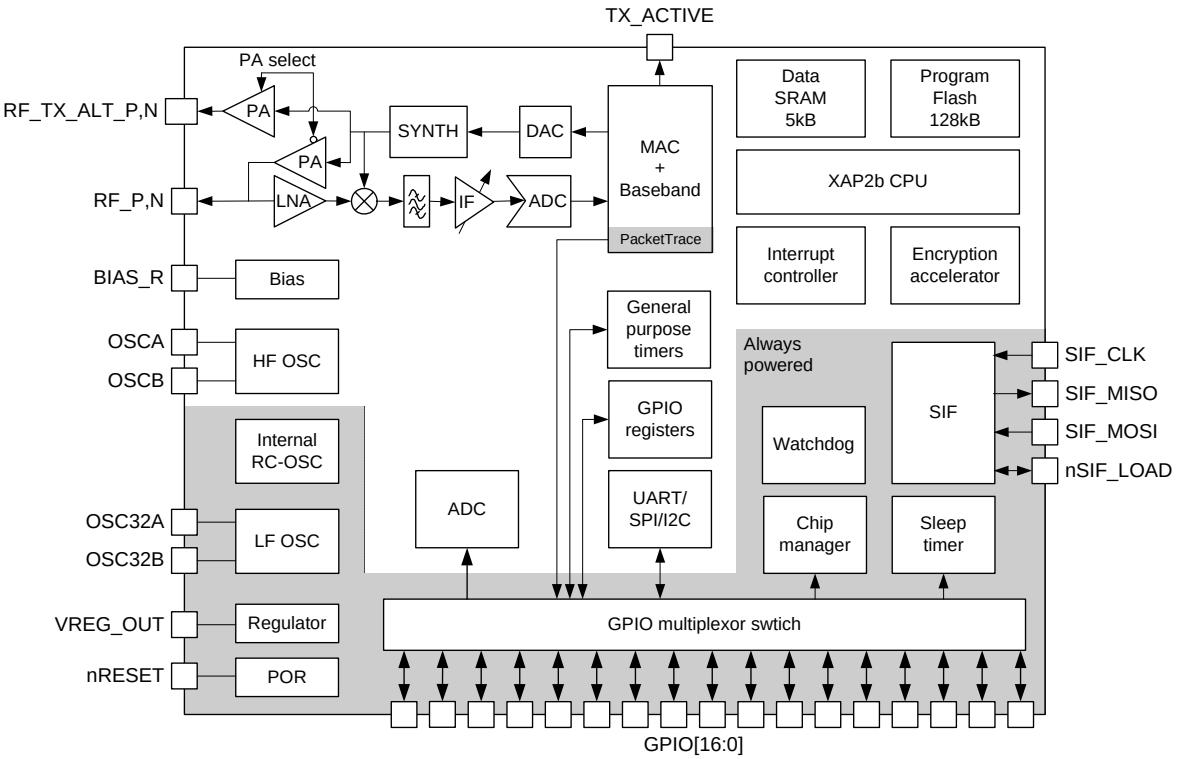
EM250

Single-Chip ZigBee/802.15.4 Solution

- *Integrated 2.4GHz, IEEE 802.15.4-compliant transceiver:*
  - Robust RX filtering allows co-existence with IEEE 802.11g and Bluetooth devices
  - - 99dBm RX sensitivity (1% PER, 20byte packet)
  - + 3dBm nominal output power
  - Increased radio performance mode (boost mode) gives - 100dBm sensitivity and + 5dBm transmit power
  - Integrated VCO and loop filter
- *Integrated IEEE 802.15.4 PHY and lower MAC with DMA*
- *Integrated hardware support for Packet Trace Interface for the Ember development environment*
- *Provides integrated RC oscillator for low power operation*
- *Supports optional 32.768kHz crystal oscillator for higher accuracy needs*
- *16-bit XAP2b microprocessor*
- *Integrated memory:*
  - 128kB of Flash
  - 5kB of SRAM
- *Configurable memory protection scheme*
- *Two sleep modes:*
  - Processor idle
  - Deep sleep—1.0µA (1.5µA with optional 32.768kHz oscillator enabled)
- *Seventeen GPIO pins with alternate functions*
- *Two Serial Controllers with DMA*
  - SC1: I^2C master, SPI master + UART
  - SC2: I^2C master, SPI master/slave
- *Two 16-bit general-purpose timers; one 16-bit sleep timer*
- *Watchdog timer and power-on-reset circuitry*
- *Non-intrusive debug interface (SIF)*
- *Integrated AES encryption accelerator*
- *Integrated ADC module first-order, sigma-delta converter with 12-bit resolution*
- *Integrated 1.8V voltage regulator*



Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032
www.silabs.com

May 29, 2013
120-0082-000V Rev 1.1

General Description

The EM250 is a single-chip solution that integrates a 2.4GHz, IEEE 802.15.4-compliant transceiver with a 16-bit XAP2b microprocessor. It contains integrated Flash and RAM memory and peripherals of use to designers of ZigBee-based applications.

The transceiver utilizes an efficient architecture that exceeds the dynamic range requirements imposed by the IEEE 802.15.4-2003 standard by over 15dB. The integrated receive channel filtering allows for co-existence with other communication standards in the 2.4GHz spectrum such as IEEE 802.11g and Bluetooth. The integrated regulator, VCO, loop filter, and power amplifier keep the external component count low. An optional high performance radio mode (boost mode) is software selectable to boost dynamic range by a further 3dB.

The XAP2b microprocessor is a power-optimized core integrated in the EM250. It supports two different modes of operation—System Mode and Application Mode. The EmberZNet PRO stack runs in System Mode with full access to all areas of the chip. Application code runs in Application Mode with limited access to the EM250 resources; this allows for the scheduling of events by the application developer while preventing modification of restricted areas of memory and registers. This architecture results in increased stability and reliability of deployed solutions.

The EM250 has 128kB of embedded Flash memory and 5kB of integrated RAM for data and program storage. The EM250 software stack employs an effective wear-leveling algorithm in order to optimize the lifetime of the embedded Flash.

To maintain the strict timing requirements imposed by ZigBee and the IEEE 802.15.4-2003 standard, the EM250 integrates a number of MAC functions into the hardware. The MAC hardware handles automatic ACK transmission and reception, automatic backoff delay, and clear channel assessment for transmission, as well as automatic filtering of received packets. In addition, the EM250 allows for true MAC level debugging by integrating the Packet Trace Interface.

To support user-defined applications, a number of peripherals such as GPIO, UART, SPI, I^2C, ADC, and general-purpose timers are integrated. Also, an integrated voltage regulator, power-on-reset circuitry, sleep timer, and low-power sleep modes are available. The deep sleep mode draws less than 1µA, allowing products to achieve long battery life.

Finally, the EM250 utilizes the non-intrusive SIF module for powerful software debugging and programming of the XAP2b microcontroller.

Target applications for the EM250 include:

- *Building automation and control*
- *Home automation and control*
- *Home entertainment control*
- *Asset tracking*

The EM250 is purchased with EmberZNet PRO, the Silicon Labs ZigBee-compliant software stack, providing a ZigBee profile-ready, platform-compliant solution. This technical datasheet details the EM250 features available to customers using it with the EmberZNet PRO stack.

Contents

1 Pin Assignment

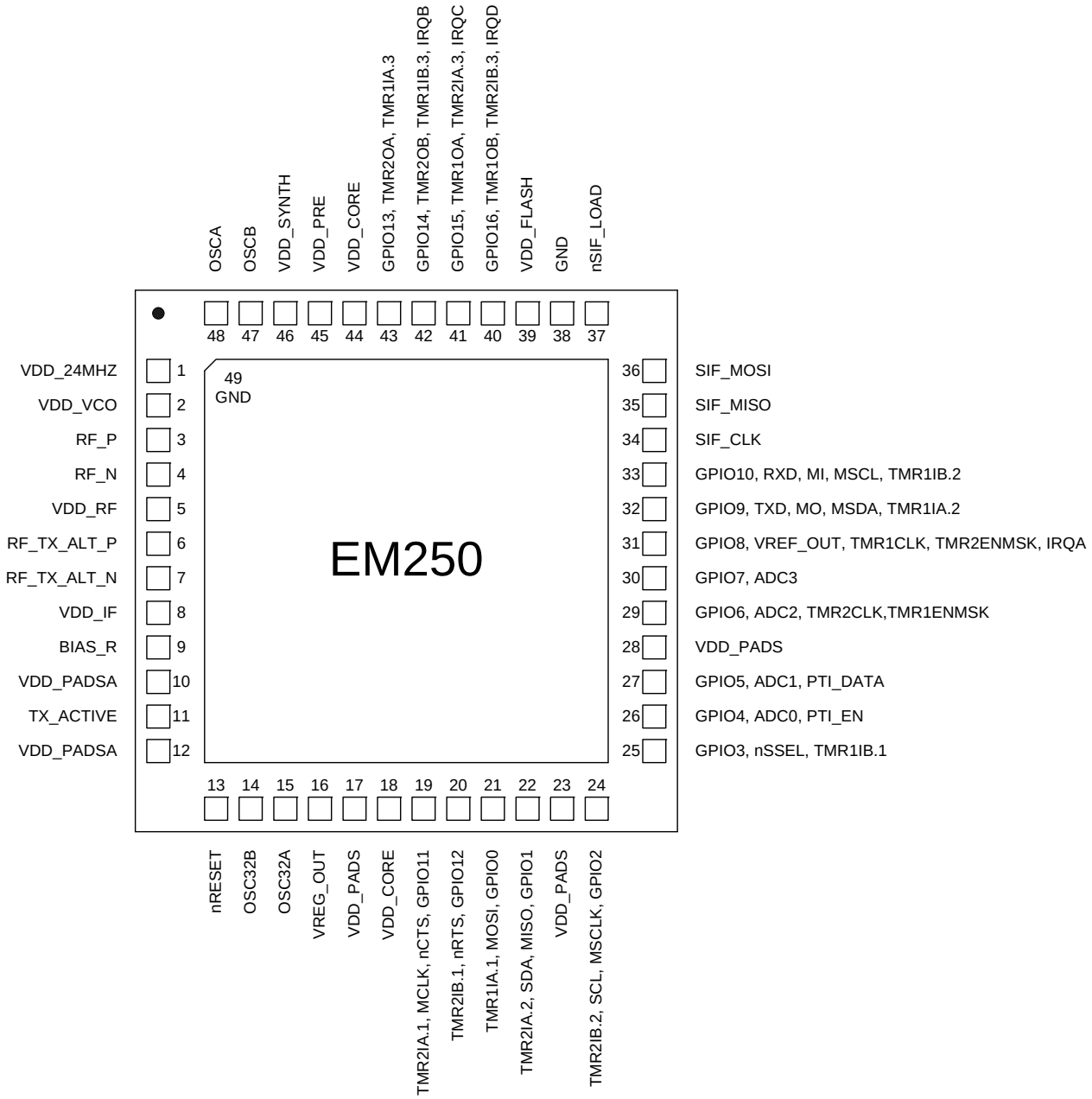


Figure 1. EM250 Pin Assignment

Refer to Table 17 and Table 18 for selecting alternate pin functions.

Table 1. Pin Descriptions

Pin #	Signal	Direction	Description
1	VDD_24MHZ	Power	1.8V high-frequency oscillator supply; should be connected to VREG_OUT
2	VDD_VCO	Power	1.8V VCO supply; should be connected to VREG_OUT
3	RF_P	I/O	Differential (with RF_N) receiver input/transmitter output
4	RF_N	I/O	Differential (with RF_P) receiver input/transmitter output
5	VDD_RF	Power	1.8V RF supply (LNA and PA); should be connected to VREG_OUT
6	RF_TX_ALT_P	O	Differential (with RF_TX_ALT_N) transmitter output (optional)
7	RF_TX_ALT_N	O	Differential (with RF_TX_ALT_P) transmitter output (optional)
8	VDD_IF	Power	1.8V IF supply (mixers and filters); should be connected to VREG_OUT
9	BIAS_R	I	Bias setting resistor
10	VDD_PADSA	Power	Analog pad supply (1.8V); should be connected to VREG_OUT
11	TX_ACTIVE	O	Logic-level control for external RX/TX switch The EM250 baseband controls TX_ACTIVE and drives it high (1.8V) when in TX mode. (Refer to Table 6 and section 4.2.2.)
12	VDD_PADSA	Power	Analog pad supply (1.8V); should be connected to VREG_OUT
13	nRESET	I	Active low chip reset (internal pull-up)
14	OSC32B	I/O	32.768kHz crystal oscillator. This pin should be left open when using external clock on OSC32A or when using the internal RC Oscillator
15	OSC32A	I/O	32.768kHz crystal oscillator or digital clock input. This pin can be left open when using the internal RC Oscillator.
16	VREG_OUT	Power	Regulator output (1.8V)
17	VDD_PADS	Power	Pads supply (2.1-3.6V)
18	VDD_CORE	Power	1.8V digital core supply; should be connected to VREG_OUT
19	GPIO11	I/O	Digital I/O Enable GPIO11 with `GPIO_CFG[7:4]`
	nCTS	I	UART CTS handshake of Serial Controller SC1 Enable SC1-4A with `GPIO_CFG[7:4]`, select UART with `SC1_MODE`
	MCLK	O	SPI master clock of Serial Controller SC1 Enable SC1-3M with `GPIO_CFG[7:4]`, select SPI with `SC1_MODE`, enable master with `SC1_SPICFG[4]`
	TMR2IA.1	I	Capture Input A of Timer 2 Enable CAP2-0 with `GPIO_CFG[7:4]`
20	GPIO12	I/O	Digital I/O Enable GPIO12 with `GPIO_CFG[7:4]`
	nRTS	O	UART RTS handshake of Serial Controller SC1 Enable SC1-4A with `GPIO_CFG[7:4]`, select UART with `SC1_MODE`
	TMR2IB.1	I	Capture Input B for Timer 2 Enable CAP2-0 with `GPIO_CFG[7:4]`
21	GPIO0	I/O	Digital I/O Enable GPIO0 with `GPIO_CFG[7:4]`

