	DR_CMD
ADDR	0x5	DR_ADDR
WR_DATA	0x7	DR_WR_DATA
RD_DATA	0x8	DR_RD_DATA
REQ_STAT	0x9	DR_REQ_STAT
STREAM_TRANS	0xA	DR_STREAM_TRANS

8.2.2 Data Registers (DRs)

The debug controller features the data registers listed in table above. These are described in the following sections.

8.2.2.1 Device Identification Data Register (DR_DEVICE_ID)

This 32-bit register is intended to provide the debug tool with the TAP Controller identification code to distinguish this TAP controller from others in a JTAG chain.

Table 38. Data Register

Mnemonic	Access	Bits	Description	Default
ID	R	31:4	Identification Code. For debugging purposes, the field is loaded by constant code 0x053A1013.	0x053A1013
MBZ	W	3:1	Must Be Zero. For debugging purposes these bits are always cleared.	0x0
MBO	W	0	Must Be One. In accordance with the IEEE Std 1149.1, the bit must be always 1.	0x1

8.2.2.2 Bypass Data Register (DR_BYPASS)

This register is described as follows:

- Consists of a single shift-register stage
- Functions in accordance to IEEE Std 1149.1
- Loads at 0 on the rising edge of TCK in the DR-Capture TAP state

8.2.2.3 Command Data Register (DR_CMD)

This register selects the type of debug command (read or write).

Table 39. Command Data Register

Mnemonic	Access	Bits	Description	Default
RSV	-	1:2	Reserved. Must be zero.	0x0
WR	R/W	0	Write. Determines a type of a transaction on the DebugBus: 0 - read 1 - write	0x0

8.2.2.4 Address Data Register (DR_ADDR)

This register specifies the address for the debug command.

Table 40. Address Data Register

Mnemonic	Access	Bits	Description	Default
Addr	R/W	11:0	Address. Determines an address of a transaction on the DebugBus.	0x0

8.2.2.5 Write Data Register (DR_WR_DATA)

This register specifies the data for a write debug command.

Table 41. Write Data Register

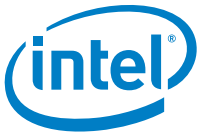
Mnemonic	Access	Bits	Description	Default
Wr_Data	R/W	31:0	Write Data. Determines 32-bit data value of a write transaction on the DebugBus	0x0

8.2.2.6 Read Data Register (DR_RD_DATA)

This register returns the data from a read debug command.

Table 42. Read Data Register

Mnemonic	Access	Bits	Description	Default
Rd_Data	R	31:0	Read Data. Represents 32-bit data value latched during the last read transaction on the DebugBus.	0x0



8.2.2.7 Request Status Data Register (DR_REQ_STAT)

This register returns the status of a debug command.

Table 43. Request Status Data Register

Mnemonic	Access	Bits	Description	Default
Status	R/W	1:0	State of the DebugBus Adapter FSM are: · "00" – idle state · "01" – start of transaction · "10" – waiting for request acknowledgment · "11" – waiting for read request data The Parallel Input value reflects the current value of the FSM state register. The Parallel Output value determines the next state of the FSM after passing through the DR-Update TAP state. Before initiating a transaction on the DebugBus, it is necessary to make sure that previous transaction has been completed as follows: · FSM is in idle state · The data shifted-out from the DR_REQ_STAT must be "00" If so, shifting in the code "01" (at the start of the transaction) changes the FSM state accordingly and then launches a transaction on the DebugBus. If the DR_REQ_STAT shifted-out value is not "00", the previous transaction is still in progress, and you should wait for the state to become idle. Most likely this means some issues on the DebugBus, as the duration of the transaction is usually much less than 1 TCK clock period.	0x0

8.2.2.8 Stream Transfer Data Register (DR_STREAM_TRANS)

This register specifies write data and returns read data for a stream transfer command.

Table 44. Stream Transfer Data Register

Mnemonic	Access	Bits	Description	Default
DATA	R/W	31:0	Specifies write data when written. Returns read data when read.	0x0
REQ_STAT	R/W	33:32	Specifies value to be loaded when written. Returns current value when read.	0x0

8.3 Debug Controller Register

This section describes the debug controller registers that are accessed using TAP commands.

8.3.1 Device Identification Register (DevID)

This register occupies address 0x000 and returns the device identification number.

Table 45. Device Identification Register

Mnemonic	Access	Bits	Description	Default
ID	RO	31:0	**Identifier**. Device Identifier of the NC Debug Controller's instance within SOC. Read-only (RO).	0x8086 DEBC

8.3.2 Build Identification Register (BldID)

This register occupies address 0x004 and returns the BCD encoded year-month-day-version code of the device.

Table 46. Build Identification Register

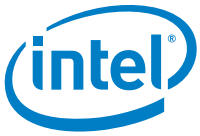
Mnemonic	Access	Bits	Description	Default
VER	RO	7:0	Version. A BCD-coded intra-day version number assigned to Debug Controller in RTL implementation.	0x00
DAY	RO	15:8	Day. A BCD-coded "day" component of current Debug Controller RTL build's date.	0x17
MONTH	RO	23:16	Month. A BCD-coded "month" component of current Debug Controller RTL build's date.	0x02
YEAR	RO	31:24	Year. A BCD-coded "year" component of current Debug Controller RTL build's date.	0x12

8.3.3 Debug Identification Register (DbgID)

This register occupies address 0x008 and returns the version code of the debug controller specification.

Table 47. Debug Identification Register

Mnemonic	Access	Bits	Description	Default
SPEC	RO	0:11	**Specification Version.** Always returns zero.	0x0
Reserved	RAZ	12:31	Reserved. Reserved RO bit field. Must be zero for reads. Writes are ignored.	0x0



8.3.4 Debug Control and Status Register (DbgCSR)

This register occupies address 0x100 and returns debug controller status.

Table 48. Debug Control and Status Register

Mnemonic	Access	Bits	Description	Default
BrkCause	RO	11:0	Break Cause. If NC is in Debug Mode (bits DbgMode and Halted are set), bits [11:8] reflect BreakID of the last redirection to the Debug Mode: 0x1 - Reset Break (redirection to the DebugMode upon reset sequence completion if DbgBrkER.RstOffBE = 1); 0x4 - DRx Breakpoint (if implemented); 0x6 - ICE Single Step (upon DbgCmdR.SStep command completion); 0x7 - Software Breakpoint (INT3) (if enabled by DbgHltER.INT3_DE bit); 0x8 - ICE Forced (as a result of DbgCmdR.DbgHlt command); 0x9 - Core Failure. All other values are illegal. Bits [7:0] are reserved.	0x0
-	RAZ	16:12	Reserved	0x0
IntDsbl	R/W	17	Interrupt Disable. If 1, EPU reaction on IRQs is disabled, and this setting overrides any other IRQ enabling states inside NC.	0x0
-	RAZ	18	Reserved	0x0
FetchPostpone	R/W	19	Fetch Postpone. If 1, the Pipeline after restart postpones instruction fetching and transits to a halted state identical to the one upon HLT instruction completion. The CPU stays in this state until the first IRQ is signaled to the EPU (unless interrupts are disabled). If 0, the Pipeline restarts as usual, with instruction fetching from the address pointed to by EIP.	0x0
Reserved	RAZ	25:20	Reserved	0x0
EpuPending	RO	26	EPU Pending Status. If 1, EPU is stopped with pending IRQ under processing.	0x0
DbgFailure	RO	27	Debug Controller Failure. If 1, violation of the Debug Controller access rules has occurred. The bit can be cleared by DbgCmdR.ClrFail command.	0x0
CoreFailure	RO	28	Core Failure	0x0
Restarted	RO	29	Restarted	0x0
Halted	RO	30	Halted	0x0
DbgMode	RO	31	Debug Mode	0x0

8.3.5 Debug Command Register (DbgCmdR)

This register occupies address 0x104 and is used to issue debug controller commands.

Table 49. Debug Command Register

Mnemonic	Access	Bits	Description	Default
Reserved	RAZ	24:0	Reserved	0x0
ClrFail	RAZ/ W1TP	25	Clear Failure	0x0
SStep	RAZ/ W1TP	26	Single Step	0x0
-	RAZ	29:27	Reserved	0x0
Restart	RAZ/ W1TP	30	Restart	0x0
DbgHlt	RAZ/ W1TP	31	Debug Halt	0x0

8.3.6 Debug Break Event Status Register (DbgEvtSR)

This register occupies address 0x108 and is used to return break event status.

Table 50. Debug Break Event Status Register

Mnemonic	Access	Bits	Description	Default
-	RAZ	30:0	Reserved.	0x0
RstOff	RAZ	31	Reset Off.	0x0

8.3.7 Debug Break Enable Register (DbgBrkER)

This register occupies address 0x10C and is used to enable break conditions.

Table 51. Debug Break Enable Register

Mnemonic	Type	Bits	Description	Default
-	RAZ	23:0	Reserved.	0x0
INT3_BE	RAO	24	INT3 Break Enable. Hardwired to 1 as INT3 is always a break event (as a software exception and as a halting event).	0x1
-	RAZ	30:25	Reserved	0x0
RstOffBE		31		Reset Off Break Enable.

8.3.8 Debug Halt Enable Register (DbgHltER)

This register occupies address 0x110 and is used to enable halt conditions.

Table 52. Debug Halt Enable Register

Mnemonic	Type	Bits	Description	Default
-	RAZ	23:0	Reserved.	0x0
INT3_DE	R/W	24	INT3 Debug Halt Enable. Enables transition of NC to Debug Mode upon INT3 instruction retirement.	0x0
-	RAZ	30:25	Reserved	0x0
RstOffDE	RAO	31	Reset Off Debug Halt Enable.	0x1

8.3.9 Power and Reset Control Register (PwrRstCR)

This register occupies address 0x114 and is used to generate resets.

Table 53. Power and Reset Control Register

Mnemonic	Type	Bits	Description	Default
-	RAZ	1:0	Reserved	0x0
HReset	RW	2	Core Hard Reset. It asserts also internal resets for all internal NC units (Pipeline, EPU, Memory Interface etc). This is reflected by the reset status bits below.	0x1
-	RAZ	7:3	Reserved	0x0
PipeRst	RW	8	Pipeline Reset	0x1
MemIfRst	RW	9	Memory Interface Reset	0x1
EpuRst	RO	10	EPU Reset	0x1
-	RAZ	11:15	Reserved	0x0
PipeRstSts	RO	16	Pipeline Reset Status. Reflects state of the internal Pipeline reset signal.	0x1
MemIfRstSts	RO	17	Memory Interface Reset Status. Reflects state of the internal MemIf reset signal.	0x1
EpuRstSts	RO	18	EPU Reset Status. Reflects state of the internal EPU reset signal.	0x1
MiscRstSts	RO	19	Miscellaneous Units Reset Status. Reflects state of the internal reset signal for miscellaneous NC units (including APIC if any).	0x1
CacheRstSts	RO	20	Cache Reset Status. While the bit is 1, cache reset sequence is in progress, and Pipeline reset de-assertion is postponed.	0x1
-	RAZ	21:31	Reserved	0x0

8.3.10 EAX Register (EAX)

This register occupies address 0x200 and is used to access the CPU's EAX register.

8.3.11 ECX Register (ECX)

This register occupies address 0x204 and is used to access the CPU's ECX register.

8.3.12 EAX Register (EDX)

This register occupies address 0x208 and is used to access the CPU's EDX register.

8.3.13 EBX Register (EBX)

This register occupies address 0x20C and is used to access the CPU's EBX register.

8.3.14 ESP Register (ESP)

This register occupies address 0x210 and is used to access the CPU's ESP register.

8.3.15 EBP Register (EBP)

This register occupies address 0x214 and is used to access the CPU's EBP register.

8.3.16 ESI Register (ESI)

This register occupies address 0x218 and is used to access the CPU's ESI register.

8.3.17 EDI Register (EDI)

This register occupies address 0x21C and is used to access the CPU's EDI register.

8.3.18 EIP Register (EIP)

This register occupies address 0x220 and is used to access the CPU's EIP register.

8.3.19 EAX Register (EFLAGS)

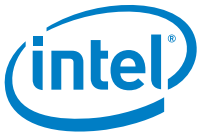
This register occupies address 0x224 and is used to access the CPU's EAX register.

8.3.20 IDTR Low Register

This register occupies address 0x2C0 and is used to access the lower 32 bits of the CPU's IDTR register.

8.3.21 IDTR High Register

This register occupies address 0x2C4 and is used to access the upper 32 bits of the CPU's IDTR register.



8.3.22 IRQ Sending Register (IRQ_SEND)

This register occupies address 0x300 and is used to generate interrupt requests.

Table 54. IRQ Sending Register

Mnemonic	Type	Bits	Description	Default
IRQ_SET	RAZ/W1TP	15:0	IRQ signals vector. Writing of 1 in certain bit position asserts an appropriate APIC's input IRQ line for 1 clock cycle.	0x0
ID_EXTRA	RAZ/WO	23:16	Extra IRQ Identification Number. The number of an IRQ line which is going to be signalled in addition to ones controlled by the IRQ_SET.	0x0
-	RAZ	29:24	Reserved	0x0
SEND_EXTRA	RAZ/W1TP	30	Send Extra IRQ. Writing of 1 asserts an IRQ line with a number specified by the ID_EXTRA field for 1 clock cycle.	0x0
NMI	RAZ/W1TP	31	NMI signal. Writing of 1 asserts APIC's NMI input for 1 clock cycle.	0x0

8.3.23 Memory Transfer Command Register (MemTransCmd)

This register occupies address 0x304 and is used to issue transfer commands.

Table 55. Memory Transfer Command Register

Mnemonic	Type	Bits	Description	Default
Stop	RAZ/W1TP	0	Stop request for the transfer being in progress.	0x0
Start	R/W	1	Writing of 1 starts memory transfer. While the transfer is in progress, the bit is set. When the transfer is completed (with or without errors), the bit is cleared.	0x0
Write	R/W	2	Transaction type parameter. 0 - read memory transfer 1- write memory transfer	0x0
Multi	R/W	3	Multi-cycle transfer. 0 - single-cycle transfer 1 - multi-cycle transfer.	0x0
WrBE	R/W	7:4	Write Byte Enable	0x0
Incr	R/W	8	Increment. 0 - all transactions use the same address (from MemTransAddr register). 1 - address is incremented by 4 at the end of each transfer cycle (MemTransAddr register content is treated as start address).	0x0
-	RAZ	14:9	Reserved	0x0
BufLock	RO	15	Buffer Lock Status. If 0, data buffer is unlocked, and DebugBus master may access MemTransWrData or MemTransRdData registers to read/write data from/to the buffer. If 1, data buffer is locked, the Memory Transfer Machine owns the buffer, and access to MemTransWrData or MemTransRdData is prohibited.	0x0
WaitAck	RO	16	Waiting Acknowledge. The Memory Transfer Machine is waiting for ACK signal.	0x0
WaitDV	RO	17	Waiting Data Valid. For memory read transfer, the Memory Transfer Machine is waiting for Rd_Data_Vd signal.	0x0
Err	RO	18	Error status. Memory transfer is completed with an error.	0x0

Buf_Ovflow	RO	19	Buffer Overflow status. If 1, there was a buffer overflow event during the last memory transfer, i.e., there was a writing to the MemTransWrData register while the data buffer is locked.	0x0
Buf_Undrun	RO	20	Buffer Underrun status. If 1, there was a buffer underrun event during the last memory transfer, i.e., there was a reading from the MemTransRdData register while the data buffer is locked.	0x0
-	RAZ	31:21	Reserved	0x0

8.3.24 Memory Transfer Address register (MemTransAddr)

This register occupies address 0x308 and is used to set the transfer command address.

Table 56. Memory Transfer Address Register

Mnemonic	Type	Bits	Description	Default
-	RAZ	1:0	Reserved	0x0
Addr	R/W	31:2	Address. Represents bits [31:2] of the start address for a memory transfer. Together with MBZ field forms full start address value aligned on DWORD boundary.	0x0

8.3.25 Memory Transfer Write Data register (MemTransWrData)

This register occupies address 0x30C and is used to set the write transfer data.

Table 57. Memory Transfer Write Data Register

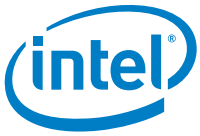
Mnemonic	Type	Bits	Description	Default
Data	R/W	31:0	The register is used as a data buffer for write memory transfers. It contains the value used as a data value during the next write transaction. Access to the register is allowed when it is unlocked, i.e., if MemTransCmd[BufLock] = 0.	0x0

8.3.26 Memory Transfer Read Data register (MemTransRdData)

This register occupies address 0x310 and is used to retrieve the read transfer data.

Table 58. Memory Transfer Read Data Register

Mnemonic	Type	Bits	Description	Default
Data	RO	31:0	The register is used as a data buffer for read memory transfers. It contains the value latched during the last read transaction. Access to the register is allowed when it is unlocked, i.e., if MemTransCmd[BufLock] = 0.	0x0



8.3.27 Memory Transfer Cycle Number Register (MemTransCycleNum)

This register occupies address 0x314 and is used to set the number of data items transferred.

Table 59. Memory Transfer Cycle Number Register

Mnemonic	Type	Bits	Description	Default
CycleNum	R/W	15:0	Cycles Number. The value determines the number of partial memory access cycles during the memory multi-cycle transfer (if MemTransCmd[Multi] = 1). Zero value corresponds to 2^16 = 65536 cycles. During single-cycle memory transfers (if MemTransCmd[Multi] = 0) content of the register is ignored.	0x0
-	RAZ	31:16	Reserved	0x0

8.3.28 Memory Transfer Last Address Register (MemTransLastAddr)

This register occupies address 0x318 and is used to read the last transfer address.

Table 60. Memory Transfer Last Address Register

Mnemonic	Type	Bits	Description	Default
-	RAZ	1:0	Reserved	0x0
Addr	RO	31:2	Address. Represents bits [31:2] of the last transfer cycle's address. Forms a full address value together with the MBZ field.	0x0

8.3.29 Memory Transfer Cycle Downcounter Register (MemTransDowncount)

This register occupies address 0x31C and is used to read the current transfer cycle.

Table 61. Memory Transfer Cycle Downcounter Register

Mnemonics	Type	Bits	Description	Default
CycleRemain	RO	15:0	Cycles Reminder. The register reflects the value of the transfer cycles downcounter, i.e., it returns a number of partial memory access cycles the machine has to carry out to complete the transfer. Zero value corresponds to zero cycles (all transfer cycles have been completed).	0x0
-	RAZ	31:1	Reserved	0x0

8.4 Break Control/Status Registers

8.4.1 Debug Address Register

Table 62. Debug Address Register

Mnemonic	Type	Bits	Description	Default
BRKPT_ADDR	RW	31:0	Breakpoint Address. The field holds 32-bit address of the corresponding breakpoint: DR0 – Breakpoint 0; DR1 – Breakpoint 1; DR2 – Breakpoint 2; DR3 – Breakpoint 3.	0x00

8.4.2 Debug Status Register

Table 63. Debug Status Register

Mnemonics	Type	Bits	Description	Default
B3:B0	RO/ WOTC	3:0	B3:B0 flags - Breakpoint n Condition Detected. If set, indicates that its associated breakpoint condition was met at least once after last clearing. These flags are set if the condition described for each breakpoint by the LEN*n*, and RW*n* flags in debug control register DR7 is true. They may or may not be set if the breakpoint is not enabled by the Gn flags in register DR7. Therefore on an ABP, a debug handler should check only those B0-B3 bits which correspond to enabled breakpoints. The field might be cleared by writing 0s. Writing 1s has no effect.	0x00
RSV	RO	11:4	Reserved. Returns 0b11111111 on read.	0xFF
RSV	RAZ	15:12	Reserved. Returns 0s on read.	0x00
RSV	RO	31:16	Reserved. Returns 1s on read.	0xFFFF

8.4.3 Debug Control Register

Table 64. Debug Control Register

Mnemonics	Type	Bits	Description	Default
RSV	RAZ	0, 2, 4,6	Reserved. Must be 0 for writes.	0x00
G0, G1, G2, G3	RW	1, 3, 5, 7	G0:G3 flags – Global Breakpoint Enable. Enables (when set) the breakpoint condition for the associated breakpoint. When a breakpoint condition is detected and its associated G*n* flag is set, a debug exception is generated.	0x00
RSV	RO	12:8	Reserved. Returns 0B00000100 on read.	0x0B00100
RSV	RAZ	13	Reserved. Returns 0B00000100 on read.	0x00

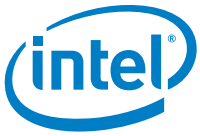


Table 64. Debug Control Register

Mnemonics	Type	Bits	Description	Default
RSV	RAZ	15:14	Reserved. Returns 0B00000100 on read.	0x0B00
RW0, RW1, RW2, RW3	RW	16, 17, 20, 21, 24, 25, 28, 29	Read/Write fields. Specifies the breakpoint condition for the corresponding breakpoint. The processor interprets the RW*n* bits the same as for the Intel386™ and Intel486™ processors, which is as follows: 00 — Break on instruction execution only. 01 — Break on data writes only. 10 — Undefined. 11 — Break on data reads or writes but not instruction fetches.	0x0B00
LEN0, LEN1, LEN2, LEN3	RW	18, 19, 22, 23, 26, 27, 30, 31	Length fields. Specify the size of the memory location at the address specified in the corresponding breakpoint address register (DR0 through DR3). These fields are interpreted as follows: 00 — 1-byte length. 01 — 2-byte length. 10 — Undefined. 11 — 4-byte length. If the corresponding RW*n* field in register DR7 is 00 (instruction execution), then the LEN*n* field should also be 00. The effect of using other lengths is undefined	0x0B00

§ §

9.0 Peripherals

9.1 Flash Controller

The MCU features a flash controller that provides the user program with a method to modify the content of flash. Operations that can be performed are:

- Data page erase
- Instruction page erase
- Mass erase
- Erase reference cell
- Data write
- Instruction write

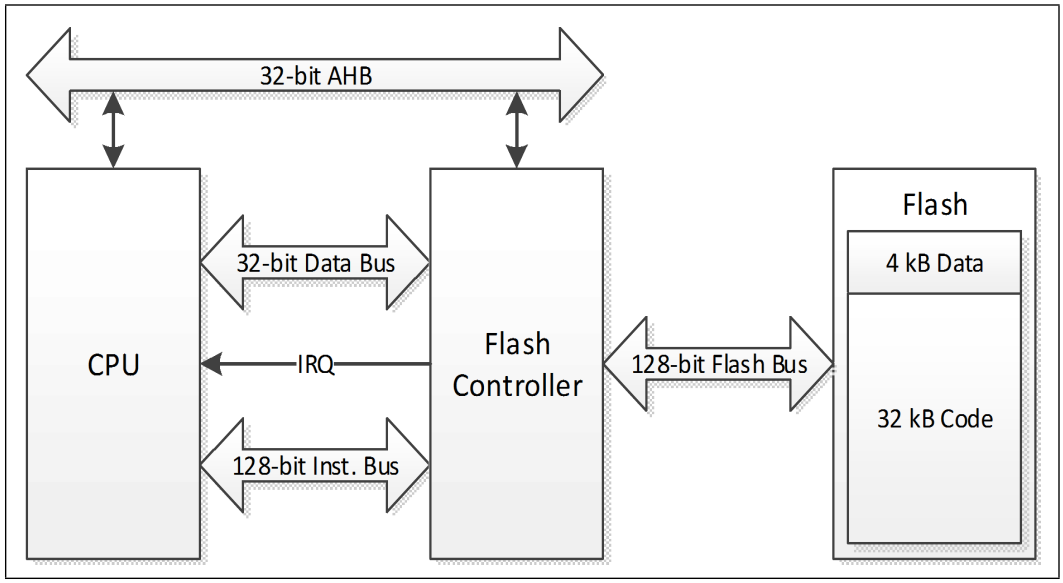
In addition to modifying the content of flash, the user program must set the flash wait states when changing CPU frequency. The rules for setting wait states are:

- Below 6.7 MHz – zero wait state for single cycle read at processor clock frequency.
- 6.7 MHz – 20 MHz – one wait state minimum for two cycles read at processor clock frequency.
- Above 20 MHz – two wait states minimum for three cycles read at processor clock frequency.
- Increasing frequency – increase wait states first.
- Decreasing frequency – decrease wait states last.

9.1.1 Functional Description

Figure 15 below shows the architecture of the flash controller. The CPU programs the flash controller via AHB. The flash controller can interrupt the CPU when a flash operation completes. Instructions are fetched over 128-bit busses between CPU and flash controller and between flash controller and flash memory. Data is fetched over a 32-bit bus between CPU and flash controller and a 128-bit bus between flash controller and flash memory. The flash controller arbitrates simultaneous requests for data and code.

Figure 15. Architecture of Flash Controller



By default un-programmed (erased) flash contains all ones. Up to 32 contiguous bits can be changed from one to zero in a single write (program) operation. Once programmed to zero, a bit can only be returned to the one state with an erase operation. The granularity of erase and program operations are different for instruction flash than data flash:

- Erasure granularity – 2 kB pages for data and 4 kB pages for instruction flash.
- Write granularity – single contiguous 32-bit DWORD for data and two noncontiguous 32-bit DWORDs for instruction flash.

The noncontiguous instruction flash DWORDs can either be both even (byte addresses 0-3 and 8-11) or both odd (byte addresses 4-7 and 12-15) DWORDS.
Modifying a particular location in flash requires a four step process:

1. The entire page containing the location to be modified must be saved in RAM.
2. The entire page must be erased.
3. The desired RAM location can be modified.
4. The entire page must be restored from RAM to flash.

The MCU features ROM utility functions to simplify and fool proof flash programming.

Flash is a single ported memory that must be arbitrated between instruction fetch, operand fetch and programming operations. When multiple operations contend for flash, the flash controller enforces the following priority:

1. Data page erase
2. Instruction page erase
3. Mass erase
4. Erase reference cell
5. Data program

6. Instruction program
7. Data read
8. Instruction read

For optimal performance, read-only and initialized data should be copied from flash to RAM during start-up. The linker script and startup files provided with the Intel® System Studio for Microcontrollers Board Support Package for reference board do this automatically.

9.1.2 Registers

Flash controller registers are aligned on 4-byte boundaries and can be accessed as bytes, words, or double words. Bit definitions are given in the following sections.

9.1.2.1 Flash Controller Wait States Register

Flash Controller Wait States Register is a readable and writable register memory mapped at address 0x8000 0040 to 0x8000 0043. This register can be used to set flash wait states.

Table 65. Flash Controller Wait States Register

Mnemonic	Access	Bits	Description	Reset
WSH	R/W	3:0	Wait State High – the flash SE high pulse width in system clocks plus one. Always set to 0.	0x0
WSL	R/W	7:4	Wait State Low – the flash SE low pulse width in system clocks plus one. This must be set to one when the system clock frequency is above 20 MHz.	0x0
MSC	R/W	13:8	Micro-Second Count – this bit field must be set equal to or greater than the number of system clocks in one micro-second. Set this register for the frequency in MHz rounded up.	0x5
-	-	14	Reserved	X
CS	R/W	15	Clock Slow – when written '1', zero wait state flash access is possible. When '0', flash accesses will always be one or more wait states. This bit must be set to zero when clock frequencies are above 6.7 MHz.	0x0

9.1.2.2 Flash Controller Erase/Write Control Register

Flash Controller Erase/Write Control Register is a readable and writable register memory mapped at address 0x8000 0044 to 0x8000 0047. It must be accessed as a double word. This register can be used to erase or write to flash. It is write protected by a pass code in bits [23:16] that must equal 0x9D to enable the write operation. Any other value inhibits the write operation. Reading is always allowed, but the pass code is not revealed. Note that the bits are not mutually exclusive and that multiple operations can be queued. When multiple operations are scheduled simultaneously, they will be serviced in the following order:

1. Data page erase
2. Instruction page erase
3. Mass erase
4. Erase reference cell
5. Data write
6. Instruction write

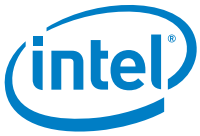


Table 66. Flash Controller Erase/Write Control Register

Mnemonic	Access	Bits	Description	Reset
IWR	R/W	0	Instruction Write Request: when written '1', an instruction write operation to the address specified in the Flash Instruction Write Address Register using data specified in the Flash Instruction Write Data [1:0] Registers is initiated. Writing '0' has no effect. Always reads '0'.	0x0
DWR	R/W	1	Data Write Request: when written '1', a data write operation to the address specified in the Flash Data Write Address Register using data specified in the Flash Data Write Data Register is initiated. Writing '0' has no effect. Always reads '0'.	0x0
IPER	R/W	2	Instruction Page Erase Request: when written '1', an instruction page erase operation of the page containing the address specified in the Flash Instruction Write Address Register is initiated. Writing '0' has no effect. Always reads '0'.	0x0
DPER	R/W	3	Data Page Erase Request: when written '1', a data page erase operation of the page containing the address specified in the Flash Data Write Address Register is initiated. Writing '0' has no effect. Always reads '0'.	0x0
MER	R/W	5:4	Mass Erase Request: when written, an odd value initiates a mass erase operation. Writing an even value has no effect. When read, an odd value indicates that an erase operation is currently in progress. Reading an even value indicates that erasure is complete. Bits are encoded thusly: 0 = No operation 1 = Erase all instruction pages 2 = Reserved 3 = Erase all instruction and data pages	0x0
-	-	6	Reserved	X
ERC	R/W	7	Erase Reference Cell: when written '1', an erase reference cell operation is initiated. Writing a '0' has no effect. Always reads '0'. ***NOTE:*** this operation need only be performed once in the flash lifetime. It is normally done at wafer sort and need not be done again.	0x0
-	-	15:8	Reserved	X
PC	W	23:16	Pass Code: when written along with bits [7:0], a value of 8'h9D will enable the write operation. Any other value inhibits the write operation.	X

9.1.2.3 Flash Controller Status Register

Flash Controller Status Register is a readable and writable register memory mapped at address 0x8000 0048 to 0x8000 004B. This register can be used to ascertain the status of the flash controller.

Table 67. Flash Controller Status Register

Mnemonic	Access	Bits	Description	Reset
-	-	0	Reserved	X
ED	R/W	1	Erase Done: when read '1', the previous erase operation is complete. A write to this register clears this bit.	0x0
WD	R/W	2	Write Done: when read '1', the previous write operation is complete. A write to this register clears this bit.	0x0
-	-	7:3	Reserved	X

9.1.2.4 Flash Controller Instruction Write Address Register

Flash Controller Instruction Write Address Register is a readable and writable register memory mapped at address 0x8000 004C to 0x8000 004F. This register can be used to configure the flash instruction write address.

Table 68. Flash Controller Instruction Write Address Register

Mnemonic	Access	Bits	Description	Reset
IWA	R/W	14:0	Instruction Write Address: the address of the instruction to be overwritten or page to be erased. In the case of erasure, the entire 4kB page containing this address shall be erased. This register must be configured before initiating an instruction write or erase operation. ***NOTE:*** if bit 2 is '0', the content of instruction write data 0 register is written to byte addresses 0-3 and instruction write data 1 register is written to byte addresses 8-11. If bit 2 is set to '1', the content of instruction write data 0 register is written to byte addresses 4-7 and instruction write data 1 register is written to byte addresses 12-15.	0x0
-	-	15	Reserved	X

9.1.2.5 Flash Controller Instruction Write Data 0 Register

Flash Controller Instruction Write Data 0 Register is a readable and writable register memory mapped at address 8000 0050 to 8000 0053. This register can be used to configure bits 31:0 of the instruction data to be written to the address specified in IWA above.

Table 69. Flash Controller Instruction Write Data 0 Register

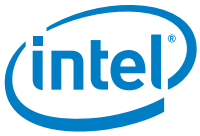
Mnemonic	Access	Bits	Description	Reset
IWD0	R/W	31:0	Instruction Write Data 0: bits 31:0 of the instruction to be overwritten at IWA. Must be configured before initiating an instruction write operation.	0x0

9.1.2.6 Flash Controller Instruction Write Data 1 Register

Flash Controller Instruction Write Data 1 Register is a readable and writable register memory mapped at address 0x8000 0054 to 0x8000 0057. This register can be used to configure bits 31:0 of the instruction data to be written to the address plus eight specified in IWA above.

Table 70. Flash Controller Instruction Write Data 1 Register

Mnemonic	Access	Bits	Description	Reset
IWD1	R/W	31:0	Instruction Write Data 1: bits 31:0 of the instruction to be overwritten at IWA + 8. Must be configured before initiating an instruction write operation.	0x0



9.1.2.7 Flash Controller Data Write Address Register

Flash Controller Data Write Address Register is a readable and writable register memory mapped at address 0x8000 0060 to 0x8000 0063. This register can be used to configure the flash data write address.

Table 71. Flash Controller Data Write Address Register

Mnemonic	Access	Bits	Description	Reset
DWA	R/W	11:0	Data Write Address: the address of the data to be overwritten or page to be erased. In the case of erasure, the entire 2 kB page containing this address shall be erased. This register must be configured before initiating a data write or erase operation.	0x0
-	-	15:12	Reserved	X

9.1.2.8 Flash Controller Data Write Data Register

Flash Controller Data Write Data Register is a readable and writable register memory mapped at address 0x8000 0064 to 0x8000 0067. This register can be used to configure bits 31:0 of the data to be written.

Table 72. Flash Controller Data Write Data Register

Mnemonic	Access	Bits	Description	Reset
DWD	R/W	31:0	Data Write Data: data to be overwritten. Must be configured before initiating a data write operation.	0x0

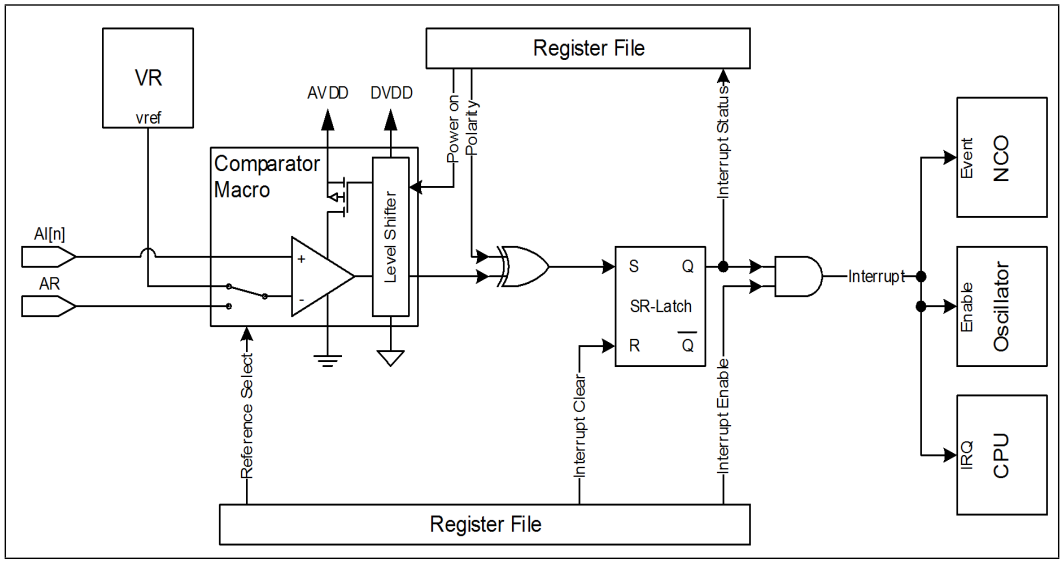
9.2 Analog Comparators

The MCU features 6 high-speed comparators with 1.6 MHz typical bandwidth and 13 low-power comparators with 0.4 µA typical static current. Due to their high bandwidth, the high-speed comparators are ideally suited to uses such as ASK signal demodulation. Due to their low static current, the low-power comparators are ideally suited to uses such as wake-up from standby or retention.

9.2.1 Functional Description

Figure 16 shows the architecture of the comparator. Device pins AI[18:0] each connect to a comparator. AI[0:5] connect to high-speed comparators while AI[6:18] connect to low-power comparator. By default all comparators use an internal 0.95 V threshold reference voltage produced by the voltage regulator. However, an additional device pin AR can optionally be used to supply a different threshold reference voltage. The threshold reference voltage in use is selected under software control. The comparators have rail-to-rail common mode input range which means that they will function properly with a threshold reference voltage from 0 to AVDD voltage.

Figure 16. Analog Comparator



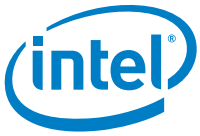
The polarity of each comparator can be selected under software control. If above threshold polarity is selected, a comparator's interrupt status will be asserted when the input voltage is above the threshold voltage. If below threshold polarity is selected, a comparator's interrupt status will be asserted when the input voltage is below the threshold voltage. Once asserted, interrupt status is latched until the interrupt condition is removed and the interrupt status bit is cleared by software. If the interrupt condition is true while the interrupt status is cleared, the interrupt will reassert immediately after clearing.

Each comparator can be powered on or off under software control. By default, comparators are powered off. Powered off comparators output a logic zero. However, if polarity is set to below threshold, a powered off comparator will assert its interrupt status. When a comparator is initially powered on, the output state is unknown for a brief period until they stabilize. The user should delay one millisecond after power on before attempting to use the comparators.