Pin #	Signal	Direction	Description
	MOSI	O	SPI master data out of Serial Controller SC2 Enable SC2-3M with GPIO_CFG[7:4], select SPI with SC2_MODE, enable master with SC2_SPICFG[4]
	MOSI	I	SPI slave data in of Serial Controller SC2 Enable SC2-4S with GPIO_CFG[7:4], select SPI with SC2_MODE, enable slave with SC2_SPICFG[4]
	TMR1IA.1	I	Capture Input A of Timer 1 Enable CAP1-0 with GPIO_CFG[7:4]
22	GPIO1	I/O	Digital I/O Enable GPIO1 with GPIO_CFG[7:4]
	MISO	I	SPI master data in of Serial Controller SC2 Enable SC2-3M with GPIO_CFG[7:4], select SPI with SC2_MODE, enable master with SC2_SPICFG[4]
	MISO	O	SPI slave data out of Serial Controller SC2 Enable SC2-4S with GPIO_CFG[7:4], select SPI with SC2_MODE, enable slave with SC2_SPICFG[4]
	SDA	I/O	I^2C data of Serial Controller SC2 Enable SC2-2 with GPIO_CFG[7:4], select I^2C with SC2_MODE
	TMR2IA.2	I	Capture Input A of Timer 2 Enable CAP2-1 with GPIO_CFG[7:4]
23	VDD_PADS	Power	Pads supply (2.1-3.6V)
24	GPIO2	I/O	Digital I/O Enable GPIO2 with GPIO_CFG[7:4]
	MSCLK	O	SPI master clock of Serial Controller SC2 Enable SC2-3M with GPIO_CFG[7:4], select SPI with SC2_MODE, enable master with SC2_SPICFG[4]
	MSCLK	I	SPI slave clock of Serial Controller SC2 Enable SC2-4S with GPIO_CFG[7:4], select SPI with SC2_MODE, enable slave with SC2_SPICFG[4]
	SCL	I/O	I^2C clock of Serial Controller SC2 Enable SC2-2 with GPIO_CFG[7:4], select I^2C with SC2_MODE
	TMR2IB.2	I	Capture Input B of Timer 2 Enable CAP2-1 with GPIO_CFG[7:4]
25	GPIO3	I/O	Digital I/O Enable GPIO3 with GPIO_CFG[7:4]
	nSSEL	I	SPI slave select of Serial Controller SC2 Enable SC2-4S with GPIO_CFG[7:4], select SPI with SC2_MODE, enable slave with SC2_SPICFG[4]
	TMR1IB.1	I	Capture Input B of Timer 1 Enable CAP1-0 with GPIO_CFG[7:4]
26	GPIO4	I/O	Digital I/O Enable GPIO4 with GPIO_CFG[12] and GPIO_CFG[8]
	ADC0	Analog	ADC Input 0 Enable ADC0 with GPIO_CFG[12] and GPIO_CFG[8]

Pin #	Signal	Direction	Description
	PTI_EN	O	Frame signal of Packet Trace Interface (PTI) Enable PTI with GPIO_CFG[12]
27	GPIO5	I/O	Digital I/O Enable GPIO5 with GPIO_CFG[12] and GPIO_CFG[9]
	ADC1	Analog	ADC Input 1 Enable ADC1 with GPIO_CFG[12] and GPIO_CFG[9]
	PTI_DATA	O	Data signal of Packet Trace Interface (PTI) Enable PTI with GPIO_CFG[12]
28	VDD_PADS	Power	Pads supply (2.1-3.6V)
29	GPIO6	I/O	Digital I/O Enable GPIO6 with GPIO_CFG[10]
	ADC2	Analog	ADC Input 2 Enable ADC2 with GPIO_CFG[10]
	TMR2CLK	I	External clock input of Timer 2
	TMR1ENMSK	I	External enable mask of Timer 1
30	GPIO7	I/O	Digital I/O Enable GPIO7 with GPIO_CFG[13] and GPIO_CFG[11]
	ADC3	Analog	ADC Input 3 Enable ADC3 with GPIO_CFG[13] and GPIO_CFG[11]
31	GPIO8	I/O	Digital I/O Enable GPIO8 with GPIO_CFG[14]
	VREF_OUT	Analog	ADC reference output Enable VREF_OUT with GPIO_CFG[14]
	TMR1CLK	I	External clock input of Timer 1
	TMR2ENMSK	I	External enable mask of Timer 2
	IRQA	I	External interrupt source A
32	GPIO9	I/O	Digital I/O Enable GPIO9 with GPIO_CFG[7:4]
	TXD	O	UART transmit data of Serial Controller SC1 Enable SC1-4A or SC1-2 with GPIO_CFG[7:4], select UART with SC1_MODE
	MO	O	SPI master data out of Serial Controller SC1 Enable SC1-3M with GPIO_CFG[7:4], select SPI with SC1_MODE, enable master with SC1_SPICFG[4]
	MSDA	I/O	I^2C data of Serial Controller SC1 Enable SC1-2 with GPIO_CFG[7:4], select I^2C with SC1_MODE
	TMR1IA.2	I	Capture Input A of Timer 1 Enable CAP1-1 or CAP1-1h with GPIO_CFG[7:4]
33	GPIO10	I/O	Digital I/O Enable GPIO10 with GPIO_CFG[7:4]
	RXD	I	UART receive data of Serial Controller SC1 Enable SC1-4A or SC1-2 with GPIO_CFG[7:4], select UART with SC1_MODE

Pin #	Signal	Direction	Description
	MI	I	SPI master data in of Serial Controller SC1 Enable SC1-3M with GPIO_CFG[7:4], select SPI with SC1_MODE, enable master with SC1_SPICFG[4]
	MSCL	I/O	I^2C clock of Serial Controller SC1 Enable SC1-2 with GPIO_CFG[7:4], select I^2C with SC1_MODE
	TMR1IB.2	I	Capture Input B of Timer 1 Enable CAP1-1 with GPIO_CFG[7:4]
34	SIF_CLK	I	Programming and debug interface, clock (internal pull-down)
35	SIF_MISO	O	Programming and debug interface, master in/slave out
36	SIF_MOSI	I	Programming and debug interface, master out/slave in (external pull-down required to guarantee state in Deep Sleep Mode)
37	nSIF_LOAD	I/O	Programming and debug interface, load strobe (open-collector with internal pull-up)
38	GND	Power	Ground supply
39	VDD_FLASH	Power	1.8V Flash memory supply; should be connected to VREG_OUT
40	GPIO16	I/O	Digital I/O Enable GPIO16 with GPIO_CFG[3]
	TMR1OB	O	Waveform Output B of Timer 1 Enable TMR1OB with GPIO_CFG[3]
	TMR2IB.3	I	Capture Input B of Timer 2 Enable CAP2-2 with GPIO_CFG[7:4]
	IRQD	I	External interrupt source D
41	GPIO15	I/O	Digital I/O Enable GPIO15 with GPIO_CFG[2]
	TMR1OA	O	Waveform Output A of Timer 1 Enable TMR1OA with GPIO_CFG[2]
	TMR2IA.3	I	Capture Input A of Timer 2 Enable CAP2-2 with GPIO_CFG[7:4]
	IRQC	I	External interrupt source C
42	GPIO14	I/O	Digital I/O Enable GPIO14 with GPIO_CFG[1]
	TMR2OB	O	Waveform Output B of Timer 2 Enable TMR2OB with GPIO_CFG[1]
	TMR1IB.3	I	Capture Input B of Timer 1 Enable CAP1-2 with *GPIO_CFG[7:4]*
	IRQB	I	External interrupt source B
43	GPIO13	I/O	Digital I/O Enable GPIO13 with GPIO_CFG[0]
	TMR2OA	O	Waveform Output A of Timer 2 Enable TMR2OA with GPIO_CFG[0]
	TMR1IA.3	I	Capture Input A of Timer 1 Enable CAP1-2 or CAP1-2h with GPIO_CFG[7:4]

Pin #	Signal	Direction	Description
44	VDD_CORE	Power	1.8V digital core supply; should be connected to VREG_OUT
45	VDD_PRE	Power	1.8V prescaler supply; should be connected to VREG_OUT
46	VDD_SYNTH	Power	1.8V synthesizer supply; should be connected to VREG_OUT
47	OSCB	I/O	24MHz crystal oscillator or left open when using external clock input on OSCA
48	OSCA	I/O	24MHz crystal oscillator or external clock input
49	GND	Ground	Ground supply pad in the bottom center of the package forms Pin 49 (See the *EM250 Reference Design* for PCB considerations.)

2 Top-Level Functional Description

Figure 2 shows a detailed block diagram of the EM250.

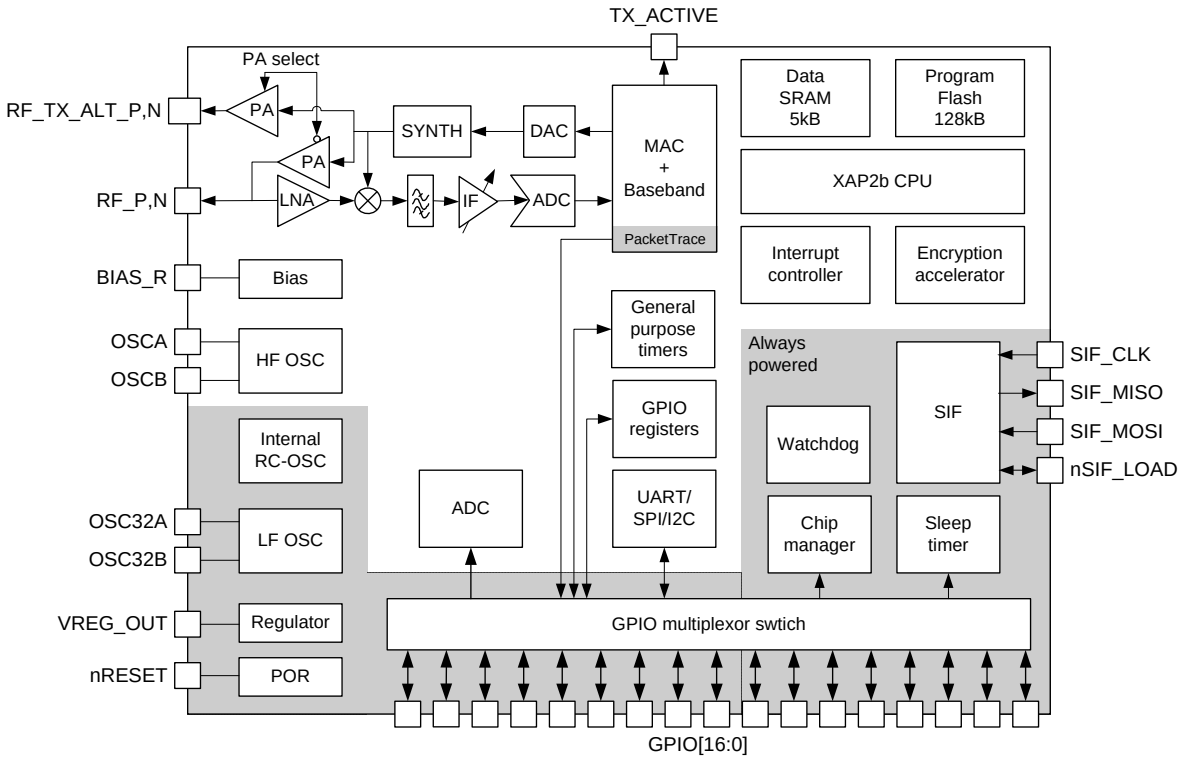


Figure 2. EM250 Block Diagram

The radio receiver is a low-IF, super-heterodyne receiver. It utilizes differential signal paths to minimize noise interference, and its architecture has been chosen to optimize co-existence with other devices within the 2.4GHz band (namely, IEEE 802.11g and Bluetooth). After amplification and mixing, the signal is filtered and combined prior to being sampled by an ADC.

The digital receiver implements a coherent demodulator to generate a chip stream for the hardware-based MAC. In addition, the digital receiver contains the analog radio calibration routines and control of the gain within the receiver path.

The radio transmitter utilizes an efficient architecture in which the data stream directly modulates the VCO. An integrated PA boosts the output power. The calibration of the TX path as well as the output power is controlled by digital logic. If the EM250 is to be used with an external PA, the TX_ACTIVE signal should be used to control the timing of the external switching logic.

The integrated 4.8 GHz VCO and loop filter minimize off-chip circuitry. Only a 24MHz crystal with its loading capacitors is required to properly establish the PLL reference signal.

The MAC interfaces the data memory to the RX and TX baseband modules. The MAC provides hardware-based IEEE 802.15.4 packet-level filtering. It supplies an accurate symbol time base that minimizes the synchronization effort of the software stack and meets the protocol timing requirements. In addition, it provides timer and synchronization assistance for the IEEE 802.15.4 CSMA-CA algorithm.

The EM250 integrates hardware support for a Packet Trace module, which allows robust packet-based debug. This element is a critical component of Ember Desktop, the Ember software IDE, providing advanced network debug capability when coupled with the Ember Debug Adapter (ISA).

The EM250 integrates a 16-bit XAP2b microprocessor developed by Cambridge Consultants Ltd. This power-efficient, industry-proven core provides the appropriate level of processing power to meet the needs of ZigBee applications. In addition, 128kB of Flash and 5kB of SRAM comprise the program and data memory elements, respectively. The EM250 employs a configurable memory protection scheme usually found on larger microcontrollers. In addition, the SIF module provides a non-intrusive programming and debug interface allowing for real-time application debugging.

The EM250 contains 17 GPIO pins shared with other peripheral (or alternate) functions. Flexible routing within the EM250 lets external devices utilize the alternate functions on a variety of different GPIOs. The integrated Serial Controller SC1 can be configured for SPI (master-only), I^2C (master-only), or UART functionality, and the Serial Controller SC2 can be configured for SPI (master or slave) or I^2C (master-only) operation.

The EM250 has an ADC integrated which can sample analog signals from four GPIO pins single-ended or differentially. In addition, the unregulated voltage supply VDD_PADS, regulated supply VDD_PADSA, voltage reference VREF, and GND can be sampled. The integrated voltage reference VREF for the ADC can be made available to external circuitry.

The integrated voltage regulator generates a regulated 1.8V reference voltage from an unregulated supply voltage. This voltage is decoupled and routed externally to supply the 1.8V to the core logic. In addition, an integrated POR module allows for the proper cold start of the EM250.

The EM250 contains one high-frequency (24MHz) crystal oscillator and, for low-power operation, a second low-frequency oscillator (either an internal 10kHz RC oscillator or an external 32.768kHz crystal oscillator).

The EM250 contains two power domains. The always-powered High Voltage Supply is used for powering the GPIO pads and critical chip functions. The rest of the chip is powered by a regulated Low Voltage Supply which can be disabled during deep sleep to reduce the power consumption.

3 Electrical Characteristics

3.1 Absolute Maximum Ratings

Table 2 lists the absolute maximum ratings for the EM250.

Table 2. Absolute Maximum Ratings

Parameter	Notes	Min.	Max.	Unit
Regulator voltage (VDD_PADS)		- 0.3	3.6	V
Core voltage (VDD_24MHZ, VDD_VCO, VDD_RF, VDD_IF, VDD_PADSA, VDD_FLASH, VDD_PRE, VDD_SYNTH, VDD_CORE)		- 0.3	2.0	V
Voltage on RF_P,N; RF_TX_ALT_P,N		- 0.3	3.6	V
RF Input Power (for max level for correct packet reception see Table 7)	RX signal into a lossless balun		+15	dBm
Voltage on any GPIO[16:0], SIF_CLK, SIF_MISO, SIF_MOSI, nSIF_LOAD, OSC32A, OSC32B, nRESET, VREG_OUT		- 0.3	VDD_PADS+ 0.3	V
Voltage on TX_ACTIVE, BIAS_R, OSCA, OSCB		- 0.3	VDD_CORE+ 0.3	V
Storage temperature		- 40	+ 140	˚C

3.2 Recommended Operating Conditions

Table 3 lists the rated operating conditions of the EM250.

Table 3. Operating Conditions

Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Regulator input voltage (VDD_PADS)		2.1		3.6	V
Core input voltage (VDD_24MHZ, VDD_VCO, VDD_RF, VDD_IF, VDD_PADSA, VDD_FLASH, VDD_PRE, VDD_SYNTH, VDD_CORE)		1.7	1.8	1.9	V
Temperature range		- 40		+ 85	˚C

3.3 Environmental Characteristics

Table 4 lists the environmental characteristics of the EM250.

Table 4. Environmental Characteristics

Parameter	Test Conditions	Min.	Typ.	Max.	Unit
ESD (human body model)	On any Pin	- 2		+ 2	kV
ESD (charged device model)	Non-RF Pins	- 400		+ 400	V
ESD (charged device model)	RF Pins	- 225		+ 225	V
Moisture Sensitivity Level (MSL)			MSL3		

3.4 DC Electrical Characteristics

Table 5 lists the DC electrical characteristics of the EM250.

Note: Current Measurements were collected using the EmberZNet software stack Version 3.0.1.

Table 5. DC Characteristics

Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Regulator input voltage (VDD_PADS)		2.1		3.6	V
Power supply range (VDD_CORE)	Regulator output or external input	1.7	1.8	1.9	V
Deep Sleep Current					
Quiescent current, including internal RC oscillator	At 25° C.			1.0	µA
Quiescent current, including 32.768kHz oscillator	At 25° C.			1.5	µA
RESET Current					
Quiescent current, nRESET asserted	Typ at 25° C/3V Max at 85° C/3.6V		1.5	2.0	mA
RX Current					
Radio receiver, MAC, and baseband (boost mode)			30.0		mA
Radio receiver, MAC, and baseband			28.0		mA
CPU, RAM, and Flash memory	At 25° C and 1.8V core		8.0		mA
Total RX current (= $I_{\text{Radio receiver, MAC and baseband, CPU}}$ + $I_{\text{RAM, and Flash memory}}$)	At 25° C, VDD_PADS=3.0V		36.0		mA
TX Current					
Radio transmitter, MAC, and baseband (boost mode)	At max. TX power (+ 5dBm typical)		34.0		mA
Radio transmitter, MAC, and baseband	At max. TX power (+ 3dBm typical)		28.0		mA
	At 0 dBm typical		24.0		mA
	At min. TX power (-32dBm typical)		19.0		mA
CPU, RAM, and Flash memory	At 25° C, VDD_PADS=3.0V		8.0		mA
Total TX current (= $I_{\text{Radio transmitter, MAC and baseband, CPU}}$ + $I_{\text{RAM, and Flash memory}}$)	At 25° C and 1.8V core; max. power out		36.0		mA

Table 6 contains the digital I/O specifications for the EM250. The digital I/O power (named VDD_PADS) comes from three dedicated pins (Pins 17, 23, and 28). The voltage applied to these pins sets the I/O voltage.

Table 6. Digital I/O Specifications

Parameter	Name	Min.	Typ.	Max.	Unit
Voltage supply	VDD_PADS	2.1		3.6	V
Input voltage for logic 0	V_{IL}	0		0.2 x VDD_PADS	V
Input voltage for logic 1	V_{IH}	0.8 x VDD_PADS		VDD_PADS	V
Input current for logic 0	I_{IL}			- 0.5	µA
Input current for logic 1	I_{IH}			0.5	µA
Input pull-up resistor value	R_{IPU}		30		kΩ
Input pull-down resistor value	R_{IPD}		30		kΩ
Output voltage for logic 0	V_{OL}	0		0.18 x VDD_PADS	V
Output voltage for logic 1	V_{OH}	0.82 x VDD_PADS		VDD_PADS	V
Output source current (standard current pad)	I_{OHS}			4	mA
Output sink current (standard current pad)	I_{OLS}			4	mA
Output source current high current pad: GPIO[16:13]	I_{OHH}			8	mA
Output sink current high current pad: GPIO[16:13]	I_{OLH}			8	mA
Total output current (for I/O Pads)	$I_{OH} + I_{OL}$			40	mA
Input voltage threshold for OSC32A		0.2 x VDD_PADS		0.8 x VDD_PADS	V
Input voltage threshold for OSCA		0.2 x VDD_CORE		0.8 x VDD_CORE	V
Output voltage level (TX_ACTIVE)		0.18 x VDD_CORE		0.82 x VDD_CORE	V
Output source current (TX_ACTIVE)				1	mA

3.5 RF Electrical Characteristics

3.5.1 Receive

Table 7 lists the key parameters of the integrated IEEE 802.15.4 receiver on the EM250.

Note: Receive Measurements were collected with Silicon Labs' EM250 Lattice Balun Reference Design (Version B1) at 2440MHz and using the EmberZNet software stack Version 3.0.1. The Typical number indicates one standard deviation above the mean, measured at room temperature (25°C). The Min and Max numbers were measured over process corners at room temperature.

Note: The adjacent channel rejection (ACR) measurements were performed by using an unfiltered, ideal IEEE 802.15.4 signal of continuous pseudo-random data as the interferer. For more information on ACR measurement techniques, see document AN709, *Adjacent Channel Rejection Measurements*.

Table 7. Receive Characteristics

Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Frequency range		2400		2500	MHz
Sensitivity (boost mode)	1% PER, 20byte packet defined by IEEE 802.15.4		-100	-95	dBm
Sensitivity	1% PER, 20byte packet defined by IEEE 802.15.4		-99	-94	dBm
High-side adjacent channel rejection	IEEE 802.15.4 signal at - 82dBm		35		dB
Low-side adjacent channel rejection	IEEE 802.15.4 signal at - 82dBm		35		dB
2nd high-side adjacent channel rejection	IEEE 802.15.4 signal at - 82dBm		43		dB
2nd low-side adjacent channel rejection	IEEE 802.15.4 signal at - 82dBm		43		dB
Channel rejection for all other channels	IEEE 802.15.4 signal at - 82dBm		40		dB
802.11g rejection centered at + 12MHz or - 13MHz	IEEE 802.15.4 signal at - 82dBm		35		dB
Maximum input signal level for correct operation (low gain)		0			dBm
Image suppression			30		dB
Co-channel rejection	IEEE 802.15.4 signal at - 82dBm		- 6		dBc
Relative frequency error (2x40 ppm required by IEEE 802.15.4)		-120		+120	ppm
Relative timing error (2x40 ppm required by IEEE 802.15.4)		-120		+120	ppm
Linear RSSI range		40			dB
RSSI Range		-90		-30	dB

3.5.2 Transmit

Table 8 lists the key parameters of the integrated IEEE 802.15.4 transmitter on the EM250.

Note: Transmit Measurements were collected with Silicon Labs' EM250 Lattice Balun Reference Design (Version B1) at 2440MHz and using the EmberZNet software stack Version 3.0.1. The Typical number indicates one standard deviation below the mean, measured at room temperature (25°C). The Min and Max numbers were measured over process corners at room temperature.

Table 8. Transmit Characteristics

Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Maximum output power (boost mode)	At highest power setting		5		dBm
Maximum output power	At highest power setting	0	3		dBm
Minimum output power	At lowest power setting		- 32		dBm
Error vector magnitude	As defined by IEEE 802.15.4, which sets a 35% maximum		5	15	%
Carrier frequency error		- 40		+ 40	ppm
Load impedance			200+j90		Ω
PSD mask relative	3.5MHz away	- 20			dB
PSD mask absolute	3.5MHz away	- 30			dBm

3.5.3 Synthesizer

Table 9 lists the key parameters of the integrated synthesizer on the EM250.

Table 9. Synthesizer Characteristics

Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Frequency range		2400		2500	MHz
Frequency resolution			11.7		kHz
Lock time	From off, with correct VCO DAC setting			100	µs
Relock time	Channel change or RX/TX turnaround (IEEE 802.15.4 defines 192µs turnaround time)			100	µs
Phase noise at 100kHz			- 71		dBc/Hz
Phase noise at 1MHz			- 91		dBc/Hz
Phase noise at 4MHz			- 103		dBc/Hz
Phase noise at 10MHz			- 111		dBc/Hz

4 Functional Description—System Modules

The EM250 contains a dual-thread mode of operation—System Mode and Application Mode—to guarantee microcontroller bandwidth to the application developer and protect the developer from errant software access.

During System Mode, all areas including the RF Transceiver, MAC, Packet Trace Interface, Sleep Timer, Power Management Module, Watchdog Timer, and Power on Reset Module are accessible.

Since the EM250 comes with a license to EmberZNet PRO, the Silicon Labs ZigBee-compliant software stack, these areas are not available to the application developer in Application Mode. The following brief description of these modules provides the necessary background on the operation of the EM250. For more information, contact support at www.silabs.com/zigbee-support.

4.1 Receive (RX) Path

The EM250 RX path spans the analog and digital domains. The RX architecture is based on a low-IF, super-heterodyne receiver. It utilizes differential signal paths to minimize noise interference. The input RF signal is mixed down to the IF frequency of 4MHz by I and Q mixers. The output of the mixers is filtered and combined prior to being sampled by a 12Msps ADC. The RX filtering within the RX path has been designed to optimize the co-existence of the EM250 with other 2.4GHz transceivers, such as the IEEE 802.11g and Bluetooth.

4.1.1 RX Baseband

The EM250 RX baseband (within the digital domain) implements a coherent demodulator for optimal performance. The baseband demodulates the O-QPSK signal at the chip level and synchronizes with the IEEE 802.15.4-2003 preamble. An automatic gain control (AGC) module adjusts the analog IF gain continuously (every ¼ symbol) until the preamble is detected. Once the packet preamble is detected, the IF gain is fixed during the packet reception. The baseband de-spreads the demodulated data into 4-bit symbols. These symbols are buffered and passed to the hardware-based MAC module for filtering.

In addition, the RX baseband provides the calibration and control interface to the analog RX modules, including the LNA, RX Baseband Filter, and modulation modules. The EmberZNet PRO software includes calibration algorithms which use this interface to reduce the effects of process and temperature variation.

4.1.2 RSSI and CCA

The EM250 calculates the RSSI over an 8-symbol period as well as at the end of a received packet. It utilizes the RX gain settings and the output level of the ADC within its algorithm. The linear range of RSSI is specified to be 40dB over all temperatures. At room temperature, the linear range is approximately 60dB (-90 dBm to -30dBm).

The EM250 RX baseband provides support for the IEEE 802.15.4-2003 required CCA methods summarized in Table 10. Modes 1, 2, and 3 are defined by the 802.15.4-2003 standard; Mode 0 is a proprietary mode.

Table 10. CCA Mode Behavior

CCA Mode	Mode Behavior
0	Clear channel reports busy medium if either carrier sense OR RSSI exceeds their thresholds.
1	Clear channel reports busy medium if RSSI exceeds its threshold.
2	Clear channel reports busy medium if carrier sense exceeds its threshold.
3	Clear channel reports busy medium if both RSSI AND carrier sense exceed their thresholds.

The EmberZNet PRO Software Stack sets the CCA Mode, and it is not configurable by the Application Layer. For software versions beginning with EmberZNet 2.5.4, CCA Mode 1 is used, and a busy channel is reported if the RSSI exceeds its threshold. For software versions prior to 2.5.4, the CCA Mode was set to 0.

At RX input powers higher than –25dBm, there is some compression in the receive chain where the gain is not properly adjusted. In the worst case, this has resulted in packet loss of up to 0.1%. This packet loss can be seen in range testing measurements when nodes are closely positioned and transmitting at high power or when receiving from test equipment. There is no damage to the EM250 from this problem. This issue will rarely occur in the field as ZigBee Nodes will be spaced far enough apart. If nodes are close enough for it to occur in the field, the MAC and networking software treat the packet as not having been received and therefore the MAC level and network level retries resolve the problem without upper level application being notified.

4.2 Transmit (TX) Path

The EM250 transmitter utilizes both analog circuitry and digital logic to produce the O-QPSK modulated signal. The area-efficient TX architecture directly modulates the spread symbols prior to transmission. The differential signal paths increase noise immunity and provide a common interface for the external balun.

4.2.1 TX Baseband

The EM250 TX baseband (within the digital domain) performs the spreading of the 4-bit symbol into its IEEE 802.15.4-2003-defined 32-chip I and Q sequence. In addition, it provides the interface for software to perform the calibration of the TX module in order to reduce process, temperature, and voltage variations.

4.2.2 TX_ACTIVE Signal

Even though the EM250 provides an output power suitable for most ZigBee applications, some applications will require an external power amplifier (PA). Due to the timing requirements of IEEE 802.15.4-2003, the EM250 provides a signal, TX_ACTIVE, to be used for external PA power management and RF Switching logic. When in TX, the TX Baseband drives TX_ACTIVE high (as described in Table 6). When in RX, the TX_ACTIVE signal is low. If an external PA is not required, then the TX_ACTIVE signal should be connected to GND through a 100k Ohm resistor, as shown in the application circuit in Figure 16.

The TX_ACTIVE signal can only source 1mA of current, and it is based upon the 1.8V signal swing. If the PA Control logic requires greater current or voltage potential, then TX_ACTIVE should be buffered externally to the EM250.

4.3 Integrated MAC Module

The EM250 integrates critical portions of the IEEE 802.15.4-2003 MAC requirements in hardware. This allows the microcontroller to provide greater bandwidth to application and network operations. In addition, the hardware acts as a first-line filter for non-intended packets. The EM250 MAC utilizes a DMA interface to RAM memory to further reduce the overall microcontroller interaction when transmitting or receiving packets.

When a packet is ready for transmission, the software configures the TX MAC DMA by indicating the packet buffer RAM location. The MAC waits for the backoff period, then transitions the baseband to TX mode and performs channel assessment. When the channel is clear, the MAC reads data from the RAM buffer, calculates

the CRC, and provides 4-bit symbols to the baseband. When the final byte has been read and sent to the baseband, the CRC remainder is read and transmitted.

The MAC resides in RX mode most of the time, and different format and address filters keep non-intended packets from using excessive RAM buffers, as well as preventing the CPU from being interrupted. When the reception of a packet begins, the MAC reads 4-bit symbols from the baseband and calculates the CRC. It assembles the received data for storage in a RAM buffer. A RX MAC DMA provides direct access to the RAM memory. Once the packet has been received, additional data is appended to the end of the packet in the RAM buffer space. The appended data provides statistical information on the packet for the software stack.

The primary features of the MAC are:

- CRC generation, appending, and checking
- Hardware timers and interrupts to achieve the MAC symbol timing
- Automatic preamble, and SFD pre-pended to a TX packet
- Address recognition and packet filtering on received packets
- Automatic acknowledgement transmission
- Automatic transmission of packets from memory
- Automatic transmission after backoff time if channel is clear (CCA)
- Automatic acknowledgement checking
- Time stamping of received and transmitted messages
- Attaching packet information to received packets (LQI, RSSI, gain, time stamp, and packet status)
- IEEE 802.15.4 timing and slotted/unslotted timing

4.4 Packet Trace Interface (PTI)

The EM250 integrates a true PHY-level PTI for effective network-level debugging. This two-signal interface monitors all the PHY TX and RX packets (in a non-intrusive manner) between the MAC and baseband modules. It is an asynchronous 500kbps interface and cannot be used to inject packets into the PHY/MAC interface. The two signals from the EM250 are the frame signal (PTI_EN) and the data signal (PTI_DATA). The PTI is supported by Ember Desktop.