Each comparator's interrupt can be enabled or disabled under software control. When an enabled comparator interrupt status asserts, the following actions occur automatically in hardware:

1. The system oscillator powers up.
2. The AHB and CPU clocks restart.
3. The NCO switches to the pending event frequency immediately.
4. An interrupt is generated if the CPU's comparator interrupt is unmasked and its IDT entry is present.

Each comparator's interrupt status can be read regardless of the interrupt enable state. Writing a logic one to a bit in the comparator interrupt status register clears the corresponding comparator's interrupt status.



9.2.2 Registers

Comparator registers are aligned on 4-byte boundaries and can be accessed as bytes, words, or double words. Bit definitions are given in the following sections.

9.2.2.1 Comparator Interrupt Enable Register (CE)

Comparator Interrupt Enable Register is a readable and writable register memory mapped at address 0x8000 00C0 to 0x8000 00C3. This register is used to enable interrupts. The interrupt status register is logically AND'd to this register and the result reduction OR'd to generate an interrupt.

Table 73. Comparator Interrupt Enable Register

Mnemonic	Access	Bits	Description	Reset
CE	R/W	18:0	Comparator Interrupt Enable: a 1 enables the interrupt.	0x0
-	-	31:19	Reserved	X

9.2.2.2 Comparator Reference Register (CR)

Comparator Reference Register is a readable and writable register memory mapped at address 0x8000 00C4 to 0x8000 00C7. This register is used for comparator reference voltage selection.

Table 74. Comparator Reference Register

Mnemonic	Access	Bits	Description	Reset
CR	R/W	18:0	Comparator Reference: when 1, the internal voltage reference supplied by the voltage regulator is used. When 0, an external voltage reference on the AR pin is used.	0x0
-	-	31:19	Reserved	X

9.2.2.3 Comparator Polarity Register (CP)

Comparator Polarity Register is a readable and writable register memory mapped at address 0x8000 00C8 to 0x8000 00CB. This register is used for comparator polarity selection.

Table 75. Comparator Polarity Register

Mnemonic	Access	Bits	Description	Reset
CP	R/W	18:0	Comparator Polarity: when 0, an input above the threshold triggers an event. When 1, an input below the threshold triggers an event.	0x0
-	-	31:19	Reserved	X

9.2.2.4 Comparator Power Register (CU)

Comparator Power Register is a readable and writable register memory mapped at address 0x8000 00CC to 0x8000 00CF. This register is used for comparator power up.

Table 76. Comparator Power Register

Mnemonic	Access	Bits	Description	Reset
CU	R/W	18:0	Comparator Power Up: when 1, the comparator is powered up.	0x0
-	-	31:19	Reserved	X

9.2.2.5 Comparator Interrupt Status Register (CS)

Comparator Interrupt Status Register is a readable and writable register memory mapped at address 0x8000 00D0 to 0x8000 00D3. This register is used for comparator interrupt status read and clear. The content of this register is unaffected by the interrupt enable mask. This register can be used in both polled and interrupt driven applications.

Table 77. Comparator Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
CS	R/W	18:0	Comparator Status: when read, a 1 indicates that the comparator has detected a threshold crossing since last reset. When written, a 1 clears the bit. ***NOTE:*** status is latched using a transparent latch. Therefore, if the analog input is across the threshold during and after the reset, the status will remain 1 and an interrupt will be generated immediately if enabled.	X
-	-	31:19	Reserved	X

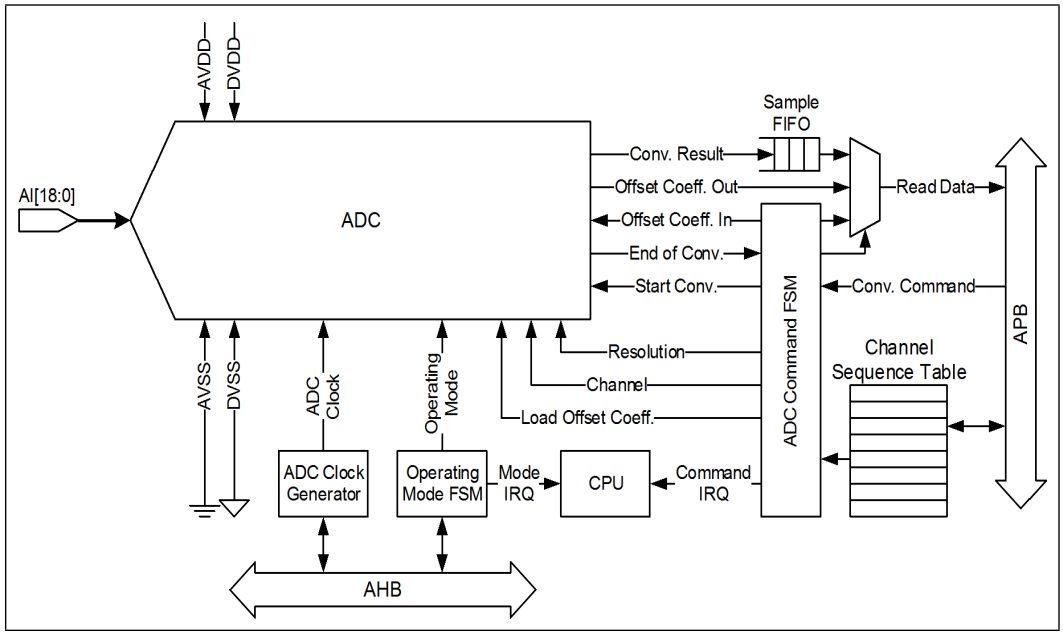
9.3 Analog to Digital Converter (ADC)

The MCU features a SAR ADC supporting resolutions from 6 to 12 bits and sample rates from DC up to 4.2 MSps. Use of the ADC is simplified by two finite state machines: (1) controlling operating mode and (2) controlling analog to digital conversions. When used along with a crystal oscillator, this ADC is ideally suited for uses such as spectrum analysis that require low harmonic distortion and uniform continuous sampling.

9.3.1 Functional Description

A schematic diagram of the ADC and its finite state machines is shown in Figure 17. Device pins AI[18:0] connect to the ADC. Device pin AVDD supplies power to the front end analog circuitry as well as the reference voltage. Device pin DVDD supplies power to the back end digital circuitry. The ADC has been carefully designed to deliver low total harmonic distortion. Analog inputs and power supplies have been carefully routed to deliver high effective number of bits (ENOB) and signal to noise ratio (SNR). The ADC has been augmented with finite state machines to simply the user interface.

Figure 17. Schematic Diagram of the ADC and its Finite State Machines (FSM)



The operating mode FSM sequences the ADC through its various operating modes from deep power down to operational and vice versa. The user simply writes a command to the FSM to initiate a mode transition. The command includes a delay parameter that specifies the time in clock cycles that the ADC delays between mode changes. Other parameters include whether or not to generate an interrupt when the mode change completes and the ADC NCO phase increment, which controls the ADC clock frequency. The various operating modes are:

- Operational with Calibration – the ADC can execute conversion and calibration commands in this mode. Offset calibration coefficients are used to reduce offset error.
- Operational without Calibration – the ADC can execute conversion commands in this mode. Offset calibration coefficients are not used.
- Standby – ADC power consumption is reduced to leakage current in this mode. The ADC cannot execute commands in this mode, but can be transitioned to operational mode within a few clocks.
- Power Down – the ADC's internal low dropout voltage regulator is powered down in this mode significantly reducing leakage current. The ADC cannot execute commands in this mode, and transition to standby or operational modes requires at least 10 µs.
- Deep power down – in addition to powering down the ADC's internal low dropout voltage regulator, the offset calibration coefficient storage is also powered down further reducing leakage current. The ADC cannot execute commands in this mode, and transition to standby or operational modes requires at least 10 µs. Furthermore, offset coefficients are lost in this mode. Therefore, if operating with calibration, a start calibration or load offset coefficient command will be needed before issuing any start conversion commands.

The ADC command FSM sequences the ADC through one or more analog to digital conversions. The user simply writes a command to the FSM to start conversions. A built-in analog multiplexor allows the ADC to sample any of the AI[·] device pins. The channel sequence table provides the FSM with the list of channels (i.e. device pins) to sample. When a start conversion command is initiated, the FSM will direct the ADC to sample channels in the order that they are listed in the channel sequence table. There are two variants of the start conversion command: (1) for a fixed number of passes through the channel sequence table and (2) for continuous passes through the channel sequence table. The number of entries in the channel sequence table can be from 1 to 32. Channels can be listed in any order and repeated multiple times.

Start conversion commands require the following parameters:

1. Whether or not to generate an interrupt when the command completes.
2. The number of passes to make through the channel sequence table or the number of samples between interrupts in the case of continuous conversions.
3. The width of the sample window, which needs to be lengthened for high impedance sources or can be used to fine tune the sample rate.
4. The starting point in the channel sequence table.
5. The resolution (i.e. number of bits).

Other ADC commands include reset, start calibration, load offset coefficient, and stop continuous conversions. The load offset coefficient command requires the coefficient as a parameter.

9.3.2 Registers

ADC registers are aligned on 4-byte boundaries and must be accessed as double words. Bit definitions are given in the following sections.

9.3.2.1 ADC Operating Mode Register (OM)

The ADC Operating mode register is a readable and writable register memory mapped at address 0x8000_00E0 to 0x8000_00E3. The register is used to change the current ADC operating mode and clock frequency.

Table 78. ADC Operating Mode Register (OM)

Mnemonic	Access	Bits	Description	Reset
Mode	R/W	2:0	Operating Mode: writing to this bit field along with delay initiates a change in the operating mode. Reading this bit field returns the current operating mode. It is encoded thusly: 0 = Deep power down 1 = Power down 2 = Standby 3 = Normal w/calibration 4 = Normal wo/calibration 5 = No change 6 = No change 7 = No change	0x0
Delay	R/W	15:3	Delay: writing to this bit field along with operating mode sets the delay between mode transitions in system clock cycles. Reading this bit field returns the current delay value.	0x500

PI	R/W	23:16	Phase Increment: writing to this bit field initiates a change in the clock frequency. Reading this bit field returns the current NCO phase increment. It is encoded thusly: $PI = \begin{cases} 256\frac{adcFreq}{oscFreq}, adcFreq \neq oscFreq \\ 0, otherwise \end{cases}$	0x1
Enctrl	R/W	26:24	Enable Direct Control: writing a non-zero value places the ADC into special test mode. Set this field to zero for normal operation.	0x0
IE	R/W	27	Interrupt Enable: writing a '1' to this bit enables an interrupt at the completion of the operating mode change. Reading this bit returns the current interrupt enable setting.	0x0
-	-	31:28	Reserved	X

9.3.2.2 ADC Channel Sequence Table Register (CS)

The ADC Channel Sequence Table is a readable and writable register file memory mapped at addresses 0x9500_0000 to 0x9500_001F. The register file is used to specify the channel sequence used for conversions. A start conversion command points to the beginning of the table. Each conversion advances the pointer until an entry with the Last bit asserted is found. The pointer then recycles to the beginning of the table. There are 4 entries per double word for a total of 32 table entries. Each double word has the following format:

Table 79. ADC Channel Sequence Table Register (CS)

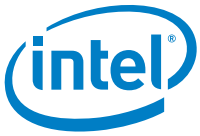
Mnemonic	Access	Bits	Description	Reset
Channel[4N]	R/W	4:0	ADC Channel: this bit field defines the ADC channel to sample	0x0
-	-	6:5	Reserved	X
Last[4N]	R/W	7	Last Channel: when set, this bit indicates that this is the last channel in the sequence. Subsequent conversions will begin at the start of the table.	0x1
Channel[4N+1]	R/W	12:8	ADC Channel: this bit field defines the ADC channel to sample	0x0
-	-	14:13	Reserved	X
Last[4N+1]	R/W	15	Last Channel: when set, this bit indicates that this is the last channel in the sequence. Subsequent conversions will begin at the start of the table.	0x1
Channel[4N+2]	R/W	20:16	ADC Channel: this bit field defines the ADC channel to sample	0x0
-	-	22:21	Reserved	X
Last[4N+2]	R/W	23	Last Channel: when set, this bit indicates that this is the last channel in the sequence. Subsequent conversions will begin at the start of the table.	0x1
Channel[4N+3]	R/W	28:24	ADC Channel: this bit field defines the ADC channel to sample	0x0
-	-	30:29	Reserved	X
Last[4N+3]	R/W	31	Last Channel: when set, this bit indicates that this is the last channel in the sequence. Subsequent conversions will begin at the start of the table.	0x1

9.3.2.3 ADC Command Register (Command)

The ADC Command Register is a readable and writable register memory mapped at address 0x9500 0020 to 0x9500 0023. Writing to this register initiates an ADC operation. Reading this register returns non-zero if and only if an ADC operation is currently in progress (i.e. the finite state machine is not idle). If the register is written while an ADC operation is in progress, the command is ignored.

Table 80. ADC Command Register (Command)

Mnemonic	Access	Bits	Description	Reset
Command	W	2:0	ADC Command: this bit field defines ADC operation. It is encoded thusly: 0 = Start single conversion 1 = Start continuous conversion 2 = Reset calibration 3 = Start calibration 4 = Load calibration 5 = Stop continuous conversion 6 = No operation 7 = No operation	X
IE	W	3	Interrupt Enable: when set, this bit enables an interrupt on completion of the ADC operation.	X
NS	W	8:4	Number of Samples: when the ADC command is start single conversion, this bit field defines the number of repeated conversions before stopping. When the ADC command is start continuous conversions, this bit field defines the number conversion between interrupts. This bit field is ignored otherwise. It encoded thusly: $numConversions = NS + 1$	X
CSTI	W	13:9	Channel Sequence Table Index: when the ADC command is a start conversion command, this bit field defines the first entry in the channel sequence table to be sampled. If repeated or continuous conversions are commanded, subsequent conversions will follow the channel sequence table.	X
Resolution	W	15:14	Resolution: this bit field defines the number of bits of precision. It is defined thusly: 0 = 6-bits 1 = 8-bits 2 = 10-bits 3 = 12-bits	X
CD	W	22:16	Calibration Data: when the ADC command is a load calibration command, this bit field defines the digital offset calibration data to be loaded.	X
-	-	23	Reserved	X
SW	W	31:24	Sample Window: when the ADC command is a conversion command, this bit field defines the length of the sample interval in ADC clocks. When repeated or continuous conversions are commanded, the next sample interval begins on the N^{th} + 2 ADC clock following start of conversion and ends after the first clock of the next conversion where N is the number of bits of precision. The sample interval should be lengthened in case of a high impedance source. This field is encoded thusly: $sampleInterval = SW + 2$ $sampleRate = \frac{adcFreq}{N + sampleInterval}$	X



9.3.2.4 ADC Interrupt Status Register (IS)

The ADC Interrupt Status Register is a readable and writable register for both Command Complete (CC) and FIFO Overrun (FO) memory mapped at address 0x9500 0024 to 0x9500 0027. The register is used to read and reset status interrupts. The Interrupt Enable Register is logically AND'd to this register and the result reduction OR'd to generate an interrupt. However, the interrupt enable mask has no effect on the content of this register. This register can be used in both polled and interrupt driven applications.

Table 81. ADC Interrupt Status Register (IS)

Mnemonic	Access	Bits	Description	Reset
CC	R/W	0	Command Complete: when read, a 1 indicates that an ADC command has completed since the bit was last reset. When written, a 1 clears the bit.	0x0
FO	R/W	1	FIFO Overrun: when read, a 1 indicates a FIFO overrun. When written, a 1 clears the bit.	0x0
-	-	2:31	Reserved	X

9.3.2.5 ADC Interrupt Enable Register (IE)

The ADC Interrupt Enable Register is a readable and writable register for both Command Complete (CC) and FIFO Overrun (FO) memory mapped at address 0x9500 0028 to 0x9500 002B. This register is used to enable interrupts. The Interrupt Status Register is logically AND'd to this register and the result reduction OR'd to generate an interrupt.

Table 82. ADC Interrupt Enable Register (IE)

Mnemonic	Access	Bits	Description	Reset
CC	R/W	0	Command Complete: a 1 enables the interrupt.	0x0
FO	R/W	1	FIFO Overrun: a 1 enables the interrupt.	0x0
-	-	2:31	Reserved	X

9.3.2.6 ADC Sample Register (Sample)

The ADC Sample Register is a readable and writable register memory mapped at address 0x9500 002C to 0x9500 002F. This register is used to read ADC samples or flush the sample FIFO. Read access unloads one sample. Write access flushes the FIFO.

Table 83. ADC Sample Register (Sample)

Mnemonic	Access	Bits	Description	Reset
Sample	R/W	11:0	ADC Sample: when read, this bit-field contains the ADC sample. The FIFO is flushed on a write.	X
-	-	31:12	Reserved	X

9.3.2.7 ADC Calibration Data Register (CD)

The ADC Calibration Data Register is read-only register memory mapped at address 0x9500 0030 to 0x9500 0033. This register is used to read ADC calibration data.

Table 84. ADC Calibration Data Register (CD)

Mnemonic	Access	Bits	Description	Reset
CD	R	6:0	Calibration Data: when read, this bit field contains the ADC digital offset calibration data.	X
-	-	31:7	Reserved	X

9.3.2.8 ADC FIFO Count Register (Count)

The ADC FIFO Count Register is a read-only register memory mapped at address 0x9500 0034 to 0x9500 0037. This register is used to read the number of samples store in the ACD FIFO.

Table 85. ADC FIFO Count Register (Count)

Mnemonic	Access	Bits	Description	Reset
Count	R	5:0	ADC FIFO Count: when read, this bit-field contains the number of samples stored in the ADC FIFO. Writing has no effect.	0x0
-	-	31:6	Reserved	X

9.4 Real Time Clock (RTC)

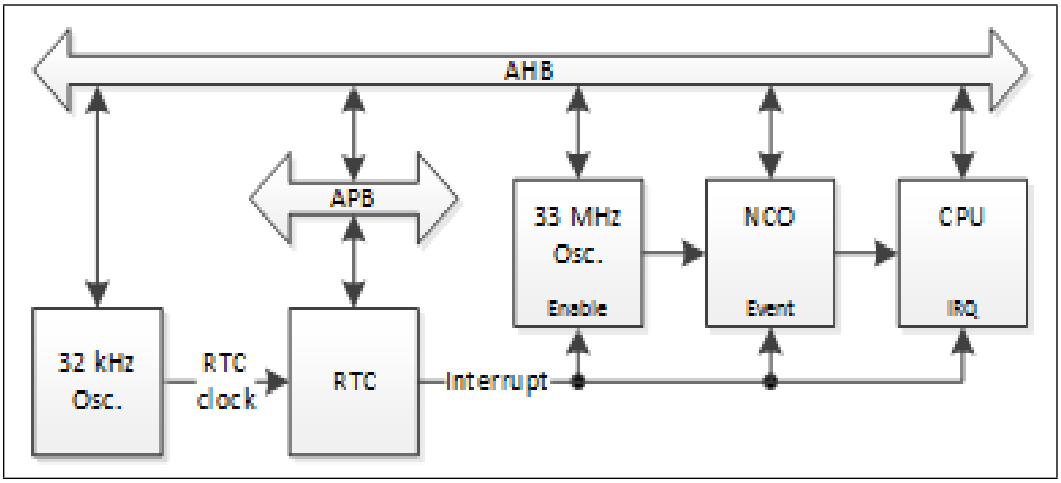
The MCU features a 32-bit real-time clock driven by the 32 kHz oscillator that can be used to keep track of time even when all other system clocks are disabled or the core voltage is reduced to retention levels. Other features and benefits include:

- Programmable match register to trigger an interrupt at the specified time.
- Programmable counter load register to initialize the current time.
- Programmable wrap mode controls when the counter wraps to zero – when match time is reached or when all ones count is reached.
- Maskable interrupt will automatically restart the system oscillator when match time is reached.
- Component version register.

9.4.1 Functional Description

Figure 18 shows the architecture of the real time clock. The RTC peripheral receives a 32 kHz clock from the 32 kHz crystal oscillator and increments a 32-bit counter on each clock. The RTC features registers that allow the user application to load the up counter with a specific value, request notification when the up counter reaches a specific value, enable or disable counting, enable or disable interrupts, and select the wrap mode (i.e. wrap to zero when an all ones count is reached or when a specific value is reached).

Figure 18. Architecture of the Real Time Clock



The RTC does not need the APB clock running to keep time or generate interrupts. It needs only the 32 kHz RTC clock. However, the APB clock must be running in order for the CPU to access registers within the RTC. Also, register values must be transferred between APB and RTC clock domains. The consequence of this is that a specific value to be loaded into the RTC up counter programmed from APB will read back for up to two 32 kHz clock cycles. Also, the APB clock must remain running until the register values transfer from APB to RTC clock domains.

The RTC can generate an interrupt when the up counter reaches a specific value. The interrupt can be enabled, which means that it will appear in the raw interrupt status register, but masked, which means that the CPU will not be interrupted. If interrupts are both enabled and unmasked, an interrupt will be generated when the up counter reaches the value specified in the Counter Match Register. In addition to generating an interrupt, the 33 MHz hybrid system oscillator will be powered on if it is not already on. This enables wake-up at a specified time from standby and retention states where all clocks except the RTC clock are off.

The RTC also provides a memory mapped register for reading the version code.

9.4.2 Registers

RTC registers are aligned on 4-byte boundaries and can be accessed only as double words. Bit definitions are given in the following sections.

9.4.2.1 Current Counter Value Register

The RTC Current Counter Value Register is a read-only register memory mapped at address 0x9100 0000 to 0x9200 0003. When read, this register is the current value of the internal counter. This value always is read coherently.

Table 86. RTC Current Counter Value Register

Mnemonic	Access	Bits	Description	Reset
RTC_CCVR	R	31:0	Returns the current up counter value.	0x0

9.4.2.2 Counter Match Register

The RTC Counter Match Register is a readable and writable register memory mapped at address 0x9100 0004 to 0x9200 0007. When the internal counter matches this register, an interrupt is generated, provided interrupt generation is enabled.

Table 87. RTC Counter Match Register

Mnemonic	Access	Bits	Description	Reset
RTC_CMR	R/W	31:0	This value is transferred to the RTC clock domain then compared against the up counter. An interrupt is generated when the internal up counter matches this register. When read, this register returns the last value written.	0x0

9.4.2.3 Counter Load Register

The RTC Counter Load Register is a readable and writable register memory mapped at address 0x9100 0008 to 0x9200 000B. This register load the counter the loaded value.

Table 88. RTC Counter Load Register

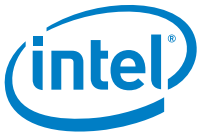
Mnemonic	Access	Bits	Description	Reset
RTC_CLR	R/W	31:0	When written, this value is transferred to the RTC clock domain then loaded into the up counter. When read, this register returns the last value written.	0x0

9.4.2.4 Counter Control Register

The RTC Counter Control Register is a readable and writable register memory mapped at address 0x9100 000C to 0x9200 000F. This register is used to control counter of the RTC.

Table 89. RTC Counter Control Register

Mnemonic	Access	Bits	Description	Reset
RTC_IEN	R/W	0	Allows the user to disable interrupt generation. 0 = Interrupt disabled 1 = Interrupt enabled	0x0
RTC_MSK	R/W	1	Allows the user to mask a generated interrupt. 0 = Interrupt unmasked 1 = Interrupt masked	0x0
RTC_EN	R/W	2	Allows the user to control counting in the counter. 0 = Counter disabled 1 = Counter enabled	0x0
RTC_WEN	R/W	3	Allows the user to force the counter to wrap when a match occurs instead of waiting until the maximum count is reached. 0 = Wrap disabled 1 = Wrap enabled	0x0
-	-	31:4	Reserved	X



9.4.2.5 Interrupt Status Register

The RTC Interrupt Status Register is a read-only register memory mapped at address 0x9100 0010 to 0x9200 0013. This register is used to read the interrupt status.

Table 90. RTC Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
RTC_STAT	R	0	This register is the masked raw status 0 = Interrupt is inactive 1 = Interrupt is active (regardless of polarity).	0x0
-	-	31:1	Reserved	X

9.4.2.6 Interrupt Raw Status Register

The RTC Interrupt Status Register is a read-only register memory mapped at address 0x9100 0014 to 0x9200 0017. This register is used to read the interrupt raw status.

Table 91. RTC Interrupt Raw Status Register

Mnemonic	Access	Bits	Description	Reset
RTC_RSTAT	R	0	0 = Interrupt is inactive 1 = Interrupt is active (regardless of polarity).	0x0
-	-	31:1	Reserved	X

9.4.2.7 End of Interrupt Register

The RTC End of Interrupt Register is a read-only register memory mapped at address 0x9100 0018 to 0x9200 001B. This register is used to clear the interrupt.

Table 92. RTC End of Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
RTC_EOI	R	0	By reading this location, the match interrupt is cleared. Performing read-to-clear on interrupts, the interrupt is cleared at the end of the read.	0x0
-	-	31:1	Reserved	X

9.4.2.8 Component Version Register

The RTC Component Version Register is a read-only register memory mapped at address 0x9100 001C to 0x9200 001F. This register is used to read the RTC version code.

Table 93. RTC Component Version Register

Mnemonic	Access	Bits	Description	Reset
RTC_COMP_VERSION	R	31:0	Always returns the version code 2.03 as the big endian ASCII string "203*" or 0x3230332A.	0x3230 332A

9.5 Watchdog Timer (WDT)

The MCU features a 32-bit watchdog timer driven by *apb_clk* that can be used to reset the microcontroller in the event of a software error that makes the system unresponsive. Other features and benefits include:

- Programmable timeout mode
  - — Reset on timeout.
  - — Interrupt on first timeout then reset on second timeout.
- Programmable timeout period.
- Write inhibit after enablement to prevent accidental disabling or changing of timeout period.
- Component version register.

9.5.1 Functional Description

Once enabled, the watchdog timer counts down from a user specified timeout value. The counter can be restarted at any time by writing 0x76 to the Counter Restart Register. If the counter reaches zero, one of two things will happen:

1. If the timer is programmed for interrupt first mode and this is the first timeout, an interrupt will be generated and the down counter will restart from the user specified timeout value.
2. If this is the second timeout or the timer is programmed for reset only mode, a system reset will be generated.

Once enabled, the timer cannot be stopped. The timeout value can be changed, but the new value won't take effect until the counter is restarted. As a general rule, the timer should be configured prior to enablement. Memory mapped registers are provided for this purpose.

Memory mapped registers are also provided for managing interrupts and reading type code, version code, and configuration parameters.

9.5.2 Registers

WDT registers are aligned on 4-byte boundaries and can be accessed only as double words. Bit definitions are given in the following sections.

9.5.2.1 WDT Control Register (CR)

The WDT Control Register is a readable and writable register memory mapped at address 0x9200 0000 to 0x9200 0003. This register is used to control the WDT.

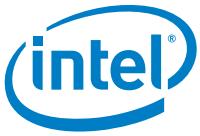


Table 94. WDT Control Register

Mnemonic	Access	Bits	Description	Reset
WDT_EN	R/W	0	WDT Enable: when '1', the WDT is enabled. This bit can only be cleared on system reset.	0x0
RMOD	R/W	1	Response Mode: when '1', the WDT will interrupt on the first timeout then reset the system on the second timeout. When '0', the WDT will reset the system on the first timeout.	0x0
RPL	R/W	4:2	Reset Pulse Length: selects the width of the reset pulse encoded thusly: 0 = 2 APB clock cycles 1 = 4 APB clock cycles 2 = 8 APB clock cycles 3 = 16 APB clock cycles 4 = 32 APB clock cycles 5 = 64 APB clock cycles 6 = 128 APB clock cycles 7 = 256 APB clock cycles	0x0
SCR	R/W	5	Scratch Pad: returns the last value written.	0x0
-	-	31:6	Reserved	X

9.5.2.2 WDT Timeout Range Register (TORR)

The WDT Timeout Range Register is a readable and writable register memory mapped at address 0x9200 0004 to 0x9200 0007. This register is used to select the timeout.

Table 95. WDT Timeout Range Register

Mnemonic	Access	Bits	Description	Reset
TOP	R/W	3:0	Timeout Period: selects the timeout period in APB clock cycles encoded thusly: $timeout\ period = 2^{16+TOP}$ Changes in the timeout value take effect at enablement or counter restart.	0x0
-	-	31:6	Reserved	X

9.5.2.3 WDT Current Counter Value Register (CCVR)

The WDT Current Counter Value Register is a read only register memory mapped at address 0x9200 0008 to 0x9200 000B. This register is used to read the current timeout counter value.

Table 96. WDT Current Counter Value Register

Mnemonic	Access	Bits	Description	Reset
CCVR	R	31:0	Current Counter Value Register: returns the current timeout counter value.	0x65535

9.5.2.4 WDT Counter Restart Register (CRR)

The WDT Counter Restart Register is a write only register memory mapped at address 0x9200 000C to 0x9200 000F. This register is used to restart the timeout counter.

Table 97. WDT Counter Restart Register

Mnemonic	Access	Bits	Description	Reset
CRR	W	7:0	Counter Restart Register: writing the value 0x76 will restart the timeout counter and clear the interrupt if it is active. Writing any other value has no effect.	0x0

9.5.2.5 WDT Interrupt Status Register (STAT)

The WDT Interrupt Status Register is a read only register memory mapped at address 0x9200 0010 to 0x9200 0013. This register is used to read the interrupt status.

Table 98. WDT Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
ISR	R	0	Interrupt Status Register: when '1', the interrupt is active. When '0', the interrupt is inactive.	0x0
-	-	31:1	Reserved	X

9.5.2.6 WDT End of Interrupt Register (EOI)

The WDT End of Interrupt Register is a read only register memory mapped at address 0x9200 0014 to 0x9200 0017. This register is used to clear the interrupt.

Table 99. WDT End of Interrupt Register

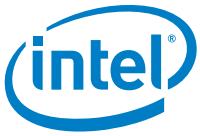
Mnemonic	Access	Bits	Description	Reset
EOI	R	0	End of Interrupt Register: reading this register clears the interrupt. The returned value has no meaning and can be ignored.	0x0
-	-	31:1	Reserved	X

9.5.2.7 WDT Component Parameters Register 1 (COMP_PARAMS_1)

The WDT Component Parameters Register 1 is a read only register memory mapped at address 0x9200 00F4 to 0x9200 00F7. This register is used to read the peripheral's hardware configuration.

Table 100. WDT Component Parameters Register 1

Mnemonic	Access	Bits	Description	Reset
CP_WDT_ALWAYS_EN	R	0	Enable WDT from reset: always returns 0.	0x0
CP_WDT_DFLT_RMOD	R	1	Output response mode default value: always returns 0.	0x0



CP_WDT_DUAL_TOP	R	2	Include a second timeout period for initialization: always returns 0.	0x0
CP_WDT_HC_RMOD	R	3	Hard code output response mode: always returns 0.	0x0
CP_WDT_HC_RPL	R	4	Hard code reset pulse length: always returns 0.	0x0
CP_WDT_HC_TOP	R	5	Hard code timeout period range selection:	0x0
CP_WDT_USE_FIX_TOP	R	6	Use predefined timeout period ranges:	0x1
CP_WDT_PAUSE	R	7	Include pause enable input on I/F:	0x0
CP_WDT_APB_DATA_WIDTH	R	9:8	APB Data bus width:	0x2
CP_WDT_DFLT_RPL	R	12:10	Default reset pulse length:	0x0
-	-	15:13	Reserved.	X
CP_WDT_DFLT_TOP	R	19:16	Main timeout period range to be selected as default:	0x0
CP_WDT_DFLT_TOP_INIT	R	23:20	Initial timeout period range to be selected as default:	0x0
CP_WDT_CNT_WIDTH	R	28:24	WDT counter width:	0x32
-	-	31:29	Reserved	X

9.5.2.8 WDT Component Parameters Register 2 (COMP_PARAMS_2)

The WDT Component Parameters Register 2 is a read only register memory mapped at address 0x9200 00F0 to 0x9200 00F3. This register is used to read the peripheral's hardware configuration.

Table 101. WDT Component Parameters Register 2

Mnemonic	Access	Bits	Description	Reset
CP_WDT_CNT_RST	R	31:0	WDT Counter Reset value: always returns 65535.	0x65535

9.5.2.9 WDT Component Parameters Register 3 (COMP_PARAMS_3)

The WDT Component Parameters Register 3 is a read only register memory mapped at address 0x9200 00EC to 0x9200 00EF. This register is used to read the peripheral's hardware configuration.

Table 102. WDT Component Parameters Register 3

Mnemonic	Access	Bits	Description	Reset
CD_WDT_TOP_RST	R	31:0	WDT Timeout Period Reset value: always returns 0.	0x0

9.5.2.10 WDT Component Parameters Register 4 (COMP_PARAMS_4)

The WDT Component Parameters Register 4 is a read only register memory mapped at address 0x9200 00E8 to 0x9200 00EB. This register is used to read the peripheral's hardware configuration.

Table 103. WDT Component Parameters Register 4

Mnemonic	Access	Bits	Description	Reset
CP_WDT_USER_TOP_INIT_MAX	R	31:0	Upper limit of Initial Timeout Period parameters: always returns 0.	0x0

9.5.2.11 WDT Component Parameters Register 5 (COMP_PARAMS_5)

The WDT Component Parameters Register 5 is a read only register memory mapped at address 0x9200 00E4 to 0x9200 00E7. This register is used to read the peripheral's hardware configuration.

Table 104. WDT Component Parameters Register 5

Mnemonic	Access	Bits	Description	Reset
CP_WDT_USER_TOP_MAX	R	31:0	Upper limit of Timeout Period parameters: always returns 0.	0x0

9.5.2.12 WDT Component Version Register (COMP_VERSION)

The WDT Component Version Register is a read only register memory mapped at address 0x9200 00F8 to 0x9200 00FB. This register is used to read the WDT version code.

Table 105. WDT Version Code Register

Mnemonic	Access	Bits	Description	Reset
COMP_VERSION	R	31:0	Version Code Register: always returns the version code 1.06 as the big endian ASCII string "106*" or 0x3130362A.	0x3130 362A

9.5.2.13 WDT Component Type Register (COMP_TYPE)

The WDT Component Type Register is a read only register memory mapped at address 0x9200 00FC to 0x9200 00FF. This register is used to read the peripheral's identification code.

Table 106. WDT Component Type Register

Mnemonic	Access	Bits	Description	Reset
COMP_TYPE	R	31:0	Component Type Register: always returns the identification code 0x4457 0120.	0x4457 0120

9.6 General Purpose Timers

The MCU features two 32-bit general purpose timers in addition to the CPU's built-in timer. These timers are driven by independent clocks allowing CPU clock frequency changes without affecting timer rate. Other features and benefits include:

- Independent load count registers.
- Independent current count registers.
- Independently maskable interrupts.
- Programmable wrap mode controls what value the counter wraps to after counting down to zero – all ones or the value on the load count register.
- Component version register.

9.6.1 Functional Description

The two general purpose timers are completely independent. They each have their own clock source (timer_0_clk and timer_1_clk). These clocks can run at different frequencies (see Section 6.4.2.2 above for details). Each timer has its own set of memory mapped registers to program the timeout period, select the operating mode, enable the timer, and manage interrupts. In addition, a set of memory mapped registers is provided to manage interrupts from both timers at once. A memory mapped register for reading the component version is also provided.

The timeout period should be programmed prior to enabling the timer. The operating mode can be selected at the same time the timer is enabled. The timeout period and mode cannot be changed once the timer is enabled. The timer should first be disabled if the timeout period or mode is to be changed.

The two operating modes are:

1. Free running – after the timer counts down to zero, it reloads the maximum count 0xFFFF FFFF. This mode is useful for implementing one shot timers, where the interrupt handler disables the timer.
2. Periodic – after the timer counts down to zero, it reloads the value in the Load Count register. This mode is useful for implementing periodic timers, where the interrupt handler does not disable the timer.

9.6.2 Registers

Timer registers are aligned on 4-byte boundaries and can be accessed only as double words. Bit definitions are given in the following sections.

9.6.2.1 Timer Load Count Register (Tn_LOAD_CNT)

Timer Load Count Registers are readable and writable registers memory mapped at addresses 0x9300 0000 to 0x9300 0003 for timer[0] and 0x9300 0014 to 0x9300 0017 for timer[1]. These registers are used to program the initial value loaded into the timers when they start or, if operating in periodic mode, restart.

Table 107. Timer Load Count Register

Mnemonic	Access	Bits	Description	Reset
LOAD_CNT	R/W	31:0	Load Count: the value loaded into the timer on start or, if operating in periodic mode, restart. The timer must be disabled for changes in load count to have any effect.	0xFFFF FFFF

9.6.2.2 Timer Current Value Register (Tn_CRNT_VAL)

Timer Current Value Registers are read only registers memory mapped at addresses 0x9300 0004 to 0x9300 0007 for timer[0] and 0x9300 0018 to 0x9300 001B for timer[1]. These registers are used to read the timer's current countdown value.

Table 108. Timer Current Value Register

Mnemonic	Access	Bits	Description	Reset
CRNT_VAL	R	31:0	Current Value: returns the timer's current countdown value.	0xFFFF FFFF

9.6.2.3 Timer Control Register (Tn_CTL_REG)

Timer Control Registers are readable and writable registers memory mapped at addresses 0x9300 0008 to 0x9300 000B for timer[0] and 0x9300 001C to 0x9300 001F for timer[1]. These registers are used to configure the operating mode, mask or unmask interrupts, and enable or disable the timers.

Table 109. Timer Control Register

Mnemonic	Access	Bits	Description	Reset
ENABLE	R/W	0	Enable: when '1', the timer is enabled.	0x0
MODE	R/W	1	Mode: when '1', the timer reloads from the Load Count register after counting down to zero. When '0', the time loads 0xFFFF FFFF after counting down to zero.	0x0
INT_MASK	R/W	2	Interrupt Mask: when '1', interrupts are masked. When '0', the CPU will be interrupted after counting down to zero.	0x0
-	-	31:3	Reserved	X

9.6.2.4 Timer End-of-Interrupt Register (Tn_EOI)

Timer End-of-Interrupt Registers are read only registers memory mapped at addresses 0x9300 000C to 0x930 0000F for timer[0] and 0x9300 0020 to 0x9300 0023 for timer[1]. These registers are used to clear interrupts.

Table 110. Timer End-of-Interrupt Register

Mnemonic	Access	Bits	Description	Reset
EOI	R	31:0	End-of-Interrupt: clears the timer interrupt and always returns zero.	0x0

9.6.2.5 Timer Interrupt Status Register (Tn_INTSTAT)

Timer Interrupt Status Registers are read only registers memory mapped at addresses 0x9300 0010 to 0x9300 0013 for timer[0] and 0x9300 0024 to 0x9300 0027 for timer[1]. These registers are used to read interrupt status.

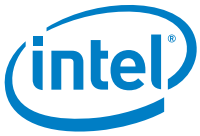


Table 111. Timer Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
INTSTAT	R	0	Interrupt Status: when '1', the timer has expired (i.e. counted down to zero) since interrupts were last cleared. This register only returns '1' if the interrupt is unmasked.	0x0

9.6.2.6 Timers Interrupt Status Register (TM_INTSTAT)

The Timers Interrupt Status Registers is a read only register memory mapped at addresses 0x9300 00A0 to 0x9300 00A3. This register is used to read interrupt status from both timers at once.

Table 112. Timers Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
INTSTAT0	R	0	Timer[0] Interrupt Status: when '1', the timer has expired (i.e. counted down to zero) since interrupts were last cleared. This register only returns '1' if the interrupt is unmasked.	0x0
INTSTAT1	R	0	Timer[1] Interrupt Status: when '1', the timer has expired (i.e. counted down to zero) since interrupts were last cleared. This register only returns '1' if the interrupt is unmasked.	0x0

9.6.2.7 Timers End-of-Interrupt Register (TM_EOI)

The Timers End-of-Interrupt Register is a read only registers memory mapped at addresses 0x9300 00A4 to 0x9300 00A7. This register is used to clear interrupts from both timers at once.

Table 113. Timer End-of-Interrupt Register

Mnemonic	Access	Bits	Description	Reset
EOI	R	31:0	End-of-Interrupt: clears any active timer interrupts and always returns zero.	0x0

9.6.2.8 Timers Raw Interrupt Status Register (TM_RAW_INTSTAT)

The Timers Raw Interrupt Status Registers is a read only register memory mapped at addresses 0x9300 00A8 to 0x9300 00AB. This register is used to read unmasked interrupt status from both timers at once.

Table 114. Timers Raw Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
RAW_INTSTAT0	R	0	Timer[0] Raw Interrupt Status: when '1', the timer has expired (i.e. counted down to zero) since interrupts were last cleared. The interrupt mask has no effect on this register.	0x0
RAW_INTSTAT1	R	0	Timer[1] Raw Interrupt Status: when '1', the timer has expired (i.e. counted down to zero) since interrupts were last cleared. The interrupt mask has no effect on this register.	0x0

9.6.2.9 Timers Component Version Register (TM_CMPVER)

The Timers Component Version Register is a read only register memory mapped at addresses 0x9300 00AC to 0x9300 00AF. This register is used to read the general purpose timers version code.

Table 115. Timers Version Code Register

Mnemonic	Access	Bits	Description	Reset
TM_CMPVER	R	31:0	Version Code Register: always returns the version code 2.05 as the big endian ASCII string "205*" or 0x3230 352A.	0x3230 352A

9.7 General Purpose I/O (GPIO)

The MCU features 24 general purpose I/O directly accessible by the CPU via memory mapped registers. Each general purpose I/O can be used as an input, a push-pull output, an open drain output, an open source output, or bidirectional. In addition, interrupts can be generated on a pin state or on a change of pin state. GPIO are ideally suited for use with devices such as relays, LEDs, switches, H-bridges and servos.

9.7.1 Functional Description

The GPIO peripheral features the following memory mapped registers for reading and writing pin state:

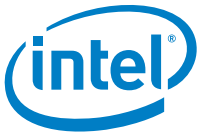
- Selecting input or output direction on each pin.
- Selecting the output state on each output pin
- Reading the input state on each input pin.

In addition, the GPIO peripheral can generate interrupts on the following conditions:

- If an input pin is high.
- If an input pin is low.
- If an input pin sees a rising edge.
- If an input pin sees a falling edge.

The GPIO peripheral features the following memory mapped registers for managing interrupts:

- Enabling or disabling interrupts from a pin.
- Masking interrupts from a pin.
- Configure the condition that results in an interrupt from a pin (i.e. high, low, rising, or falling).
- Synchronizing inputs from a pin.
- Reading the raw interrupt status.
- Reading the masked interrupt status.
- Clearing interrupts.



Finally, the GPIO peripheral features memory mapped registers for reading the ID code, version code, and configuration of the GPIO peripheral.

9.7.2 Registers

GPIO registers are aligned on 4-byte boundaries and can be accessed only as double words. Bit definitions are given in the following sections.

9.7.2.1 GPIO Data Register (DR)

The GPIO Data Register is a readable and writable register memory mapped at address 0x9000 0000 to 0x9000 0003. This register is used to set the state of output pins.

Table 116. GPIO Data Register

Mnemonic	Access	Bits	Description	Reset
DR	R/W	23:0	Data Register: when '1', the output state of GPIO[*n*] pin, where *n* corresponds to the bit number, is high. When '0', the output state of GPIO[*n*] pin is low. Reading returns the last value written. Note that a pin's function must be programmed as GPIO in its I/O Pin Function Register and it must be configured as an output in the GPIO Data Direction Register in order for the state programmed in this register to be driven out on to the pin.	0x0
-	-	31:24	Reserved	X

9.7.2.2 GPIO Data Direction Register (DDR)

The GPIO Data Direction Register is a readable and writable register memory mapped at address 0x9000 0004 to 0x9000 0007. This register is used to set the direction of GPIO pins.

Table 117. GPIO Data Direction Register

Mnemonic	Access	Bits	Description	Reset
DDR	R/W	23:0	Data Direction Register: when '1', the GPIO[*n*] pin, where *n* corresponds to the bit number, is an output. When '0', the GPIO[*n*] pin is an input. Reading returns the last value written. Note that a pin's function must be programmed as GPIO in its I/O Pin Function Register for this register to have any effect.	0x0
-	-	31:24	Reserved	X

9.7.2.3 GPIO Interrupt Enable Register (INTEN)

The GPIO Interrupt Enable Register is a readable and writable register memory mapped at address 0x9000 0030 to 0x9000 0033. This register is used to enable interrupts from a GPIO pin.

Table 118. GPIO Interrupt Enable Register

Mnemonic	Access	Bits	Description	Reset
INTEN	R/W	23:0	Interrupt Enable Register: when '1', interrupts from the GPIO[*n*] pin, where *n* corresponds to the bit number, are enabled. When '0', interrupts from the GPIO[*n*] pin are disabled. Reading returns the last value written. Note that a pin must be configured as an output in the GPIO Data Direction Register in order for it to generate interrupts. Also, pending interrupts cannot be cleared by disabling the interrupt from that pin.	0x0
-	-	31:24	Reserved	X

9.7.2.4 GPIO Interrupt Mask Register (INTMASK)

The GPIO Interrupt Mask Register is a readable and writable register memory mapped at address 0x9000 0034 to 0x9000 0037. This register is used to mask interrupts from a GPIO pin.

Table 119. GPIO Interrupt Mask Register

Mnemonic	Access	Bits	Description	Reset
INTMASK	R/W	23:0	Interrupt Mask Register: when '1', interrupts from the GPIO[*n*] pin, where *n* corresponds to the bit number, are masked. When '0', interrupts from the GPIO[*n*] pin are unmasked. Reading returns the last value written. Note that a pin must be configured as an output in the GPIO Data Direction Register and interrupts must be enabled in the GPIO Interrupt Enable Register in order for the pin to generate interrupts. Also, pending interrupts cannot be cleared by masking them.	0x0
-	-	31:24	Reserved	X

9.7.2.5 GPIO Interrupt Type Register (INTTYPE)

The GPIO Interrupt Type Register is a readable and writable register memory mapped at address 0x9000 0038 to 0x9000 003B. This register is used to select the type of interrupt (level or edge) from a GPIO pin.

Table 120. GPIO Interrupt Type Register

Mnemonic	Access	Bits	Description	Reset
INTTYPE	R/W	23:0	Interrupt Type Register: when '1', interrupts from the GPIO[*n*] pin, where *n* corresponds to the bit number, are edge sensitive. When '0', interrupts from the GPIO[*n*] pin are level sensitive. Reading returns the last value written. Note that a pin must be configured as an output in the GPIO Data Direction Register, interrupts must be enabled in the GPIO Interrupt Enable Registers, and unmasked in the GPIO Interrupt Mask Register in order for the pin to generate interrupts.	0x0
-	-	31:24	Reserved	X

9.7.2.6 GPIO Interrupt Polarity Register (INTPOL)

The GPIO Interrupt Polarity Register is a readable and writable register memory mapped at address 0x9000 003C to 0x9000 003F. This register is used to select the polarity of interrupt (high or low; rise or fall) from a GPIO pin.

Table 121. GPIO Interrupt Polarity Register

Mnemonic	Access	Bits	Description	Reset
INTPOL	R/W	23:0	Interrupt Polarity Register: when '1', interrupts from the GPIO[*n*] pin, where *n* corresponds to the bit number, are active when the pin is high or after a rising edge. When '0', interrupts from the GPIO[*n*] pin are active when the pin is low or after a falling edge. Reading returns the last value written. Note that a pin must be configured as an output in the GPIO Data Direction Register, interrupts must be enabled in the GPIO Interrupt Enable Registers, and unmasked in the GPIO Interrupt Mask Register in order for the pin to generate interrupts.	0x0
-	-	31:24	Reserved	X

9.7.2.7 GPIO Interrupt Status Register (INTSTATUS)

The GPIO Interrupt Status Register is a read only register memory mapped at address 0x9000 0040 to 0x9000 0043. This register is used to read the GPIO pin interrupt status.

Table 122. GPIO Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
INTSTATUS	R	23:0	Interrupt Status Register: when '1', the GPIO[*n*] pin, where *n* corresponds to the bit number, has generated an interrupt. When '0', the GPIO[*n*] pin has not generated an interrupt since last cleared. Note that a pin must be configured as an output in the GPIO Data Direction Register, interrupts must be enabled in the GPIO Interrupt Enable Registers, and unmasked in the GPIO Interrupt Mask Register in order for the pin to generate interrupts.	0x0
-	-	31:24	Reserved	X

9.7.2.8 GPIO Raw Interrupt Status Register (RAWSTATUS)

The GPIO Raw Interrupt Status Register is a read only register memory mapped at address 0x9000 0044 to 0x9000 0047. This register is used to read the raw GPIO pin interrupt status.

Table 123. GPIO Raw Interrupt Status Register

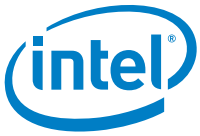
Mnemonic	Access	Bits	Description	Reset
RAWSTATUS	R	23:0	Raw Interrupt Status Register: when '1', the GPIO[*n*] pin, where *n* corresponds to the bit number, has detected an interrupt condition. When '0', the GPIO[*n*] pin has not detected an interrupt condition since last cleared. Note that a pin must be configured as an output in the GPIO Data Direction Register, and interrupts must be enabled in the GPIO Interrupt Enable Registers for interrupt conditions to be detected. The GPIO Interrupt Mask Register has no effect on this register.	0x0
-	-	31:24	Reserved.	X

9.7.2.9 GPIO End of Interrupt Register (EOI)

The GPIO End of Interrupt Register is a write only register memory mapped at address 0x9000 004C to 0x9000 004F. This register is used to clear the GPIO pin interrupt status.

Table 124. GPIO End of Interrupt Register

Mnemonic	Access	Bits	Description	Reset
EOI	W	23:0	End of Interrupt Register: when written '1', pending interrupts from the GPIO[*n*] pin, where *n* corresponds to the bit number, are cleared. When written '0', pending interrupts on the GPIO[*n*] pin are unaffected.	0x0
-	-	31:24	Reserved.	X



9.7.2.10 GPIO External Pin State Register (EXT)

The GPIO External Pin State Register is a read only register memory mapped at address 0x9000 0050 to 0x9000 0053. This register is used to read the GPIO external pin state.

Table 125. GPIO External Pin State Register

Mnemonic	Access	Bits	Description	Reset
EXT	R	23:0	External Pin State Register: when '1', the GPIO[*n*] pin, where *n* corresponds to the bit number, is high. When '0', the GPIO[*n*] pin is low.	0x0
-	-	31:24	Reserved.	X

9.7.2.11 GPIO Interrupt Synchronization Register (SYNC)

The GPIO Interrupt Synchronization Register is a readable and writable register memory mapped at address 0x9000 0060 to 0x9000 0063. This register is used to enable synchronizers for interrupts from a GPIO pin.

Table 126. GPIO Interrupt Synchronization Register

Mnemonic	Access	Bits	Description	Reset
SYNC	R/W	23:0	Interrupt Synchronization Register: when '1' interrupts from the GPIO[*n*] pin, where *n* corresponds to the bit number, are synchronized to *apb_clk* using double flip-flop synchronizers. When '0', interrupts from the GPIO[*n*] pin are not synchronized. Reading returns the last value written.	0x0
-	-	31:24	Reserved	X

9.7.2.12 GPIO Identification Code Register (ID)

The GPIO ID Code Register is a read only register memory mapped at address 0x9000 0064 to 0x9000 0067. This register is used to read the GPIO ID code.

Table 127. GPIO ID Code Register

Mnemonic	Access	Bits	Description	Reset
ID	R	31:0	ID Code Register: always returns zero.	0x0

9.7.2.13 GPIO Version Code Register (VER)

The GPIO Version Code Register is a read only register memory mapped at address 0x9000 0068 to 0x9000 006B. This register is used to read the GPIO version code.

Table 128. GPIO Version Code Register

Mnemonic	Access	Bits	Description	Reset
VER	R	31:0	Version Code Register: always returns the version code 2.08 as the big endian ASCII string "208*" or 0x3230 382A.	0x3230 382A

9.7.2.14 GPIO Configuration Register 1 (CONFIG1)

The GPIO Configuration Register 1 is a read only register memory mapped at address 0x9000 0074 to 0x9000 0077. This register is used to read the peripheral's hardware configuration.

Table 129. GPIO Configuration Register 1

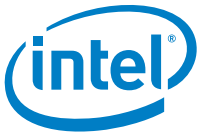
Mnemonic	Access	Bits	Description	Reset
APB_DATA_WIDTH	R	1:0	APB Data Width: always returns 2.	0x2
NUM_PORTS	R	3:2	Number of Ports: always returns 1.	0x1
PORTA_SINGLE_CTL	R	4	Port A Single Control: always returns 1.	0x1
PORTB_SINGLE_CTL	R	5	Port B Single Control: always returns 1.	0x1
PORTC_SINGLE_CTL	R	6	Port C Single Control: always returns 1.	0x1
PORTD_SINGLE_CTL	R	7	Port D Single Control: always returns 1.	0x1
HW_PORTA	R	8	Port A Hardware Control: always returns 0.	0x0
HW_PORTB	R	9	Port B Hardware Control: always returns 0.	0x0
HW_PORTC	R	10	Port C Hardware Control: always returns 0.	0x0
HW_PORTD	R	11	Port D Hardware Control: always returns 0.	0x0
PORTA_INTR	R	12	Port A Interrupts: always returns 1.	0x1
DEBOUNCE	R	13	Port A has De-bounce: always returns 0.	0x0
ADD_ENCODED_PARAMS	R	14	Configuration Registers are Present: always returns 1.	0x1
GPIO_ID	R	15	ID Code Register Present: always returns 1.	0x1
ENCODED_ID_WIDTH	R	20:16	ID Code Register Width: always returns 31.	0x31
-	-	31:21	Reserved	X

9.7.2.15 GPIO Configuration Register 2 (CONFIG2)

The GPIO Configuration Register 2 is a read only register memory mapped at address 0x9000 0070 to 0x9000 0073. This register is used to read the peripheral's hardware configuration.

Table 130. GPIO Configuration Register 2

Mnemonic	Access	Bits	Description	Reset
ENCODED_ID_PWIDTH_A	R	4:0	Port A Width: always returns 1.	0x24
ENCODED_ID_PWIDTH_B	R	9:5	Port B Width: always returns 1.	0x8
ENCODED_ID_PWIDTH_C	R	14:10	Port C Width: always returns 1.	0x8
ENCODED_ID_PWIDTH_D	R	19:15	Port D Width: always returns 1.	0x8
-	-	31:20	Reserved	X



9.8 Serial Peripheral Interface (SPI) Master

The MCU features a SPI master peripheral that supports a variety of 3- and 4-wire serial protocols making it easy to connect with most popular SPI devices. Software can choose from Motorola* SPI, Texas Instruments* Synchronous Serial Protocol (SSP) or National Semiconductor* Microwire transfer types on a per transfer basis. Software can also choose from transmit only, receive only, transmit-receive, and EEPROM transfer modes. Other features and benefits include:

- Master collision, transmit FIFO overflow, transmit FIFO empty, receive FIFO full, receive FIFO underflow, and receive FIFO overflow interrupts. All can be masked independently.
- Interrupt or polled-mode operation.
- Multi-master contention detection informs the processor of multiple serial-master accesses on the serial bus.
- Programmable baud rates up to 16.5 Mbps.
- Programmable word size from 4-16 bits.
- Eight word transmit and receive FIFO buffers with programmable interrupt thresholds.
- Four automatically generated slave-select outputs.
- Independent SPI clock allows CPU clock frequency changes without affecting in progress transfers.
- Component version register.

9.8.1 Functional Description

This peripheral is a full duplex SPI master. This section describes its functional operation. The Intel® Quark™ microcontroller D1000 pin names used in this section map to commonly used alternative pin names as shown in Table 131.

Table 131. Intel® Quark™ microcontroller D1000 to commonly used alternative pin name mapping

Intel® Quark™ microcontroller D1000 pin name	Defacto industry standard pin name
MST_M2SC	SCLK
MST_M2SD	MOSI
MST_M2SS	SS
MST_S2MD	MISO

9.8.1.1 SPI Overview

In order to connect this SPI peripheral to a serial-slave device, the device must have a least one of the following interfaces:

- Motorola Serial Peripheral Interface (SPI) – a 4-wire, full-duplex serial protocol, which uses an enveloping slave select.
- Texas Instruments Serial Protocol (SSP) – a 4-wire, full-duplex serial protocol, which uses a pulsed slave select.
- National Semiconductor Microwire – a 3- or 4-wire half-duplex serial protocol, which uses a control word transmitted from the serial master to the target serial slave.

The software selects which protocol is used by programming the FRF (frame format) bit field in the Control Register 0 (CTRLR0). The software also selects which slave(s) is involved in the transfer by programming the Slave Enable (SER) register. Other registers and bit fields are provided to control protocol specific options.

This SPI master features four mutually exclusive slave select outputs that assert automatically during a transfer. A schematic diagram of a four slave system is shown in Figure 19. In systems with more than four slaves, three of the slave select outputs can be configured as general purpose I/O (GPIO) and used along with an external one-of-eight decoder such as the 74LVC138 to select up to eight slaves. A schematic diagram of an eight slave system is shown in Figure 20. Additional GPIOs can be used to increase the number of slaves to 16 or more. Finally, if some of the slaves are programmable devices such as micro controllers or FPGAs, one slave select can be bussed to multiple slaves and a command word prefixed to each transfer addresses the targeted slave. The unaddressed slaves must keep their MISO output tri-stated and ignore their MOSI input. A schematic diagram of such a system is shown in Figure 21.

Figure 19. Schematic diagram for connecting up to four SPI slaves.

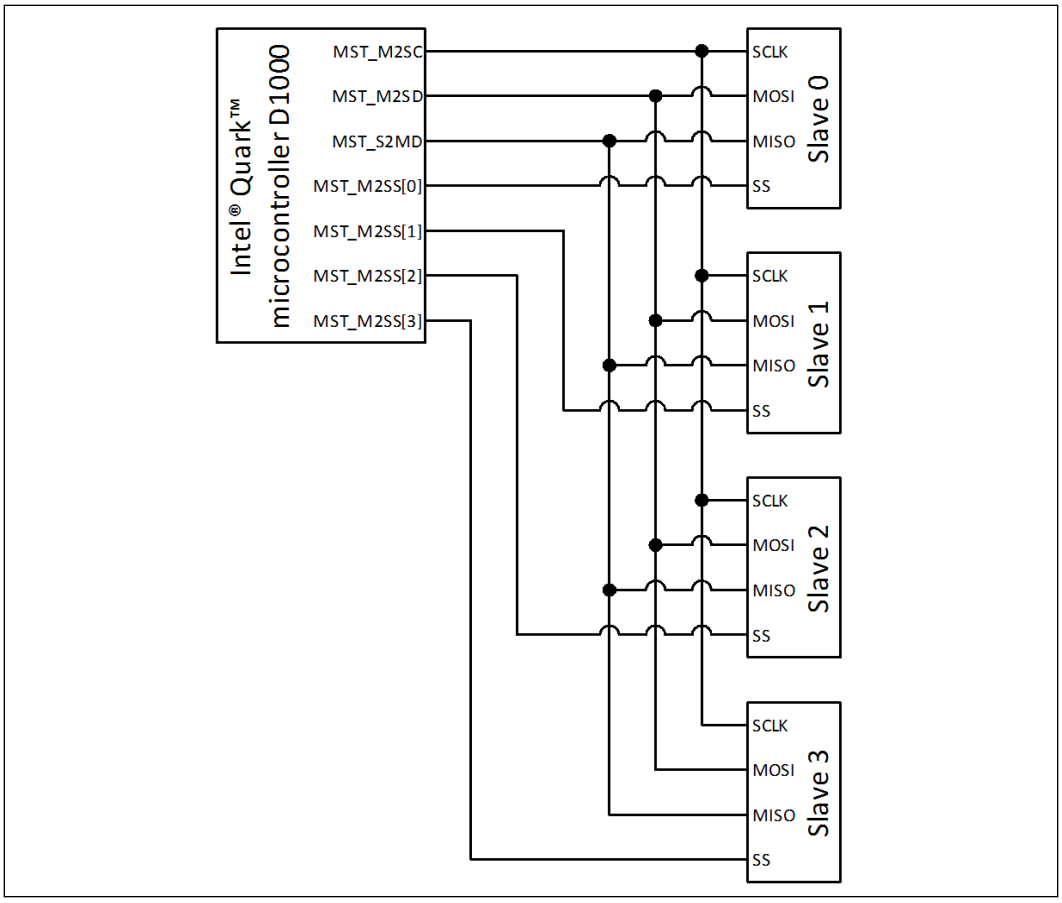


Figure 20. Schematic diagram for connecting up to eight SPI slaves

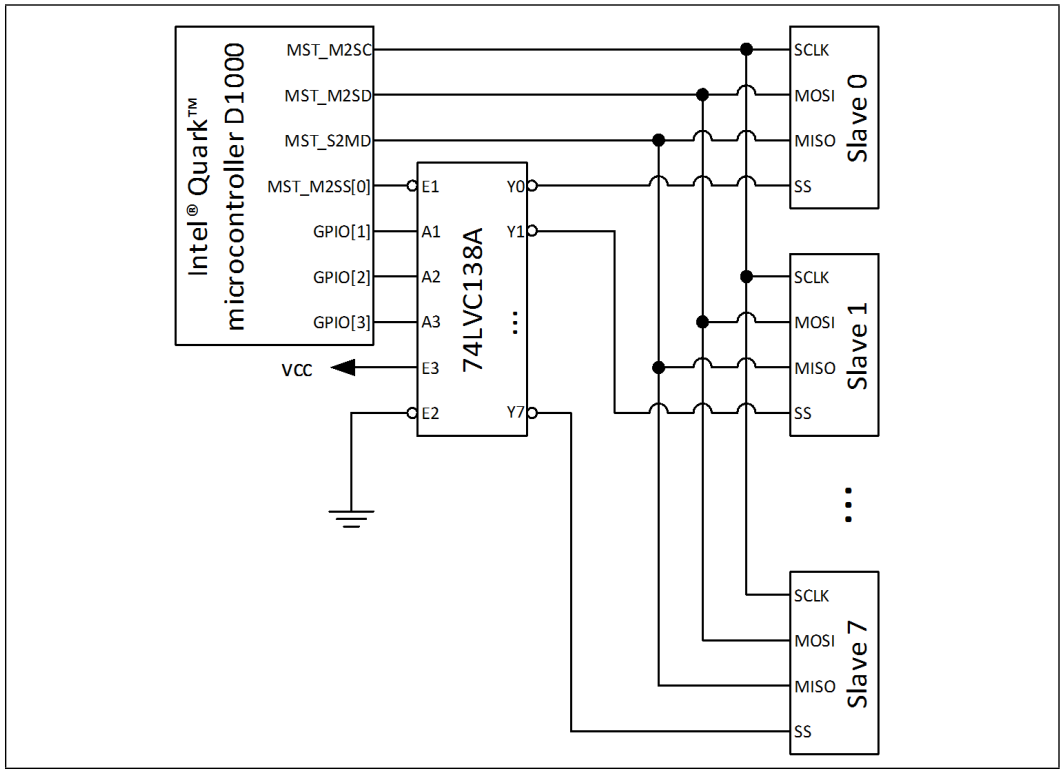
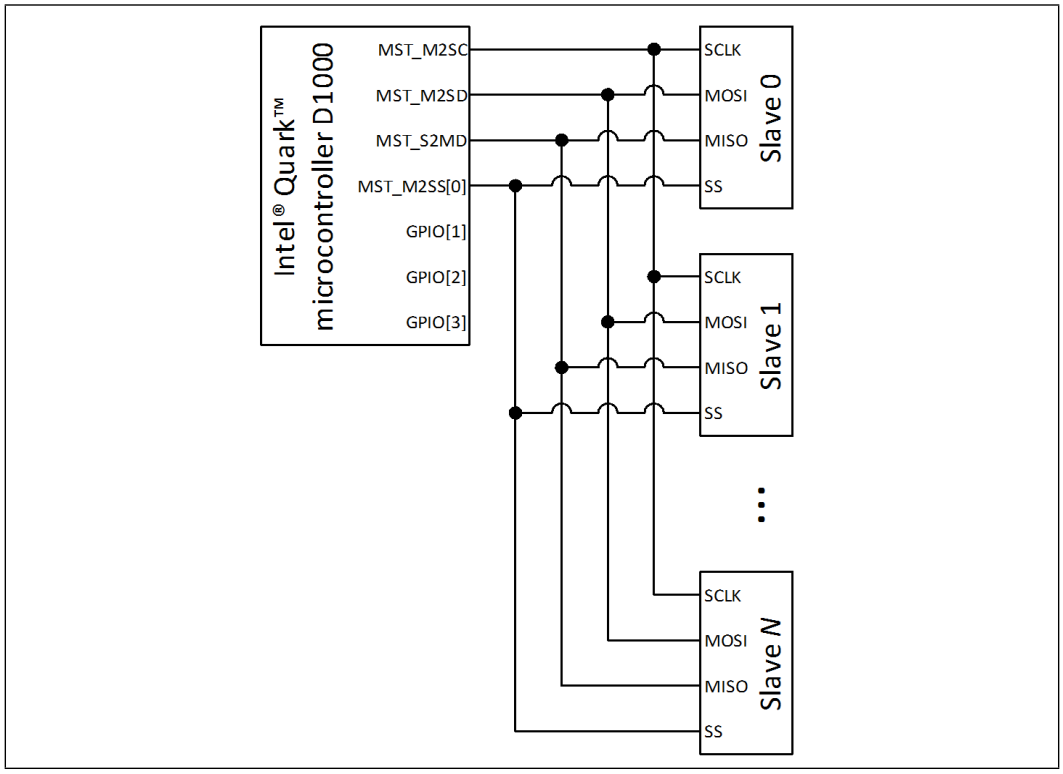


Figure 21. Schematic diagram for connecting any number of programmable SPI slaves



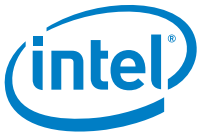
9.8.1.2 Clock Ratios

The frequency of the SPI master clock (*mst_spi_clk*) must be equal to or less than the APB clock (*apb_clk*). The frequency of the MST_M2SC clock may be from one half to one 65,534th of the SPI master clock frequency. The MST_M2SC clock frequency is programmed using the BAUDR register.

9.8.1.3 Transmit and Receive FIFO Buffers

This SPI master peripheral features eight word transmit and receive FIFOs. The following table shows the differences between transmit and receive FIFOs.

Transmit FIFO	Receive FIFO
Transmit data are popped from the transmit FIFO and serialized.	Receive data are de-serialized and pushed into the receive FIFO.
Registers are provided to set transmit FIFO empty threshold levels.	Registers are provided to set receive FIFO full threshold levels.
When data in the transmit FIFO reaches the threshold level or below, an interrupt is generated.	When data in the receive FIFO reaches the threshold level or above, an interrupt is generated.
Additional interrupts are provided to indicate when the CPU has written too much data overflowing the transmit FIFO.	Additional interrupts are provided to indicate when the CPU has read too much data underrunning the receive FIFO or the deserialization logic has pushed the data faster than the CPU has read the data overflowing the receive FIFO.



Data words are always right justified in the FIFOs. Both FIFOs are flushed when this peripheral is disabled using the SSI_EN register.

9.8.1.4 Interrupts

This peripheral features the following maskable interrupts:

- Transmit FIFO Empty Interrupt – Set when the transmit FIFO is equal to or below its threshold value and requires service to prevent an under-run. The threshold value, set through a software-programmable register, determines the level of transmit FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are written into the transmit FIFO buffer, bringing it over the threshold level.
- Transmit FIFO Overflow Interrupt – Set when the CPU attempts to write into the transmit FIFO after it has been completely filled. Data written while the FIFO is full, is discarded. This interrupt is cleared by reading the transmit FIFO overflow interrupt clear register (TXOICR).
- Receive FIFO Full Interrupt – Set when the receive FIFO is above its threshold value and requires service to prevent an overflow. The threshold value, set through a software-programmable register, determines the level of receive FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are read from the receive FIFO buffer, bringing it below the threshold level.
- Receive FIFO Overflow Interrupt – Set when the receive logic attempts to place data into the receive FIFO after it has been completely filled. Data received while the FIFO is full are discarded. This interrupt is cleared by reading the receive FIFO overflow interrupt clear register (RXOICR).
- Receive FIFO Underflow Interrupt – Set when the CPU attempts to read from the receive FIFO when it is empty. Zeros are read back from the receive FIFO when empty. This interrupt is cleared by reading the receive FIFO underflow interrupt clear register (RXUICR).
- Multi-Master Contention Interrupt – Set when MST_S2M_SS is asserted indicating that another master is actively transferring data on the bus. This informs the processor of possible contention. This interrupt is cleared by reading the multi-master interrupt clear register (MSTICR).

9.8.1.5 Transfer Modes

This SPI peripheral features the transfer modes discussed in this section. Transfer modes are selected in the CTRLR0.TMOD register bit field for all protocols except Microwire. For Microwire, the transfer mode is selected in the MWCR register.

9.8.1.5.1 Transmit and Receive

When TMOD = 2‘b00, both transmit and receive data are valid. The full-duplex data transfer occurs as normal according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the MST_M2SD line to the target device, which replies with data on the MST_S2MD line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

9.8.1.5.2 Transmit Only

When TMOD = 2‘b01, the receive data are invalid and should not be stored in the receive FIFO. The half-duplex data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the MST_M2SD line to the target device, and data on the MST_S2MD line are ignored. Interrupts originating from the receive logic should be masked in this mode.

9.8.1.5.3 Receive Only

When TMOD = 2'b10, the transmit data are invalid. The half-duplex data transfer occurs as normal according to the selected frame format (serial protocol). After the first write to the transmit FIFO, the same word is retransmitted for the duration of the transfer. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. Interrupts originating from the transmit logic should be masked in this mode.

9.8.1.5.4 EEPROM Read

When TMOD = 2'b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. Typically this takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). During the transmission of the opcode and address, no data is captured by the receive logic (as long as this SPI master is transmitting data on the MST_M2SD line, data on the MST_S2MD line is ignored). This master continues to transmit data until the transmit FIFO is empty. Therefore, the only data in the transmit FIFO should be the opcode and address to the EEPROM. If more data are in the transmit FIFO than are needed, then read data is lost.

When the transmit FIFO becomes empty (all control information has been sent), data on the MST_S2MD line is stored in the receive FIFO; the MST_M2SD output is held at a constant logic level. The serial transfer continues until the number of data frames specified in the CTRL1.NDF register bit field are received.

Note: EEPROM read mode is not supported when this SPI peripheral is configured for SSP protocol.

9.8.1.6 Master Contention Input

This SPI peripheral features a serial slave select input, MST_S2M_SS, that can be used to inform this master that another serial master is active on the bus. When this input is active—the active level depends on the serial protocol—the master remains in an IDLE state and holds off any pending serial transfer until the MST_S2M_SS input is returned to an in-active level.

If this SPI peripheral is the only master device on the serial bus, glue logic may be needed to control the logic level on the master MST_S2M_SS input. Glue logic is required if both of the following are true:

- Serial protocol changes dynamically.
- One of the protocols being used is SSP

Glue logic is not required under either of the following conditions:

- If the serial protocol does not change.
- If the serial protocol changes, but does not include the SSP protocol.

Under these conditions, the MST_S2M_SS signal can be tied high or low depending on which serial protocol you use.

- If the serial protocol is SPI or MicroWire, the MST_S2M_SS signal should be tied high.
- If the serial protocol being used is SSP, the MST_S2M_SS signal should be tied low.

9.8.1.7 Serial Protocols

This SPI peripheral features the serial protocols discussed in this section. Serial protocol is selected in the CTRLR0. FRF register bit field.

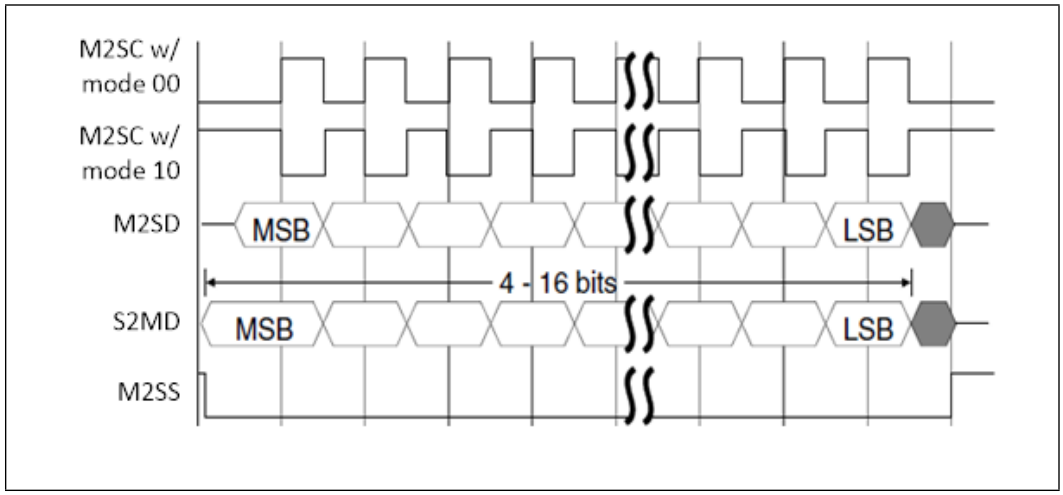
9.8.1.7.1 Motorola Serial Peripheral Interface (SPI)

When Motorola SPI protocol is selected, the serial clock phase (CTRL0.SCPH) register bit field determines which clock edge coincides with data transmission and the serial clock polarity (CTRL0.SCPOL) register bit field determines whether the inactive state of the serial clock is high or low. To transmit data, both master and slave SPI peripherals must have identical serial clock phase and clock polarity. The data frame can be 4 to 16 bits in length.

When the register bit field CTRL0.SCPH = 0, data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the M2SD and S2MD lines prior to the first serial clock edge. Figure 22 shows a timing diagram for a single SPI data transfer in modes 00 and 10.