4.5 XAP2b Microprocessor

The EM250 integrates the XAP2b microprocessor developed by Cambridge Consultants Ltd., making it a true system-on-a-chip solution. The XAP2b is a 16-bit Harvard architecture processor with separate program and data address spaces. The word width is 16 bits for both the program and data sides. Data-side addresses are always specified in bytes, though they can be accessed as either bytes or words, while program-side addresses are always specified and accessed as words. The data-side address bus is effectively 15 bits wide, allowing for an address space of 32kB; the program-side address bus is 16 bits wide, addressing 64k words.

The standard XAP2 microprocessor and accompanying software tools have been enhanced to create the XAP2b microprocessor used in the EM250. The XAP2b adds data-side byte addressing support to the XAP2 by utilizing the 15^{th} bit of the data-side address bus to indicate byte or word accesses. This allows for more productive usage of RAM, optimized code, and a more familiar architecture for Silicon Labs customers when compared to the standard XAP2.

The XAP2b clock speed is 12MHz. When used with the EmberZNet PRO stack, code is loaded into Flash memory over the air or by a serial link using a built-in bootloader in a reserved area of the Flash. Alternatively, code may be loaded via the SIF interface with the assistance of RAM-based utility routines also loaded via SIF.

The XAP2b in the EM250 has also been enhanced to support two separate protection levels. The EmberZNet stack runs in System Mode, which allows full, unrestricted access to all areas of the chip, while application code runs in Application Mode. When running in Application Mode, writing to certain areas of memory and registers is restricted to prevent common software bugs from interfering with the operation of the EmberZNet

stack. These errant writes are captured and details are reported to the developer to assist in tracking down and fixing these issues.

4.6 Embedded Memory

As shown in Figure 3, the program side of the address space contains mappings to both integrated Flash and RAM blocks.

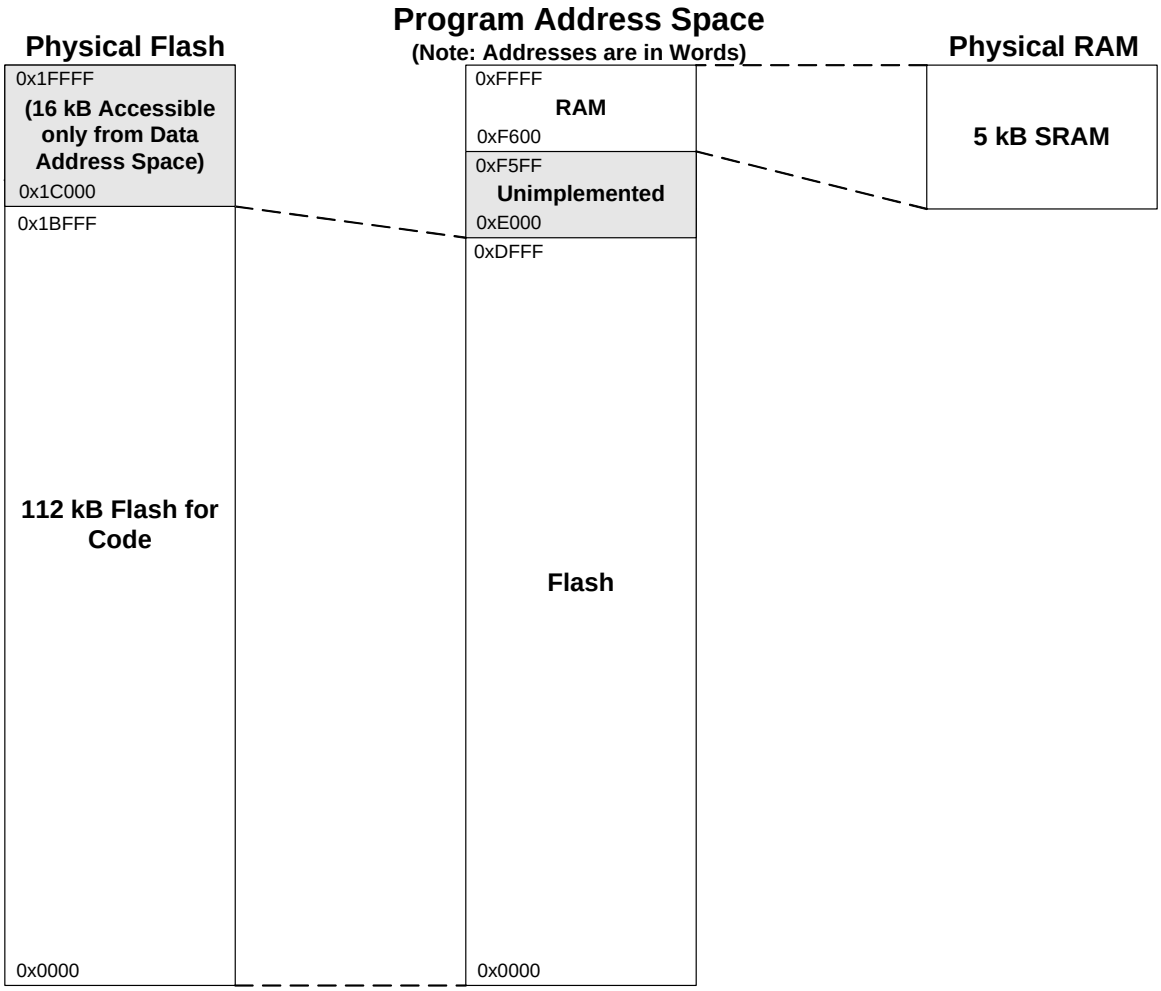


Figure 3. Program Address Space

The data side of the address space contains mappings to the same Flash and RAM blocks, as well as registers and a separate Flash information area, as shown in Figure 4.

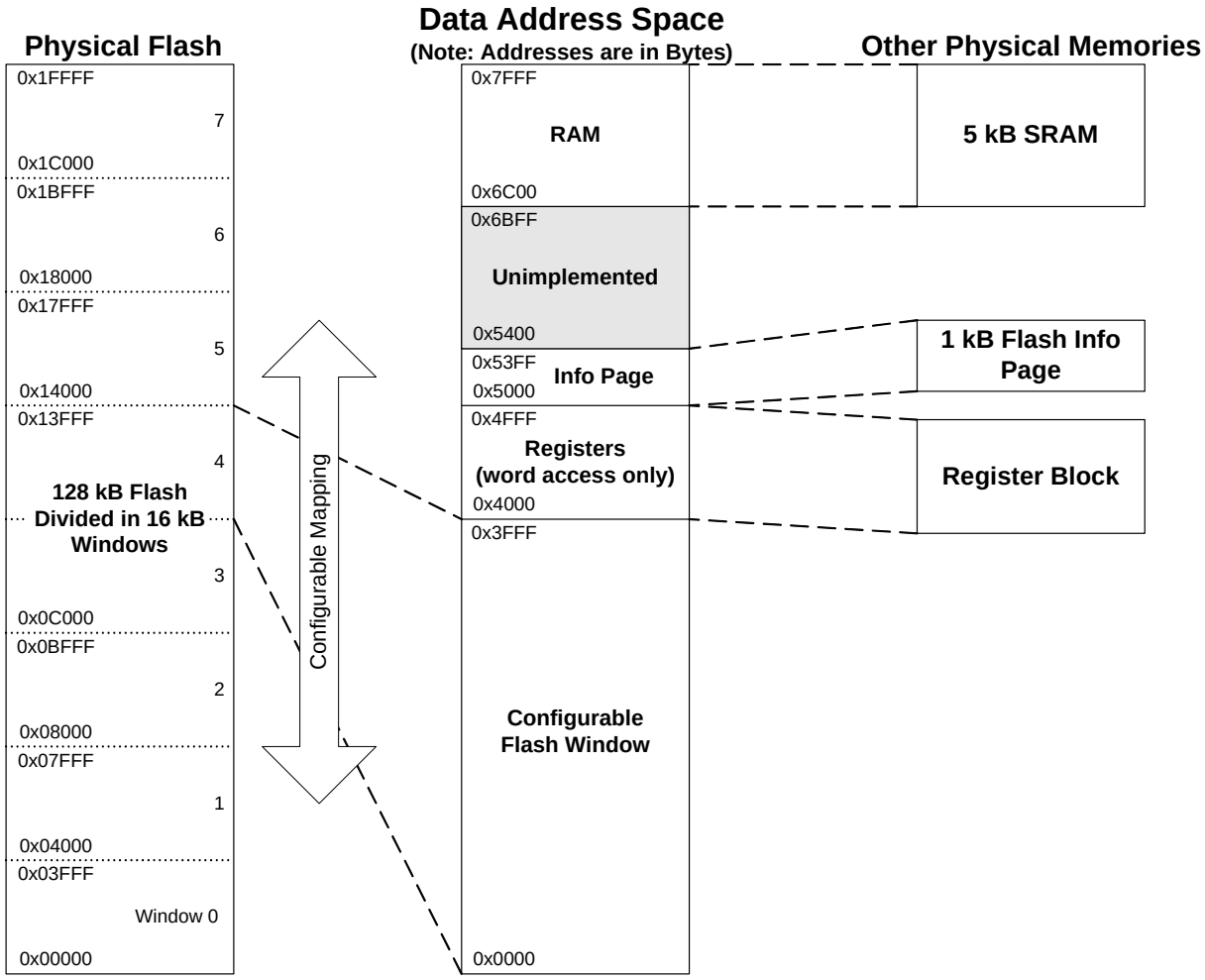


Figure 4. Data Address Space

4.6.1 Flash Memory

The EM250 integrates 128kB of Flash memory. The Flash cell has been qualified for a data retention time of >100 years at room temperature. Each Flash page size is 1024 bytes and is rated to have a guaranteed 1,000 write/erase cycles.

The Flash memory has mappings to both the program and data side address spaces. On the program side, the first 112kB of the Flash memory are mapped to the corresponding first 56k word addresses to allow for code storage, as shown in Figure 3.

On the program side, the Flash is always read as whole words. On the data side, the Flash memory is divided into eight 16kB sections, which can be separately mapped into a Flash window for the storage of constant data and the Simulated EEPROM. As shown in Figure 4, the Flash window corresponds to the first 16kB of the data-side address space. On the data side, the Flash may be read as bytes, but can only be written to one word at a time using utility routines in the EmberZNet PRO stack and HAL.

4.6.2 Simulated EEPROM

The EmberZNet PRO stack reserves a section of Flash memory to provide Simulated EEPROM storage area for stack and customer tokens. Therefore, the EM250 utilizes 8kB of upper Flash storage. This section of Flash is only accessible when mapped to the Flash window in the data-side address space. Because the Flash cells are qualified for up to 1,000 write cycles, the Simulated EEPROM implements an effective wear-leveling algorithm which effectively extends the number of write cycles for individual tokens.

4.6.3 Flash Information Area (FIA)

The EM250 also includes a separate 1024-byte FIA that can be used for storage of data during manufacturing, including serial numbers and calibration values. This area is mapped to the data side of the address space, starting at address 0x5000. While this area can be read as individual bytes, it can only be written to one word at a time, and may only be erased as a whole. Programming of this special Flash page can only be enabled using the SIF interface to prevent accidental corruption or erasure. The EmberZNet PRO stack reserves a small portion of this space for its own use, but the rest is available to the application.

4.6.4 RAM

The EM250 integrates 5kB of SRAM. Like the Flash memory, this RAM is also mapped to both the program and data-side address spaces. On the program side, the RAM is mapped to the top 2.5k words of the program address space. The program-side mapping of the RAM is used for code when writing to or erasing the Flash memory. On the data side, the RAM is also mapped to the top of the address space, occupying the last 5kB, as shown in Figure 3 and Figure 4.

Additionally, the EM250 supports a protection mechanism to prevent application code from overwriting system data stored in the RAM. To enable this, the RAM is segmented into 32-byte sections, each with a configurable bit that allows or denies write access when the EM250 is running in Application Mode. Read access is always allowed to the entire RAM, and full access is always allowed when the EM250 is running in System Mode. The EmberZNet PRO stack intelligently manages this protection mechanism to assist in tracking down many common application errors.

4.6.5 Registers

Table 41 provides a short description of all application-accessible registers within the EM250. Complete descriptions are provided at the end of each applicable Functional Description section. The registers are mapped to the data-side address space starting at address 0x4000. These registers allow for the control and configuration of the various peripherals and modules. The registers may only be accessed as whole word quantities; attempts to access them as bytes may result in undefined behavior. There are additional registers used by the EmberZNet PRO stack when the EM250 is running in System Mode, allowing for control of the MAC, baseband, and other internal modules. These system registers are protected from being modified when the EM250 is running in Application Mode.

4.7 Encryption Accelerator

The EM250 contains a hardware AES encryption engine that is attached to the CPU using a memory-mapped interface. NIST-based CCM, CCM*, CBC-MAC, and CTR modes are implemented in hardware. These modes are described in the IEEE 802.15.4-2003 specification, with the exception of CCM*, which is described in the ZigBee Security Services Specification 1.0. The EmberZNet PRO stack implements a security API for applications that require security at the application level.

4.8 nRESET Signal

When the asynchronous external reset signal, nRESET (Pin 13), is driven low for a time greater than 200ns, the EM250 resets to its default state. An integrated glitch filter prevents noise from causing an inadvertent reset to occur. If the EM250 is to be placed in a noisy environment, an external LC Filter or supervisory reset circuit is recommended to guarantee the integrity of the reset signal.

When nRESET asserts, all EM250 registers return to their reset state as defined by Table 41. In addition, the EM250 consumes 1.5mA (typical) of current when held in RESET.

4.9 Reset Detection

The EM250 contains multiple reset sources. The reset event is logged into the reset source register, which lets the CPU determine the cause of the last reset. The following reset causes are detected:

- Power-on-Reset
- Watchdog
- PC rollover
- Software reset
- Core Power Dip

4.10 Power-on-Reset (POR)

Each voltage domain (1.8V Digital Core Supply VDD_CORE and Pads Supply VDD_PADS) has a power-on-reset (POR) cell.

The VDD_PADS POR cell holds the always-powered high-voltage domain in reset until the following conditions have been met:

- The high-voltage Pads Supply VDD_PADS voltage rises above a threshold.
- The internal RC clock starts and generates three clock pulses.
- The 1.8V POR cell holds the main digital core in reset until the regulator output voltage rises above a threshold.

Additionally, the digital domain counts 1,024 clock edges on the 24MHz crystal before releasing the reset to the main digital core.

Table 11 lists the features of the EM250 POR circuitry.

Table 11. POR Specifications

Parameter	Min.	Typ.	Max.	Unit
VDD_PADS POR release	1.0	1.2	1.4	V
VDD_PADS POR assert	0.5	0.6	0.7	V
1.8V POR release	1.35	1.5	1.65	V
1.8V POR hysteresis	0.08	0.1	0.12	V

4.11 Clock Sources

The EM250 integrates three oscillators: a high-frequency 24MHz crystal oscillator, an optional low-frequency 32.768kHz crystal oscillator, and a low-frequency internal 10kHz RC oscillator.

4.11.1 High-Frequency Crystal Oscillator

The integrated high-frequency crystal oscillator requires an external 24MHz crystal with an accuracy of ±40ppm. Based upon the application Bill of Materials and current consumption requirements, the external crystal can cover a range of ESR requirements. For a lower ESR, the cost of the crystal increases but the overall current consumption decreases. Likewise, for higher ESR, the cost decreases but the current consumption increases. Therefore, the designer can choose a crystal to fit the needs of the application.

Table 12 lists the specifications for the high-frequency crystal.

Table 12. High-Frequency Crystal Specifications

Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Frequency			24		MHz
Duty cycle		40		60	%
Phase noise from 1kHz to 100kHz				- 120	dBc/Hz
Accuracy	Initial, temperature, and aging	- 40		+ 40	Ppm
Crystal ESR	Load capacitance of 10pF			100	Ω
Crystal ESR	Load capacitance of 18pF			60	Ω
Start-up time to stable clock (max. bias)				1	Ms
Start-up time to stable clock (optimum bias)				2	Ms
Current consumption	Good crystal: 20Ω ESR, 10pF load		0.2	0.3	mA
Current consumption	Worst-case crystals (60Ω, 18pF or 100Ω, 10pF)			0.5	mA
Current consumption	At maximum bias			1	mA

4.11.2 Low-Frequency Oscillator

The optional low-frequency crystal source for the EM250 is a 32.768kHz crystal. Table 13 lists the requirements for the low-frequency crystal. The low-frequency crystal may be used for applications that require greater accuracy than can be provided by the internal RC oscillator. When using the internal RC Oscillator, the pins OSC32A and OSC32B can be left open (or not connected). If the designer would like to implement the low frequency clock source with an external digital logic source, then the OSC32A pin should be connected to the clock source with OSC32B left open.

The crystal oscillator has been designed to accept any standard watch crystal with an ESR of 100 kΩ (max). In order to keep the low frequency oscillator from being overdriven by the 32.768kHz crystal, Silicon Labs recommends the PCB designer asymmetrically load the capacitor with 18pF on OSC32A and 27pf on OSC32B. For more information on this design recommendation, please review document AN695, *Designing with an EM250.*

Table 13. Low-Frequency Crystal Specifications

Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Frequency			32.768		kHz
Accuracy	Initial, temperature, and aging	- 100		+ 100	Ppm
Load capacitance (18pF on OSC32A and 27pF on OSC32B)			12.5		pF
Crystal ESR				100	kΩ
Start-up time				1	S
Current consumption	At 25°C, VDD_PADS=3.0V		0.6		µA

4.11.3 Internal RC Oscillator

The EM250 has a low-power, low-frequency RC oscillator that runs all the time. Its nominal frequency is 10kHz.

The RC oscillator has a coarse analog trim control, which is first adjusted to get the frequency as close to 10kHz as possible. This raw clock is used by the chip management block. It is also divided down to 1kHz using a variable divider to allow software to accurately calibrate it. This calibrated clock is available to the sleep timer.

Timekeeping accuracy depends on temperature fluctuations the chip is exposed to, power supply impedance, and the calibration interval, but in general it will be better than 150ppm (including crystal error of 40ppm). If this tolerance is accurate enough for the application, then there is no need to use an external 32.768kHz crystal oscillator. By removing the 32.768kHz oscillator, the external component count further decreases as does the Bill of Material cost.

Note: If the 32.768kHz crystal is not needed, then OSC32A and OSC32B pins should be left open or not connected.

Table 14 lists the specifications of the RC oscillator.

Table 14. RC Oscillator Specifications

Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Frequency			10		kHz
Analog trim steps			1		kHz
Frequency variation with supply	For a voltage drop from 3.6V to 3.1V or 2.6V to 2.1V		0.75	1.5	%

4.12 Random Number Generator

The EM250 allows for the generation of random numbers by exposing a randomly generated bit from the RX ADC. Analog noise current is passed through the RX path, sampled by the receive ADC, and stored in a register. The value contained in this register could be used to seed a software-generated random number. The EmberZNet PRO stack utilizes these random numbers to seed the Random MAC Backoff and Encryption Key Generators.

4.13 Watchdog Timer

The EM250 contains a watchdog timer clocked from the internal oscillator. The watchdog is disabled by default, but can be enabled or disabled by software.

If the timer reaches its time-out value of approximately 2 seconds, it will generate a reset signal to the chip.

When software is running properly, the application can periodically restart this timer to prevent the reset signal from being generated.

The watchdog will generate a low watermark interrupt in advance of actually resetting the chip. This low watermark interrupt occurs approximately 1.75 seconds after the timer has been restarted. This interrupt can be used to assist during application debug.

4.14 Sleep Timer

The 16-bit sleep timer is contained in the always-powered digital block. It has the following features:

- Two output compare registers, with interrupts
- Only Compare A Interrupt generates Wake signal
- Further clock divider of 2^N, for N = 0 to 10

The clock source for the sleep timer can be either the 32.768 kHz clock or the calibrated 1kHz clock (see Table 15). After choosing the clock source, the frequency is slowed down with a 2^N prescaler to generate the final timer clock (see Table 16). Legal values for *N* are 0 to 10. The slowest rate the sleep timer counter wraps is $2^{16} * 2^{10}$ / 1kHz ≈ 67109 sec. ≈ about 1118.48 min. ≈ 18.6 hrs.

Table 15. Sleep Timer Clock Source Selection

CLK_SEL	Clock Source
0	Calibrated 1kHz clock
1	32.768kHz clock

Table 16. Sleep Timer Clock Source Prescaling

CLK_DIV[3:0]	Clock Source Prescale Factor
N = 0..10	2^N
N = 11..15	2^{10}

The EmberZNet PRO software allows the application to define the clock source and prescaler value. Therefore, a programmable sleep/wake duty cycle can be configured according to the application requirements.

4.15 Power Management

The EM250 supports three different power modes: processor ACTIVE, processor IDLE, and DEEP SLEEP.

The IDLE power mode stops code execution of the XAP2b until any interrupt occurs or an external SIF wakeup command is seen. All peripherals of the EM250 including the radio continue to operate normally.

The DEEP SLEEP power mode powers off most of the EM250 but leaves the critical chip functions, such as the GPIO pads and RAM powered by the High Voltage Supply (VDD_PADS). The EM250 can be woken by configuring the sleep timer to generate an interrupt after a period of time, using an external interrupt, or with the SIF interface. Activity on a serial interface may also be configured to wake the EM250, though actual reception of data is not re-enabled until the EM250 has finished waking up. Depending on the speed of the serial data, it is possible to finish waking up in the middle of a byte. Care must be taken to reset the serial interface between bytes and discard any garbage data before the rest. Another condition for wakeup is general activity on GPIO pins. The GPIO activity monitoring is described in section 5.1.

When in DEEP SLEEP, the internal regulator is disabled and VREG_OUT is turned off. All GPIO output signals are maintained in a frozen state. Additionally, the state of all registers in the powered-down low-voltage domain of the EM250 is lost. Register settings for application peripherals should be preserved by the application as desired. The operation of DEEP SLEEP is controlled by EmberZNet PRO APIs which automatically preserve the state of necessary system peripherals. The internal XAP2b CPU registers are automatically saved and restored to RAM by hardware when entering and leaving the DEEP SLEEP mode, allowing code execution to continue from where it left off. The event that caused the wakeup and any additional events that occurred while waking up are reported to the application via the EmberZNet PRO APIs. Upon waking from DEEP SLEEP, the internal regulator is re-enabled.

5 Functional Description—Application Modules

In Application Mode, access to privileged areas is blocked while access to application-specific modules such as GPIO, Serial Controllers (SC1 and SC2), General Purpose Timers, ADC, and Event Manager are enabled.

5.1 GPIO

The EM250 has 17 multi-purpose GPIO pins that can be configured in a variety of ways. All pins have the following programmable features:

- Selectable as input, output, or bi-directional.
- Output can be totem-pole, used as open drain or open source output for wired-OR applications.
- Can have internal pull-up or pull-down.

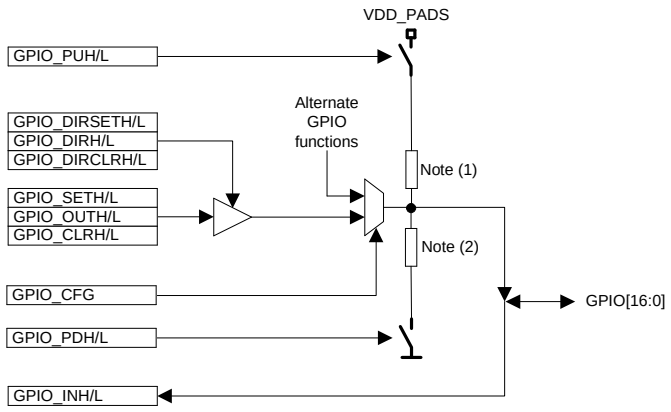
The information flow between the GPIO pin and its source are controlled by separate GPIO Data registers. The GPIO_INH and GPIO_INL registers report the input level of the GPIO pins. The GPIO_DIRH and GPIO_DIRL registers enable the output signals for the GPIO Pins. The GPIO_PUH and GPIO_PUL registers enable pull-up resistors while GPIO_PDH and GPIO_PDL registers enable pull-down resistors on the GPIO Pins. The GPIO_OUTH and GPIO_OUTL control the output level.

To configure a GPIO as an open source output, the GPIO_OUT register should be set to 0, the GPIO_PD register bit should be enabled, and the GPIO_DIR register can be used for the data. To configure a GPIO as an open drain, the GPIO_OUT register should be set to 0, the GPIO_PU register bit should be enabled, and the GPIO_DIR register can be used for the data.

Instead of changing the entire contents to the OUT/DIR registers with one write access, a limited change can be applied. Writing to the GPIO_SETH/L or GPIO_DIRSETH/L register changes individual register bits from 0 to 1, while data bits that are already 1 are maintained. Writing to the GPIO_CLRH/L or GPIO_DIRCLRH/L register changes individual register bits from 1 to 0, while data bits that are already 0 are maintained.

Note that the value read from GPIO_OUTH/L, GPIO_SETH/L, and GPIO_CLRH/L registers may not reflect the current pin state. To observe the pin state, the GPIO_INH/L registers should be read.

All registers controlling the GPIO pin definitions are unaffected by power cycling the main core voltage (VDD_CORE).



Note (1) : Pull-up resistor is always disabled for Alternate GPIO functions ADC0, ADC1, ADC2, ADC3, VREF_OUT, PTI_EN and PTI_DATA.

Note (2) : Pull-down resistor is always disabled for Alternate GPIO functions VREF_OUT, PTI_EN and PTI_DATA.

Figure 5. GPIO Control Logic

The GPIO_DBG register must always remain set to zero. The GPIO_CFG register controls the GPIO signal routing for alternate GPIO functions as listed in Table 17. Refer to Table 1 for individual pin alternate functions.

Table 18 defines the alternate functions routed to the GPIO. To allow more flexibility, the timer signals can come from alternative sources (e.g., TIM1IA.1, TIM1IA.2, TIM1IA.3), depending on what serial controller functions are used.

The Always Connected input functions labeled IRQA, IRQB, IRQC, and IRQD refer to the external interrupts. GPIO8, GPIO14, GPIO15, and GPIO16 are the only pins designed to operate as external interrupts (IRQs). These pins offer individual filtering options, triggering options, and interrupt configurations. The minimum width needed to latch an unfiltered external interrupt in both level and edge triggered mode is 80ns. With the filter engaged via the GPIO_INTFILT bit, the minimum width needed is 450ns. Other alternate functions such as timer input captures are capable of generating an interrupt based upon external signals, but these other alternate functions do not contain the flexibility offered on the four external interrupts (IRQs).

When the core is powered down, peripherals stop driving correct output signals. To maintain correct output signals, the system software will ensure that the GPIO output signals are frozen before going into deep sleep.

Note: Enabling alternate functions do not overwrite the pull-up and pull-down configurations, but the alternate function outputs are all forced to be totem-pole (except I^2C).

Monitoring circuitry is in place to detect when the logic state of GPIO input pins change. The lower 16 GPIO pins that should be monitored can be chosen by software with the GPIO_WAKEL register. The resulting event can be used for waking up from deep sleep as described in section 4.15.

Table 17. GPIO Pin Configurations

GPIO_CFG[15:0]	Mode
0010 0000 0000 0000	DEFAULT
---1 ---- ---- ----	Enable PTI_EN + PTI_DATA
---0 ---1 ---- ----	Enable analog input ADC0
---0 ---0 ---- ----	Enable GPIO4
---0 --1- ---- ----	Enable analog input ADC1
---0 --0- ---- ----	Enable GPIO5
---- -1-- ---- ----	Enable analog input ADC2
---- -0-- ---- ----	Enable GPIO6
--0- 1--- ---- ----	Enable analog input ADC3
--0- 0--- ---- ----	Enable GPIO7
-1-- ---- ---- ----	Enable VREF_OUT
-0-- ---- ---- ----	Enable GPIO8
---- ---- 0000 ----	Enable + CAP2-0 + CAP1-0 mode+GPIO[12,11,10,9,3,2,1,0]
---- ---- 0001 ----	Enable SC1-2 + SC2-2 + CAP2-0 + CAP1-0 mode+GPIO[12,11, 3, 0]
---- ---- 0010 ----	Enable SC1-4A + SC2-4S + CAP2-2 + CAP1-2h mode
---- ---- 0011 ----	Enable SC1-3M + SC2-3M + CAP2-2 + CAP1-2 mode+GPIO[12, 3]
---- ---- 0100 ----	Enable SC2-2 + CAP2-0 + CAP1-0 mode+GPIO[12,11,10,9,3, 0]
---- ---- 0101 ----	Enable SC1-2 + SC2-4S + CAP2-0 + CAP1-2h mode+GPIO[12,11]
---- ---- 0110 ----	Enable SC1-4A + SC2-3M + CAP2-2 + CAP1-2 mode+GPIO[3]
---- ---- 0111 ----	Enable SC1-3M + CAP2-1 + CAP1-0 mode+GPIO[12 3,2,1,0]
---- ---- 1000 ----	Enable SC2-4S + CAP2-0 + CAP1-1h mode+GPIO[12,11,10,9]
---- ---- 1001 ----	Enable SC1-2 + SC2-3M + CAP2-0 + CAP1-2 mode+GPIO[12,11, 3]
---- ---- 1010 ----	Enable SC1-4A + CAP2-1 + CAP1-0 mode+GPIO[3,2,1,0]
---- ---- 1011 ----	Enable SC1-3M + SC2-2 + CAP2-2 + CAP1-0 mode+GPIO[12 3, 0]
---- ---- 1100 ----	Enable SC2-3M + CAP2-0 + CAP1-1 mode+GPIO[12,11,10,9,3]
---- ---- 1101 ----	Enable SC1-2 + CAP2-0 + CAP1-0 mode+GPIO[12,11, 3,2,1,0]
---- ---- 1110 ----	Enable SC1-4A + SC2-2 + CAP2-2 + CAP1-0 mode+GPIO[3, 0]
---- ---- 1111 ----	Enable SC1-3M + SC2-4S + CAP2-2 + CAP1-2h mode+GPIO[12]
---- ---- ---- ---1	Enable TMR2OA
---- ---- ---- ---0	Enable GPIO13
---- ---- ---- --1-	Enable TMR2OB
---- ---- ---- --0-	Enable GPIO14
---- ---- ---- -1--	Enable TMR1OA
---- ---- ---- -0--	Enable GPIO15
---- ---- ---- 1---	Enable TMR1OB
---- ---- ---- 0---	Enable GPIO16

Table 18. GPIO Pin Functions

GPIO Pin	Always Connected Input Functions	Timer Functions	Serial Digital Functions	Analog Function	Output Current Drive
0	IO	TMR1IA.1 (when CAP1-0 mode)	MOSI		Standard
1	IO	TMR2IA.2 (when CAP2-1 mode)	MISO / SDA		Standard
2	IO	TMR2IB.2 (when CAP2-1 mode)	MSCLK / SCL		Standard
3	IO	TMR1IB.1 (when CAP1-0 mode)	nSSEL (input)		Standard
4	IO		PTI_EN	ADC0 input	Standard
5	IO		PTI_DATA	ADC1 input	Standard
6	IO	TMR2CLK, TMR1ENMSK		ADC2 input	Standard
7	IO			ADC3 input	Standard
8	IO / IRQA	TMR1CLK, TMR2ENMSK		VREF_OUT	Standard
9	IO	TMR1IA.2 (when CAP1-1 or CAP1-1h mode)	TXD / MO / MSDA		Standard
10	IO	TMR1IB.2 (when CAP1-1 mode)	RXD / MI / MSCL		Standard
11	IO	TMR2IA.1 (when CAP2-0 mode)	nCTS / MCLK		Standard
12	IO	TMR2IB.1 (when CAP2-0 mode)	nRTS		Standard
13	IO	TMR2OA TMR1IA.3 (when CAP1-2h or CAP1-2 mode)			High
14	IO / IRQB	TMR2OB TMR1IB.3 (when CAP1-2 mode)			High
15	IO / IRQC	TMR1OA TMR2IA.3 (when CAP2-2 mode)			High
16	IO / IRQD	TMR1OB TMR2IB.3 (when CAP2-2 mode)			High

5.1.1 Registers

GPIO_CFG [0x4712]

15	14	13	12	11	10	9	8
0-R	0-RW	1-RW	0-RW	0-RW	0-RW	0-RW	0-RW
0	GPIO_CFG						
GPIO_CFG							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

GPIO_CFG [14:0] GPIO configuration modes. Refer to Table 1 and Table 17 for mode settings.