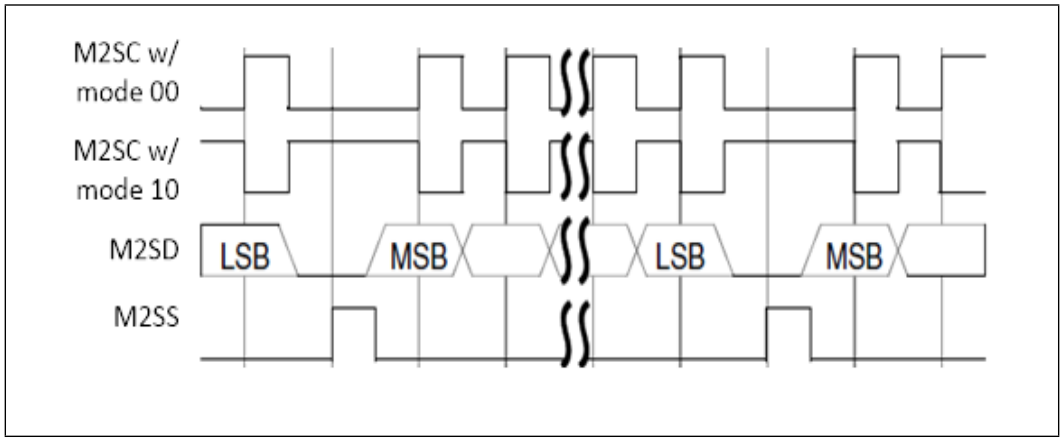
Note: Mode is an industry standard term meaning {SCPOL, SCPH}.

Figure 22. Motorola SPI serial protocol for a single transfer (SCPH = 0)



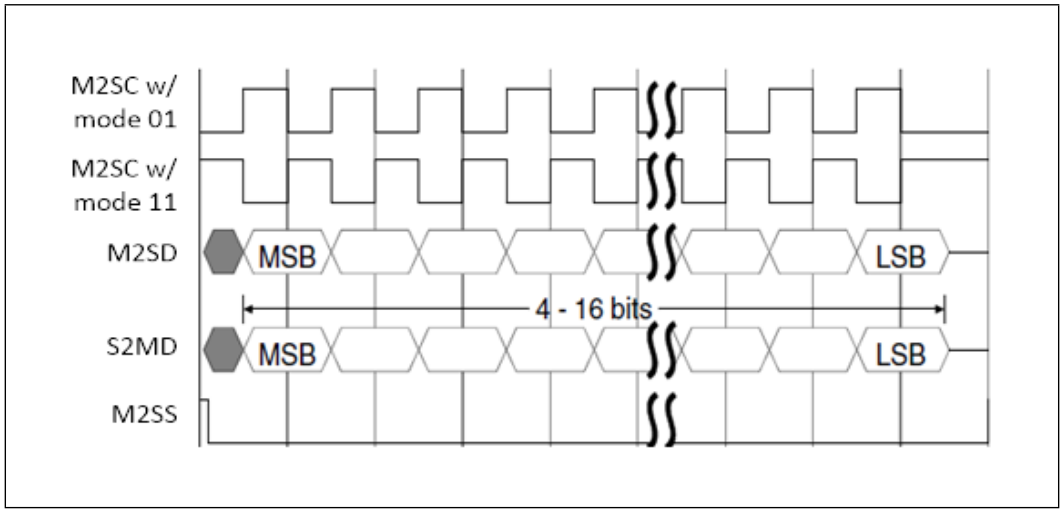
Since data transmission starts on the falling edge of the slave select signal when CTRL0.SCPH = 0, continuous data transfers require the slave select signal to toggle before beginning the next data frame. This is illustrated in Figure 23.

Figure 23. Motorola SPI serial protocol for continuous transfers (SCPH = 0)



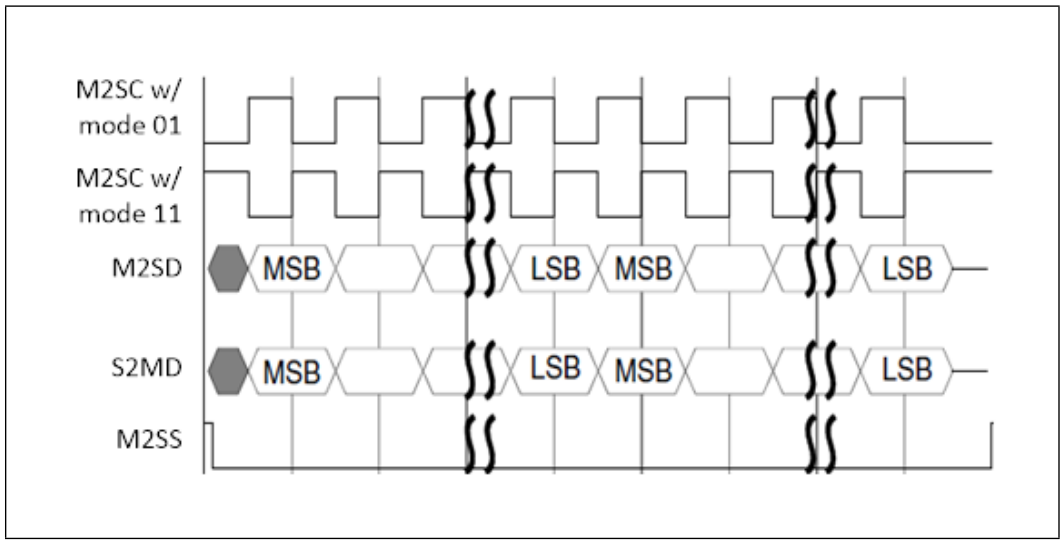
When the register bit field CTRL0.SCPH = 1, both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the master and slave peripherals on the leading edge of the serial clock. During continuous data frame transfers, the slave select line may be held active-low until the last bit of the last frame has been captured. Figure 24 shows the timing diagram for the SPI format when the configuration parameter CTRL0.SCPH = 1.

Figure 24. Motorola SPI serial protocol for a single transfer (SCPH = 1)



Continuous data frames are transferred in the same way as single frames, with the MSB of the next frame following directly after the LSB of the current frame. The slave select signal is held active for the duration of the transfer. Figure 25 shows the timing diagram for continuous SPI transfers when the configuration parameter CTRL0.SCPH = 1.

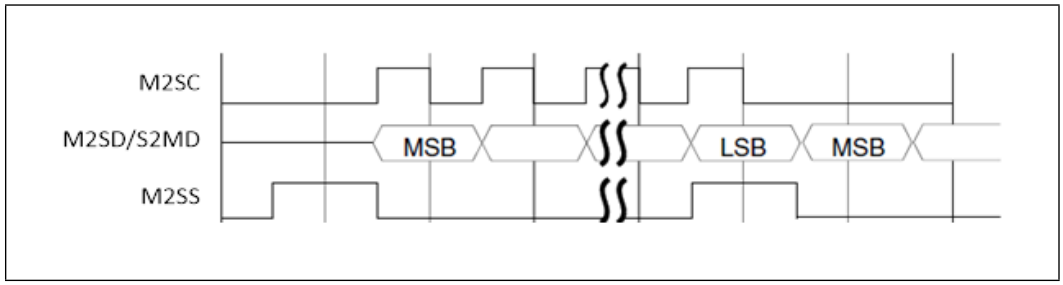
Figure 25. Motorola SPI serial protocol for continuous transfers (SCPH = 1)



9.8.1.7.2 Texas Instruments Synchronous Serial Protocol (SSP)

Data transfers begin by asserting the frame indicator line MST_M2SS for one serial clock period. Data to be transmitted are driven onto the MST_M2SD line one serial clock cycle later. Similarly data from the slave are driven onto the MST_S2MD line. Data are propagated on the rising edge of the serial clock MST_M2SC and captured on the falling edge. The length of the data frame ranges from 4 to 16 bits and frames may follow immediately after each other. Texas Instruments SPP is transferred. Figure 26 shows the timing diagram for the SSP serial transfer.

Figure 26. Texas Instruments SSP transfer



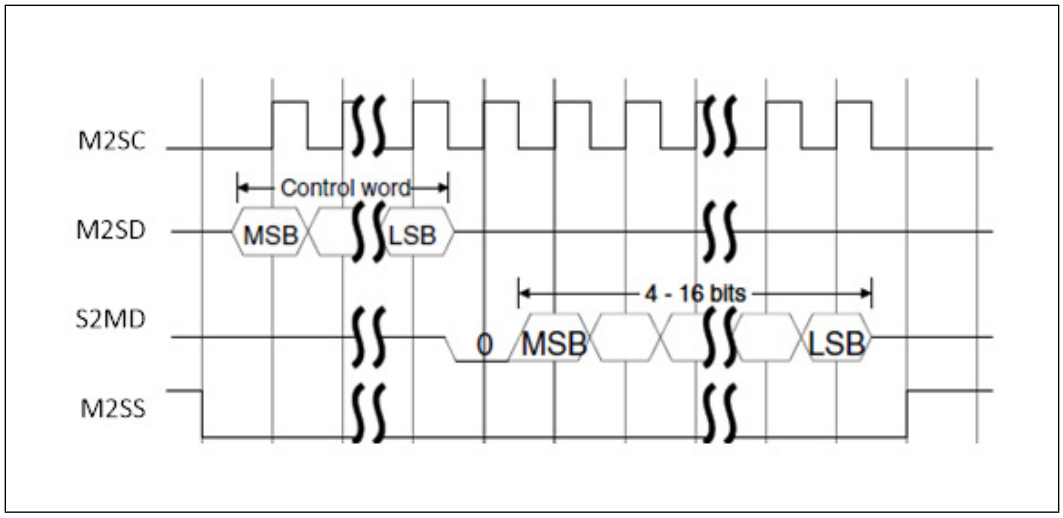
9.8.1.7.3 National Semiconductor Microwire

Data transmission begins with the falling edge of the slave-select signal (MST_M2SS). One-half serial clock (MST_M2SC) period later, the first bit of the control is sent out on the MST_M2SD line. The length of the control word can be in the range 1 to 16 bits and is set by writing register bit field CTRL0.CFS. The remainder of the control word is transmitted (propagated on the falling edge of MST_M2SC) by this SSP master. During this transmission, no data are present (high impedance) on the MST_S2MD line.

The direction of the data word is controlled by the MWCR.MDD register bit field. When MWCR.MDD = 0, this master receives data from the external serial slave. One clock cycle after the LSB of the control word is transmitted, the slave peripheral responds with a dummy 0 bit, followed by the data frame, which can be 4 to 16 bits in length. Data are propagated on the falling edge of the serial clock and captured on the rising edge.

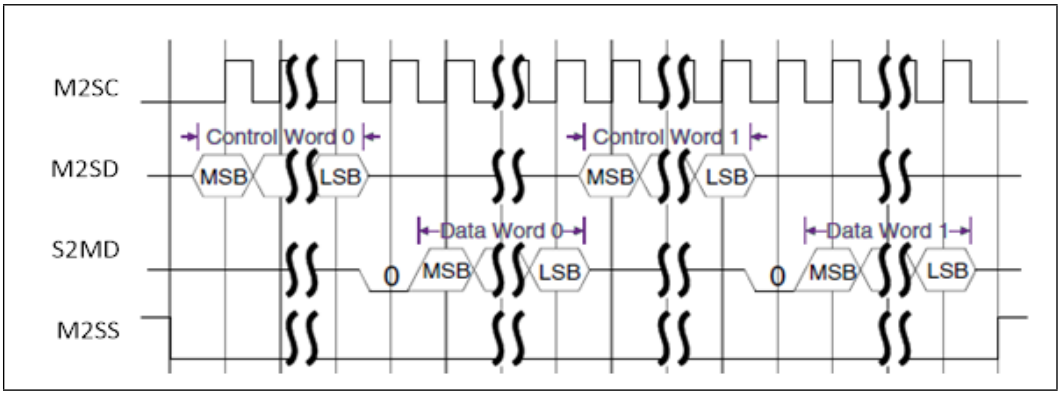
The slave-select signal is held active-low during the transfer and is de-asserted one-half clock cycle later, after the data are transferred. Figure 27 shows the timing diagram for a single read transfer.

Figure 27. National Semiconductor Microwire single read transfer (MHS = 0, MDD = 0, MWMOD = 0)



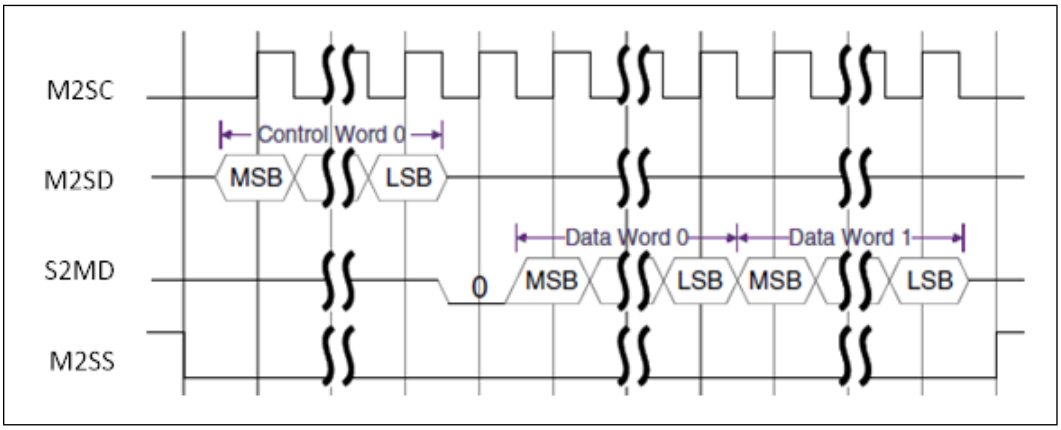
Continuous transfers can be sequential or non-sequential, and are controlled by the MCWR.MWMOD register bit. Non-sequential continuous read transfers occur as illustrated in Figure 27, with the control word for the next transfer following immediately after the LSB of the received data word. The only modification needed to perform a continuous non-sequential transfer is to write more control words into the transmit FIFO buffer.

Figure 28. National Semiconductor Microwire continuous non-sequential read transfer (MHS = 0, MDD = 0, MWMOD = 0)



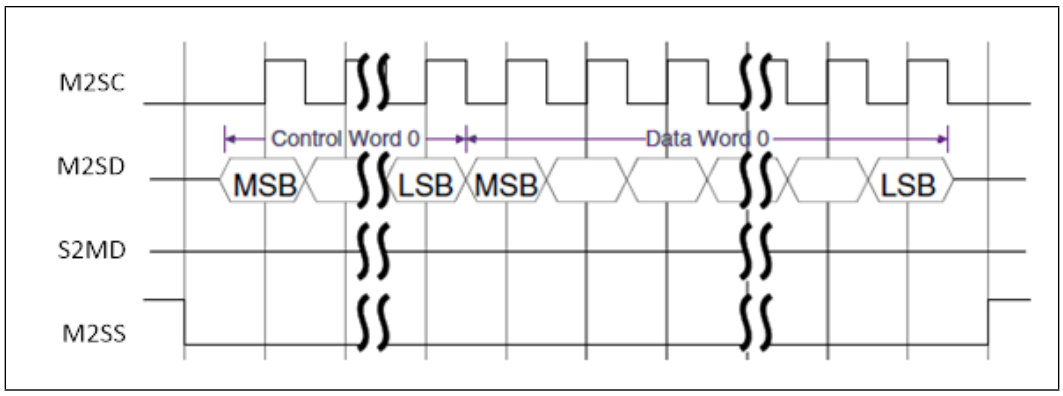
During sequential continuous transfers, only one control word is transmitted from this master. The transfer is started in the same manner as with non-sequential read operations, but the cycle is continued to read further data. The slave device automatically increments its address pointer to the next location and continues to provide data from that location. Any number of locations can be read in this manner. This master terminates the transfer when the number of words received is equal to the value in the CTRLR1.NDF register bit field plus 1. The timing diagram in Figure 29 shows a continuous sequential read of two data frames.

Figure 29. National Semiconductor Microwire continuous sequential read transfer (MHS = 0, MDD = 0, MWMOD = 1)



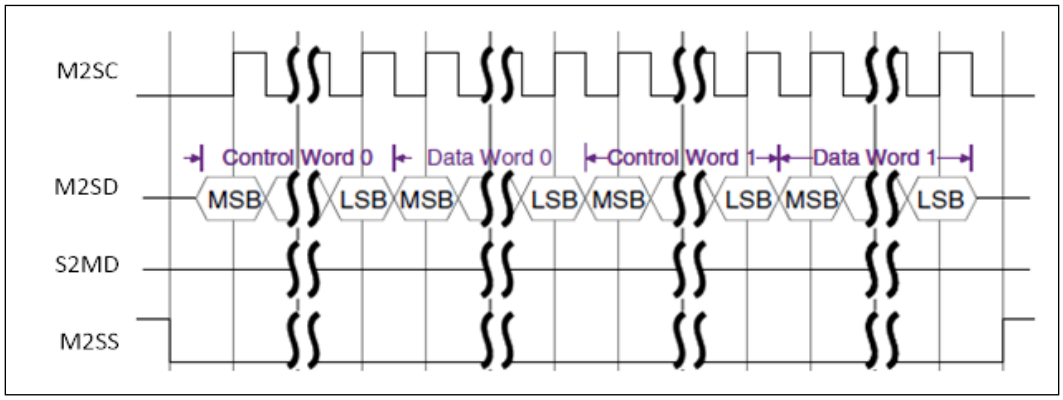
When MCWR.MDD = 1, this master transmits data to the external serial slave. Immediately after the LSB of the control word is transmitted, this master begins transmitting the data frame to the slave peripheral. Figure 30 shows the timing diagram for a single write transfer.

Figure 30. National Semiconductor Microwire single write transfer (MHS = 0, MDD = 1, MWMOD = 0)



Continuous transfers occur as shown in Figure 31, with the control word for the next transfer following immediately after the LSB of the write data word. This example shows two data words are written to the external serial slave device. The only modification needed to perform a continuous transfer is to write more control and data words into the transmit FIFO buffer.

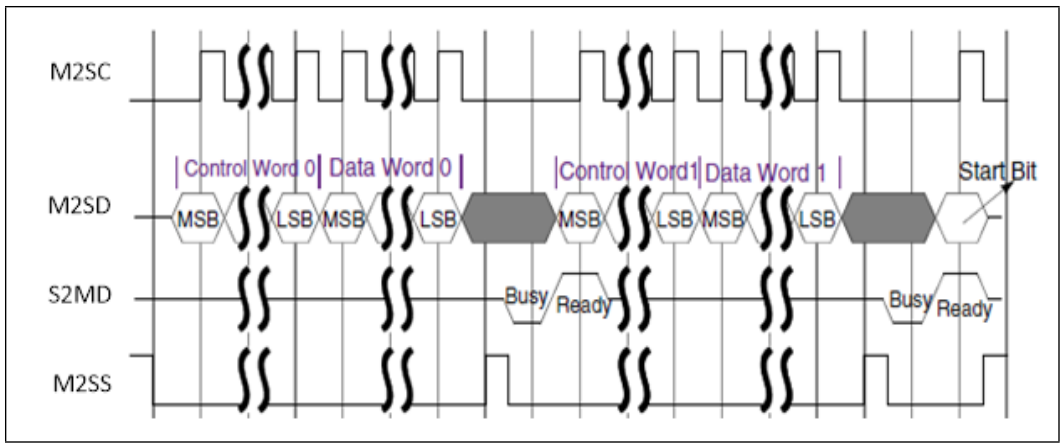
Figure 31. National Semiconductor Microwire continuous write transfer (MHS = 0, MDD = 1, MWMOD = 0)



The Microwire handshaking interface can also be enabled for master write operations to external serial-slave devices. To enable the handshaking interface, you must write 1 into the MWCR.MHS register bit. When MWCR.MHS is set to 1, this master checks for a ready status from the slave device before completing the transfer, or transmitting the next control word for continuous transfers.

Figure 32 shows an example of a continuous Microwire transfer with the handshaking interface enabled. After the first data word has been transmitted to the serial-slave device, this master polls the MST_S2MD input waiting for a ready status from the slave device. Upon reception of the ready status, this master begins transmission of the next control word. After transmission of the last data frame has completed, this master transmits a start bit to clear the ready status of the slave device before completing the transfer.

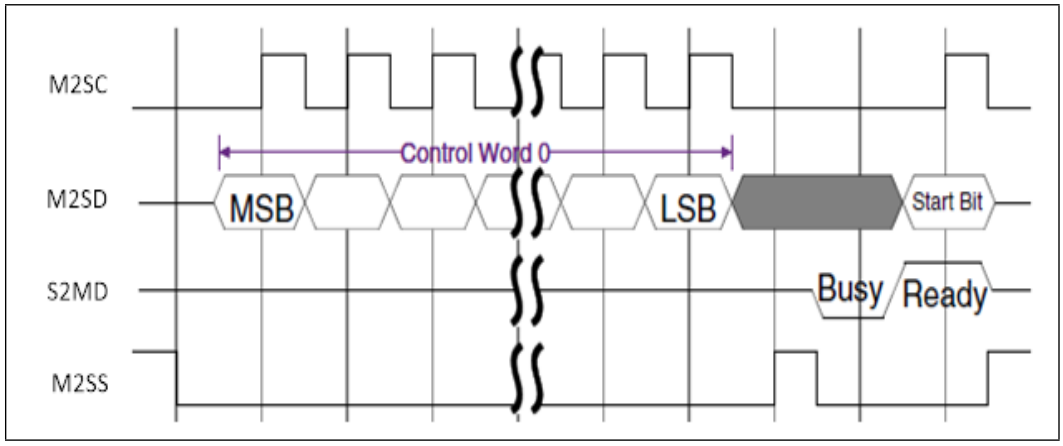
Figure 32. National Semiconductor Microwire continuous write transfer with handshaking (MHS = 1, MDD = 1, MWMOD = 0)



To transmit a control word (not followed by data) to a serial-slave device from this master, there must be only one entry in the transmit FIFO buffer. It is impossible to transmit two control words in a continuous transfer, as the shift logic in this Microwire peripheral treats the second control word as a data word. When this master transmits only a control word, the MWCR.MDD register bit must be set (1).

In the example shown in the timing diagram of the Figure 33, the handshaking interface is enabled. If the handshaking interface is disabled (MHS=0), the transfer is terminated by this master one MST_M2SC clock cycle after the LSB of the control word is captured by the slave device.

Figure 33. National Semiconductor Microwire control word transfer (MHS = 1, MDD = 1, MWMOD = 0)



9.8.1.8 Data Transfers

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

Note: The BUSY status is not set when data are written into the transmit FIFO. This bit gets set only when the target slave has been selected and the transfer is underway. After writing data into the transmit FIFO, the shift logic does not begin the serial transfer until a positive edge of the MST_M2SC signal is present. The delay in waiting for this positive edge depends on the baud rate of the serial transfer. Before polling the BUSY status, software should first poll the transmit FIFO empty status (SR.TFE) or wait for at least one MST_M2SC clock cycle.

9.8.1.8.1 Master SPI and SSP Serial Transfers

The SPI and SSP serial protocols are described in Section 9.8.1.7.1 and Section 9.8.1.7.2 respectively. They include timing diagrams for the various transfers. The transfer modes supported by this SPI master are described in Section 9.8.1.5 above.

When the transfer mode is “transmit and receive” or “transmit only” (TMOD = 2'b00 or TMOD = 2'b01 respectively), transfers are terminated by the shift control logic when the transmit FIFO is empty. For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level (TXFTLR) can be used to interrupt the processor before the transmit FIFO empties.

When the transfer mode is “receive only” (TMOD = 2'b10), a serial transfer is started by writing one “dummy” data word into the transmit FIFO when a serial slave is selected. The MST_M2SD output from tis master is held at a constant logic level for the duration of the serial transfer. The transmit FIFO is popped only once at the beginning and may remain empty for the duration of the serial transfer. The end of the serial transfer is controlled by the “number of data frames” (CTRLR1.NDF) register bit field.

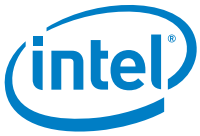
If, for example, you want to receive 24 data frames from a serial-slave peripheral, you should program the NDF field with the value 23. The receive logic terminates the serial transfer when the number of frames received is equal to the NDF value plus 1. The receive FIFO buffer should be read each time the receive FIFO generates a FIFO full interrupt request to prevent an overflow. The receive FIFO threshold level (RXFTLR) can be used to interrupt the processor before the receive FIFO fills.

When the transfer mode is “eeprom_read” (TMOD = 2‘b11), a serial transfer is started by writing the opcode and/or address into the transmit FIFO when a serial slave (EEPROM) is selected. The opcode and address are transmitted to the EEPROM device, after which read data is received from the EEPROM device and stored in the receive FIFO. The end of the serial transfer is controlled by the CTRLR.NDF register bit field.

Note: EEPROM read mode is not supported when this master is configured to be in the SSP mode.

A typical software flow for completing an SPI or SSP serial transfer from this master is outlined as follows:

1. If this master is enabled, disable it by writing 0 to the SSI Enable register (SSIENR).
2. Set up the control registers for the transfer. These registers can be set in any order.



- Write Control Register 0 (CTRLR0). For SPI transfers, the serial clock polarity and serial clock phase parameters must be set identical to target slave device.
- If the transfer mode is receive only, write CTRLR1 (Control Register 1) with the number of frames in the transfer minus 1. For example, if you want to receive four data frames, write this register with 3.
- Write the Baud Rate Select Register (BAUDR) to set the baud rate for the transfer.
- Write the Transmit and Receive FIFO Threshold Level registers (TXFTLR and RXFTLR, respectively) to set FIFO threshold levels.
- Write the IMR register to set up interrupt masks.
- The Slave Enable Register (SER) register can be written here to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the Data Register (DR), the transfer does not begin until a slave is enabled.

3. Enable this master by writing 1 to the SSIENR register.
4. Write data for transmission to the target slave into the transmit FIFO (write DR). If no slaves were enabled in the SER register at this point, enable it now to begin the transfer.
5. If the transfer is larger than eight frames, interrupts should be enabled to keep the transmit FIFO from emptying and the receive FIFO from overflowing.

   •If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR).

   •If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
6. The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is receive only (TMOD = 2'b10), the transfer is stopped by the shift control logic when the specified number of frames have been received. When the transfer is done, the BUSY status is reset to 0. Poll the BUSY status to wait for completion of the transfer, but do not poll it prior to at least one MST_M2SC clock cycle from the start of the transfer.
7. If the transfer mode is not transmit only (TMOD != 01), read the receive FIFO until it is empty.
8. Disable this master by writing 0 to SSIENR.

9.8.1.8.2 Master Microwire Serial Transfers

The Microwire serial protocol is described in Section 9.8.1.8.2. It includes timing diagrams for the various transfers.

Microwire serial transfers from this master are controlled by the Microwire Control Register (MWCR). The MWHS bit field enables and disables the Microwire handshaking interface. The MDD bit field controls the direction of the data frame (the control frame is always transmitted by the master and received by the slave). The MWMOD bit field defines whether the transfer is sequential or non-sequential.

All Microwire transfers are started by this master when there is at least one control word in the transmit FIFO and a slave is enabled. When this master transmits the data frame (MDD = 1), the transfer is terminated by the shift logic when the transmit FIFO is empty. When this master receives the data frame (MDD = 1), the termination of the transfer depends on the setting of the MWMOD bit field. If the transfer is non-sequential (MWMOD = 0), it is terminated when the transmit FIFO is empty after shifting in the data frame from the slave. When the transfer is sequential (MWMOD = 1), it is terminated by the shift logic when the number of data frames received is equal to the value in the CTRLR1 register plus 1.

When the handshaking interface on this master is enabled (MWHS =1), the status of the target slave is polled after transmission. Only when the slave reports a ready status does this master complete the transfer and clear its BUSY status. If the transfer is continuous, the next control/data frame is not sent until the slave device returns a ready status.

A typical software flow for completing a Microwire serial transfer from this master is outlined as follows:

1. If the SPI Master is enabled, disable it by writing 0 to SSIENR.
2. Set up the SPI Master control registers for the transfer. These registers can be set in any order.
   - Write CTRLR0 and MWCR to set transfer parameters.
   - If the transfer is sequential and this master receives data, write CTRLR1 with the number of frames in the transfer minus 1. For example, if you want to receive four data frames, write this register with 3.
   - Write BAUDR to set the baud rate for the transfer.
   - Write TXFTLR and RXFTLR to set FIFO threshold levels.
   - Write the IMR register to set up interrupt masks.
   - Write the SER register to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the DR register, the transfer does not begin until a slave is enabled.
3. Enable this master by writing 1 to the SSIENR register.
4. If this master transmits data, write the control and data words into the transmit FIFO (write DR). If this master receives data, write the control word(s) into the transmit FIFO. If no slaves were enabled in the SER register at this point, enable now to begin the transfer.
5. If the transfer is larger than eight frames, interrupts should be enabled to keep the transmit FIFO from emptying and the receive FIFO from overflowing.
   - If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR).
   - If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
6. The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is sequential and this master receives data, the transfer is stopped by the shift control logic when the specified number of data frames is received. When the transfer is done, the BUSY status is reset to 0. Poll the BUSY status to wait for completion of the transfer, but do not poll it prior to at least one MST_M2SC clock cycle from the start of the transfer.
7. If this master receives data, read the receive FIFO until it is empty.
8. Disable this master by writing 0 to SSIENR.

9.8.2 Registers

SPI master registers are aligned on 4-byte boundaries and can be accessed only as double words. Bit definitions are given in the following sections. Some registers may be written only when the SPI master is disabled. The SPI Master Enable Register (SM_SSIENR) is used to enable or disable SPI master. Software should not disable the SPI master while it is active. The SPI Master Status Register (SM_SR) can be used to determine if SPI master is active. If the SPI master is in the process of transmitting when it is disabled, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. The slave continues receiving until the remote master aborts the transfer, in which case the SPI master could be disabled. Registers that cannot be written to when the SPI master is enabled are indicated in their descriptions.

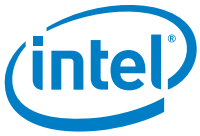
9.8.2.1 SPI Master Control Register 0 (SM_CTRLR0)

The SPI Master Control Register 0 is a read/write register memory mapped at addresses 0x9400 0000 to 0x9400 0003. This register is used to control the SPI master peripheral. This register can only be written when SPI is disabled, which corresponds to the SM_SSIENR register being set to 0.

Table 132. SPI Master Control Register 0

Mnemonic	Access	Bits	Description	Reset
DFS	R/W	3:0	Data Frame Size. Selects the data frame length. When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data. This field is encoded as follows: 0000 Reserved – undefined operation 0001 Reserved – undefined operation 0010 Reserved – undefined operation 0011 4-bit serial data transfer 0100 5-bit serial data transfer 0101 6-bit serial data transfer 0110 7-bit serial data transfer 0111 8-bit serial data transfer 1000 9-bit serial data transfer 1001 10-bit serial data transfer 1010 11-bit serial data transfer 1011 12-bit serial data transfer 1100 13-bit serial data transfer 1101 14-bit serial data transfer 1110 15-bit serial data transfer 1111 16-bit serial data transfer	7
FRF	R/W	5:4	Frame Format. Selects which serial protocol transfers the data. 0x0 = Motorola SPI 0x1 = Texas Instruments SSP 0x2 = National Semiconductors Microwire 0x3 = Reserved	0

SCPH	R/W	6	Serial Clock Phase. Valid when the frame format (FRF) is set to Motorola SPI. The serial clock phase selects the relationship of the serial clock with the slave select signal. When SCPH = 0, data are captured on the first edge of the serial clock. When SCPH = 1, the serial clock starts toggling one cycle after the slave select line is activated, and data are captured on the second edge of the serial clock. 0 = Serial clock toggles in middle of first data bit 1 = Serial clock toggles at start of first data bit	0
SCPOL	R/W	7	Serial Clock Polarity. Valid when the frame format (FRF) is set to Motorola SPI. Used to select the polarity of the inactive serial clock, which is held inactive when the SPI master is not actively transferring data on the serial bus. 0 = Inactive state of serial clock is low 1 = Inactive state of serial clock is high	0
TMOD	R/W	9:8	Transfer Mode. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. In transmit-only mode, data received from the external device is not valid and is not stored in the receive FIFO memory; it is overwritten on the next transfer. In receive-only mode, transmitted data are not valid. After the first write to the transmit FIFO, the same word is retransmitted for the duration of the transfer. In transmit-and-receive mode, both transmit and receive data are valid. The transfer continues until the transmit FIFO is empty. Data received from the external device are stored into the receive FIFO memory, where it can be accessed by the host processor. In EEPROM-read mode, receive data is not valid while control data is being transmitted. When all control data is sent to the EEPROM, receive data becomes valid and transmit data becomes invalid. All data in the transmit FIFO is considered control data in this mode. 00 = Transmit & Receive 01 = Transmit only 10 = Receive only 11 = EEPROM read	0
-	-	10	Reserved	X
SRL	R/W	11	Shift Register Loop. Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. 0 = Normal Mode Operation 1 = Test Mode Operation	0



CFS	R/W	15:12	Control Frame Size. Selects the length of the control word for the Microwire frame format. 0000 1-bit control word 0001 2-bit control word 0010 3-bit control word 0011 4-bit control word 0100 5-bit control word 0101 6-bit control word 0110 7-bit control word 0111 8-bit control word 1000 9-bit control word 1001 10-bit control word 1010 11-bit control word 1011 12-bit control word 1100 13-bit control word 1101 14-bit control word 1110 15-bit control word 1111 16-bit control word	0
-	-	31:16	Reserved	X

9.8.2.2 SPI Master Control Register 1 (SM_CTRL1)

The SPI Master Control Register 1 is a read/write register memory mapped at addresses 0x9400 0004 to 0x9400 0007. This register is used to control the end of serial transfers when in receive-only mode. This register can only be written when SPI is disabled, which corresponds to the SM_SSIENR register being set to 0.

Table 133. SPI Master Control Register 1

Mnemonic	Access	Bits	Description	Reset
NDF	R/W	15:0	Number of Data Frame. When TMOD = 10 or TMOD = 11 this register field sets the number of data frames to continuously receive data. The SPI master continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables it to receive up to 64 kB of data in a continuous transfer.	0
-	-	31:16	Reserved	X

9.8.2.3 SPI Master Enable Register (SM_SSIENR)

The SPI Master Enable Register is a read/write register memory mapped at addresses 0x9400 0008 to 0x9400 000B. This register enables and disables the SPI master peripheral.

Table 134. SPI Mater SSI Enable Register

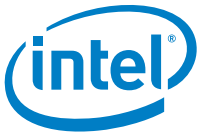
Mnemonic	Access	Bits	Description	Reset
SSI_EN	R/W	0	SSI Enable. Enables and disables all SPI master operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the control registers when the SPI master is enabled. 1 = Enabled. 0 = Disabled.	0
-	-	31:1	Reserved	X

9.8.2.4 SPI Master Microwire Control Register (SM_MWCR)

The SPI Master Microwire Control Register is a read/write register memory mapped at address 0x9400 000C to 0x9400 000F. This register controls the direction of the data word for the half-duplex Microwire serial protocol. This register can only be written when SPI is disabled, which corresponds to the SM_SSIENR register being set to 0.

Table 135. SPI Master Microwire Control Register

Mnemonic	Access	Bits	Description	Reset
MWMOD	R/W	0	Microwire Transfer Mode. Defines whether the Microwire transfer is sequential or non-sequential. When sequential mode is used, only one control word is needed to transmit or receive a block of data words. When non-sequential mode is used, there must be a control word for each data word that is transmitted or received. 0 = Non-sequential transfer 1 = Sequential transfer	0
MDD	R/W	1	Microwire Control. Defines the direction of the data word when the Microwire serial protocol is used. When this bit is set to 0, the data word is received by the SPI master from the external serial device. When this bit is set to 1, the data word is transmitted from the SPI master to the external serial device.	0
MHS	R/W	2	Microwire Handshaking. Used to enable and disable the "busy/ready" handshaking interface for the Microwire protocol. When enabled, the SPI master checks for a ready status from the target slave after the transfer of the last data/control bit before clearing the BUSY status in the SR register. 0 = Handshaking interface is disabled 1 = Handshaking interface is enabled	0
-	-	31:3	Reserved	X



9.8.2.5 SPI Master Slave Enable Register (SM_SER)

The SPI Master Slave Enable Register is a read/write register memory mapped at addresses 0x9400 0010 to 0x9400 0013. This register enables the individual slave select output lines from this master. This register can only be written when SPI is not busy, which corresponds to the SM_SR.BUSY = 0.

Table 136. SPI Master Slave Enable Register

Mnemonic	Access	Bits	Description	Reset
SER	R/W	3:0	Slave Select Enable Flag. When a bit in this register is set (1), the corresponding slave select line from the master is activated when a serial transfer begins. It should be noted that setting or clearing bits in this register have no effect on the corresponding slave select outputs until a transfer is started. Before beginning a transfer, you should enable the bit in this register that corresponds to the slave device with which the master wants to communicate. When not operating in broadcast mode, only one bit in this field should be set. 1 = Selected 0 = Not Selected	0
-	-	31:4	Reserved	X

9.8.2.6 SPI Master Baud Rate Select (SM_BAUDR)

The SPI Master Baud Rate Select Register is a read/write register memory mapped at addresses 0x9400 0014 to 0x9400 0017. This register derives the frequency of the serial clock that regulates the data transfer. This register can only be written when SPI is disabled, which corresponds to the SM_SSIENR register being set to 0.