GPIO_INH [0x4700]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	GPIO_INH
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

GPIO_INH [0] Read the input level of GPIO[16] pin.

GPIO_INL [0x4702]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
GPIO_INL							
GPIO_INL							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

GPIO_INL [15:0] Read the input level of GPIO[15:0] pins.

GPIO_OUTH [0x4704]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	GPIO_ OUTH
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
7	6	5	4	3	2	1	0

GPIO_OUTH [0] Write the output level of GPIO[16] pin. The value read may not match the actual value on the pin.

GPIO_OUTL [0x4706]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
GPIO_OUTL							
GPIO_OUTL							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

GPIO_OUTL [15:0] Write the output level of GPIO[15:0] pins. The value read may not match the actual value on the pin.

GPIO_SETH [0x4708]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	GPIO_SETH
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-W
7	6	5	4	3	2	1	0

GPIO_SETH [0] Set the output level of GPIO[16] pin. Only writing ones into this register will have an effect. Any bit that has one written to it will cause the corresponding bit in GPIO_OUTH to become 1.

GPIO_SETL [0x470A]

15	14	13	12	11	10	9	8
0-W	0-W	0-W	0-W	0-W	0-W	0-W	0-W
GPIO_SETL							
GPIO_SETL							
0-W	0-W	0-W	0-W	0-W	0-W	0-W	0-W
7	6	5	4	3	2	1	0

GPIO_SETL [15:0] Set the output level of GPIO[15:0] pins. Only writing ones into this register will have an effect. Any bit that has one written to it will cause the corresponding bit in GPIO_OUTL to become 1.

GPIO_CLRH [0x470C]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	GPIO_ CLRH
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-W
7	6	5	4	3	2	1	0

GPIO_CLRH [0] Clear the output level of GPIO[16] pin. Only writing ones into this register will have an effect. Any bit that has one written to it will cause the corresponding bit in GPIO_OUTH to become 0.

GPIO_CLRL [0x470E]

15	14	13	12	11	10	9	8
0-W	0-W	0-W	0-W	0-W	0-W	0-W	0-W
GPIO_CLRL							
GPIO_CLRL							
0-W	0-W	0-W	0-W	0-W	0-W	0-W	0-W
7	6	5	4	3	2	1	0

GPIO_CLRL [15:0] Clear the output level of GPIO[15:0] pins. Only writing ones into this register will have an effect. Any bit that has one written to it will cause the corresponding bit in GPIO_OUTL to become 0.

GPIO_DIRH [0x4714]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	GPIO_DIRH
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
7	6	5	4	3	2	1	0

GPIO_DIRH [0] Enable the output of GPIO[16] pin.

GPIO_DIRL [0x4716]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
GPIO_DIRL							
GPIO_DIRL							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

GPIO_DIRL [15:0] Enable the output of GPIO[15:0] pins.

GPIO_DIRSETH [0x4718]

15 0-R	14 0-R	13 0-R	12 0-R	11 0-R	10 0-R	9 0-R	8 0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	GPIO_ DIRSETH
0-R 7	0-R 6	0-R 5	0-R 4	0-R 3	0-R 2	0-R 1	0-W 0

GPIO_DIRSETH [0] Set the output enable of GPIO[16] pin. Only writing ones into this register will have an effect. Any bit that has one written to it will cause the corresponding bit in GPIO_DIRH to become 1.

GPIO_DIRSETL [0x471A]

15 0-W	14 0-W	13 0-W	12 0-W	11 0-W	10 0-W	9 0-W	8 0-W
GPIO_DIRSETL							
GPIO_DIRSETL							
0-W 7	0-W 6	0-W 5	0-W 4	0-W 3	0-W 2	0-W 1	0-W 0

GPIO_DIRSETL [15:0] Set the output enable of GPIO[15:0] pins. Only writing ones into this register will have an effect. Any bit that has one written to it will cause the corresponding bit in GPIO_DIRL to become 1.

GPIO_DIRCLRH [0x471C]

15 0-R	14 0-R	13 0-R	12 0-R	11 0-R	10 0-R	9 0-R	8 0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	GPIO_ DIRCLRH
0-R 7	0-R 6	0-R 5	0-R 4	0-R 3	0-R 2	0-R 1	0-W 0

GPIO_DIRCLRH [0] Clear the output enable of GPIO[16] pin. Only writing ones into this register will have an effect. Any bit that has one written to it will cause the corresponding bit in GPIO_DIRH to become 0.

GPIO_DIRCLRL [0x471E]

15	14	13	12	11	10	9	8
0-W	0-W	0-W	0-W	0-W	0-W	0-W	0-W
GPIO_DIRCLRL							
GPIO_DIRCLRL							
0-W	0-W	0-W	0-W	0-W	0-W	0-W	0-W
7	6	5	4	3	2	1	0

GPIO_DIRCLRL [15:0] Clear the output enable of GPIO[15:0] pins. Only writing ones into this register will have an effect. Any bit that has one written to it will cause the corresponding bit in `GPIO_DIRL` to become 0.

GPIO_PDH [0x4720]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	GPIO_PDH
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
7	6	5	4	3	2	1	0

GPIO_PDH [0] Set this bit to enable pull-down resistors on GPIO[16] pin.

GPIO_PDL [0x4722]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
GPIO_PDL							
GPIO_PDL							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

GPIO_PDL [15:0] Set this bit to enable pull-down resistors on GPIO[15:0] pins.

GPIO_PUH [0x4724]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	GPIO_PUH
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
7	6	5	4	3	2	1	0

GPIO_PUH [0] Set this bit to enable pull-up resistors on GPIO[16] pin.

GPIO_PUL [0x4726]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
GPIO_PUL							
GPIO_PUL							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

GPIO_PUL [15:0] Set this bit to enable pull-up resistors on GPIO[15:0] pins.

GPIO_WAKEL [0x4728]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
GPIO_WAKEL							
GPIO_WAKEL							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

GPIO_WAKEL [15:0] Setting bits will enable GPIO wakeup monitoring for changing states on GPIO[15:0] pins.

GPIO_INTCFGA [0x4630]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
0	0	0	0	0	0	0	GPIO_INTFILT
GPIO_INTMOD			0	0	0	0	0
0-RW	0-RW	0-RW	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

GPIO_INTFILT [8] Set this bit to enable GPIO IRQA filter.

GPIO_INTMOD [7:5] GPIO IRQA input edge triggering selection: 0 = disabled; 1 = rising; 2 = falling; 3 = both edges; 4 = active high triggered; 5 = active low trigger; 6,7 = reserved.

GPIO_INTCFGB [0x4632]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
0	0	0	0	0	0	0	GPIO_INTFILT
GPIO_INTMOD			0	0	0	0	0
0-RW	0-RW	0-RW	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

GPIO_INTFILT [8] Set this bit to enable GPIO IRQB filter

GPIO_INTMOD [7:5] GPIO IRQB input edge triggering selection: 0 = disabled; 1 = rising; 2 = falling; 3 = both edges; 4 = active high triggered; 5 = active low trigger; 6,7 = reserved.

GPIO_INTCFGC [0x4634]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
0	0	0	0	0	0	0	GPIO_ INTFILT
GPIO_INTMOD			0	0	0	0	0
0-RW	0-RW	0-RW	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

GPIO_INTFILT [8] Set this bit to enable GPIO IRQC filter.

GPIO_INTMOD [7:5] GPIO IRQC input edge triggering selection: 0 = disabled; 1 = rising; 2 = falling; 3 = both edges; 4 = active high triggered; 5 = active low trigger; 6,7 = reserved.

GPIO_INTCFGD [0x4636]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
0	0	0	0	0	0	0	GPIO_ INTFILT
GPIO_INTMOD			0	0	0	0	0
0-RW	0-RW	0-RW	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

GPIO_INTFILT [8] Set this bit to enable GPIO IRQD filter.

GPIO_INTMOD [7:5] GPIO IRQD input edge triggering selection: 0 = disabled; 1 = rising; 2 = falling; 3 = both edges; 4 = active high triggered; 5 = active low trigger; 6,7 = reserved.

INT_GPIOCFG [0x4628]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	INT_GPIOD	INT_GPIOC	INT_GPIOB	INT_GPIOA
0-R	0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

INT_GPIOD [3] GPIO IRQD interrupt enable.

INT_GPIOC [2] GPIO IRQC interrupt enable.

INT_GPIOB [1] GPIO IRQB interrupt enable.

INT_GPIOA [0] GPIO IRQA interrupt enable.

INT_GPIOFLAG [0x4610]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	INT_GPIOD	INT_GPIOC	INT_GPIOB	INT_GPIOA
0-R	0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

INT_GPIOD [3] GPIO IRQD interrupt pending.

INT_GPIOC [2] GPIO IRQC interrupt pending.

INT_GPIOB [1] GPIO IRQB interrupt pending.

INT_GPIOA [0] GPIO IRQA interrupt pending.

GPIO_DBG [0x4710]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	GPIO_DBG	
0-R	0-R	0-R	0-R	0-R	0-R	0-RW	0-RW
7	6	5	4	3	2	1	0

GPIO_DBG [1:0] This register must remain zero.

5.2 Serial Controller SC1

The EM250 SC1 module provides asynchronous (UART) or synchronous (SPI or I^2C) serial communications. Figure 6 is a block diagram of the SC1 module.

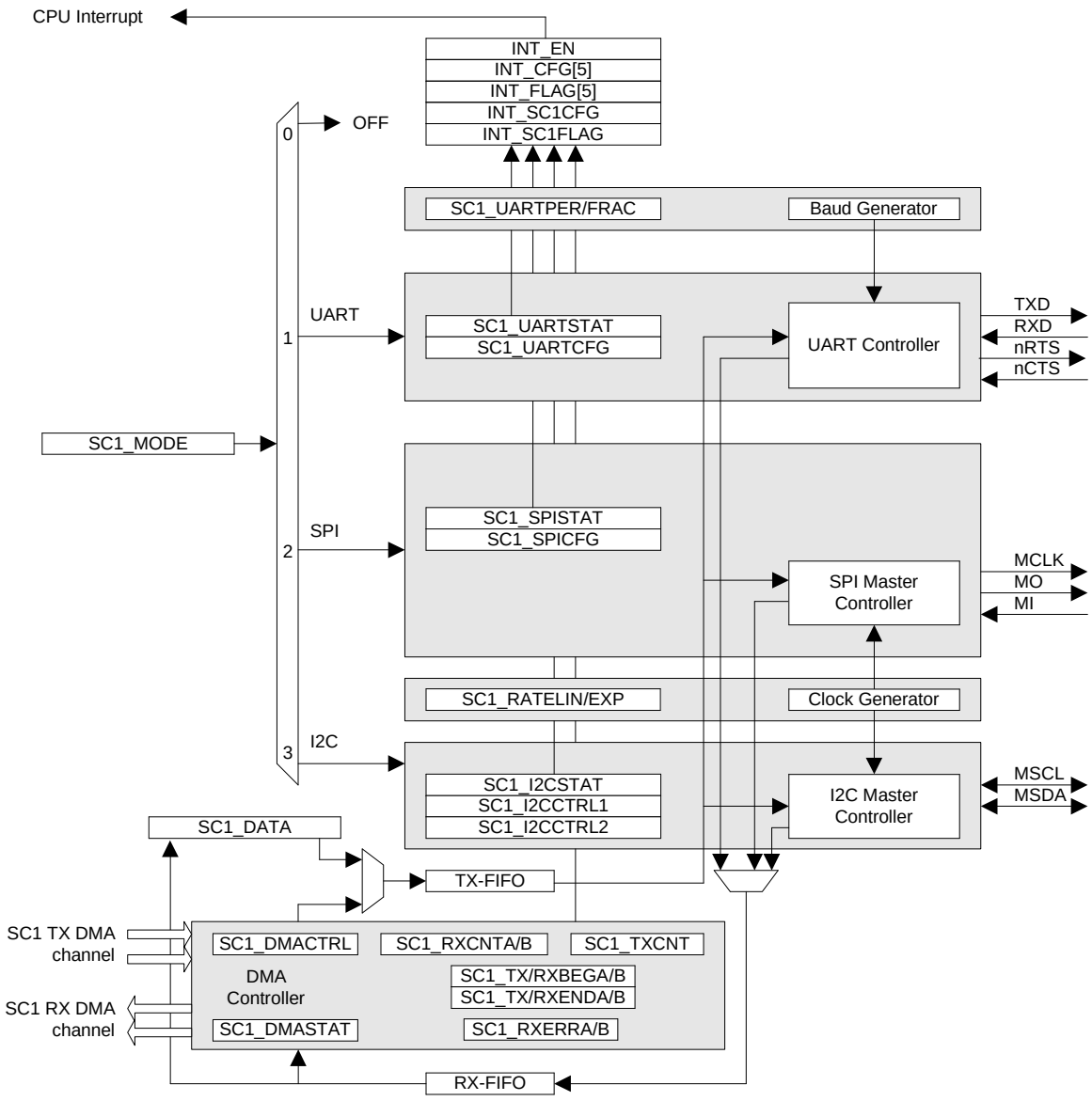


Figure 6. SC1 Block Diagram

The full-duplex interface of the SC1 module can be configured into one of these three communication modes, but it cannot run them simultaneously. To reduce the interrupt service requirements of the CPU, the SC1 module contains buffered data management schemes for the three modes. A dedicated, buffered DMA controller is available to the SPI and UART controllers while a FIFO is available to all three modes. In addition, a SC1 data register allows the software application direct access to the SC1 data within all three modes. Finally, the SC1 routes the interface signals to GPIO pins. These are shared with other functions and are controlled by the GPIO_CFG register. For selecting alternate pin functions, refer to Table 17 and Table 18.

5.2.1 UART Mode

The SC1 UART controller is enabled with SC1_MODE set to 1.

The UART mode contains the following features:

- Baud rate (300bps up to 921kbps)
- Data bits (7 or 8)
- Parity bits (none, odd, or even)
- Stop bits (1 or 2)

The following signals can be made available on GPIO pins:

- TXD
- RXD
- nRTS (optional)
- nCTS (optional)

The SC1 UART module obtains its reference baud-rate clock from a programmable baud generator. Baud rates are set by a clock division ratio from the 24MHz clock:

rate = 24MHz / (2 * (N + (0.5 * F)))

The integer portion, N, is written to the SC1_UARTPER register and the fractional remainder, F, to the SC1_UARTFRAC register. Table 19 lists the supported baud rates with associated baud rate error. The minimum allowable setting for SC1_UARTPER is 8.

Table 19. UART Baud Rates

Baud Rate (bps)	SC1_UARTPER	SC1_UARTFRAC	Baud Rate Error (%)
300	40000	0	0
4800	2500	0	0
9600	1250	0	0
19200	625	0	0
38400	312	1	0
57600	208	1	- 0.08
115200	104	0	0.16
460800	26	0	0.16
921600	13	0	0.16

The UART module supports various frame formats depending upon the number of data bits (SC1_UART8BIT), the number of stop bits (SC1_UART2STP), and the parity (SC1_UARTPAR plus SC1_UARTODD). The register bits SC1_UART8BIT, SC1_UART2STP, SC1_UARTPAR, and SC1_UARTODD are defined within the SC1_UARTCFG register. In addition, the UART module supports flow control by setting SC1_UARTFLOW, SC1_UARTAUTO, and SC1_UARTRTS in the SC1_UARTCFG register (see Table 20).

Table 20. Configuration Table for the UART Module

SC1_UARTCFG					
SC1_MODE	SC1_UARTFLOW	SC1_UARTAUTO	SC1_UARTRTS	GPIO_CFG[7:4]	GPIO Pin Function
1	0	-	-	SC1-2 mode	TXD/RXD output/input
1	1	-	-	SC1-2 mode	Illegal
1	1	0	0/1	SC1-4A mode	TXD/RXD/nCTS output/input/input nRTS output = ON/OFF
1	1	1	-	SC1-4A mode	TXD/RXD/nCTS output/input/input nRTS output = ON if 2 or more bytes will fit in receive buffer
1	0	1	-	SC1-4A mode	Reserved
1	0	0	-	SC1-4A mode	Illegal
1	-	-	-	SC1-3M mode	Illegal

Characters transmitted and received are passed through transmit and receive FIFOs. The transmit and receive FIFOs are 4 bytes deep. The FIFOs are accessed under software control by accessing the SC1_DATA data register or under hardware control by the SC1 DMA.

When a transmit character is written to the (empty) transmit FIFO, the register bit SC1_UARTTXIDLE in the SC1_UARTSTAT register clears to indicate that not all characters are transmitted yet. Further transmit characters can be written to the transmit FIFO until it is full, which causes the register bit SC1_UARTTXFREE in the SC1_UARTSTAT register to clear. After shifting one transmit character to the TXD pin, space for one transmit character becomes available in the transmit FIFO. This causes the register bit SC1_UARTTXFREE in the SC1_UARTSTAT register to get set. After all characters are shifted out, the transmit FIFO empties, which causes the register bit SC1_UARTTXIDLE in the SC1_UARTSTAT register to get set.

A received character is stored with its parity and frame error status in the receive FIFO. The register bit SC1_UARTRXVAL in the SC1_UARTSTAT register is set to indicate that not all received characters are read out from the receive FIFO. The error status of a received byte is available with the register bits SC1_UARTPARERR and SC1_UARTFRMERR in the SC1_UARTSTAT register. When the DMA controller is transferring the data from the receive FIFO to a memory buffer, it checks the stored parity and frame error status flags. When an error is flagged, the SC1_RXERRA/B register is updated, marking the offset to the first received character with parity or frame error.

When the 4-character receive FIFO contains 3 characters, flow control needs to be used to avoid an overflow event. One method is to use software handshaking by transmitting reserved XON/XOFF characters which are interpreted by the transmitting terminal to pause further transmissions (to the receive FIFO). Another method is to use hardware handshaking using XOFF assertion through the nRTS signal.

There are two schemes available to assert the nRTS signal. The first scheme is to initiate nRTS assertion with software by setting the register bit SC1_UARTRTS in the SC1_UARTCFG register. The second scheme is to assert nRTS automatically depending on the fill state of the receive FIFO. This is enabled with the register bit SC1_UARTAUTO in the SC1_UARTCFG register.

The UART also contains overrun protection for both the FIFO and DMA options. If the transmitting terminal continues to transmit characters to the receive FIFO, only 4 characters are stored in the FIFO. Additional characters are dropped, and the register bit SC1_UARTRXOVF in the SC1_UARTSTAT register is set. Should this

receive overrun occur during DMA operation, the SC1_RXERRA/B registers mark the error-offset. The RX FIFO hardware generates the INT_SCRXOVF interrupt, but the DMA register will not indicate the error condition until the RX FIFO is drained. Once the DMA marks a RX error, there are two conditions that will clear the error indication: setting the appropriate SC_TX/RXDMARST bit in the SC1_DMACTRL register, or loading the appropriate DMA buffer after it has unloaded.

Interrupts are generated on the following events:

- Transmit FIFO empty and last character shifted out (0 to 1 transition of SC1_UARTTXIDLE)
- Transmit FIFO changed from full to not full (0 to 1 transition of SC1_UARTTXFREE)
- Receive FIFO changed from empty to not empty (0 to 1 transition of SC1_UARTRXVAL)
- Transmit DMA buffer A/B complete (1 to 0 transition of SC_TXACTA/B)
- Receive DMA buffer A/B complete (1 to 0 transition of SC_RXACTA/B)
- Character received with Parity error
- Character received with Frame error
- Received and lost character while receive FIFO was full (Receive overrun error)

To generate interrupts to the CPU, the interrupt masks in the INT_SC1CFG and INT_CFG registers must be enabled.

5.2.2 SPI Master Mode

The SPI mode of the SC1 is master mode only. It has a fixed word length of 8 bits. The SC1 SPI controller is enabled with SC1_MODE set to 2 and register bit SC_SPIMST set in the SC1_SPICFG register.

The SPI mode has the following features:

- Full duplex operation
- Programmable clock frequency (12MHz max.)
- Programmable clock polarity and clock phase
- Selectable data shift direction (either LSB or MSB first)

The following signals can be made available on the GPIO pins:

- MO (master out)
- MI (master in)
- MCLK (serial clock)

The SC1 SPI module obtains its reference clock from a programmable clock generator. Clock rates are set by a clock division ratio from the 24MHz clock:

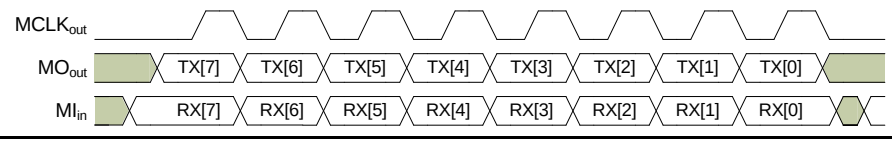
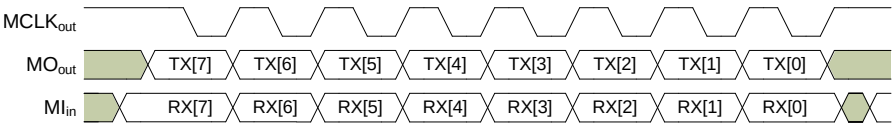
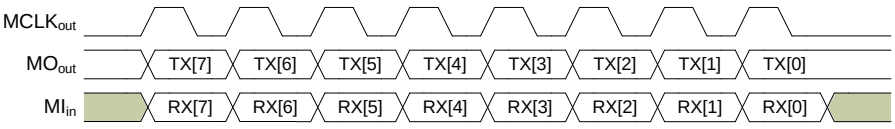
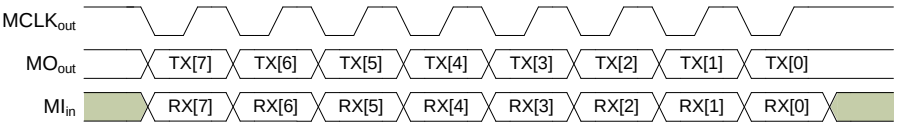
$$\text{rate} = 24\text{MHz} / (2 * (\text{LIN} + 1) * 2^{\text{EXP}})$$

EXP is written to the SC1_RATEEXP register and LIN to the SC1_RATELIN register. Since the range for both values is 0 to 15, the fastest data rate is 12Mbps and the slowest rate is 22.9bps.

The SC1 SPI master supports various frame formats depending upon the clock polarity (SC_SPIPOL), clock phase (SC_SPIPHA), and direction of data (SC_SPIORD) (see Table 21). The register bits SC_SPIPOL, SC_SPIPHA, and SC_SPIORD are defined within the SC1_SPICFG register.

Note: Switching the SPI configuration from SC_SPIPOL=1 to SC_SPIPOL=0 without subsequently setting SC1_MODE=0 and reinitializing the SPI will cause an extra byte (0xFE) to be transmitted immediately before the first intended byte.

Table 21. SC1 SPI Master Frame Format

	SC1_SPICFG					
SC1_MODE	SC_SPIMST	SC_SPIORD	SC_SPIPHA	SC_SPIPOL	GPIO_CFG[7:4]	Frame Format
2	1	0	0	0	SC1-3M mode	
2	1	0	0	1	SC1-3M mode	
2	1	0	1	0	SC1-3M mode	
2	1	0	1	1	SC1-3M mode	
2	1	1	-	-	SC1-3M mode	Same as above except LSB first instead of MSB first
2	1	-	-	-	SC1-2 mode	Illegal
2	1	-	-	-	SC1-4A mode	Illegal

Serialized SC1 SPI transmit data is driven to the output pin MO. SC1 SPI master data is received from the input pin MI. To generate slave select signals to SPI slave devices, other GPIO pins have to be used and their assertion must be controlled by software.

Characters transmitted and received are passed through transmit and receive FIFOs. The transmit and receive FIFOs are 4 bytes deep. These FIFOs are accessed under software control by accessing the SC1_DATA data register or under hardware control using a DMA controller.

When a transmit character is written to the (empty) transmit FIFO, the register bit SC_SPITXIDLE in the SC1_SPISTAT register clears and indicates that not all characters are transmitted yet. Further transmit characters can be written to the transmit FIFO until it is full, which causes the register bit SC_SPITXFREE in the SC1_SPISTAT register to clear. After shifting one transmit character to the MO pin, space for one transmit character becomes available in the transmit FIFO. This causes the register bit SC_SPITXFREE in the SC1_SPISTAT register to get set. After all characters are shifted out, the transmit FIFO empties, which causes the register bit SC_SPITXIDLE in the SC1_SPISTAT register to get set also.

Any character received is stored in the (empty) receive FIFO. The register bit SC_SPIRXVAL in the SC1_SPISTAT register is set to indicate that not all received characters are read out from receive FIFO. If software or DMA is not reading from the receive FIFO, the receive FIFO will store up to 4 characters. Any

further reception is dropped and the register bit SC_SPIRXOVF in the SC1_SPISTAT register is set. The RX FIFO hardware generates the INT_SCRXOVF interrupt, but the DMA register will not indicate the error condition until the RX FIFO is drained. Once the DMA marks a RX error, there are two conditions that will clear the error indication: setting the appropriate SC_TX/RXDMARST bit in the SC1_DMACTRL register, or loading the appropriate DMA buffer after it has unloaded.

Receiving a character always requires transmitting a character. In a case when a long stream of receive characters is expected, a long sequence of (dummy) transmit characters must be generated. To avoid software or transmit DMA initiating these transfers (and consuming unnecessary bandwidth), the SPI serializer can be instructed to retransmit the last transmitted character, or to transmit a busy token (0xFF), which is determined by the register bit SC_SPIRPT in the SC1_SPICFG register. This functionality can only be enabled (or disabled) when the transmit FIFO is empty and the transmit serializer is idle, as indicated by a cleared SC_SPITXIDLE register bit in the SC1_SPISTAT register.

Every time an automatic character transmission is started, a transmit underrun is detected (as there is no data in transmit FIFO), and the register bit INT_SCTXUND in the INT_SC1FLAG register is set. Note that after disabling the automatic character transmission, the reception of new characters stops and the receive FIFO holds characters just received.

Note: The event Receive DMA complete does not automatically mean receive FIFO empty.

Interrupts are generated on the following events:

- Transmit FIFO empty and last character shifted out (0 to 1 transition of SC_SPITXIDLE)
- Transmit FIFO changed from full to not full (0 to 1 transition of SC_SPITXFREE)
- Receive FIFO changed from empty to not empty (0 to 1 transition of SC_SPIRXVAL)
- Transmit DMA buffer A/B complete (1 to 0 transition of SC_TXACTA/B)
- Receive DMA buffer A/B complete (1 to 0 transition of SC_RXACTA/B)
- Received and lost character while receive FIFO was full (Receive overrun error)
- Transmitted character while transmit FIFO was empty (Transmit underrun error)

To generate interrupts to the CPU, the interrupt masks in the INT_SC1CFG and INT_CFG registers must be enabled.

5.2.3 I^2C Master Mode

The SC1 I^2C controller is only available in master mode. The SC1 I^2C controller is enabled with SC1_MODE set to 3. The I^2C Master controller supports Standard (100kbps) and Fast (400kbps) I^2C modes. Address arbitration is not implemented, so multiple master applications are not supported. The I^2C signals are pure open-collector signals, and external pull-up resistors are required.

The SC1 I^2C mode has the following features:

- Programmable clock frequency (400kHz max.)
- Supports both 7-bit and 10-bit addressing

The following signals can be made available on the GPIO pins:

- MSDA (serial data)
- MSCL (serial clock)

The I^2C Master controller obtains its reference clock from a programmable clock generator. Clock rates are set by a clock division ratio from the 24MHz clock:

$$\text{Nominal Rate} = 24\text{MHz} / (2 * (\text{LIN} + 1) * 2^{\text{EXP}})$$

EXP is written to the SC1_RATEEXP register and LIN to the SC1_RATELIN register. Table 22 shows the rate settings for Standard I^2C (100kbps) and Fast I^2C (400kbps) operation.