Table 137. SPI Master Baud Rate Select

Mnemonic	Access	Bits	Description	Reset
SCKDV	R/W	15:0	SSI Clock Divider. The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (MST_M2SC) is disabled. The frequency of the MST_M2SC is derived from the following equation: $f_{MST\_M2SC} = \frac{f_{MST\_SPI\_CLK}}{SCKDIV}$ where *SCKDIV* is any even integer between 2 and 65,534.	0
-	-	31:16	Reserved	X

9.8.2.7 SPI Master Transmit FIFO Threshold Level (SM_ TXFTLR)

The SPI Master Transmit FIFO Threshold Level is a read/write register memory mapped at addresses 0x9400 0018 to 0x9400 001B. This register controls the threshold level for the transmit FIFO empty interrupt. This register can only be written when SPI is disabled, which corresponds to the SM_SSIENR register being set to 0.

Table 138. SPI Master Transmit FIFO Threshold Level

Mnemonic	Access	Bits	Description	Reset
TFT	R/W	2:0	Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an empty interrupt.	0
-	-	31:3	Reserved	X

9.8.2.8 SPI Master Receive FIFO Threshold Level (SM_RXFTLR)

The SPI Master Receive FIFO Threshold Level is a read/write register memory mapped at addresses 0x9400 001C to 0x9400 001F. This register controls the threshold level for the receive FIFO full interrupt. This register can only be written when SPI is disabled, which corresponds to the SM_SSIENR register being set to 0.

Table 139. SPI Master Receive FIFO Threshold Level

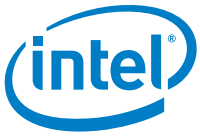
Mnemonic	Access	Bits	Description	Reset
RFT	R/W	2:0	Receive FIFO Threshold. Controls the level of entries above which the receive FIFO controller triggers a full interrupt.	0
-	-	31:3	Reserved	X

9.8.2.9 SPI Master Transmit FIFO Level Register (SM_ TXFLR)

The SPI Master Transmit FIFO Level Register is a read only register memory mapped at addresses 0x9400 0020 to 0x9400 0023. This register contains the number of valid data entries in the transmit FIFO memory.

Table 140. SPI Master Transmit FIFO Level Register

Mnemonic	Access	Bits	Description	Reset
TFL	R	3:0	Transmit FIFO Level. Returns the number of valid entries in the transmit FIFO.	0
-	-	31:3	Reserved	X



9.8.2.10 SPI Master Receive FIFO Level Register (SM_RXFLR)

The SPI Master Receive FIFO Level Register is a read only register memory mapped at addresses 0x9400 0024 to 0x9400 0027. This register contains the number of valid data entries in the receive FIFO memory.

Table 141. SPI Master Receive FIFO Level Register

Mnemonic	Access	Bits	Description	Reset
RFL	R	3:0	Receive FIFO Level. Returns the number of valid entries in the receive FIFO.	0
-	-	31:3	Reserved	X

9.8.2.11 SPI Master Status Register (SM_SR)

The SPI Master Status Register is a read only register memory mapped at addresses 0x9400 0028 to 0x9400 002B. This register is used to indicate the current transfer status, FIFO status, and any transmission/reception errors that may have occurred. The status register may be read at any time. None of the bits in this register request an interrupt.

Table 142. SPI Master Status Register

Mnemonic	Access	Bits	Description	Reset
BUSY	R	0	Busy Flag. When set, indicates that a serial transfer is in progress; when cleared indicates that this master is idle or disabled. 0 = this master is idle or disabled 1 = this master is actively transferring data	0
TFNF	R	1	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full	1
TFE	R	2	Transmit FIFO Empty. When the transmit FIFO is completely empty, this bit is set. When the transmit FIFO contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty	1
RFNE	R	3	Receive FIFO Not Empty. Set when the receive FIFO contains one or more entries and is cleared when the receive FIFO is empty. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty	0
RFF	R	4	Receive FIFO Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. 0 = Receive FIFO is not full 1 = Receive FIFO is full	0
-	-	5	Reserved	X
DCOL	R	6	Data Collision Error. This bit is set if this master is actively transmitting while another master is actively transmitting. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. 0 = No error 1 = Transmit data collision error	0
-	-	31:7	Reserved	X

9.8.2.12 SPI Master Interrupt Mask Register (SM_IMR)

The SPI Master Interrupt Mask Register is a read/write register memory mapped at addresses 0x9400 002C to 0x9400 002F. This register masks or enables all interrupts generated by this master.

Table 143. SPI Master Interrupt Mask Register

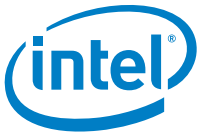
Mnemonic	Access	Bits	Description	Reset
TXEIM	R/W	0	Transmit FIFO Empty Interrupt Mask 0 = interrupt is masked 1 = interrupt is not masked	1
TXOIM	R/W	1	Transmit FIFO Overflow Interrupt Mask 0 = interrupt is masked 1 = interrupt is not masked	1
RXUIM	R/W	2	Receive FIFO Underflow Interrupt Mask 0 = interrupt is masked 1 = interrupt is not masked	1
RXOIM	R/W	3	Receive FIFO Overflow Interrupt Mask 0 = interrupt is masked 1 = interrupt is not masked	1
RXFIM	R/W	4	Receive FIFO Full Interrupt Mask 0 = interrupt is masked 1 = interrupt is not masked	1
MSTIM	R/W	5	Multi-Master Contention Interrupt Mask. 0 = interrupt is masked 1 = interrupt is not masked	1
-	-	31:6	Reserved	X

9.8.2.13 SPI Master Interrupt Status Register (SM_ISR)

The SPI Master Interrupt Status Register is a read only register memory mapped at addresses 0x9400 0030 to 0x9400 0033. This register reports the status of the interrupts after they have been masked.

Table 144. SPI Master Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
TXEIS	R	0	Transmit FIFO Empty Interrupt Status 0 = interrupt is not active after masking 1 = interrupt is active after masking	0
TXOIS	R	1	Transmit FIFO Overflow Interrupt Status 0 = interrupt is not active after masking 1 = interrupt is active after masking	0
RXUIS	R	2	Receive FIFO Underflow Interrupt Status 0 = interrupt is not active after masking 1 = interrupt is active after masking	0
RXOIS	R	3	Receive FIFO Overflow Interrupt Status 0 = interrupt is not active after masking 1 = interrupt is active after masking	0



RXFIS	R	4	Receive FIFO Full Interrupt Status 0 = interrupt is not active after masking 1 = interrupt is full after masking	0
MSTIS	R	5	Multi-Master Contention Interrupt Status. 0 = interrupt not active after masking 1 = interrupt is active after masking	0
-	-	31:6	Reserved	X

9.8.2.14 SPI Master Raw Interrupt Status Register (SM_RISR)

The SPI Master Raw Interrupt Status Register is a read only register memory mapped at addresses 0x9400 0034 to 0x9400 0037. This register reports the status of the interrupts prior to masking.

Table 145. SPI Master Raw Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
TXEIR	R	0	Transmit FIFO Empty Interrupt Status 0 = interrupt is not active prior to masking 1 = interrupt is active prior to masking	0
TXOIR	R	1	Transmit FIFO Overflow Interrupt Status 0 = interrupt is not active prior to masking 1 = interrupt is active prior to masking	0
RXUIR	R	2	Receive FIFO Underflow Interrupt Status 0 = interrupt is not active prior to masking 1 = interrupt is active prior to masking	0
RXOIR	R	3	Receive FIFO Overflow Interrupt Status 0 = interrupt is not active prior to masking 1 = interrupt is active prior to masking	0
RXFIR	R	4	Receive FIFO Full Interrupt Status 0 = interrupt is not active prior to masking 1 = interrupt is full prior to masking	0
MSTIR	R	5	Multi-Master Contention Interrupt Status. 0 = interrupt not active prior to masking 1 = interrupt is active prior to masking	0
-	-	31:6	Reserved	X

9.8.2.15 SPI Master Transmit FIFO Overflow Interrupt Clear Register (SM_TXOICR)

The SPI Master Transmit FIFO Overflow Interrupt Clear Register is a read only register memory mapped at address 0x9400 0038 to 0x9400 003B. This register clears the transmit FIFO overflow interrupt when read. The data returned has no meaning.

Table 146. SPI Master Transmit FIFO Overflow Interrupt Clear Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	X

9.8.2.16 SPI Master Receive FIFO Overflow Interrupt Clear Register (SM_RXOICR)

The SPI Master Receive FIFO Overflow Interrupt Clear Register is a read only register memory mapped at addresses 0x9400 003C to 0x9400 003F. This register clears the receive FIFO overflow interrupt when read. The data returned has no meaning.

Table 147. SPI Master Receive FIFO Overflow Interrupt Clear Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	X

9.8.2.17 SPI Master Receive FIFO Underflow Interrupt Clear Register (SM_RXUICR)

The SPI Master Receive FIFO Underflow Interrupt Clear Register is a read only register memory mapped at address 0x9400 0040 to 0x9400 0043. This register clears the receive FIFO underflow interrupt when read. The data returned has no meaning.

Table 148. SPI Master Receive FIFO Underflow Interrupt Clear Register

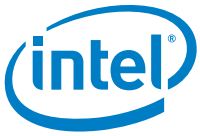
Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	X

9.8.2.18 SPI Master Multi-Master Interrupt Clear Register (SM_MSTICR)

The SPI Master Multi-Master Interrupt Clear Register is a read only register memory mapped at addresses 0x9400 0044 to 0x9400 0047. This register clears the multi-master contention interrupt when read. The data returned has no meaning.

Table 149. SPI Master Multi-Master Interrupt Clear Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	X



9.8.2.19 SPI Master Interrupt Clear Register (SM_ICR)

The SPI Master Interrupt Clear Register is a read only register memory mapped at addresses 0x9400 0048 to 0x9400 004B. This register clears all active interrupts when read. The data returned has no meaning.

Table 150. SPI Master Interrupt Clear Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	X

9.8.2.20 SPI Master Identification Register (SM_IDR)

The SPI Master Identification Register is a read only register memory mapped at addresses 0x9400 0058 to 0x9400 005B. This register returns the peripheral identification code.

Table 151. SPI Master Identification Register

Mnemonic	Access	Bits	Description	Reset
IDCODE	R	31:0	Identification Code. Always returns 0xFFFF FFFF.	0xFFFF FFFF

9.8.2.21 SPI Master Component Version Register (SM_COMP_VERSION)

The SPI Master Component Version Register is a read only register memory mapped at addresses 0x9400 005C to 0x9400 005F. This register returns the component version code.

Table 152. SPI Master Component Version Register

Mnemonic	Access	Bits	Description	Reset
COMP_VERSION	R	31:0	Version Code Register: always returns the version code 3.20 as the big endian ASCII string “320*” or 0x3332 302A.	0x3332 302A

9.8.2.22 SPI Master Data Register (SM_DR)

The SPI Master Data Register is a read/write register memory mapped at addresses 0x9400 0060 to 0x9400 00EC. This register is a read/write buffer for the 16-bit transmit/receive FIFOs. When the register is read, data are popped from the receive FIFO. When it is written, data are pushed into the transmit FIFO. Accessing this register is only meaningful when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.

Note: The SM_DR register occupies ninety six 32-bit address locations. Writing to any of these address locations pushes data into the transmit FIFO. Reading from any of these locations pops data from the receive FIFO. The FIFO buffers are not addressable.

Table 153. SPI Master Data Register

Mnemonic	Access	Bits	Description	Reset
DR	R/W	15:0	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer	0
-	-	31:16	Reserved	X

9.9 Serial Peripheral Interface (SPI) Slave

The MCU features a SPI slave peripheral that supports a variety of 3- and 4-wire serial protocols making it easy to connect with most popular SPI masters. Software can choose from Motorola* SPI, Texas Instruments* Synchronous Serial Protocol (SSP) or National Semiconductor* Microwire* transfer types on a per transfer basis. Software can also choose from transmit only, receive only, transmit-receive, and EEPROM transfer modes. Other features and benefits include:

- Transmit FIFO overflow, transmit FIFO underflow, transmit FIFO empty, receive FIFO full, receive FIFO underflow, and receive FIFO overflow interrupts. All can be masked independently.
- Interrupt or polled-mode operation.
- Programmable word size from 4-16 bits.
- Eight word transmit and receive FIFO buffers with programmable interrupt thresholds.
- Independent SPI clock allows CPU clock frequency changes without affecting in progress transfers.
- Component version register.

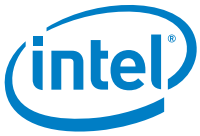
9.9.1 Functional Description

This peripheral is a full duplex SPI slave. Please read Section 9.8.1 above for a complete discussion of the supported SPI serial protocols and transfer modes. The following sections only describe functional operation that is unique to the SPI slave.

9.9.1.1 Clock Ratios

The minimum frequency of *slv_spi_clk* depends on the operation of this SPI slave. If it is receive only, the minimum frequency of *slv_spi_clk* is six times the maximum expected frequency of the bit-rate clock from the master device (SLV_M2SC). The SLV_M2SC signal is double synchronized to the *slv_spi_clk* domain, and then edge detected. This synchronization requires three *slv_spi_clk* periods.

If this SPI slave is transmit and receive, the minimum frequency of *slv_spi_clk* is eight times the maximum expected frequency of the bit-rate clock from the master device (SLV_M2SC). This ensures that data on the master SLV_M2SD line is stable before the master shift control logic captures the data.



9.9.1.2 Data Transfers

All serial transfers are initiated and controlled by the external serial bus master. When this SPI slave is selected (i.e. SLV_M2SS is asserted), it enables its output data (SLV_S2MD) onto the serial bus. All data transfers to and from this slave are regulated by the serial clock line (SLV_M2SC) driven by the serial-master. Data are propagated from the serial slave on one edge of the serial clock line and sampled on the opposite edge.

The serial clock that regulates the data transfer is generated by the SPI master device and is input to this SPI slave on the SLV_M2SC pin. This SPI slave remains in an idle state until selected by the SPI master. When not actively transmitting data, this SPI slave holds its SLV_S2MD output in a high impedance state to avoid interference with serial transfers with other slave devices. When this SPI slave's SLV_M2SS input is asserted, it participates in the serial transfer and continues to transfer data to and from the master device as long as it is selected. If the master transmits to all serial slaves, the CTRLR0.SLV_OE register bit can be programmed to control whether or not this SPI slave responds with data from the its SLV_S2MD line.

9.9.1.2.1 Slave SPI and SSP Serial Transfers

The SPI and SSP serial protocols are described in Section 9.8.1.7.1 and Section 9.8.1.7.2 above respectively. They include timing diagrams for the various transfers. The transfer modes supported by this SPI slave are described in Section 9.8.1.5 above.

If this SPI slave is receive only (TMOD=10), the transmit FIFO need not contain valid data because the data currently in the transmit shift register is resent each time the slave device is selected. The TXE error flag in the status register (SR) is not set when TMOD=01. The transmit FIFO empty interrupt should be masked when this mode is used.

If this SPI slave transmits data to the master, software must ensure that data exists in the transmit FIFO before a transfer is initiated by the external SPI master. If the master initiates a transfer to this slave when no data exists in the transmit FIFO, an error flag (TXE) is set in the status register, and the previously transmitted data frame is resent on SLV_S2MD. For continuous data transfers, software must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level register (TXFTLR) can be used to interrupt the processor when the transmit FIFO buffer is nearly empty. The FIFO can then be refilled with data to continue the serial transfer.

The receive FIFO buffer should be read each time the receive FIFO generates a FIFO full interrupt request to prevent an overflow. The receive FIFO threshold level register (RXFTLR) can be used to interrupt the processor when the receive FIFO is nearly full.

A typical software flow for completing a continuous serial transfer from an external SPI master to this SPI slave is described as follows:

1. If this slave is enabled, disable it by writing 0 to SSIENR.
2. Set up the control registers for the transfer. These registers can be set in any order.
   a. Write CTRLR0 (for SPI transfers SCPH and SCPOL must be set identical to the master device).
   b. Write TXFTLR and RXFTLR to set FIFO threshold levels.
   c. Write the IMR register to set up interrupt masks.
3. Enable this slave by writing 1 to the SSIENR register.

4. If the transfer mode is transmit and receive (TMOD=2'b00) or transmit only (TMOD=2'b01), write data for transmission into the transmit FIFO (Write DR). If the transfer mode is receive only (TMOD=2'b10), there is no need to write data into the transmit FIFO; the current value in the transmit shift register is retransmitted.
5. This slave is now ready for the serial transfer. The transfer begins when this slave is selected by a serial-master device.
6. If the transfer is larger than eight frames, interrupts should be enabled to keep the transmit FIFO from underflowing and the receive FIFO from overflowing.
   a. If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR).
   b. If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
7. The transfer ends when the master removes the select input to this slave. When the transfer is completed, the BUSY status is reset to 0.
8. When the transfer is done, the BUSY status is reset to 0.
9. Poll the BUSY status to wait for completion of the transfer, but do not poll it prior to least one SLV_M2SC clock cycle from the start of the transfer.
10. If the transfer mode is not transmit only (TMOD != 01), read the receive FIFO until empty.
11. Disable this slave by writing 0 to SSIENR.

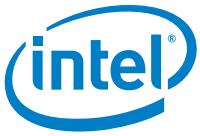
9.9.1.2.2 Slave Microwire Serial Transfers

The Microwire serial protocol is described in Section 9.8.1.8.2 above. It includes timing diagrams for the various transfers. The Microwire protocol operates in much the same way as the SPI protocol and a similar to that in Section 9.9.1.2.1 above will suffice.

Note: There is no decode of the control frame by this slave device. Decoding of the control frame must be done by software.

9.9.2 Registers

SPI slave registers are aligned on 4-byte boundaries and can be accessed only as double words. Bit definitions are given in the following sections. Some registers may be written only when the SPI slave is disabled. The SPI Slave Enable Register(SS_SSIENR)is used to enable or disable SPI slave. Software should not disable the SPI slave while it is active. The SPI Slave Status Register (SS_SR)can be used to determine if SPI slave is active. If the SPI slave is in the process of transmitting when it is disabled, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. Registers that cannot be written to when the SPI slave is enabled are indicated in their descriptions.



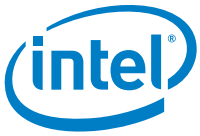
9.9.2.1 SPI Slave Control Register 0 (SS_CTRLR0)

The SPI Slave Control Register 0 is a read/write register memory mapped at address 0x9800 0000 to 0x9800 0003. This register is used to control the SPI slave peripheral. This register can only be written when SPI is disabled, which corresponds to the SS_SSIENR register being set to 0.

Table 154. SPI Slave Control Register 0

Mnemonic	Access	Bits	Description	Reset
DFS	R/W	3:0	Data Frame Size. Selects the data frame length. When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data. The field is encoded as follows: 0000 Reserved – undefined operation 0001 Reserved – undefined operation 0010 Reserved – undefined operation 0011 4-bit serial data transfer 0100 5-bit serial data transfer 0101 6-bit serial data transfer 0110 7-bit serial data transfer 0111 8-bit serial data transfer 1000 9-bit serial data transfer 1001 10-bit serial data transfer 1010 11-bit serial data transfer 1011 12-bit serial data transfer 1100 13-bit serial data transfer 1101 14-bit serial data transfer 1110 15-bit serial data transfer 1111 16-bit serial data transfer	7
FRF	R/W	5:4	Frame Format. Selects which serial protocol transfers the data. 0x0 = Motorola SPI 0x1 = Texas Instruments SSP 0x2 = National Semiconductors Microwire 0x3 = Reserved	0
SCPH	R/W	6	Serial Clock Phase. Valid when the frame format (FRF) is set to Motorola SPI. The serial clock phase selects the relationship of the serial clock with the slave select signal. When SCPH = 0, data are captured on the first edge of the serial clock. When SCPH = 1, the serial clock starts toggling one cycle after the slave select line is activated, and data are captured on the second edge of the serial clock. 0 = Serial clock toggles in middle of first data bit 1 = Serial clock toggles at start of first data bit	0

SCPOL	R/W	7	Serial Clock Polarity. Valid when the frame format (FRF) is set to Motorola SPI. Used to select the polarity of the inactive serial clock, which is held inactive when the SPI slave is not actively transferring data on the serial bus. 0 = Inactive state of serial clock is low 1 = Inactive state of serial clock is high	0
TMOD	R/W	9:8	Transfer Mode. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. In transmit-only mode, data received from the external device is not valid and is not stored in the receive FIFO memory; it is overwritten on the next transfer. In receive-only mode, transmitted data are not valid. After the first write to the transmit FIFO, the same word is retransmitted for the duration of the transfer. In transmit-and-receive mode, both transmit and receive data are valid. The transfer continues until the transmit FIFO is empty. Data received from the external device are stored into the receive FIFO memory, where it can be accessed by the host processor. 00 = Transmit & Receive 01 = Transmit only 10 = Receive only 11 = Reserved	0
SLV_OE	R/W	10	Slave Output Enable. This bit enables or disables serial output data pin SLV_S2MD. This is useful when the master transmits in broadcast mode (master transmits data to all slave devices). Only one slave may respond with data on the master SLV_S2MD pin. This bit is enabled after reset and must be disabled by software (when broadcast mode is used), if you do not want this device to respond with data. 0 = SLV_S2MD is enabled 1 = SLV_S2MD is disabled	0
SRL	R/W	11	Shift Register Loop. Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. The SLV_M2SS and SLV_M2SC signals must be provided by an external source. The slave cannot generate these signals. 0 = Normal Mode Operation 1 = Test Mode Operation	0



CFS	R/W	15:12	Control Frame Size. Selects the length of the control word for the Microwire frame format. 0000 1-bit control word 0001 2-bit control word 0010 3-bit control word 0011 4-bit control word 0100 5-bit control word 0101 6-bit control word 0110 7-bit control word 0111 8-bit control word 1000 9-bit control word 1001 10-bit control word 1010 11-bit control word 1011 12-bit control word 1100 13-bit control word 1101 14-bit control word 1110 15-bit control word 1111 16-bit control word	0
-	-	31:16	Reserved	X

9.9.2.2 SPI Slave Enable Register (SS_SSIENR)

The SPI Slave Enable Register is a read/write register memory mapped at address 0x9800 0008 to 0x9800 000B. This register enables and disables the SPI slave peripheral.

Table 155. SPI Slave Enable Register

Mnemonic	Access	Bits	Description	Reset
SSI_EN	R/W	0	SSI Enable. Enables and disables all SPI slave operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the control registers when the SPI slave is enabled. 1 = Enabled. 0 = Disabled.	0
-	-	31:1	Reserved	X

9.9.2.3 SPI Slave Microwire Control Register (SS_MWCR)

The SPI Slave Microwire Control Register is a read/write register memory mapped at address 0x9800 000C to 0x9800 000F. This register controls the direction of the data word for the half-duplex Microwire serial protocol. This register can only be written when SPI is disabled, which corresponds to the SS_SSIENR register being set to 0.

Table 156. SPI Slave Microwire Control Register

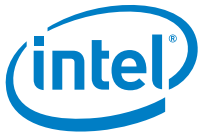
Mnemonic	Access	Bits	Description	Reset
MWMOD	R/W	0	Microwire Transfer Mode. Defines whether the Microwire transfer is sequential or non-sequential. When sequential mode is used, only one control word is needed to transmit or receive a block of data words. When non-sequential mode is used, there must be a control word for each data word that is transmitted or received. 0 = Non-sequential transfer 1 = Sequential transfer	0
MDD	R/W	1	Microwire Control. Defines the direction of the data word when the Microwire serial protocol is used. When this bit is set to 0, the data word is received by the SPI slave from the external serial device. When this bit is set to 1, the data word is transmitted from the SPI slave to the external serial device.	0
-	-	31:2	Reserved	X

9.9.2.4 SPI Slave Transmit FIFO Threshold Level (SS_ TXFTLR)

The SPI Slave Transmit FIFO Threshold Level is a read/write register memory mapped at address 0x9800 0018 to 0x9800 001B. This register controls the threshold level for the transmit FIFO empty interrupt. This register can only be written when SPI is disabled, which corresponds to the SS_SSIENR register being set to 0.

Table 157. SPI Slave Transmit FIFO Threshold Level

Mnemonic	Access	Bits	Description	Reset
TFT	R/W	2:0	Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an empty interrupt.	0
-	-	31:3	Reserved	X



9.9.2.5 SPI Slave Receive FIFO Threshold Level (SS_RXFTLR)

The SPI Slave Receive FIFO Threshold Level is a read/write register memory mapped at address 0x9800 001C to 0x9800 001F. This register controls the threshold level for the receive FIFO full interrupt. This register can only be written when SPI is disabled, which corresponds to the SS_SSIENR register being set to 0.

Table 158. SPI Slave Receive FIFO Threshold Level

Mnemonic	Access	Bits	Description	Reset
RFT	R/W	2:0	Receive FIFO Threshold. Controls the level of entries above which the receive FIFO controller triggers a full interrupt.	0
-	-	31:3	Reserved	X

9.9.2.6 SPI Slave Transmit FIFO Level Register (SS_ TXFLR)

The SPI Slave Transmit FIFO Level Register is a read only register memory mapped at address 0x9800 0020 to 0x9800 0023. This register contains the number of valid data entries in the transmit FIFO memory.

Table 159. SPI Slave Transmit FIFO Level Register

Mnemonic	Access	Bits	Description	Reset
TFL	R	3:0	Transmit FIFO Level. Returns the number of valid entries in the transmit FIFO.	0
-	-	31:3	Reserved	X

9.9.2.7 SPI Slave Receive FIFO Level Register (SS_RXFLR)

The SPI Slave Receive FIFO Level Register is a read only register memory mapped at address 0x9800 0024 to 0x9800 0027. This register contains the number of valid data entries in the receive FIFO memory.

Table 160. SPI Slave Receive FIFO Level Register

Mnemonic	Access	Bits	Description	Reset
RFL	R	3:0	Receive FIFO Level. Returns the number of valid entries in the receive FIFO.	0
-	-	31:3	Reserved	X

9.9.2.8 SPI Slave Status Register (SS_SR)

The SPI Slave Status Register is a read only register memory mapped at address 0x9800 0028 to 0x9800 002B. This register is used to indicate the current transfer status, FIFO status, and any transmission/reception errors that may have occurred. The status register may be read at any time. None of the bits in this register request an interrupt.

Table 161. SPI Slave Status Register

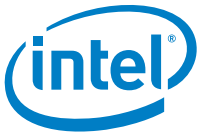
Mnemonic	Access	Bits	Description	Reset
BUSY	R	0	Busy Flag. When set, indicates that a serial transfer is in progress; when cleared indicates that this slave is idle or disabled. 0 = this slave is idle or disabled 1 = this slave is actively transferring data	0
TFNF	R	1	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full	1
TFE	R	2	Transmit FIFO Empty. When the transmit FIFO is completely empty, this bit is set. When the transmit FIFO contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty	1
RFNE	R	3	Receive FIFO Not Empty. Set when the receive FIFO contains one or more entries and is cleared when the receive FIFO is empty. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty	0
RFF	R	4	Receive FIFO Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. 0 = Receive FIFO is not full 1 = Receive FIFO is full	0
TXE	R	5	Transmission Error. Set if the transmit FIFO is empty when a transfer is started. Data from the previous transmission is resent on the SLV_S2MD pin. This bit is cleared when read. 0 – No error 1 – Transmission error	0
-	-	31:6	Reserved	X

9.9.2.9 SPI Slave Interrupt Mask Register (SS_IMR)

The SPI Slave Interrupt Mask Register is a read/write register memory mapped at address 0x9800 002C to 0x9800 002F. This register masks or enables all interrupts generated by this slave.

Table 162. SPI Slave Interrupt Mask Register

Mnemonic	Access	Bits	Description	Reset
TXEIM	R/W	0	Transmit FIFO Empty Interrupt Mask 0 = interrupt is masked 1 = interrupt is not masked	1
TXOIM	R/W	1	Transmit FIFO Overflow Interrupt Mask 0 = interrupt is masked 1 = interrupt is not masked	1
RXUIM	R/W	2	Receive FIFO Underflow Interrupt Mask 0 = interrupt is masked 1 = interrupt is not masked	1
RXOIM	R/W	3	Receive FIFO Overflow Interrupt Mask 0 = interrupt is masked 1 = interrupt is not masked	1



RXFIM	R/W	4	Receive FIFO Full Interrupt Mask 0 = interrupt is masked 1 = interrupt is not masked	1
-	-	31:5	Reserved	X

9.9.2.10 SPI Slave Interrupt Status Register (SS_ISR)

The SPI Slave Interrupt Status Register is a read only register memory mapped at address 0x9800 0030 to 0x9800 0033. This register reports the status of the interrupts after they have been masked.

Table 163. SPI Slave Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
TXEIS	R	0	Transmit FIFO Empty Interrupt Status 0 = interrupt is not active after masking 1 = interrupt is active after masking	0
TXOIS	R	1	Transmit FIFO Overflow Interrupt Status 0 = interrupt is not active after masking 1 = interrupt is active after masking	0
RXUIS	R	2	Receive FIFO Underflow Interrupt Status 0 = interrupt is not active after masking 1 = interrupt is active after masking	0
RXOIS	R	3	Receive FIFO Overflow Interrupt Status 0 = interrupt is not active after masking 1 = interrupt is active after masking	0
RXFIS	R	4	Receive FIFO Full Interrupt Status 0 = interrupt is not active after masking 1 = interrupt is full after masking	0
-	-	31:5	Reserved	X

9.9.2.11 SPI Slave Raw Interrupt Status Register (SS_RISR)

The SPI Slave Raw Interrupt Status Register is a read only register memory mapped at address 0x9800 0034 to 0x9800 0037. This register reports the status of the interrupts prior to masking.

Table 164. SPI Slave Raw Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
TXEIR	R	0	Transmit FIFO Empty Interrupt Status 0 = interrupt is not active prior to masking 1 = interrupt is active prior to masking	0
TXOIR	R	1	Transmit FIFO Overflow Interrupt Status 0 = interrupt is not active prior to masking 1 = interrupt is active prior to masking	0
RXUIR	R	2	Receive FIFO Underflow Interrupt Status 0 = interrupt is not active prior to masking 1 = interrupt is active prior to masking	0
RXOIR	R	3	Receive FIFO Overflow Interrupt Status 0 = interrupt is not active prior to masking 1 = interrupt is active prior to masking	0

RXFIR	R	4	Receive FIFO Full Interrupt Status 0 = interrupt is not active prior to masking 1 = interrupt is full prior to masking	0
-	-	31:5	Reserved	X

9.9.2.12 SPI Slave Transmit FIFO Overflow Interrupt Clear Register (SS_TXOICR)

The SPI Slave Transmit FIFO Overflow Interrupt Clear Register is a read only register memory mapped at address 0x9800 0038 to 0x9800 003B. This register clears the transmit FIFO overflow interrupt when read. The data returned has no meaning.

Table 165. SPI Slave Transmit FIFO Overflow Interrupt Clear Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	X

9.9.2.13 SPI Slave Receive FIFO Overflow Interrupt Clear Register (SS_RXOICR)

The SPI Slave Receive FIFO Overflow Interrupt Clear Register is a read only register memory mapped at address 0x9800 003C to 0x9800 003F. This register clears the receive FIFO overflow interrupt when read. The data returned has no meaning.

Table 166. SPI Slave Receive FIFO Overflow Interrupt Clear Register

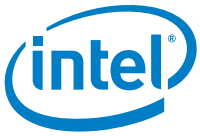
Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	X

9.9.2.14 SPI Slave Receive FIFO Underflow Interrupt Clear Register (SS_RXUICR)

The SPI Slave Receive FIFO Underflow Interrupt Clear Register is a read only register memory mapped at address 0x9800 0040 to 0x9800 0043. This register clears the receive FIFO underflow interrupt when read. The data returned has no meaning.

Table 167. SPI Slave Receive FIFO Underflow Interrupt Clear Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	X



9.9.2.15 SPI Slave Interrupt Clear Register (SS_ICR)

The SPI Slave Interrupt Clear Register is a read only register memory mapped at address 0x9800 0048 to 0x9800 004B. This register clears all active interrupts when read. The data returned has no meaning.

Table 168. SPI Slave Interrupt Clear Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	X

9.9.2.16 SPI Slave Identification Register (SS_IDR)

The SPI Slave Identification Register is a read only register memory mapped at address 0x9800 0058 to 0x9800 005B. This register returns the peripheral identification code.

Table 169. SPI Slave Identification Register

Mnemonic	Access	Bits	Description	Reset
IDCODE	R	31:0	Identification Code. Always returns 0xFFFFFFFF.	0xFFFFFFFF

9.9.2.17 SPI Slave Component Version Register (SS_COMP_VERSION)

The SPI Slave Component Version Register is a read only register memory mapped at address 0x9800 005C to 0x9800 005F. This register returns the component version code.

Table 170. SPI Slave Component Version Register

Mnemonic	Access	Bits	Description	Reset
COMP_VERSION	R	31:0	Version Code Register: always returns the version code 3.20 as the big endian ASCII string “320*” or 0x3332 302A.	0x3332 302A

9.9.2.18 SPI Slave Data Register (SS_DR)

The SPI Slave Data Register is a read/write register memory mapped at address 0x9800 0060 to 0x9800 00EC. This register is a read/write buffer for the 16-bit transmit/receive FIFOs. When the register is read, data are popped from the receive FIFO. When it is written, data are pushed into the transmit FIFO. Accessing this register is only meaningful when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.

Note: The SM_DR register occupies ninety six 32-bit address locations. Writing to any of these address locations pushes data into the transmit FIFO. Reading from any of these locations pops data from the receive FIFO. The FIFO buffers are not addressable.

Table 171. SPI Slave Data Register

Mnemonic	Access	Bits	Description	Reset
DR	R/W	15:0	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer	0
-	-	31:16	Reserved	X

9.10 Inter-Integrated Circuit Bus (I^2C)

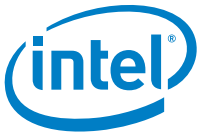
The MCU features an I^2C peripheral interface that can be programmed as master or slave. It complies with the requirements for standard and fast operating modes as defined in the I^2C-Bus Specification (NXP Semiconductors, 2007). Other features and benefits include:

- Standard (up to 100 kbps) and fast (up to 400 kbps) modes.
- Programmable master or slave operation.
- Programmable 7- or 10-bit addressing with combined format transfers.
- Bulk transmit mode
- Ignores CBUS addresses (an older ancestor of I^2C that used to share the I^2C bus).
- Eight byte transmit and receive FIFOs with programmable threshold levels.
- Eleven independently maskable interrupts.
- Interrupt or polled-mode operation
- Handles bit and byte waiting at all bus speeds.
- Programmable SDA hold time.
- Component version register.

9.10.1 Functional Description

The I^2C bus is a two-wire serial interface, consisting of a serial data line (SDA) and a serial clock (SCL). These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a "transmitter" or "receiver," depending on the function of the device. A master is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

This I^2C peripheral can operate in standard mode (with data rates from 0 to 100 kb/s) and fast mode (with data rates less than or equal to 400 kb/s). High speed mode is not supported by this peripheral. However, high-speed mode devices can communicate with fast and standard mode devices in a mixed-speed bus system, and fast mode devices can communicate with standard mode devices in 0 to 100 kb/s I^2C bus system. However, standard mode devices are not upward compatible and should not be incorporated in a fast-mode I^2C bus system as they cannot follow the higher transfer rate and unpredictable states will occur.



Any I2C device can be attached to an I²C-bus, and every device can talk with any master, passing information back and forth. There needs to be at least one master (such as a microcontroller or DSP) on the bus but there can be multiple masters, which require them to arbitrate for ownership.

9.10.1.1 I2C Bus Terms

The following terms relate to how the role of the I2C device and how it interacts with other I2C devices on the bus.

- Transmitter – the device that sends data to the bus. A transmitter can either be a device that initiates the data transmission to the bus (a master-transmitter) or responds to a request from the master to send data to the bus (a slave-transmitter).
- Receiver – the device that receives data from the bus. A receiver can either be a device that receives data on its own request (a master-receiver) or in response to a request from the master (a slave-receiver).
- Master -– the component that initializes a transfer (START command), generates the clock (SCL) signal and terminates the transfer (STOP command). A master can be either a transmitter or a receiver.
- Slave – the device addressed by the master. A slave can be either receiver or transmitter.
- Multi-master – the ability for more than one master to co-exist on the bus at the same time without collision or data loss.
- Arbitration – the predefined procedure that authorizes only one master at a time to take control of the bus.
- Synchronization – the predefined procedure that synchronizes the clock signals provided by two or more masters.
- SDA – data signal line (Serial DAta)
- SCL – clock signal line (Serial CLock)

9.10.1.2 Bus Transfer Terms

The following terms are specific to data transfers that occur to/from the I2C bus.

START (RESTART) – data transfer begins with a START or RESTART condition. The level of the SDA data line changes from high to low, while the SCL clock line remains high. When this occurs, the bus becomes busy.

STOP – data transfer is terminated by a STOP condition. This occurs when the level on the SDA data line passes from the low state to the high state, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free or idle once again. The bus stays busy if a RESTART is generated instead of a STOP condition.

9.10.1.3 I2C Behavior

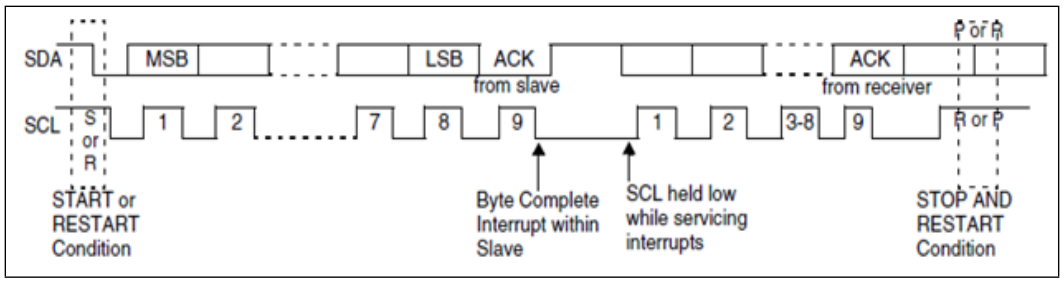
This I2C peripheral can be programmed via software to be an I2C master only, communicating with one or more I2C slaves or an I2C slave only, communicating with one or more I2C masters. The master is responsible for generating the clock and controlling the transfer of data.

The acknowledgement of data is sent by the device that is receiving data, which can be either a master or a slave. As mentioned previously, the I2C protocol also allows multiple masters to reside on the I2C bus and uses an arbitration procedure to determine bus ownership.

Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that is then followed by the slave's address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge (ACK) pulse after the address.

If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver gets one byte of data. This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with the ACK pulse. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition. This behavior is illustrated in Figure 34.

Figure 34. Data transfer on the I²C Bus



The I²C device is a synchronous serial interface. The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP, START, and RESTART conditions. The output drivers are open-drain or open-collector to perform wire-AND functions on the bus. The maximum number of devices on the bus is limited by only the maximum capacitance specification of 400 pF. Data is transmitted in byte packages.

9.10.1.3.1 START and STOP Generation

When operating as an I²C master, putting data into the transmit FIFO causes this I²C peripheral to generate a START condition on the I²C bus. Allowing the transmit FIFO to empty causes this I²C peripheral to generate a STOP condition on the I²C bus.

When operating as a slave, the peripheral does not generate START and STOP conditions, as per the protocol. However, if a read request is made to this peripheral, it holds the SCL line low until the CPU supplies the requested read data or disables the peripheral by writing a 0 to the IC_ENABLE register. This is referred to as stalling the I²C bus.

9.10.1.3.2 Combined Formats

This I²C peripheral supports mixed read and write combined format transactions in both 7-bit and 10-bit addressing modes. It does not support mixed address and mixed address format—that is, a 7-bit address transaction followed by a 10-bit address transaction or vice versa—combined format transactions.

To initiate a mixed read and write combined format transfer, IC_CON.IC_RESTART_EN should be set to 1. With this value set and operating as a master, when this peripheral completes an I^2C transfer, it checks the transmit FIFO and executes the next transfer. If the direction of this transfer differs from the previous transfer, the combined format is used to issue the transfer. If the transmit FIFO is empty when the current I^2C transfer completes, a STOP is issued and the next transfer is issued following a START condition.

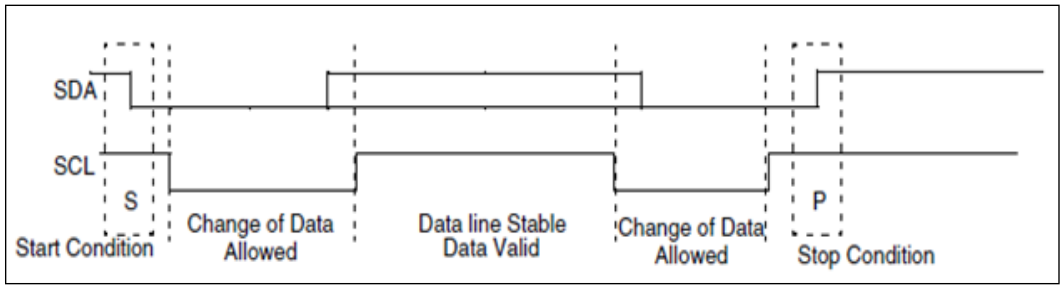
9.10.1.4 I^2C Protocols

This section discusses the protocols supported by this I^2C peripheral.

9.10.1.4.1 START and STOP Conditions

When the bus is idle, both the SCL and SDA signals are pulled high through external pull-up resistors on the bus. When the master wants to start a transmission on the bus, the master issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is 1. When the master wants to terminate the transmission, the master issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1. Figure 35 shows the timing of the START and STOP conditions. When data is being transmitted on the bus, the SDA line must be stable when SCL is 1.

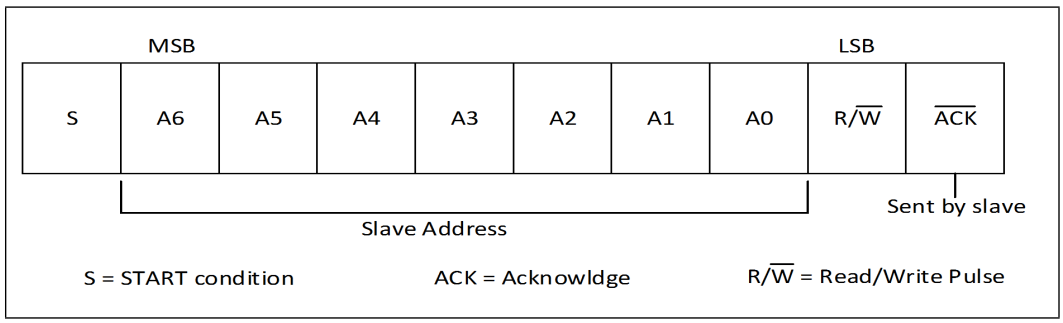
Figure 35. START and STOP Condition



9.10.1.4.2 Addressing Slave Protocol

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the slave address and the LSB bit (bit 0) is the R/W bit as shown in Figure 36. When bit 0 (R/W) is set to 0, the master writes to the slave. When bit 0 (R/W) is set to 1, the master reads from the slave.

Figure 36. 7-bit Address Format



During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the slaves address bits 9:8, and the LSB bit (bit 0) is the R/W bit. The second byte transferred sets bits 7:0 of the slave address. Figure 37 and Table 172 shows the 10-bit address format, and I2C Definition of Bits in First Byte. The table defines the special purpose and reserved first byte addresses.

Figure 37. 10-bit Address Format

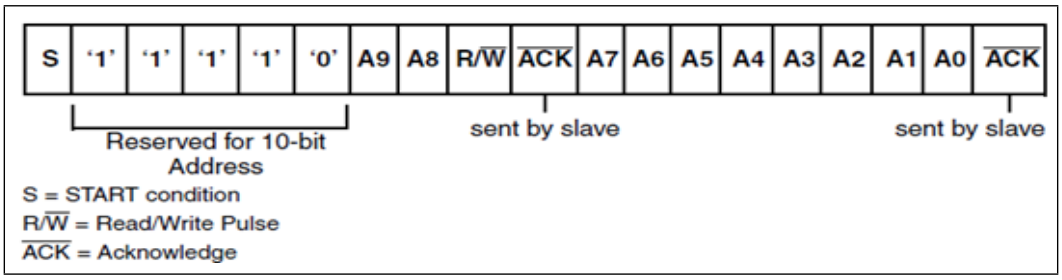


Table 172. I2C Definition of Bits in First Byte

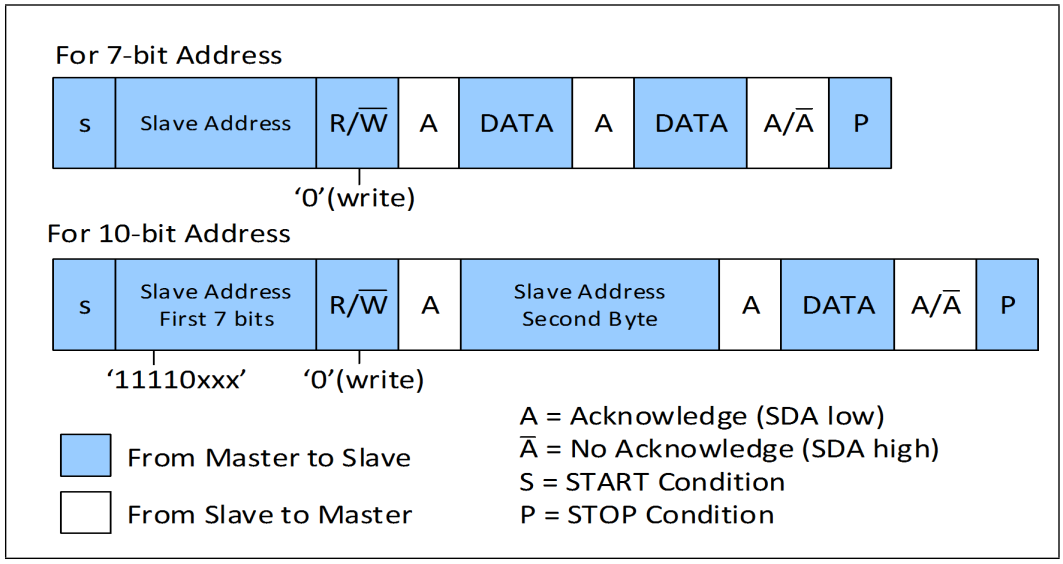
Slave	Address	R/W Bit	Description
0000	000	0	General Call Address. I2C module places the data in the receive buffer and issues a General Call interrupt.
0000	000	1	START byte.
0000	001	X	CBUS address. I2C module ignores these accesses.
0000	010	X	Reserved.
0000	011	X	Reserved.
0000	1XX	X	High-speed master code
1111	1XX	X	Reserved.
1111	0XX	X	10-bit slave addressing.

This I2C peripheral does not restrict you from using these reserved addresses. However, if you use these reserved addresses, you may run into incompatibilities with other I2C components.

9.10.1.4.3 Master-Transmitter and Slave-Receiver

The master initiates a write transfer to the slave using the addressing slave protocol defined above. All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer. After the master sends the address and R/W bit or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so that the master can abort the transfer. Master-transmitter protocol examples for 7 and 10-bit addressing are illustrated in Figure 38.

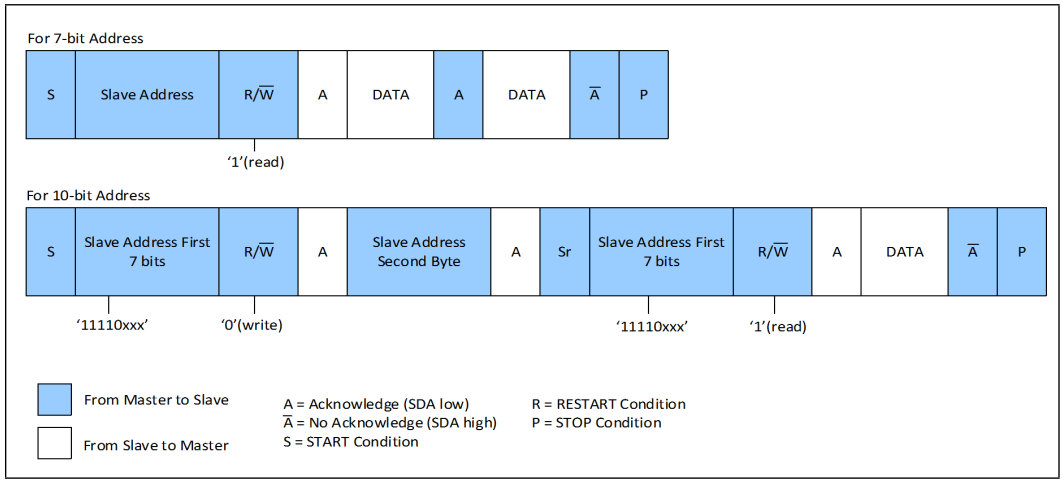
Figure 38. Master-Transmitter and Slave-Receiver Protocol



9.10.1.4.4 Master-Receiver and Slave-Transmitter

The master initiates the read transfer to the slave using the address slave protocol defined above. The master-receiver then responds to the slave-transmitter with an acknowledge pulse after each byte of data, except for the last byte. By not acknowledging (NACK), the master-receiver notifies the slave-transmitter that this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the NACK so that the master can issue a STOP condition. Master-receiver protocol examples for 7 and 10-bit addressing are illustrated in Figure 39.

Figure 39. Master-Receiver and Slave-Transmitter Protocol

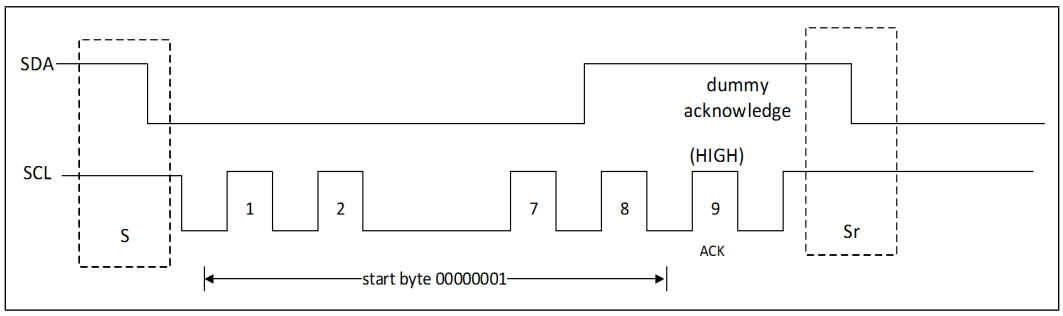


When a master does not want to relinquish the bus with a STOP condition, the master can issue a RESTART condition. This is identical to a START condition except it occurs after the ACK pulse. Operating in master mode, this I^2C peripheral can then communicate with the same slave using a transfer of a different direction. This is referred to as "combined format" and is describe in Section 9.10.1.3.2 above.

9.10.1.4.5 START BYTE Transfer Protocol

The START BYTE transfer protocol is set up for AHB slaves that under-sample the bus while it is idle. When this I^2C peripheral is addressed as a slave, it always samples the I^2C bus at the highest speed supported so that it never requires a START BYTE transfer. However, when this peripheral is a master, it supports the generation of START BYTE transfers at the beginning of every transfer in case a slave device requires it. This protocol consists of seven zeros being transmitted followed by a 1, as illustrated in Figure 40.

Figure 40. START BYTE Transfer



The START BYTE procedure is as follows:

1. Master generates a START (S) condition.
2. Master transmits the START byte (0000 0001).
3. Master transmits the ACK clock pulse. (Present only to conform with the byte handling format used on the bus)
4. No slave sets the ACK signal to 0.
5. Master generates a RESTART (Sr) condition.

A hardware receiver does not respond to the START BYTE because it is a reserved address and resets after the RESTART condition is generated.

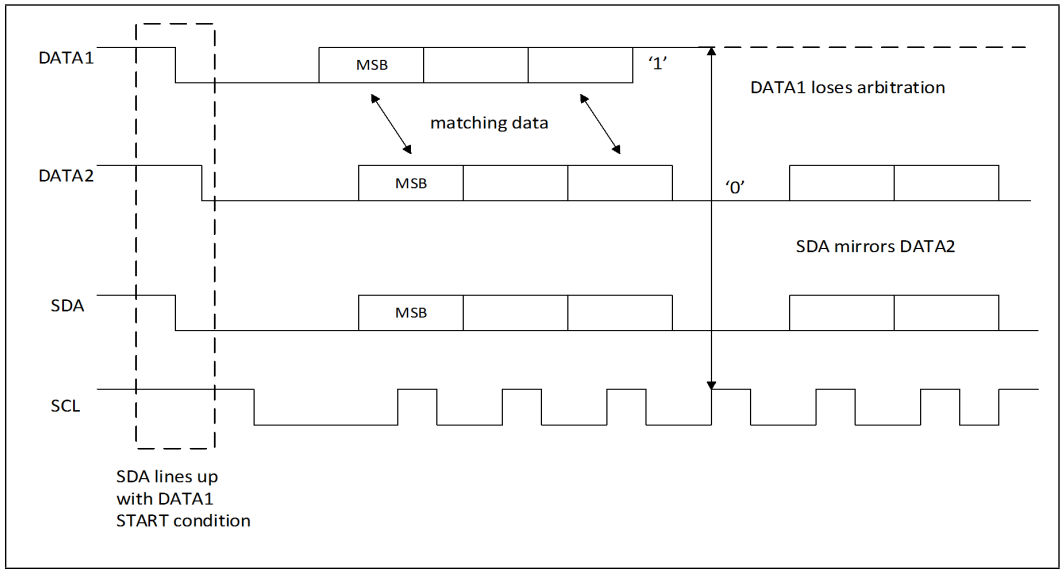
9.10.1.4.6 Multiple Master Arbitration

The I^2C bus protocol allows multiple masters to reside on the same bus. If two masters try to take control of the bus at the same time by generating a START condition at the same time, there is an arbitration procedure. Once a master has control of the bus, no other master can take control until the first master sends a STOP condition and places the bus in an idle state.

Arbitration takes place on the SDA line, while the SCL line is 1. The master, which transmits a 1 while the other master transmits 0, loses arbitration and turns off its data output stage. The master that lost arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase.

Upon detecting that it has lost arbitration to another master, this I^2C peripheral will stop generating SCL. Figure 41 illustrates the timing when two masters are arbitrating on the bus.

Figure 41. Multiple Master Arbitration



Arbitration is not allowed between the following conditions:

- A RESTART condition and a data bit.
- A STOP condition and a data bit.
- A RESTART condition and a STOP condition.

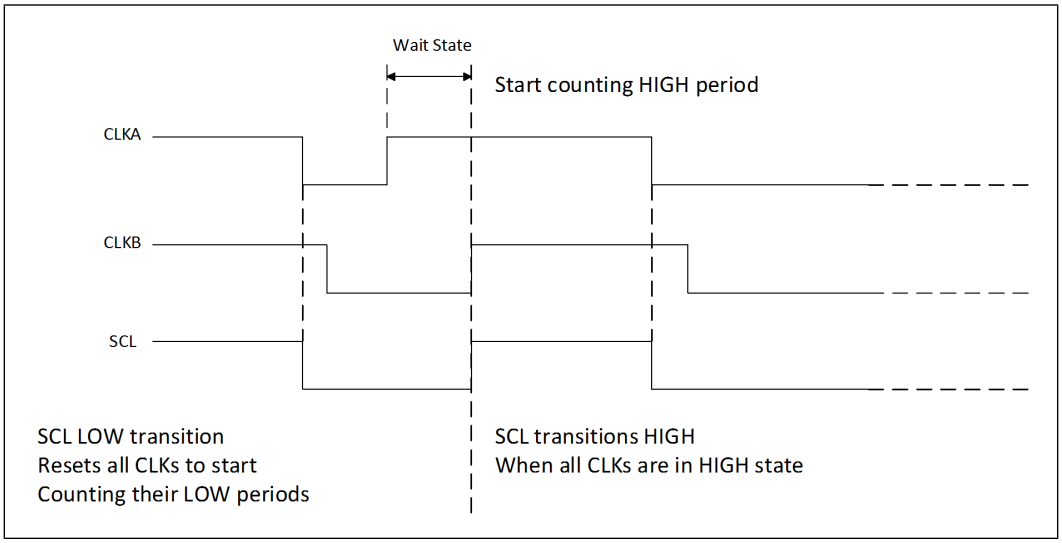
Slaves are not involved in the arbitration process.

9.10.1.4.7 Clock Synchronization

When two or more masters try to transfer information on the bus at the same time, they must arbitrate and synchronize the SCL clock. All masters generate their own clock to transfer messages. Data is valid only during the high period of SCL clock. Clock synchronization is performed using the wired-AND connection to the SCL signal. When the master transitions the SCL clock to 0, the master starts counting the low time of the SCL clock and transitions the SCL clock signal to 1 at the beginning of the next clock period. However, if another master is holding the SCL line to 0, then the master goes into a HIGH wait state until the SCL clock line transitions to 1.

All masters then count off their high time, and the master with the shortest high time transitions the SCL line to 0. The masters then counts out their low time and the one with the longest low time forces the other master into a HIGH wait state. Therefore, a synchronized SCL clock is generated, which is illustrated in Figure 42. Optionally, slaves may hold the SCL line low to slow down the timing on the I^2C bus.

Figure 42. Multi-Master Clock Synchronization



9.10.1.5 Slave Mode Operation

The following sections present the methods for using this I^2C peripheral as a slave device. Note that for any given bus transfer, this I^2C peripheral can be used as slave or master, but not both at once.

9.10.1.5.1 Initial Configuration

To use this I^2C peripheral as a slave, perform the following steps:

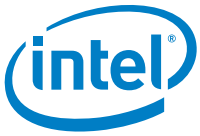
1. Disable this peripheral by writing a '0' to the IC_ENABLE register.
2. Poll the IC_ENABLE_STATUS.IC_EN register bit until it equals '0'.
3. Write this I^2C peripheral's slave address to the IC_SAR register.
4. Select the RX FIFO full threshold in the IC_RX_TL register (the default is 1 byte).
5. Select the slave address type (7 or 10-bit) and enable slave mode operation by programming the following IC_CON register bits:
   a. IC_10BITADDR_SLAVE = '0' for 7-bit or '1' for 10-bit addressing.
   b. IC_SLAVE_DISABLE = '0'.
   c. MASTER_MODE = '0'.
6. Enable this peripheral by writing a '1' to the IC_ENABLE register.

Note: A different addressing mode may be used when operating as a slave from when operating as a master.

9.10.1.5.2 Slave-Transmitter Operation for a Single Byte

When this peripheral is operating in slave mode and an I^2C master addresses it for reading, this peripheral acts as a slave-transmitter and the following events occur:

1. The I^2C master initiates an I^2C transfer with a read address that matches the slave address in the IC_SAR register of this slave.



2. This slave recognizes the direction of the transfer and acknowledges the address to indicate that it is acting as a slave-transmitter.
3. This slave asserts the RD_REQ interrupt and holds the SCL line low. This stalls the transfer until software services the interrupt. If the RD_REQ interrupt has been masked, then the CPU may poll the IC_RAW_INTR_STAT register to detect the interrupt.
4. The CPU must write the requested data to the IC_DATA_CMD with the CMD bit set to '0'.
5. The CPU must clear the RD_REQ interrupts by reading the IC_CLR_RD_REQ register.
6. This slave releases the SCL and transmits the byte.
7. The master may hold the I^2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

Note: If there is any data remaining in the TX FIFO before receiving the read request, then this peripheral asserts a TX_ABRT interrupt to flush the old data from the TX FIFO. The CPU must clear the interrupt by reading the IC_CLR_TX_ABRT register before depositing data in the TX_FIFO.

9.10.1.5.3 Slave-Receiver Operation for a Single Byte

When this peripheral is operating in slave mode and an I^2C master addresses it for writing, this peripheral acts as a slave-receiver and the following events occur:

1. The I^2C master initiates an I^2C transfer with a write address that matches the slave address in the IC_SAR register of this slave.
2. This slave recognizes the direction of the transfer and acknowledges the address to indicate that it is acting as a slave-receiver.
3. This slave receives the transmitted byte and places it in the receive buffer.
4. Assuming that the RX FIFO threshold is set to 1 in the
5. IC_RX_TL register, this slave asserts the RX_FULL interrupt. If the RX_FULL interrupt has been masked, then the CPU may poll the IC_RAW_INTR_STAT register to detect the interrupt.
6. The CPU may read the data byte from the IC_DATA_CMD register.
7. The master may hold the I^2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

Note: If the RX FIFO is completely filled with data when a byte is transferred, then an overflow occurs and this peripheral generates an RX_OVER interrupt while continuing subsequent I^2C transfers. Because a NACK is not generated, the CPU must recognize the overflow when interrupted and take appropriate actions to recover from lost data. Hence, there is a real time constraint on software to service the RX FIFO before the latter overflow as there is no way to re-apply pressure to the remote transmitting master. The software developer must select an RX FIFO threshold using the IC_RX_TL register that leaves room enough to satisfy the interrupt latency of their system.

9.10.1.5.4 Slave-Transmitter Operation for Bulk Transfers

In the standard I^2C protocol, all transactions are single byte transactions and the CPU responds to a remote master read request by writing one byte into this slave's TX FIFO. However, this I^2C peripheral is designed to store additional data in the TX FIFO so that subsequent read requests can take that data without raising an interrupt to get more. This eliminates the overhead incurred in raising an interrupt for each data byte requested. This situation occurs only when this I^2C peripheral is acting as a slave-transmitter.

If the master-receiver acknowledges the data sent by this slave-transmitter and there is no data in the slave's TX FIFO, this slave will hold the SCL line low while it raises the read request interrupt (RD_REQ) and waits for the CPU to write data into the TX FIFO.

The RD_REQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt handler. The interrupt handler may write one or more bytes into the TX FIFO. During the transmission of these bytes to the master, if the FIFO is emptied and the master acknowledges the last byte, this slave will raise the RD_REQ interrupt again because the master is requesting for more data.

If the software developer knows in advance that the remote master is going to request a multiple bytes, the TX FIFO can be written with multiple bytes and the remote master will receive them as a continuous stream of data. This slave will not hold the SCL line low or issue RD_REQ again unless the TX FIFO empties and the master indicates that more is requested by acknowledging the last byte. If the master NACKs a data byte before this slave's TX FIFO empties, this slave flushes the TX FIFO of any excess data bytes remaining in it and raises the transmit abort (TX_ABRT) interrupt.

9.10.1.6 Master Mode Operation

The following sections present the methods for using this I^2C peripheral as a master device. Note that for any given bus transfer, this I^2C peripheral can be used as slave or master, but not both at once.

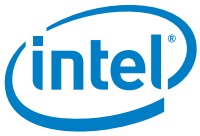
9.10.1.6.1 Initial Configuration

To use this I^2C peripheral as a master, perform the following steps:

1. Disable this peripheral by writing '0' to the IC_ENABLE register.
2. Select the speed, address type (7 or 10-bit), whether or not restart is enabled and enable master mode operation by programming the following IC_CON register bits:
   a. IC_RESTART_EN = '0' to disable RESTART or '1' to enable RESTART.
   b. IC_10BITADDR_MASTER = '0' for 7-bit or '1' for 10-bit addressing.
   c. SPEED = '1' for standard (up to 100 kb/s) or '2' for fast (up to 400 kb/s).
   d. IC_SLAVE_DISABLE = '1'.
   e. MASTER_MODE = '1'.
3. Write the target address of the I^2C slave to be addressed to the IC_TAR register. This register also indicates whether a General Call or a START BYTE command is going to be performed by I^2C.
4. Enable this master by writing a '1' to the IC_ENABLE register.

9.10.1.6.2 Master-Transmitter and Master Receiver Operation

This I^2C peripheral will dynamically and automatically switch back and forth between reading and writing. Write data to be transmitted to the lower byte of the IC_DATA_CMD register with the CMD bit set to '0'. Queue up one or more read commands by writing "don't cares" to the lower byte of the IC_DATA_CMD register with the CMD bit set to '1'. This master will continue to initiate transfers switching back and forth between reading and writing automatically so long as there are commands present in the transmit FIFO. If the transmit FIFO becomes empty, this master inserts a STOP condition after completing any in progress transfers.



9.10.1.7 Disabling I²C

This I²C peripheral features an IC_ENABLE_STATUS register to allow software to unambiguously determine when the hardware has completely shut down in response to the IC_ENABLE register being set from '1' to '0'. To disable this I²C peripheral, perform the following steps:

1. Set IC_ENABLE to '0'.
2. Poll the IC_ENABLE_STATUS.IC_EN register bit until it returns '0'.

Note: Since the time required to send or receive a byte over I²C is many CPU clock cycles, a timer can be used to schedule a suitable polling interval.

9.10.1.8 IC_CLK Frequency Configuration

In order to ensure proper I/O timing, the *CNT registers must be programmed before using this I²C peripheral as a master. There is no need to program the *CNT registers when operating as a slave. The *CNT registers are:

- IC_SS_SCL_HCNT – APB clock cycles in the minimum standard speed SCL high time.
- IC_SS_SCL_LCNT – APB clock cycles in the minimum standard speed SCL low time.
- IC_FS_SCL_HCNT – APB clock cycles in the minimum fast speed SCL high time.
- IC_FS_SCL_LCNT – APB clock cycles in the minimum fast speed SCL low time.

The following code snippet can be used to calculate the correct values for these registers where HCLK_FREQUENCY is the APB clock frequency in Hz.

```
// These minimum high and low times are in nanoseconds.  They represent

// the minimum amount of time a bus signal must remain either high or

// low to be interpreted as a logical high or low as per the I2C bus

// protocol.  These values are used in conjunction with an I2C input

// clock frequency to determine the correct values to be written to the

// clock count registers.

#define SS_MIN_SCL_HIGH  4000

#define SS_MIN_SCL_LOW   4700

#define FS_MIN_SCL_HIGH   600

#define FS_MIN_SCL_LOW   1300


// These are the slow and fast clock periods in nanoseconds

#define SS_PERIOD       10000

#define FS_PERIOD        2500


// I2C peripheral input clock frequency in kHz
```

```
#define I2C_CLOCK    (HCLK_FREQUENCY/1000)


// These are the slow and fast low and high count values calculated such that

// the high phase of SCL is maximized.

#define SS_SCL_LCNT  (uint32_t)((SS_MIN_SCL_LOW)*I2C_CLOCK/1000000 - 1)

#define SS_SCL_HCNT  (uint32_t)((SS_PERIOD - SS_MIN_SCL_LOW)*I2C_CLOCK/1000000 - 8)

#define FS_SCL_LCNT  (uint32_t)((FS_MIN_SCL_LOW)*I2C_CLOCK/1000000 - 1)

#define FS_SCL_HCNT  (uint32_t)((FS_PERIOD - FS_MIN_SCL_LOW)*I2C_CLOCK/1000000 - 8)
```

9.10.2 Registers

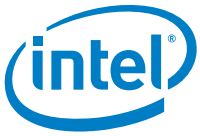
I^{2}C registers are aligned on 4-byte boundaries and can be accessed only as double words. Bit definitions are given in the following sections. Some registers may be written only when the I^{2}C is disabled. The I^{2}C Enable Register (IC_ENABLE) is used to enable or disable I^{2}C. Software should not disable the I^{2}C while it is active. The I^{2}C Status Register (IC_STATUS) can be used to determine if I^{2}C is active. If the I^{2}C is in the process of transmitting when it is disabled, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. The slave continues receiving until the remote master aborts the transfer, in which case the I^{2}C could be disabled. Registers that cannot be written to when the I^{2}C is enabled are indicated in their descriptions.

9.10.2.1 I^{2}C Control Register (IC_CON)

The I^{2}C Control Register is a read/write register memory mapped at address 0x9600 0000 to 0x9600 0003. This register is used to control the I^{2}C register. This register can only be written when I^{2}C is disabled, which corresponds to the IC_ENABLE register being set to 0.

Table 173. I^{2}C Control Register

Mnemonic	Access	Bits	Description	Reset
MASTER_MODE	R/W	0	Controls whether I^{2}C master is enabled. Software should ensure that if this bit is written with '1,' then bit 6 should also be written with a '1'. 0 = master disabled 1 = master enabled	0x1
SPEED	R/W	2:1	Control at which speed the I^{2}C operates; its setting is relevant only if master mode is enabled. Hardware protects against illegal values being programmed by software. 0 = reserved 1 = standard mode 2 = fast mode 3 = reserved	0x2



IC_10BITADDR_SLAVE	R/W	3	Controls whether I^2C slave supports 7 or 10 bit addressing on the I^2C interface reset when acting as a slave. The I^2C module will ignore 10-bit addresses when acting as a 7-bit address slave. 0 = 7-bit addressing 1 = 10-bit addressing	0x1
IC_10BITADDR_MASTER	R/W	4	Controls whether I^2C supports 7 or 10 bit addressing on the I^2C interface when acting as a master. Master generated transfers will use this number of address bits. 0 = 7-bit addressing 1 = 10-bit addressing	0x1
IC_RESTART_EN	R/W	5	Determines whether RESTART conditions may be sent when acting as a master. 0 = disable 1 = enable	0x1
IC_SLAVE_DISABLE	R/W	6	Controls whether I^2C slave is enabled. If this bit is set, I^2C functions only as a master and does not perform any action that requires a slave. 0 = slave enabled 1 = slave disabled	0x1
-	-	31:7	Reserved	X

9.10.2.2 I^2C Target Address Register (IC_TAR)

The I^2C Target Address Register is a read/write register memory mapped at address 0x9600 0004 to 0x9600 0007. This register is used to target address for any master transaction. This register can only be written when I^2C is disabled, which corresponds to the IC_ENABLE register being set to 0.

Table 174. I^2C Target Address Register

Mnemonic	Access	Bits	Description	Reset
IC_TAR	R/W	9:0	Target address for any master transaction. Note that if IC_TAR == IC_SAR, loopback is enabled. However, the FIFOs are shared between master and slave, so full loopback is not feasible. Only one direction loopback mode is supported (simplex), not duplex. A master cannot transmit to itself; it can transmit to only a slave.	0x055
GC_OR_START	R/W	10	If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed. 0 = General Call Address – after issuing a General Call, only write commands may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register. The I^2C remains in General Call mode until SPECIAL (bit 11) is cleared. 1 = START BYTE.	0x0
SPECIAL	R/W	11	This bit indicates whether software performs a General Call or START BYTE command. 0 = ignore GC_OR_START (bit 10) and use IC_TAR normally. 1 = perform special I2C command as specified in GC_OR_START bit.	0x0
-	-	31:12	Reserved	X

9.10.2.3 I²C Slave Address Register (IC_SAR)

The I²C Slave Address Register is a read/write register memory mapped at address 0x9600 0008 to 0x9600 000C. This register can only be written when I²C is disabled, which corresponds to the IC_ENABLE register being set to 0.

Table 175. I²C Slave Address Register

Mnemonic	Access	Bits	Description	Reset
IC_SAR	R/W	9:0	Holds the slave address when the I²C is operating as a slave. Set IC_SAR = IC_TAR for loopbackt.	0x055
-	-	15:10	Reserved	X

9.10.2.4 I²C Rx/Tx Data Buffer and Command Register (IC_DATA_CMD)

The I²C Rx/Tx Data Buffer and Command Register is a read/write register memory mapped at address 0x9600 0010 to 0x9600 0013. This is the register the CPU writes to when filling the TX FIFO and the CPU reads from when retrieving bytes from RX FIFO.

Table 176. I²C Rx/Tx Data Buffer and Command Register

Mnemonic	Access	Bits	Description	Reset
DAT	R/W	7:0	Contains the data to be transmitted or received on the I²C bus. If you are writing a read command (i.e. CMD = 1) to the TX FIFO, bits 7:0 (DAT) are ignored. When you read this register, these bits return the value of data received on the I²C interface.	0x0
CMD	W	8	Controls whether a read or a write is performed. It controls the direction when it acts as a master. 1 = Read 0 = Write When a command is entered in the TX FIFO, this bit distinguishes the write and read commands.	0x0
-	-	31:9	Reserved	X

9.10.2.5 I²C Standard Speed Clock SCL High Count Register (IC_SS_SCL_HCNT)

The I²C Standard Speed Clock SCL High Count Register is a read/write register memory mapped at address 0x9600 0014 to 0x9600 0017. This register is used to set the width of the SCL high pulse for standard speed.

Table 177. I²C Standard Speed Clock SCL High Count Register

Mnemonic	Access	Bits	Description	Reset
IC_SS_SCL_HCNT	R/W	15:0	This register must be set before any I²C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed. This register can be written only when the I²C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. ***NOTE:*** This register must not be programmed to a value higher than 65,525 because the I²C peripheral uses a 16-bit counter to flag an I²C bus idle condition when this counter reaches a value of IC_SS_SCL_HCNT + 10.	0x400
-	-	31:8	Reserved	X

9.10.2.6 I²C Standard Speed Clock SCL Low Count Register (IC_SS_SCL_LCNT)

The I²C Standard Speed Clock SCL Low Count Register is a read/write register memory mapped at address 0x9600 0018 to 0x9600 001B. This register is used to set the width of the SCL low pulse for standard speed.

Table 178. I²C Standard Speed Clock SCL Low Count Register

Mnemonic	Access	Bits	Description	Reset
IC_SS_SCL_LCNT	R/W	15:0	This register must be set before any I²C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low-period count for standard speed. This register can be written only when the I²C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set.	0x470
-	-	31:8	Reserved	X

9.10.2.7 I²C Fast Speed Clock SCL High Count Register (IC_FS_SCL_HCNT)

The I²C Fast Speed Clock SCL High Count Register is a read/write register memory mapped at address 0x9600 001C to 0x9600 001F. This register is used to set the width of the SCL high pulse for fast speed.

Table 179. I²C Fast Speed Clock SCL High Count Register

Mnemonic	Access	Bits	Description	Reset
IC_FS_SCL_HCNT	R/W	15:0	This register must be set before any I²C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed. This register can be written only when the I²C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set.	0x60
-	-	31:8	Reserved	X

9.10.2.8 I²C Fast Speed Clock SCL Low Count Register (IC_FS_SCL_LCNT)

The I²C Fast Speed Clock SCL Low Count Register is a read/write register memory mapped at address 0x9600 0020 to 0x9600 0023. This register is used to set the width of the SCL low pulse for fast speed.

Table 180. I²C Fast Speed Clock SCL Low Count Register

Mnemonic	Access	Bits	Description	Reset
IC_FS_SCL_LCNT	R/W	15:0	This register must be set before any I²C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low-period count for standard speed. This register can be written only when the I²C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set.	0x130
-	-	31:8	Reserved	X

9.10.2.9 I²C Interrupt Status Register (IC_INTR_STAT)

The I²C Interrupt Status Register is a read only register memory mapped at address 0x9600 002C to 0x9600 002F. This register is used to read the masked interrupt status. The CPU is interrupted when any bit is set.

Table 181. I²C Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
RX_UNDER	R	0	Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register. This interrupt is cleared by reading the IC_CLR_RX_UNDER register. If the peripheral is disabled (IC_ENABLE==0), this bit keeps its level until the master or slave state machines go into idle. This interrupt is then cleared.	0x0
RX_OVER	R	1	Set if the receive buffer is completely full and an additional byte is received from an external I²C device. This I²C peripheral acknowledges the received byte, but any data bytes received while the FIFO is full are lost. This interrupt is cleared by reading the IC_CLR_RX_OVER register. If the module is disabled (IC_ENABLE==0), this bit keeps its level until the master or slave state machines go into idle. This interrupt is then cleared.	0x0
RX_FULL	R	2	Set when the receive buffer reaches or goes above the RX_TL threshold set in the IC_RX_TL register. Unlike other interrupts, this interrupt is automatically cleared by hardware when the receive buffer level goes below the threshold. If the module is disabled (IC_ENABLE==0), the RX FIFO is flushed and held in reset. Therefore, the RX FIFO is not full, and this interrupt is cleared immediately.	0x0
TX_OVER	R	3	Set during transmit if the transmit buffer is full and the processor attempts to issue another I²C command by writing to the IC_DATA_CMD register. This interrupt is cleared by reading the IC_CLR_TX_OVER register. If the module is disabled (IC_ENABLE==0), this bit keeps its level until the master or slave state machines go into idle. This interrupt is then cleared.	0x0
TX_EMPTY	R	4	Set when the transmit buffer is at or below the threshold value set in the IC_TX_TL register. Unlike other interrupts, this interrupt is automatically cleared by hardware when the buffer level goes above the threshold. If the module is disabled (IC_ENABLE==0), the TX FIFO is flushed and held in reset. Therefore, the TX FIFO is empty, so this bit is set until activity in the master or slave state machines cease. This interrupt is then cleared.	0x0
RD_REQ	R	5	Set when this I²C peripheral is acting as a slave and another I²C master is attempting to read data from it. This I²C slave holds the I²C bus in a wait state (SCL=0) until this interrupt is serviced. To service this interrupt, the CPU must write the requested data to the IC_DATA_CMD register and read the IC_CLR_RD_REQ register. This interrupt is then cleared.	0x0

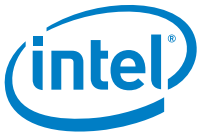
TX_ABRT	R	6	Set when this I2C peripheral is acting as a transmitter and is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or a slave and is referred to as a "transmit abort". When this interrupt is set, the IC_TX_ABRT_SOURCE register indicates the reason for the "transmit abort." ***NOTE:*** This peripheral flushes the TX FIFO whenever this interrupt is set. The TX FIFO remains in this flushed state until the IC_CLR_TX_ABRT register is read. Once this read is performed, the TX FIFO is ready to accept more data from the CPU.	0x0
RX_DONE	R	7	Set when this I2C peripheral is acting as a slave-transmitter and the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission indicating that the transmission is done. This interrupt is cleared by reading the IC_CLR_RX_DONE register.	0x0
ACTIVITY	R	8	Set when there is activity on the bus, and stays set until it is cleared. There are four ways to clear it: · Disabling the I2C peripheral. · Reading the IC_CLR_ACTIVITY register · Reading the IC_CLR_INTR register · System reset	0x0
STOP_DET	R	9	Set when a STOP condition has occurred on the I2C interface regardless of whether this I2C peripheral is acting as a slave or master. This interrupt is cleared by reading the IC_CLR_STOP_DET register.	0x0
START_DET	R	10	Set when a START or RESTART condition has occurred on the I2C interface regardless of whether this I2C peripheral is acting as a slave or master. This interrupt is cleared by reading the IC_CLR_START_DET register.	0x0
GEN_CALL	R	11	Set only when a General Call address is received and is acknowledged. This interrupt stays set until it is cleared either by disabling this I2C peripheral or when the CPU reads the IC_CLR_GEN_CALL register. The received data is stored in the RX buffer.	0x0
-	-	31:12	Reserved	X

9.10.2.10 I2C Interrupt Mask Register (IC_INTR_MASK)

The I2C Interrupt Mask Register is a read/write register memory mapped at address 0x9600 0030 to 0x9600 0033. This register is used to program the interrupt masks.

Table 182. I2C Interrupt Mask Register

Mnemonic	Access	Bits	Description	Reset
M_RX_UNDER	R	0	When '0', RX underflow interrupts are masked. Returns last value written when read.	0x1
M_RX_OVER	R	1	When '0', RX overflow interrupts are masked. Returns last value written when read.	0x1
M_RX_FULL	R	2	When '0', RX full interrupts are masked. Returns last value written when read.	0x1
M_TX_OVER	R	3	When '0', TX overflow interrupts are masked. Returns last value written when read.	0x1
M_TX_EMPTY	R	4	When '0', TX empty interrupts are masked. Returns last value written when read.	0x1
M_RD_REQ	R	5	When '0', read request interrupts are masked. Returns last value written when read.	0x1



M_TX_ABRT	R	6	When '0', TX abort interrupts are masked. Returns last value written when read.	0x1
M_RX_DONE	R	7	When '0', RX done interrupts are masked. Returns last value written when read.	0x1
M_ACTIVITY	R	8	When '0', activity interrupts are masked. Returns last value written when read.	0x0
M_STOP_DET	R	9	When '0', STOP interrupts are masked. Returns last value written when read.	0x0
M_START_DET	R	10	When '0', START interrupts are masked. Returns last value written when read.	0x0
M_GEN_CALL	R	11	When '0', general call interrupts are masked. Returns last value written when read.	0x1
-	-	31:12	Reserved	X

9.10.2.11 I²C Raw Interrupt Status Register (IC_RAW_INTR_STAT)

The I²C Raw Interrupt Status Register is a read/write register memory mapped at address 0x9600 0034 to 0x9600 0037. This register is used to read the raw interrupt status before masking.

Table 183. I²C Raw Interrupt Status Register

Mnemonic	Access	Bits	Description	Reset
R_RX_UNDER	R	0	Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register. This interrupt is cleared by reading the IC_CLR_RX_UNDER register. If the peripheral is disabled (IC_ENABLE==0), this bit keeps its level until the master or slave state machines go into idle. This interrupt is then cleared.	0x0
R_RX_OVER	R	1	Set if the receive buffer is completely full and an additional byte is received from an external I²C device. This I²C peripheral acknowledges the received byte, but any data bytes received while the FIFO is full are lost. This interrupt is cleared by reading the IC_CLR_RX_OVER register. If the module is disabled (IC_ENABLE==0), this bit keeps its level until the master or slave state machines go into idle. This interrupt is then cleared.	0x0
R_RX_FULL	R	2	Set when the receive buffer reaches or goes above the RX_TL threshold set in the IC_RX_TL register. Unlike other interrupts, this interrupt is automatically cleared by hardware when receive buffer level goes below the threshold. If the module is disabled (IC_ENABLE==0), the RX FIFO is flushed and held in reset. Therefore, the RX FIFO is not full, and this interrupt is cleared immediately.	0x0
R_TX_OVER	R	3	Set during transmit if the transmit buffer is full and the processor attempts to issue another I²C command by writing to the IC_DATA_CMD register. This interrupt is cleared by reading the IC_CLR_TX_OVER register. If the module is disabled (IC_ENABLE==0), this bit keeps its level until the master or slave state machines go into idle. This interrupt is then cleared.	0x0
R_TX_EMPTY	R	4	Set when the transmit buffer is at or below the threshold value set in the IC_TX_TL register. Unlike other interrupts, this interrupt is automatically cleared by hardware when the buffer level goes above the threshold. If the module is disabled (IC_ENABLE==0), the TX FIFO is flushed and held in reset. Therefore, the TX FIFO is empty, so this bit is set until activity in the master or slave state machines cease. This interrupt is then cleared.	0x0

R_RD_REQ	R	5	Set when this I^2C peripheral is acting as a slave and another I^2C master is attempting to read data from it. This I^2C slave holds the I^2C bus in a wait state (SCL=0) until this interrupt is serviced. To service this interrupt, the CPU must write the requested data to the IC_DATA_CMD register and read the IC_CLR_RD_REQ register. This interrupt is then cleared.	0x0
R_TX_ABRT	R	6	Set when this I^2C peripheral is acting as a transmitter and is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I^2C master or a slave and is referred to as a "transmit abort". When this interrupt is set, the IC_TX_ABRT_SOURCE register indicates the reason for the "transmit abort." ***NOTE:*** This peripheral flushes the TX FIFO whenever this interrupt is set. The TX FIFO remains in this flushed state until the IC_CLR_TX_ABRT register is read. Once this read is performed, the TX FIFO is ready to accept more data from the CPU.	0x0
R_RX_DONE	R	7	Set when this I^2C peripheral is acting as a slave-transmitter and the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission indicating that the transmission is done. This interrupt is cleared by reading the IC_CLR_RX_DONE register.	0x0
R_ACTIVITY	R	8	Set when there is activity on the bus, and stays set until it is cleared. There are four ways to clear it: · Disabling the I^2C peripheral. · Reading the IC_CLR_ACTIVITY register · Reading the IC_CLR_INTR register · System reset	0x0
R_STOP_DET	R	9	Set when a STOP condition has occurred on the I^2C interface regardless of whether this I^2C peripheral is acting as a slave or master. This interrupt is cleared by reading the IC_CLR_STOP_DET register.	0x0
R_START_DET	R	10	Set when a START or RESTART condition has occurred on the I^2C interface regardless of whether this I^2C peripheral is acting as a slave or master. This interrupt is cleared by reading the IC_CLR_START_DET register.	0x0
R_GEN_CALL	R	11	Set only when a General Call address is received and is acknowledged. This interrupt stays set until it is cleared either by disabling this I^2C peripheral or when the CPU reads the IC_CLR_GEN_CALL register. The received data is stored in the RX buffer.	0x0
-	-	31:12	Reserved	X

9.10.2.12 I²C Receive FIFO Threshold Register (IC_RX_TL)

The I²C Receive FIFO Threshold Register is a read/write register memory mapped at address 0x9600 0038 to 0x9600 003B. This register is used to program the receive FIFO Threshold Level.

Table 184. I²C Receive FIFO Threshold Register

Mnemonic	Access	Bits	Description	Reset
RX_TL	R/W	2:0	Controls the level of entries (or above) that triggers the RX_FULL interrupt. The actual threshold level equals the programmed value plus one.	0x0
-	-	31:3	Reserved	X

9.10.2.13 I²C Transmit FIFO Threshold Register (IC_TX_TL)

I²C Transmit FIFO Threshold Register is a read/write register memory mapped at address 0x9600 003C to 0x9600 003F. This register is used to program the transmit FIFO threshold level.

Table 185. I²C Transmit FIFO Threshold Register

Mnemonic	Access	Bits	Description	Reset
TX_TL	R/W	2:0	Controls the level of entries (or below) that triggers the TX_EMPTY interrupt. The actual threshold level equals the programmed value.	0x0
-	-	31:3	Reserved	X

9.10.2.14 I²C Clear Combined and Individual Interrupt Register (IC_CLR_INTR)

I²C Clear Combined and Individual Interrupt Register is a read only register memory mapped at address 0x9600 0040 to 0x9600 0043. This register is used to clear the combined interrupt, all individual interrupts, and the IC_TX_ABRT_SOURCE register. This bit does not clear hardware clearable interrupts – only software clearable interrupts. Refer to bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE. The value read has no meaning, and this register always returns zero.

Table 186. I²C Clear Combined and Individual Interrupt Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0

9.10.2.15 I²C Clear RX_UNDER Interrupt Register (IC_CLR_RX_UNDER)

I²C Clear RX_UNDER Interrupt Register is a read only register memory mapped at address 0x9600 0044 to 0x9600 0047. This register is used to clear the RX_UNDER interrupt. The value read has no meaning, and this register always returns zero.

Table 187. I²C Clear RX_UNDER Interrupt Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0

9.10.2.16 I²C Clear RX_OVER Interrupt Register (IC_CLR_RX_OVER)

I²C Clear RX_OVER Interrupt Register is a read only register memory mapped at address 0x9600 0048 to 0x9600 004B. This register is used to clear the RX_OVER interrupt. The value read has no meaning, and this register always returns zero.

Table 188. I²C Clear RX_OVER Interrupt Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0

9.10.2.17 I²C Clear TX_OVER Interrupt Register (IC_CLR_TX_OVER)

I²C Clear TX_OVER Interrupt Register is a read only register memory mapped at address 0x9600 004C to 0x9600 004F. This register is used to clear the TX_OVER interrupt. The value read has no meaning, and this register always returns zero.

Table 189. I²C Clear TX_OVER Interrupt Register

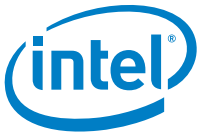
Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0

9.10.2.18 I²C Clear RD_REQ Interrupt Register (IC_CLR_RD_REQ)

I²C Clear RD_REQ Interrupt Register is a read only register memory mapped at address 0x9600 0050 to 0x9600 0053. This register is used to clear the RD_REQ interrupt. The value read has no meaning, and this register always returns zero.

Table 190. I²C Clear RD_REQ Interrupt Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0



9.10.2.19 I²C Clear TX_ABORT Interrupt Register (IC_CLR_TX_ABORT)

I²C Clear TX_ABORT Interrupt Register is a read only register memory mapped at address 0x9600 0054 to 0x9600 0057. This register is used to clear the TX_ABORT interrupt. The value read has no meaning, and this register always returns zero.

Table 191. I²C Clear TX_ABORT Interrupt Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0

9.10.2.20 I²C Clear RX_DONE Interrupt Register (IC_CLR_RX_DONE)

I²C Clear RX_DONE Interrupt Register is a read only register memory mapped at address 0x9600 0058 to 0x9600 005B. This register is used to clear the RX_DONE interrupt. The value read has no meaning, and this register always returns zero.

Table 192. I²C Clear RX_DONE Interrupt Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0

9.10.2.21 I²C Clear ACTIVITY Interrupt Register (IC_CLR_ACTIVITY)

I²C Clear ACTIVITY Interrupt Register is a read only register memory mapped at address 0x9600 005C to 0x9600 005F. This register is used to clear the ACTIVITY interrupt. The value read has no meaning, and this register always returns zero.

Table 193. I²C Clear ACTIVITY Interrupt Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0

9.10.2.22 I²C Clear STOP_DET Interrupt Register (IC_CLR_STOP_DET)

I²C Clear STOP_DET Interrupt Register is a read only register memory mapped at address 0x9600 0060 to 0x9600 0063. This register is used to clear the STOP_DET interrupt. The value read has no meaning, and this register always returns zero.

Table 194. I²C Clear STOP_DET Interrupt Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0

9.10.2.23 I²C Clear START_DET Interrupt Register (IC_CLR_START_DET)

I²C Clear START_DET Interrupt Register is a read only register memory mapped at address 0x9600 0064 to 0x9600 0067. This register is used to clear the START_DET interrupt. The value read has no meaning, and this register always returns zero.

Table 195. I²C Clear START_DET Interrupt Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0

9.10.2.24 I²C Clear GEN_CALL Interrupt Register (IC_CLR_GEN_CALL)

I²C Clear GEN_CALL Interrupt Register is a read only register memory mapped at address 0x9600 0064 to 0x9600 0067. This register is used to clear the GEN_CALL interrupt. The value read has no meaning, and this register always returns zero.

Table 196. I²C Clear GEN_CALL Interrupt Register

Mnemonic	Access	Bits	Description	Reset
-	-	31:0	Reserved	0x0

9.10.2.25 I²C Enable Register (IC_ENABLE)

The I²C Enable Register is a read/write register memory mapped at address 0x9600 006C to 9600 006F. This register is used to control whether or not the I²C is enabled.

Table 197. I²C Enable Register

Mnemonic	Access	Bits	Description	Reset
ENABLE	R/W	0	When '1', I²C is enabled. When '0' I²C is disabled. When I²C is disabled, the following occurs: · The TX FIFO and RX FIFO are flushed. · Status bits in the IC_INTR_STAT register are still active until state machines reach IDLE state. If this I²C peripheral is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer completes. If the peripheral is receiving, it stops the current transfer at the end of the current byte and does not acknowledge the transfer.	0x0
-	-	31:1	Reserved	X

9.10.2.26 I²C Status Register (IC_STATUS)

The I²C Status Register is a read only register memory mapped at address 0x9600 0070 to 0x9600 0073. This register is used to indicate the current transfer status and FIFO status. The status register may be read at any time. None of the bits in this register request an interrupt.

Table 198. I²C Status Register

Mnemonic	Access	Bits	Description	Reset
ACTIVITY	R	0	I²C Activity Status – when '1', this I²C peripheral is active. When '0', it is inactive. This bit is the logical or of SLV_ACTIVITY and MST_ACTIVITY.	0x0
TFNF	R	1	Transmit FIFO Not Full – when '0', the transmit FIFO is full. When '1' the transmit FIFO is not full. This bit clears immediately after disabling this I²C peripheral.	0x1
TFE	R	2	Transmit FIFO Completely Empty – when '0', the transmit FIFO is not empty. When '1', the transmit FIFO is empty. This bit clears immediately after disabling this I²C peripheral.	0x1
RFNE	R	3	Receive FIFO Not Empty – when '0', the receive FIFO is empty. When '1', the receive FIFO is not empty. This bit clears immediately after disabling this I²C peripheral.	0x0
RFF	R	4	Receive FIFO Completely Full – when '0', the receive FIFO is not full. When '1', the receive FIFO is full. This bit clears immediately after disabling this I²C peripheral.	0x0
MST_ACTIVITY	R	5	Master FSM Activity Status – when '1', the master FSM is not in the IDLE state. When '0', the master FSM is in the IDLE state. This bit clears immediately after the current transfer completes – not immediately after disabling this I²C peripheral.	0x0
SLV_ACTIVITY	R	5	Slave FSM Activity Status – when '1', the slave FSM is not in the IDLE state. When '0', the slave FSM is in the IDLE state. This bit clears immediately after the current transfer completes – not immediately after disabling this I²C peripheral.	0x0
-	-	31:7	Reserved	X

9.10.2.27 I²C Transmit FIFO Level Register (IC_TXFLR)

I²C Transmit FIFO Level Register is a read only register memory mapped at address 0x9600 0074 to 0x9600 0077. This register contains the number of valid data entries in the transmit FIFO buffer. It is cleared whenever:

- This I²C peripheral is disabled.
- A transmit abort has occurred (TX_ABORT == 1 in the IC_RAW_INTR_STAT register).
- A slave bulk transmit has been aborted.

Table 199. I²C Transmit FIFO Level Register

Mnemonic	Access	Bits	Description	Reset
TXFLR	R	3:0	Contains the number of valid data entries in the transmit FIFO.	0x0
-	-	31:4	Reserved	X

9.10.2.28 I²C Receive FIFO Level Register (IC_RXFLR)

The I²C Receive FIFO Level Register is a read only register memory mapped at address 0x9600 0078 to 0x9600 007B. This register contains the number of valid data entries in the receive FIFO buffer. It is cleared whenever:

- This I²C peripheral is disabled.
- A transmit abort has occurred (TX_ABORT == 1 in the IC_RAW_INTR_STAT register).

Table 200. I²C Receive FIFO Level Register

Mnemonic	Access	Bits	Description	Reset
RXFLR	R	3:0	Contains the number of valid data entries in the receive FIFO.	0x0
-	-	31:4	Reserved	X

9.10.2.29 I²C SDA Hold Time Length Register (IC_SDA_HOLD)

The I²C SDA Hold Time Length Register is a read/write register memory mapped at address 0x9600 007C to 0x9600 007F. This register programs the amount of hold time on the SDA signal after a negative edge of SCL in APB clock cycle units in both master and slave mode. The value programmed must be greater than the minimum hold time for either mode used — one cycle for master mode and seven cycles for slave mode. Writes to this register succeed only when the I²C peripheral is disabled (IC_ENABLE==0).

The programmed SDA hold time cannot exceed at any time the duration of the low part of SCL. Therefore the programmed value can be no larger than the SCL low time in APB clocks minus two.

Table 201. I²C SDA Hold Time Length Register

Mnemonic	Access	Bits	Description	Reset
IC_SDA_HOLD	R/W	15:0	Sets the required SDA hold time in APB clock units.	0x1
-	-	31:16	Reserved	X

9.10.2.30 I²C SDA Setup Register (IC_SDA_SETUP)

The I²C SDA Setup Register is a read/write register memory mapped at address 0x9600 0094 to 0x9600 0097. This register controls the amount of time delay (in APB clock units) introduced in the rising edge of SCL—relative to SDA changing—by holding SCL low when this I²C peripheral services a read request while operating as a slave-transmitter.

Table 202. I²C SDA Setup Register

Mnemonic	Access	Bits	Description	Reset
SDA_SETUP	R/W	7:0	The SDA setup time in APB clock units. The actual SDA setup time is this value minus one. This value is meaningless when the I²C peripheral is acting as a master.	0x64
-	-	31:8	Reserved	X

9.10.2.31 I²C Transmit Abort Source Register (IC_TX_ABRT_SOURCE)

The I²C Transmit Abort Source Register is a read only register memory mapped at address 0x9600 0080 to 0x9600 0083. This register is used to indicate the source of the TX_ABRT interrupt. Except for ABRT_SBYTE_NORSTRT, this register is cleared whenever the IC_CLR_TX_ABRT register or the IC_CLR_INTR register is read. To clear ABRT_SBYTE_NORSTRT, the source must be fixed first; RESTART must be enabled (IC_CON.IC_RESTART_EN = 1), the IC_TAR.SPECIAL bit must be cleared, or the IC_TAR.GC_OR_START bit must be cleared. Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, it clears for one clock cycle then re-asserts.

Table 203. I²C Transmit Abort Source Register

Mnemonic	Access	Bits	Description	Reset
ABRT_7B_ADDR_NOACK	R	0	When '1', this master is in 7-bit addressing mode and the address sent was not acknowledged by any slave.	0
ABRT_10ADDR1_NOACK	R	1	When '1', this master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave.	0
ABRT_10ADDR2_NOACK	R	2	When '1', this master is in 10-bit address mode and the second address byte of the 10-bit address was not acknowledged by any slave.	0
ABRT_TXDATA_NOACK	R	3	When '1', this master transmitter has received an acknowledgment for the address, but when it sent data byte(s) following the address, it did not receive an acknowledgment from the remote slave(s).	0
ABRT_GCALL_NOACK	R	4	When '1', this master transmitter has sent a General Call, but no slave on the bus acknowledged the General Call.	0
ABRT_GCALL_READ	R	5	When '1', this master transmitter has sent a General Call, but the CPU programmed the byte following the General Call to be a read from the bus (IC_DATA_CMD.CMD = 1).	0
-	-	6	Reserved	X
ABRT_SBYTE_ACKDET	R	7	When '1', this master has sent a START byte and the START byte was acknowledged (wrong behavior).	0
-	-	8	Reserved	X
ABRT_SBYTE_NORSTRT	R	9	When '1', this master is trying to send a START byte, but restart is disabled (IC_CON.IC_RESTART_EN = 0). To clear this bit, the source of the ABRT_SBYTE_NORSTRT must be fixed first; restart must be enabled (IC_CON IC_RESTART_EN = 1), the IC_TAR.SPECIAL bit must be cleared, or the IC_TAR.GC_OR_START bit must be cleared. Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, it clears for one clock cycle and then re-asserts.	0

ABRT_10B_RD_NORSTRT	R	10	When '1', this master receiver is trying to send a read command in 10-bit addressing mode, but restart is disabled (IC_CON.IC_RESTART_EN = 0).	0
ABRT_MASTER_DIS	R	11	When '1', the CPU is trying to initiate a master operation with master mode disabled (IC_CON.MASTER_MODE = 0)	0
ARB_LOST	R	12	When '1', this master transmitter has lost arbitration, or if ABRT_SLV_ARBLOST is also set, then this slave transmitter has lost arbitration. ***Note:*** I2C can be both master and slave at the same time.	0
ABRT_SLVFLUSH_TXFIFO	R	13	When '1', this slave transmitter has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO.	0
ABRT_SLV_ARBLOST	R	14	When '1', this slave transmitter lost the bus while transmitting data to a remote master. ARB_LOST is set at the same time. ***Note:*** Even though the slave never "owns" the bus, something could go wrong on the bus. This is a failsafe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then this slave transmitter no longer owns the bus.	0
ABRT_SLVRD_INTX	R	15	When '1', the CPU has responded to a slave mode request for data by writing '1' in IC_DATA_CMD.CMD.	0
-	-	31:16	Reserved	X

9.10.2.32 I²C ACK General Call Register (IC_ACK_GENERAL_CALL)

The I²C ACK General Call Register is a read/write register memory mapped at address 0x9600 0098 to 0x9600 009B. This register controls whether I²C responds with a ACK or NACK when it receives an I²C General Call address.

Table 204. I²C ACK General Call Register

Mnemonic	Access	Bits	Description	Reset
ACK_GEN_CALL	R/W	0	When set to 1, I²C responds with an ACK when it receives a General Call.	0x1
-	-	31:1	Reserved	X

9.10.2.33 I²C Enable Status Register (IC_ENABLE_STATUS)

The I²C Enable Status Register is a read only register memory mapped at address 0x9600 009C to 0x9600 009F. This register is used to report this I²C peripheral's hardware status when the IC_ENABLE register is set from 1 to 0; that is, when this I²C peripheral is disabled.

Table 205. I²C Enable Status Register

Mnemonic	Access	Bits	Description	Reset
IC_EN	R	0	When '1', this I²C peripheral is enabled or disabled, but finishing an in progress bus operation. When '0', this I²C peripheral is disabled and all bus operations have completed.	0
SLV_DISABLED_WHILE_BUSY	R	1	Slave Disabled While Busy (Transmit, Receive) – when '1', a potential or active slave operation has been aborted due to the setting of the IC_ENABLE register from '1' to '0'. This bit is set when the CPU writes a '0' to the IC_ENABLE register while: (a) this slave is receiving the address byte of a read operation from a remote master, or (b) address and data bytes of a write operation from a remote master. As a result, this slave has forced a NACK during some part of an I²C transfer, irrespective of whether the I²C address matches the slave address set in the IC_SAR register or the transfer completed before IC_ENABLE was set to '0' but has not taken effect. ***NOTE:*** If the remote I²C master terminates the transfer with a STOP condition before this slave has had a chance to NACK the transfer, and IC_ENABLE has been set to '0', this bit will still be set to '1'. When '0', this I²C peripheral has been disabled while there is master activity, or when the I²C bus was idle. ***NOTE:*** this bit is valid only when IC_EN is '0'.	0
SLV_RX_DATA_LOST	R	2	Slave Received Data Lost – when '1', a slave receiver operation has been aborted with at least one data byte received from an I²C transfer due to the setting of IC_ENABLE from '1' to '0'. This I2C peripheral has been actively engaged in an aborted I²C transfer (with matching address) and the data phase of the I²C transfer has been entered, even though a data byte has been responded to with a NACK. ***NOTE:*** If the remote I²C master terminates the transfer with a STOP condition before this slave has had a chance to NACK the transfer, and IC_ENABLE has been set to '0', this bit will still be set to '1'. When '0', this I²C peripheral has been disabled without being actively involved in the data phase of a slave receiver transfer. ***NOTE:*** this bit is valid only when IC_EN is '0'.	0

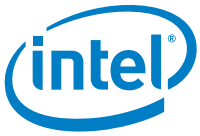
-	-	31:3	Reserved	X

9.10.2.34 I²C Component Parameter Register 1 (IC_COMP_PARAM_1)

The I²C Component Parameter Register 1 is a read only register memory mapped at addresses 0x9600 00F4 to 0x9600 00F7. This register is a constant read-only register that contains encoded information about the component's parameter settings.

Table 206. I²C Component Parameter Register 1

Mnemonic	Access	Bits	Description	Reset
APB_DATA_WIDTH	R	1:0	Width of the APB data bus. 0 = 8 bits 1 = 16 bits 2 = 32 bits 3 = Reserved	0x2
MAX_SPEED_MODE	R	3:2	IC_MAX_SPEED_MODE, maximum I²C mode supported by I²C. 0 = Reserved 1 = Standard 2 = Fast 3 = High	0x2
HC_COUNT_VALUES	R	4	IC_HC_COUNT_VALUES, Setting this parameter to 0 will allow the CNT registers to be writable. Regardless of the setting, the CNT registers are always readable and have reset values from the corresponding configuration parameters. 0 = False 1 = True	0x0
INTR_IO	R	5	IC_INTR_IO, controls which interrupt outputs are present. Individual: each interrupt source has its own output. Combined: all interrupt sources are combined in to a single output. 0 = Individual 1 = Combined	0x1
HAS_DMA	R	6	C_HAS_DMA configures the inclusion of DMA handshaking interface signals. 0 = False 1 = True	0x0
ADD_ENCODED_PARAMS	R	7	IC_ADD_ENCODED_PARAMS, by adding this in the encoded parameters gives firmware an easy and quick way of identifying the component within an I/O memory map. 0 = False 1 = True	0x1
RX_BUFFER_DEPTH	R	15:8	IC_RX_BUFFER_DEPTH, depth of receive buffer. 0x01 = 2 0x02 = 3 0x07 = 8 … 0xFF = 256	0x7



TX_BUFFER_DEPTH	R	23:16	IC_TX_BUFFER_DEPTH, depth of transmit buffer. 0x01 = 2 0x02 = 3 0x07 = 8 … 0xFF = 256	0x7
		31:24	Reserved	X

9.10.2.35 I²C Component Version (IC_COMP_VERSION)

The I²C Component Version is a read only register memory mapped at address 0x9600 00F8 to 0x9600 00FB. This register is used to read the peripheral's version code.

Table 207. I²C Component Version

Mnemonic	Access	Bits	Description	Reset
IC_COMP_VERSION	R	31:0	Version Code Register: always returns the version code 1.15 as the big endian ASCII string "115*" or 0x3131 352A.	0x3131 352A

9.10.2.36 I²C Component Type Register (IC_COMP_TYPE)

The I²C Component Type Register is a read only register memory mapped at address 0x9600 00FC to 0x9600 00FF. This register is used to read the peripheral's identification code.

Table 208. I²C Component Type Register

Mnemonic	Access	Bits	Description	Reset
IC_COMP_TYPE	R	31:0	Component Type Register: always returns the identification code 0x4457 0140.	0x4457 0140

9.11 Universal Asynchronous Receiver/Transmitter (UART)

The MCU features two National Semiconductor 16550 compatible UARTs that can be used simultaneously. They support hardware handshaking and baud rates up to 2 Mbaud. Other features and benefits include:

- 16 character transmit and receive FIFOs
- Programmable receive FIFO interrupt threshold.
- Programmable FIFO enable/disable.
- Additional FIFO status registers.
- Shadow registers to reduce software overhead.
- Software programmable reset.
- Loopback mode that enables software testing of modem control features.

- Busy functionality helps to safe guard against errors if the LCR, DLL, and/or DLH registers are changed during a transaction even though they should only be set during initialization.
- Independently controlled modem and status lines.
- False start bit detection.
- Component version register.

9.11.1 Functional Description

UART serial data is format is shown in Figure 43. UARTs are typically used to send ASCII characters. An idle signal is indicated by a constant high level. A transition from the high idle state to a low level indicates a start bit. This start bit lasts for exactly one bit time. It is followed by five to eight data bits. Data is followed by an optional parity bit, which could be even or odd. The character ends with an idle signal referred to as stop bit(s) that last for 1, 1.5, or 2 bit times. The idle signal continues until the next start bit, which could follow immediately after the stop bit(s). The UARTs feature registers to select the number of data bits, optional parity, and number of stop bits.

Figure 43. UART Serial Data Format

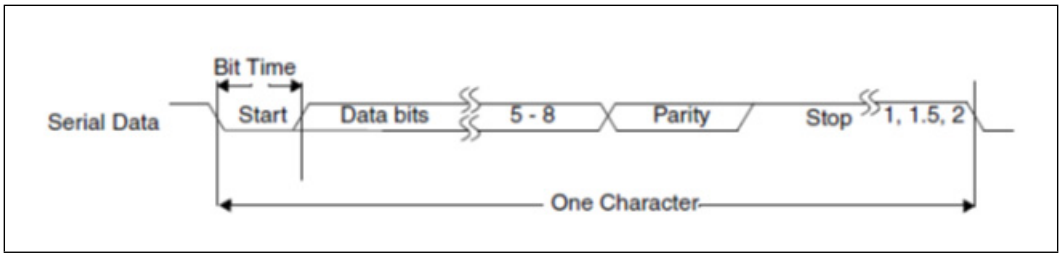
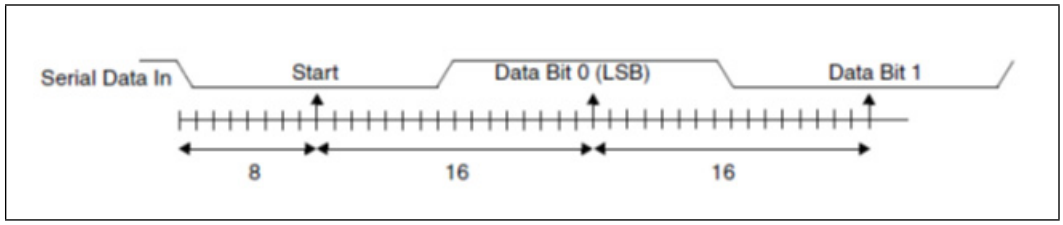


Figure 44 is a timing diagram showing the UART sampling serial data. The input is oversampled sixteen times per bit period. When the serial data input (RXD) transitions from the high idle state to the low start bit state, and eight subsequent low samples follow, the UART begins de-serializing input data. If the low start bit state lasts for less than eight samples, it is ignored. This provides some immunity to glitches. De-serialization consists of sampling the incoming serial data after every 16 sample clocks. Once all of the data bits have been sampled, the optional parity bit is validated, and a parity error flag set if parity is found to be invalid. After the optional parity bit, the stop bit is validated, and a framing error flag is set if the stop bit is found to be in the low state. The UARTs feature memory mapped registers to program the sample rate and check for errors. The sample rate must be programmed to sixteen times the baud (i.e. bit) rate prior to enabling the UART.

Figure 44. Timing Diagram showing UART Serial Data Sampling



The UARTs feature sixteen character receive and transmit FIFOs. The receive FIFO buffers de-serialized input data until the CPU can unload it from the FIFO. The transit FIFO buffers output data written by the CPU that is yet to be serialized and transmitted. These FIFOs reduce the frequency of interrupt service by allowing the CPU to read or write data in bursts. The UARTs feature memory mapped registers to select the receive FIFO fullness threshold at which interrupts will be generated.

The UART's input clock is the APB clock. The APB clock is the same frequency as the CPU and AHB clocks. The APB clock is divided down to produce the UART's sample clock. Since the sample clock frequency determines the baud rate, and the APB clock frequency determines the sample clock frequency, it is important that the CPU not adjust its frequency while the UARTs are in use. The UARTs feature memory mapped status registers to help the CPU determine when they are idle.

The UARTs feature the five prioritized interrupts shown in UART Interrupt Sources by Priority with Cause and Reset Events

Table 209. UART Interrupt Sources by Priority with Cause and Reset Events

Name	Cause	Priority	Reset Event
Receiver line status change	Overrun error, parity error, framing error, or break	1	Reading the Line Status Register
Received data is available	Character received with FIFOs disabled or receive FIFO threshold reached with FIFOs enabled	2	Reading the Receiver Buffer Register with FIFOs disabled or dropping below the receive FIFO threshold with FIFOs enabled
Character timeout	Partially full receive FIFO, but no new characters received in four character times	2	Reading the Receive Buffer Register
Transmit holding register emptied	Transmit holding register emptied	3	Reading the Interrupt Identification Register or writing to the Transmit Holding Register.
Modem status change	Clear to send, data set ready, ring indicator, or data carrier detect.	4	Reading the Modem Status Register
Busy detected	Writing to the Line Control Register while the UART is busy	5	Reading the UART Status Register

In addition to 16550 compatible registers, the UARTs feature memory mapped shadow registers that can be used to decrease interrupt service latency. These shadow registers separate the various control, status, and data bit fields into independently addressable registers. This eliminates the need for read-modify-write operations.

Finally, the UARTs feature memory mapped registers for reading the ID code, version code, and configuration of the UARTs.

9.11.2 Registers

Unlike the 16550, the MCU's UART registers are aligned on 4-byte boundaries and can be accessed only as double words. Bit definitions are given in the following sections.

9.11.2.1 UART Receive Buffer Register (RBR)

The Receive Buffer Register is a read only register memory mapped at address 0x9900 0000 to 0x9900 0003 for UART[0] and 0x9A00 0000 to 0x9A00 0003 for UART[1]. This register is used to read received character data. It shares a common address with the Divisor Latch Low (DLL) register and is selected when the DLAB bit in the Line Control Register (LCR) is zero.

Table 210. UART Receive Buffer Register

Mnemonic	Access	Bits	Description	Reset
RBR	R	7:0	Receive Buffer Register: Character received on the serial input (RXD). The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an over-run error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO is preserved, but any incoming data are lost and an over-run error occurs.	0x0
-	-	31:8	Reserved	X

9.11.2.2 UART Transmit Holding Register (THR)

The Transmit Holding Register is a write only register memory mapped at address 0x9900 0000 to 0x9900 0003 for UART[0] and 0x9A00 0000 to 0x9A00 0003 for UART[1]. This register is used to write transmit character data. It shares a common address with the Divisor Latch Low (DLL) register and is selected when the DLAB bit in the Line Control Register (LCR) is zero.

Table 211. UART Transmit Holding Register

Mnemonic	Access	Bits	Description	Reset
THR	W	7:0	Transmit Holding Register: Data to be transmitted on the serial output (TXD). Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFOs are disabled (FCR[0] = 0) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFOs are enabled (FCR[0] = 1) and THRE is set, 16 characters of data may be written to the THR before the FIFO is full. Any attempt to write data when the FIFO is full results in the write data being lost.	0x0
-	-	31:8	Reserved	X

9.11.2.3 UART Divisor Latch Low Register (DLL)

The Divisor Latch Low Register is a readable and writable register memory mapped at address 0x9900 0000 to 0x9900 0003 for UART[0] and 0x9A00 0000 to 0x9A00 0003 for UART[1]. This register is used to select the baud rate. It shares a common address with the Receive Buffer Register (RBR) and Transmit Holding Register (THR) and is selected when the DLAB bit in the Line Control Register (LCR) is one.

Table 212. UART Divisor Latch Low Register

Mnemonic	Access	Bits	Description	Reset
DLL	R/W	7:0	Divisor Latch Low Register: Lower 8 bits of the 16-bit baud rate divisor for the UART. This register may only be accessed when the DLAB bit (LCR[7]) is set and the UART is not busy (USR[0] is zero). The baud rate is equal to the APB clock frequency divided by sixteen times the value of the baud rate divisor: $$baud\ rate = \frac{APB\ clock\ freq}{16 \times baud\ rate\ divisor}$$ Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications occur. Also, once the DLL is set, at least 8 clock cycles of the APB clock should be allowed to pass before transmitting or receiving data.	0x0
-	-	31:8	Reserved	X

9.11.2.4 UART Divisor Latch High Register (DLH)

The Divisor Latch High Register is a readable and writable register memory mapped at address 0x9900 0004 to 0x9900 0007 for UART[0] and 0x9A00 0004 to 0x9A00 0007 for UART[1]. This register is used to select the baud rate. It shares a common address with the Interrupt Enable Register (IER) and is selected when the DLAB bit in the Line Control Register (LCR) is one.

Table 213. UART Divisor Latch High Register

Mnemonic	Access	Bits	Description	Reset
DLH	R/W	7:0	Divisor Latch High Register: Upper 8 bits of the 16-bit baud rate divisor for the UART. This register may only be accessed when the DLAB bit (LCR[7]) is set and the UART is not busy (USR[0] is zero). The baud rate is equal to the APB clock frequency divided by sixteen times the value of the baud rate divisor: $$baud\ rate = \frac{APB\ clock\ freq}{16 \times baud\ rate\ divisor}$$ Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications occur. Also, once the DLH is set, at least 8 clock cycles of the APB clock should be allowed to pass before transmitting or receiving data.	0x0
-	-	31:8	Reserved	X

9.11.2.5 UART Interrupt Enable Register (IER)

The Interrupt Enable Register is a readable and writable register memory mapped at address 0x9900 0004 to 0x9900 0007 for UART[0] and 0x9A00 0004 to 0x9A00 0007 for UART[1]. This register is used to enable interrupts. It shares a common address with the Divisor Latch High (DLH) register and is selected when the DLAB bit in the Line Control Register (LCR) is zero.

Table 214. UART Interrupt Enable Register

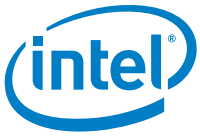
Mnemonic	Access	Bits	Description	Reset
ERBFI	R/W	0	Enable Received Data Available Interrupt: when '1', Received Data Available and Character Timeout interrupts (if in FIFO mode) are enabled. These are the second highest priority interrupts.	0x0
ETBEI	R/W	1	Enable Transmit Holding Register Empty Interrupt: when '1', Transmitter Holding Register Empty interrupts are enabled. This is the third highest priority interrupt.	0x0
ELSI	R/W	2	Enable Receiver Line Status Interrupt: when '1', Receiver Line Status interrupts are enabled. This is the highest priority interrupt.	0x0
EDSSI	R/W	3	Enable Modem Status Interrupt: when '1', Modem Status interrupts are enabled. This is the fourth highest priority interrupt.	0x0
-	-	31:4	Reserved	X

9.11.2.6 UART Interrupt Identity Register (IIR)

The Interrupt Identity Register is a read only register memory mapped at address 0x9900 0008 to 0x9900 000B for UART[0] and 0x9A00 0008 to 0x9A00 000B for UART[1]. This register is used to read interrupt status.

Table 215. UART Interrupt Identity Register

Mnemonic	Access	Bits	Description	Reset
IID	R	3:0	Interrupt ID: This indicates the highest priority pending interrupt enumerated thusly: 0 = Modem status change 1 = No interrupt pending 2 = Transmit holding register emptied 3 = Reserved 4 = Receive data is available 5 = Reserved 6 = Receiver line status change 7 = Busy detected 8 = Reserved 9 = Reserved 10 = Reserved 11 = Reserved 12 = Character timeout	0x1
-	-	5:4	Reserved	X



FIFOSE	R	7:6	FIFOs Enabled: This indicate whether the FIFOs are enabled or disabled enumerated thusly: 0 = Disabled 1 = Reserved 2 = Reserved 3 = Enable	0x0
-	-	31:8	Reserved	X

9.11.2.7 UART FIFO Control Register (FCR)

The FIFO Control Register is a write only register memory mapped at address 0x9900 0008 to 0x9900 000B for UART[0] and 0x9A00 0008 to 0x9A00 000B for UART[1]. This register is used to control the FIFOs.

Table 216. UART FIFO Control Register

Mnemonic	Access	Bits	Description	Reset
FIFOE	W	0	FIFO Enable: when '1', transmit and receive FIFOs are enabled. Whenever the value of this bit is changes, both FIFOs are reset.	0x0
RFIFOR	W	1	Receive FIFO Reset: when '1', resets the receive FIFO to empty. Note that this bit is 'self-clearing'.	0x0
XFIFOR	W	2	Transmit FIFO Reset: when '1', resets the transmit FIFO to empty. Note that this bit is 'self-clearing'.	0x0
-	-	5:3	Reserved	X
RT	W	7:6	Receive FIFO Threshold: selects the threshold level at which the Received Data Available Interrupt is generated encoded thusly: 0 = 1 characters in the FIFO 1 = 4 characters in the FIFO 2 = 8 characters in the FIFO 3 = 14 characters in the FIFO	0x0
-	-	31:8	Reserved	X

9.11.2.8 UART Line Control Register (LCR)

The Line Control Register is a readable and writable register memory mapped at address 0x9900 000C to 0x9900 000F for UART[0] and 0x9A00 000C to 0x9A00 000F for UART[1]. This register is used to control the UART. It is writable only when UART is not busy (USR[0] is zero). It is always readable.

Table 217. UART Line Control Register

Mnemonic	Access	Bits	Description	Reset
DLS	R/W	1:0	Data Length Select: used to select the number of data bits per character that the UART transmits and receives encoded thusly: 0 = 5 bits 1 = 6 bits 2 = 7 bits 3 = 8 bits	0x0

STOP	R/W	2	Number of stop bits: used to select the number of stop bits per character that the peripheral transmits and receives. If set to '0', one stop bit is transmitted in the serial data. If set to '1' and the data bits are set to 5, one and a half stop bits are transmitted. Otherwise, two stop bits are transmitted. Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit.	0x0
PEN	R/W	3	Parity Enable: when '1', parity is enabled. When '0', parity is disabled.	0x0
EPS	R/W	4	Even Parity Select: when '1', even parity is selected. When '0', odd parity is selected. Parity must be enabled for this bit to have any effect.	0x0
-	-	5	Reserved	X
BC	R/W	6	Break Control: when '1', a break (low state) is continuously transmitted until this bit is set to '1'.	0x0
DLAB	R/W	7	Divisor Latch Access Bit: when '1', access to the Divisor Latch Low and High registers is enabled. When '0', access to the Receive Buffer Register and Transmit Holding Register is enabled.	0x0
-	-	31:8	Reserved	X

9.11.2.9 UART Modem Control Register (MCR)

The Modem Control Register is a readable and writable register memory mapped at address 0x9900 0010 to 0x9900 0013 for UART[0] and 0x9A00 0010 to 0x9A00 0013 for UART[1]. This register is used to control the modem.

Table 218. UART Modem Control Register

Mnemonic	Access	Bits	Description	Reset
DTR	R/W	0	Data Terminal Ready: when '1', the DTR pin is driven low. When '0', the DTR pin is driven high. Note that in loopback mode, the DTR pin is driven high and this bit is looped back to the DSR bit in the Modem Status Register. Also, the DTR function must be selected in the Pin Function Register in order for this bit to have any effect on the pin.	0x0
RTS	R/W	1	Request to Send: when '1', the RTS pin is driven low. When '0', the RTS pin is driven high. Note that in loopback mode, the RTS pin is driven high and this bit is looped back to the CTS bit in the Modem Status Register. Also, the RTS function must be selected in the Pin Function Register in order for this bit to have any effect on the pin.	0x0
-	-	3:2	Reserved	X
LB	R/W	4	Loop Back: when '1', transmit data is looped back to receive data while TXD, DTR and RTS pins are driven high. When '0', the UART operates normally.	0x0
-	-	31:5	Reserved	X

9.11.2.10 UART Line Status Register (LSR)

The Line Status Register is a read only register memory mapped at address 0x9900 0014 to 0x9900 0017 for UART[0] and 0x9A00 0014 to 0x9A00 0017 for UART[1]. This register is used to read the UART status.

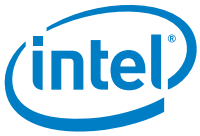


Table 219. UART Line Status Register

Mnemonic	Access	Bits	Description	Reset
DR	R	0	Data Ready: when '1', at least one character is available from the Receive Buffer Register. When '0', no characters are available. This bit clears when the RBR is read and no more characters are available.	0x0
OE	R	1	Overrun Error: when '1', the Receive Buffer Register has overrun meaning that a new character was received, but there was no room to store it. When FIFOs are enable, it means that the receive FIFO was full. When FIFOs are disabled, it means that the Receive Buffer Register was full. This bit clears after reading.	0x0
PE	R	2	Parity Error: when '1', a parity error has been detected in a received character. If FIFOs are enabled, the character with the parity error has arrived at the top of the receive FIFO. Parity must be enabled for this bit to be set. Note that a break can cause a parity error. This bit is cleared after reading.	0x0
FE	R	3	Framing Error: when '1', a framing error has occurred meaning that an invalid (i.e. low) stop bit has been detected. If FIFOs are enabled, the character with the framing error has arrived at the top of the receive FIFO. Note that a break can cause a framing error. This bit is cleared after reading.	0x0
BI	R	4	Break Interrupt: when '1', a break has been detected meaning that the RXD input has been low for the duration of a character including the start and stop bits. If FIFOs are enabled, the character with the framing error has arrived at the top of the receive FIFO. This bit is cleared after reading.	0x0
THRE	R	5	Transmit Holding Register Empty: when '1', the Transmit Holding Register is empty. If FIFOs are enabled, the transmit FIFO is empty. This bit is cleared by writing to the Transmit Holding Register.	0x1
TEMT	R	6	Transmitter Empty: when '1', both the transmitter shift register and the Transmit Holding Register are empty. This bit is cleared by writing to the Transmit Holding Register.	0x1
RFE	R	7	Receive FIFO Error: when '1', a parity error, framing error, or break is present in the receive FIFO. This bit can only be set when FIFOs are enabled. This bit is cleared after reading only if the FIFO is free of errors.	0x1
-	-	31:8	Reserved	X

9.11.2.11 UART Modem Status Register (MSR)

The Modem Status Register is a read only register memory mapped at address 0x9900 0018 to 0x9900 001B for UART[0] and 0x9A00 0018 to 0x9A00 001B for UART[1]. This register is used to read the modem status.

Table 220. UART Modem Status Register

Mnemonic	Access	Bits	Description	Reset
-	-	3:0	Reserved	X
CTS	R	4	Clear to Send: when '1', the CTS pin is low. When '0', the CTS pin is high. When in loopback mode, this bit reflects the state of the Modem Control Register RTS bit. Note that the UART must not be in loop back mode and CTS must be selected in the I/O Pin Function Register in order for the CTS pin to affect this bit.	0x0

DSR	R	5	Data Set Ready: when '1', the DSR pin is low. When '0', the DSR pin is high. When in loopback mode, this bit reflects the state of the Modem Control Register DTR bit. Note that the UART must not be in loop back mode and DSR must be selected in the I/O Pin Function Register in order for the DSR pin to affect this bit.	0x0
-	-	31:6	Reserved	X

9.11.2.12 UART Scratch Pad Register (SCR)

The Scratch Pad Register is a readable and writable register memory mapped at addresses 0x9900 001C to 0x9900 001F for UART[0] and 0x9A00 001C to 0x9A00 001F for UART[1]. This register is used to test the UART's APB interface.

Table 221. UART Scratch Pad Register

Mnemonic	Access	Bits	Description	Reset
SCR	R/W	7:0	Scratch Pad: reading returns the last value written.	0x0
-	-	31:8	Reserved	X

9.11.2.13 UART Shadow Receive Buffer Register (SRBR)

The Shadow Receive Buffer Register is a read only register memory mapped at address 0x9900 0030 to 0x9900 006F for UART[0] and 0x9A00 0030 to 0x9A00 006F for UART[1]. This register is used to read received character data. This 32-bit register is aliased throughout the allocated 64 byte memory range.

Table 222. UART Shadow Receive Buffer Register

Mnemonic	Access	Bits	Description	Reset
SRBR	R	7:0	Shadow Receive Buffer Register: Character received on the serial input (RXD). The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (LCR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the SRBR must be read before the next data arrives, otherwise it is overwritten, resulting in an over-run error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO is preserved, but any incoming data are lost and an over-run error occurs.	0x0
-	-	31:8	Reserved	X

9.11.2.14 UART Shadow Transmit Holding Register (STHR)

The Shadow Transmit Holding Register is a write only register memory mapped at addresses 0x9900 0030 to 0x9900 006F for UART[0] and 0x9A00 0030 to 0x9A00 006F for UART[1]. This register is used to write transmit character data. This 32-bit register is aliased throughout the allocated 64 byte memory range.

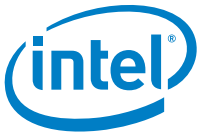


Table 223. UART Shadow Transmit Holding Register

Mnemonic	Access	Bits	Description	Reset
STHR	W	7:0	Shadow Transmit Holding Register: Data to be transmitted on the serial output (TXD). Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFOs are disabled (FCR[0] = 0) and THRE is set, writing a single character to the STHR clears the THRE. Any additional writes to the STHR before the THRE is set again causes the STHR data to be overwritten. If FIFOs are enabled (FCR[0] = 1) and THRE is set, 16 characters of data may be written to the STHR before the FIFO is full. Any attempt to write data when the FIFO is full results in the write data being lost.	0x0
-	-	31:8	Reserved	X

9.11.2.15 UART Status Register (USR)

The UART Status Register is a read only register memory mapped at address 0x9900 007C to 0x9900 007F for UART[0] and 0x9A00 007C to 0x9A00 007F for UART[1]. This register is used to read the UART status.

Table 224. UART Status Register

Mnemonic	Access	Bits	Description	Reset
BUSY	R	0	UART Busy: when '1', a serial transfer is in progress. When '0', the UART is idle. The UART is considered busy under any of the following conditions: 1. Transmission is in progress on the serial interface 2. Transmit data is present in the Transmit Holding Register, the baud divisor is non-zero, and the DLAB bit in the Line Control Register is zero. 3. Reception is in progress on the serial interface 4. Receive data is present in the Receive Buffer Register. Note that it is possible for the BUSY bit to be cleared even though a new character may have been sent from another device. That is, if the UART has no data in Transmit Holding Register and Receive Buffer Register and there is no transmission in progress and a start bit of a new character has just reached the UART. This is due to the fact that a start bit is not validated until the middle of the bit period.	0x0
TFNF	R	1	Transmit FIFO not Full: when '1', the transmit FIFO is not full. When '0', the transmit FIFO is full.	0x1
TFE	R	2	Transmit FIFO Empty: when '1', the transmit FIFO is empty. When '0', the transmit FIFO is not empty.	0x1
RFNE	R	3	Receive FIFO not Empty: when '1', the receive FIFO is not empty. When '0', the receive FIFO is empty.	0x0
RFF	R	4	Receive FIFO Full: when '1', the receive FIFO is full. When '0', the receive FIFO is not full.	0x0
-	-	31:5	Reserved	X

9.11.2.16 UART Transmit FIFO Level (TFL)

The Transmit FIFO Level is a read only register memory mapped at address 0x9900 0080 to 0x9900 0083 for UART[0] and 0x9A00 0080 to 0x9A00 0083 for UART[1]. This register is used to read the number of characters in the transmit FIFO.

Table 225. UART Transmit FIFO Level

Mnemonic	Access	Bits	Description	Reset
TFL	R	4:0	Transmit FIFO Level: returns the number of characters in the transmit FIFO.	0x0
-	-	31:5	Reserved	X

9.11.2.17 UART Receive FIFO Level (RFL)

The Receive FIFO Level is a read only register memory mapped at address 0x9900 0084 to 0x9900 0087 for UART[0] and 0x9A00 0084 to 0x9A00 0087 for UART[1]. This register is used to read the number of characters in the transmit FIFO.

Table 226. UART Receive FIFO Level

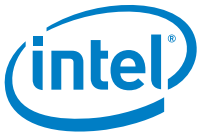
Mnemonic	Access	Bits	Description	Reset
RFL	R	4:0	Receive FIFO Level: returns the number of characters in the receive FIFO.	0x0
-	-	31:5	Reserved	X

9.11.2.18 UART Software Reset Register (SRR)

The Software Reset Register is a write only register memory mapped at address 0x9900 0088 to 0x9900 008B for UART[0] and 0x9A00 0088 to 0x9A00 008B for UART[1]. This register is used to reset the transmit FIFO, receive FIFO, or entire UART.

Table 227. UART Software Reset Register

Mnemonic	Access	Bits	Description	Reset
UR	W	0	UART Reset: when written '1', the entire UART is reset. Writing '0' has no effect. This bit is self-clearing.	0x0
RFR	W	1	Receive FIFO Reset: when written '1', the receive FIFO is reset to the empty state. Writing '0' has no effect. This bit is self-clearing.	0x0
XFR	W	2	Transmit FIFO Reset: when written '1', the transmit FIFO is reset to the empty state. Writing '0' has no effect. This bit is self-clearing.	0x0
-	-	31:3	Reserved	X



9.11.2.19 UART Shadow Request to Send (SRTS)

The Shadow Request to Send is a readable and writable register memory mapped at addresses 0x9900 008C to 0x9900 008F for UART[0] and 0x9A00 008C to 0x9A00 008F for UART[1]. This register is used to control the RTS pin.

Table 228. UART Shadow Request to Send

Mnemonic	Access	Bits	Description	Reset
SRTS	R/W	0	Shadow Request to Send: when '1', the RTS pin is driven low. When '0', the RTS pin is driven high. Note that in loopback mode, the RTS pin is driven high and this bit is looped back to the CTS bit in the Modem Status Register. Also, the RTS function must be selected in the Pin Function Register in order for this bit to have any effect on the pin.	0x0
-	-	31:1	Reserved	X

9.11.2.20 UART Shadow Break Control Register (SBCR)

The Shadow Break Control Register is a readable and writable register memory mapped at addresses 0x9900 0090 to 0x9900 0093 for UART[0] and 0x9A00 0090 to 0x9A00 0093 for UART[1]. This register is used to send a break.

Table 229. UART Shadow Break Control Register

Mnemonic	Access	Bits	Description	Reset
SBC	R/W	0	Shadow Break Control: when '1', a break (low state) is continuously transmitted until this bit is set to '1'.	0x0
-	-	31:1	Reserved	X

9.11.2.21 UART Shadow FIFO Enable (SFE)

The Shadow FIFO Enable is a readable and writable register memory mapped at addresses 0x9900 0098 to 0x9900 009B for UART[0] and 0x9A00 0098 to 0x9A00 009B for UART[1]. This register is used to enable transmit and receive FIFOs.

Table 230. UART Shadow Break Control Register

Mnemonic	Access	Bits	Description	Reset
SFE	W	0	Shadow FIFO Enable: when '1', transmit and receive FIFOs are enabled. Whenever the value of this bit is changes, both FIFOs are reset.	0x0
-	-	31:1	Reserved	X

9.11.2.22 UART Shadow Receive Threshold (SRT)

The Shadow Receive Threshold is a readable and writable register memory mapped at addresses 0x9900 009C to 0x9900 009F for UART[0] and 0x9A00 009C to 0x9A00 009F for UART[1]. This register is used to select the threshold level at which the Received Data Available Interrupt is generated.

Table 231. UART Shadow Receive Threshold

Mnemonic	Access	Bits	Description	Reset
SRT	W	1:0	Shadow Receive Threshold: selects the threshold level at which the Received Data Available Interrupt is generated encoded thusly: 0 = 1 character in the FIFO 1 = 4 characters in the FIFO 2 = 8 characters in the FIFO 3 = 14 characters in the FIFO	0x0
-	-	31:2	Reserved	X

9.11.2.23 UART Halt Transmit (HTX)

The Halt Transmit is a readable and writable register memory mapped at addresses 0x9900 00A4 to 0x9900 00A7 for UART[0] and 0x9A00 00A4 to 0x9A00 00A7 for UART[1]. This register is used to halt transmission so that the transmit FIFO can be filled and tested.

Table 232. UART Shadow Break Control Register

Mnemonic	Access	Bits	Description	Reset
HTX	R/W	0	Halt Transmit: when '1', transmission is halted. When '0', the transmitter operates normally. This bit has no effect when FIFOs are disabled.	0x0
-	-	31:1	Reserved	X

9.11.2.24 UART Component Parameter Register (CPR)

The Component Parameter Register is a read only register memory mapped at address 0x9900 00F4 to 0x9900 00F7 for UART[0] and 0x9A00 00F4 to 0x9A00 00F7 for UART[1]. This register is used to read the peripheral's hardware configuration.

Table 233. UART Component Parameter Register

Mnemonics	Access	Bits	Description	Reset
APB_DATA_WIDTH	R	1:0	APB Data Width: always returns 2.	0x2
-	-	3:2	Reserved.	X
AFCE_MODE	R	4	Auto Flow Control: always returns 0.	0x0
THRE_MODE	R	5	Programmable THRE Interrupt Mode: always returns 0.	0x0
SIR_MODE	R	6	IrDA SIR Mode Support: always returns 0.	0x0
SIR_LP_MODE	R	7	Low Power IrDA SIR Mode Support: always returns 0.	0x0
ADDITIONAL_FEATURES	R	8	Add Version and ID Registers, Enable FIFO Status, Shadow and Encoded Parameters Register Options: always returns 1.	0x1
FIFO_ACCESS	R	9	Include FIFO Access Mode: always returns 0.	0x0
FIFO_STAT	R	10	Include Software Accessible FIFO Status Registers: always returns 1.	0x1

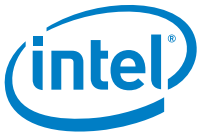


Table 233. UART Component Parameter Register

Mnemonics	Access	Bits	Description	Reset
SHADOW	R	11	Include Additional Shadow Registers for Reducing Software Overhead: always returns 1.	0x1
UART_ADD_ENCODED_PARAMS	R	12	Include Configuration Identification Register: always returns 1.	0x1
DMA_EXTRA	R	13	Include Additional DMA Signals on I/F: always returns 0.	0x0
-	-	15:14	Reserved.	X
FIFO_MODE	R	23:16	UART FIFO depth: always returns 1	0x1
-	-	31:24	Reserved.	X

9.11.2.25 UART Component Version (UCV)

The UART Component Version is a read only register memory mapped at address 0x9900 00F8 to 0x9900 00FB for UART[0] and 0x9A00 00F8 to 0x9A00 00FB for UART[1]. This register is used to read the peripheral's version code.

Table 234. UART Component Version

Mnemonic	Access	Bits	Description	Reset
UCV	R	31:0	Version Code Register: always returns the version code 3.12 as the big endian ASCII string "312*" or 0x3331 322A.	0x3331 322A

9.11.2.26 UART Component Type Register (CTR)

The Component Type Register is a read only register memory mapped at address 0x9900 00FC to 0x9900 00FF for UART[0] and 0x9A00 00FC to 0x9A00 00FF for UART[1]. This register is used to read the peripheral's identification code.

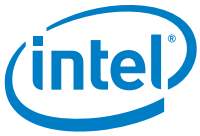
Table 235. UART Component Type Register

Mnemonic	Access	Bits	Description	Reset
CTR	R	31:0	Component Type Register: always returns the identification code 0x4457 0110.	0x4457 0110

§ §

10.0 References

- Abracon Corp. (2011, July 29). *32.768 kHz SMD Low Profile Crystal ABS06.* Retrieved March 14, 2012, from www.abracon.com
- Abracon Corp. (2011, April 20). *Ultra Miniature Ceramic Glass Sealed SMD Crystal ABM8G.* Retrieved March 14, 2012, from www.abracon.com
- NXP Semiconductors. (2007). I2C-Bus Specification and User Manual.



Appendix A Packaging Information

10.1 Package Drawing

The following figures show the package drawings and controlling dimension for the MCU. The drawings are not in scale.

Figure 45. Top View

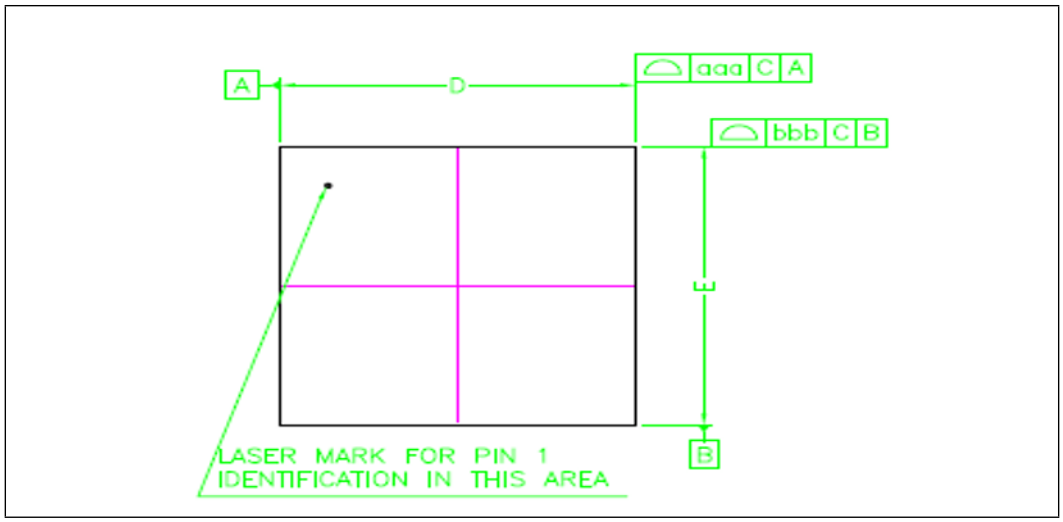


Figure 46. Side View

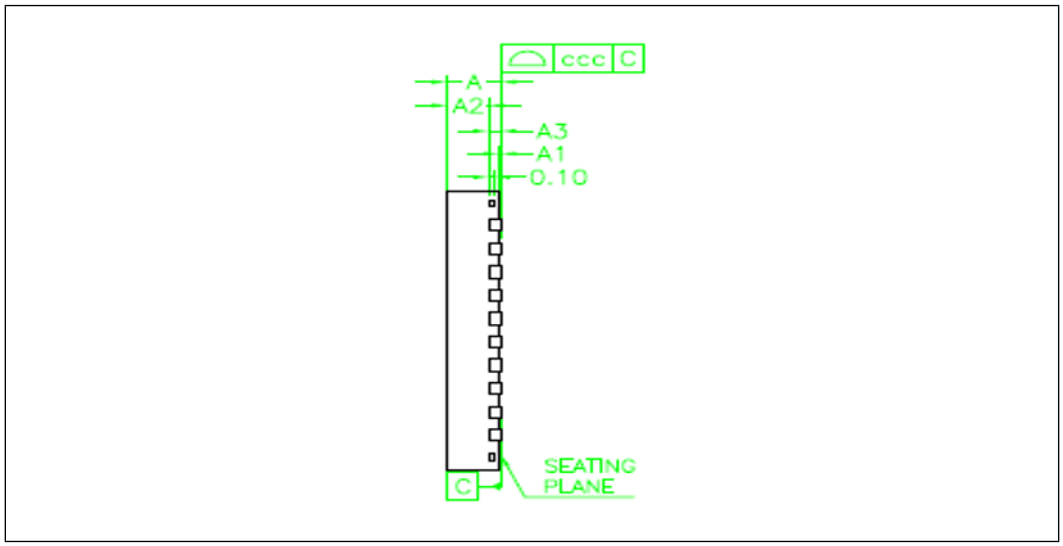


Figure 47. Bottom View

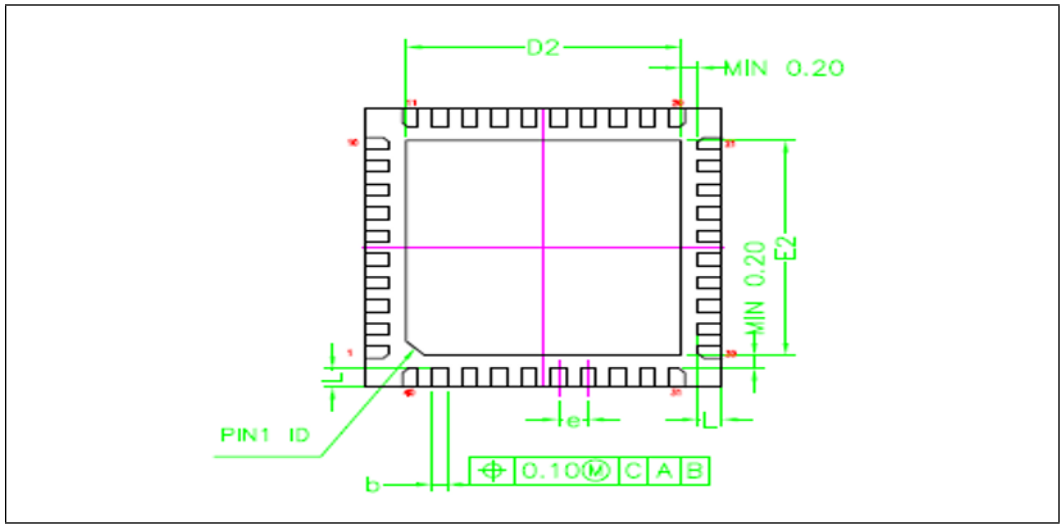


Table 236. Controlling Dimension

Symbol	Millimeter			Inch		
	Min.	Nom.	Max.	Min.	Nom.	Max.
A	0.80	0.85	0.90	0.031	0.033	0.035
A1	0.00	0.035	0.05	0.000	0.001	0.002
A2	---	0.65	0.67	---	0.026	0.026
A3	0.203 REF.			0.008 REF.		
b	0.20	0.25	0.30	0.008	0.010	0.012
D	5.90	6.00	6.05	0.232	0.236	0.238
D2	4.52	4.62	4.72	0.178	0.182	0.186
E	5.90	6.00	6.05	0,232	0.236	0.238
E2	4.52	4.62	4.72	0.178	0.182	0.186
L	0.35	0.40	0.45	0.014	0.016	0.018
e	0.50 bsc			0.020 bsc		
Tolerance of Form and Position						
aaa	0.10			0.004		
bbb	0.10			0.004		
ccc	0.05			0.002		

Notes: The following information applies to Table 236:
1. All dimensions are in millimeters.
2. Die thickness allowable is 0.305 mm maximum (0.012 inches maximum).
3. Dimensioning & tolerances conform to ASME Y14.5M. -1994.
4. The pin #1 identifier must be placed on the top surface of the package by using indentation mark or other feature of package body.
5. Exact shape and size of this feature is optional.
6. Package warpage max 0.08 mm.
7. Applied for exposed pad and terminals. Exclude embedding part of exposed pad from measuring.
8. Applied only to terminals.

A.1 Reference Design

This section shows user the external interfaces reference design for the MCU interfaces. The reference design illustrate recommended design for the Power Supply, Clocking, Reset and TAP_SEL interfaces.

Reference design provided to user for references purposes where user may implement their own design for the interfaces.

Refer to Chapter 3.0 for more details on the external interfaces configuration guidelines as shown in Figure 48.

Figure 48. External Interfaces Reference Design

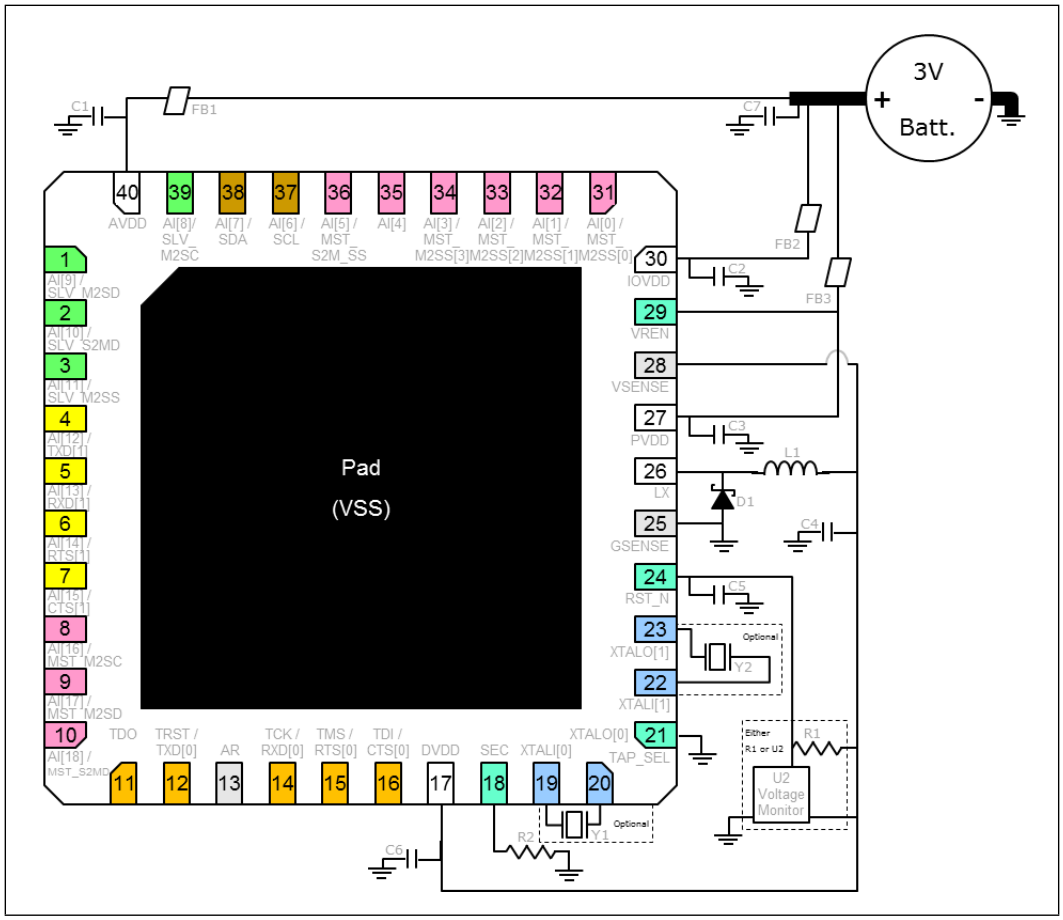


Table 237. Reference Design BOM List

Component	Schematics Reference
Schottky Diode 30V 0.1A	D1
Resistor 300k	R1
TBD	R2

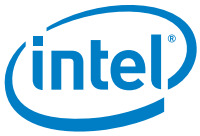


Table 237. Reference Design BOM List

Capacitor 0.1uF	C1, C2, C3, C5
Capacitor 4.7uF	C4, C7
Capacitor 10nF	C6
Inductor 47uH	L1
Ferrite Bead	FB1, FB2, FB3
Crystal Oscillator 33MHz	Y1
Crystal Oscillator 32.768kHz	Y2
Voltage Monitor 1.67V	U2

§ §