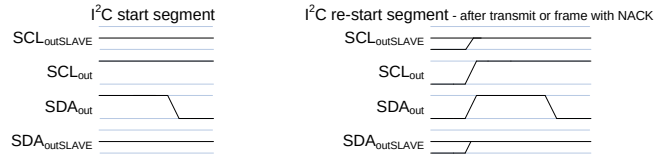
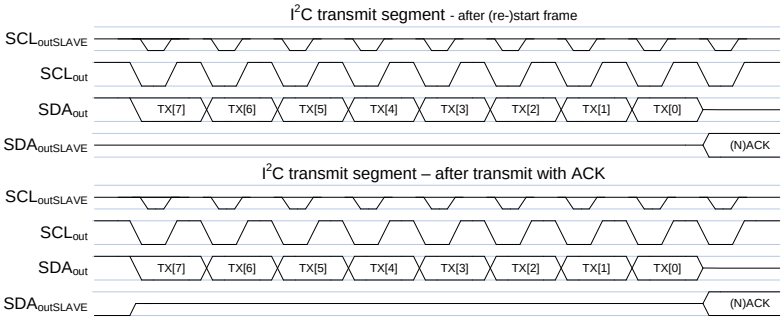
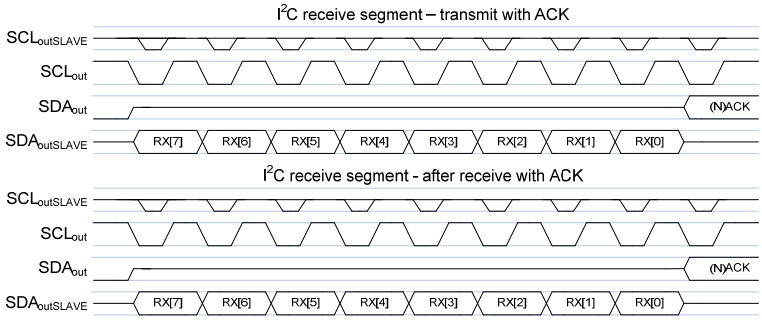
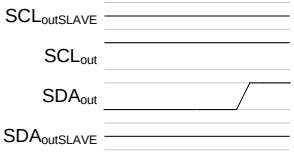
Table 22. I^2C Nominal Rate Programming

Nominal Rate	SC1_RATELIN	SC1_RATEEXP
100kbps	14	3
375kbps	15	1
400kbps	14	1

Note that, at 400kbps, the I^2C specification requires the minimum low period of SCL to be 1.3µs. To be strictly I^2C compliant, the rate needs to be lowered to 375kbps.

The I^2C Master controller supports generation of various frame segments controlled with the register bits SC_I2CSTART, SC_I2CSTOP, SC_I2CSEND, and SC_I2CRECV in the SC1_I2CCTRL1 registers. Table 23 summarizes these frames.

Table 23. SC1 I²C Master Frame Segments

SC1_MODE	SC1_I2CCTRL1				GPIO_CFG[7:4]	Frame Segments
	SC_I2CSTART	SC_I2CSEND	SC_I2CRECV	SC_I2CSTOP		
	1	0	0	0		
3	0	1	0	0	SC1-2 mode	
3	0	0	1	0	SC1-2 mode	
3	0	0	0	1	SC1-2 mode	I²C stop segment - after frame with NACK or stop 
3	0	0	0	0	SC1-2 mode	No pending frame segment
3	1 - - 1	1 1 - -	- 1 1 -	- - 1 1	SC1-2 mode	Illegal
3	-	-	-	-	SC1-4M mode	Illegal
3	-	-	-	-	SC1-4A mode	Illegal

Full I^2C frames have to be constructed under software control by generating individual I^2C segments. All necessary segment transitions are shown in Figure 7. ACK or NACK generation of an I^2C receive frame segment is determined with the register bit SC_I2CACK in the SC1_I2CCTRL2 register.

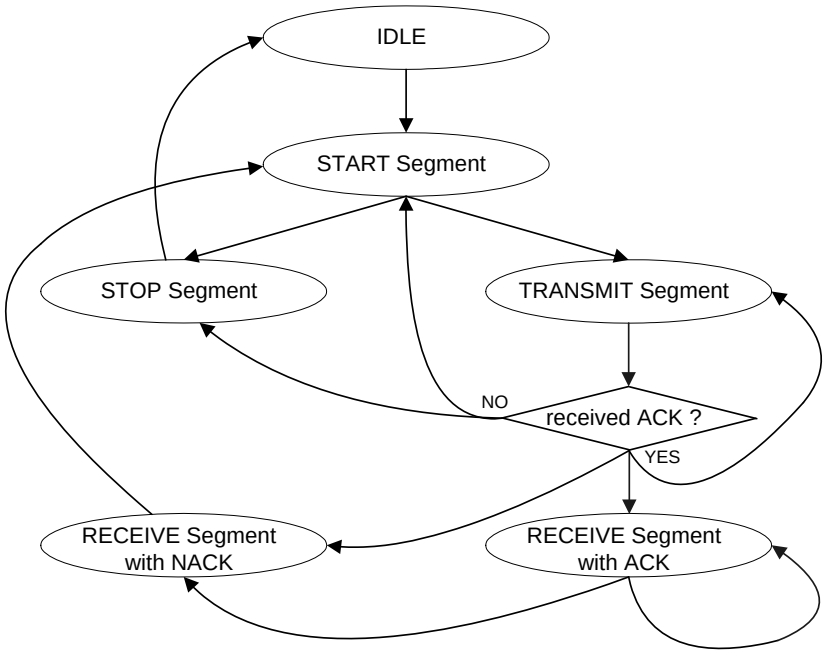


Figure 7. I2C Segment Transitions

Generation of a 7-bit address is accomplished with one transmit segment. The upper 7 bits of the transmitted character contain the 7-bit address. The remaining lower bit contains the command type ("read" or "write").

Generation of a 10-bit address is accomplished with two transmit segments. The upper 5 bits of the first transmit character must be set to 0x1E. The next 2 bits are for the 2 most significant bits of the 10-bit address. The remaining lower bit contains the command type ("read" or "write"). The second transmit segment is for the remaining 8 bits of the 10-bit address.

Characters received and transmitted are passed through receive and transmit FIFOs. The SC1 I^2C master transmit and receive FIFOs are 1-byte deep. These FIFOs are accessed under software control.

(Re)start and stop segments are initiated by setting the register bits SC_I2CSTART or SC_I2CSTOP in the SC1_I2CCTRL1 register followed by waiting until they have cleared. Alternatively, the register bit SC_I2CCMDFIN in the SC1_I2CSTAT can be used for waiting.

To initiate a transmit segment, the data have to be written to the SC1_DATA data register, followed by setting the register bit SC_I2CSEND in the SC1_I2CCTRL1 register, and completed by waiting until it clears. Alternatively, the register bit SC_I2CTXFIN in the SC1_I2CSTAT can be used for waiting.

A receive segment is initiated by setting the register bit SC_I2CRECV in the SC1_I2CCTRL1 register, waiting until it clears, and then reading from the SC1_DATA data register. Alternatively, the register bit SC_I2CRXFIN in the SC1_I2CSTAT can be used for waiting. Now the register bit SC_I2CRXNAK in the SC1_I2CSTAT register indicates if a NACK or ACK was received from an I^2C slave device.

Interrupts are generated on the following events:

- Bus command (SC_I2CSTART/SC_I2CSTOP) completed (0 to 1 transition of SC_I2CCMDFIN)
- Character transmitted and slave device responded with NACK

- Character transmitted (0 to 1 transition of SC_I2CTXFIN)
- Character received (0 to 1 transition of SC_I2CRXFIN)
- Received and lost character while receive FIFO was full (Receive overrun error)
- Transmitted character while transmit FIFO was empty (Transmit underrun error)

To generate interrupts to the CPU, the interrupt masks in the INT_SC1CFG and INT_CFG registers must be enabled.

5.2.4 Registers

SC1_MODE [0x44AA]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	SC1_MODE	
0-R	0-R	0-R	0-R	0-R	0-R	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_MODE [1:0] SC1 Mode: 0 = disabled; 1 = UART mode; 2 = SPI mode; 3 = I2C mode. NOTE: To change between modes, the previous mode must be disabled first.

SC1_DATA [0x449E]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
SC1_DATA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_DATA [7:0] Transmit and receive data register. Writing to this register pushes a byte onto the transmit FIFO. Reading from this register pulls a byte from the receive FIFO.

SC1_UARTPER [0x44B4]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
SC1_UARTPER							
SC1_UARTPER							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_UARTPER [15:0] The baud rate period (N) of the clock rate as seen in the equation: rate = 24MHz / (2 * (N + (0.5*F)))

SC1_UARTFRAC [0x44B6]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	SC1_UARTFRAC
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
7	6	5	4	3	2	1	0

SC1_UARTFRAC [0] The baud rate fractional remainder (F) of the clock rate as derived from the equation: rate = 24MHz / (2 * (N + (0.5*F)))

SC1_UARTCFG [0x44AE]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	SC1_UARTAUTO	SC1_UARTFLOW	SC1_UARTODD	SC1_UARTPAR	SC1_UART2STP	SC1_UART8BIT	SC1_UARTRTS
0-R	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_UARTAUTO [6] Set this bit to enable automatic nRTS assertion by hardware. nRTS will be deasserted when the UART can receive only one more character before the buffer is full. nRTS will be reasserted when the UART can receive more than one character before the buffer is full. The SC1_UARTRTS bit in this register has no effect when this bit is set.

SC1_UARTFLOW [5] Set this bit to enable nRTS/nCTS signals. Clear this bit to disable the signals. When this bit is cleared, the nCTS signal is asserted in hardware to enable the UART transmitter. The GPIO_CFG register should be configured for mode SC1-4A for hardware handshake with nRTS/nCTS and SC1-2 for no handshaking.

SC1_UARTODD [4] Clear this bit for even parity. Set this bit for odd parity.

SC1_UARTPAR [3] Clear this bit for no parity. Set this bit for one parity bit.

SC1_UART2STP [2] Clear this bit for one stop bit. Set this bit for two stop bits

SC1_UART8BIT [1] Clear this bit for seven data bits. Set this bit for eight data bits.

SC1_UARTRTS [0] nRTS is an output signal. When this bit is set, the signal is asserted (== TTL logic 0, GPIO is low, 'XON', RS232 positive voltage), the transmission will proceed. When this bit is cleared, the signal is deasserted (== TTL logic 1, GPIO is high, 'XOFF', RS232 negative voltage), the transmission is inhibited.

SC1_UARTSTAT [0x44A4]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	SC1_ UARTTXIDLE	SC1_ UARTPARERR	SC1_ UARTFRMERR	SC1_ UARTRXOVF	SC1_ UARTTXFREE	SC1_ UARTRXVAL	SC1_UARTCTS
0-R	1-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC1_UARTTXIDLE [6] This bit is set when the transmit FIFO is empty and the transmitter is idle.

SC1_UARTPARERR [5] This bit is set when the receive FIFO has seen a parity error. This bit clears when the data register (SC1_DATA) is read.

SC1_UARTFRMERR [4] This bit is set when the receive FIFO has seen a frame error. This bit clears when the data register (SC1_DATA) is read.

SC1_UARTRXOVF [3] This bit is set when the receive FIFO has been overrun. This bit clears when the data register (SC1_DATA) is read.

SC1_UARTTXFREE [2] This bit is set when the transmit FIFO is ready to accept at least one byte.

SC1_UARTRXVAL [1] This bit is set when the receive FIFO contains at least one byte.

SC1_UARTCTS [0] This bit shows the current state of the nCTS input signal at the nCTS pin (pin 19, GPIO11). When SC1_UARTCTS = 1, the signal is asserted (== TTL logic 0, GPIO is low, 'XON', RS232 positive voltage), the transmission will proceed. When SC1_UARTCTS = 0, the signal is deasserted (== TTL logic 1, GPIO is high, 'XOFF', RS232 negative voltage), transmission is inhibited at the end of the current character. Any characters in the transmit buffer will remain there.

SC1_RATELIN [0x44B0]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	SC1_RATELIN			
0-R	0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

NOTE: Changing the SC1_RATELIN register is only allowed when the serial controller is disabled, when SC1_MODE is 0.

SC1_RATELIN [3:0] The linear component (LIN) of the clock rate as seen in the equation: rate = 24MHz / (2 * (LIN + 1) * (2^EXP))

SC1_RATEEXP [0x44B2]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	SC1_RATEEXP			
0-R	0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

NOTE: Changing the SC1_RATEEXP register is only allowed when the serial controller is disabled, when SC1_MODE is 0.

SC1_RATEEXP [3:0] The exponential component (EXP) of the clock rate as seen in the equation: rate = 24MHz / (2 * (LIN + 1) * (2^EXP))

SC1_SPICFG [0x44AC]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	SC_SPIRXDRV	SC_SPIMST	SC_SPIRPT	SC_SPIORD	SC_SPIPHA	SC_SPIPOL
0-R	0-R	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

NOTE: Changing the SC1_SPICFG register is only allowed when the serial controller is disabled, when SC1_MODE is 0.

SC_SPIRXDRV [5] Receiver-driven mode selection bit (SPI master mode only). Clearing this bit will initiate transactions when transmit data is available. Setting this bit will initiate transactions when the receive buffer (FIFO or DMA) has space.

SC_SPIMST [4] This bit must always be set to put the SPI in master mode (slave mode is not valid).

SC_SPIRPT [3] This bit controls behavior on a transmit buffer underrun condition in slave mode. Clearing this bit will send the BUSY token (0xFF) and setting this bit will repeat the last byte. Changing this bit will only take effect when the transmit FIFO is empty and the transmit serializer is idle.

SC_SPIORD [2] Clearing this bit will result in the Most Significant Bit being transmitted first while setting this bit will result in the Least Significant Bit being transmitted first.

SC_SPIPHA [1] Clock phase configuration is selected with clearing this bit for sampling on the leading (first edge) and setting this bit for sampling on second edge.

SC_SPIPOL [0] Clock polarity configuration is selected with clearing this bit for a rising leading edge and setting this bit for a falling leading edge.

SC1_SPISTAT [0x44A0]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	SC_SPITXIDLE	SC_SPITXFREE	SC_SPIRXVAL	SC_SPIRXOVF
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC_SPITXIDLE [3] This bit is set when the transmit FIFO is empty and the transmitter is idle.

SC_SPITXFREE [2] This bit is set when the transmit FIFO is ready to accept at least one byte.

SC_SPIRXVAL [1] This bit is set when the receive FIFO contains at least one byte.

SC_SPIRXOVF [0] This bit is set when the receive FIFO has been overrun. This bit clears when the data register (SC1_DATA) is read.

SC1_I2CCTRL1 [0x44A6]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	SC_I2CSTOP	SC_I2CSTART	SC_I2CSEND	SC_I2CRECV
0-R	0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC_I2CSTOP [3] Setting this bit sends the STOP command. It auto clears when the command completes.

SC_I2CSTART [2] Setting this bit sends the START or repeated START command. It autoclears when the command completes.

SC_I2CSEND [1] Setting this bit transmits a byte. It autoclears when the command completes.

SC_I2CRECV [0] Setting this bit receives a byte. It autoclears when the command completes.

SC1_I2CCTRL2 [0x44A8]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	SC_I2CACK
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
7	6	5	4	3	2	1	0

SC_I2CACK [0] Setting this bit will signal ACK after a received byte. Clearing this bit will signal NACK after a received byte.

SC1_I2CSTAT [0x44A2]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	SC_I2CCMDFIN	SC_I2CRXFIN	SC_I2CTXFIN	SC_I2CRXNAK
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC_I2CCMDFIN [3] This bit is set when a START or STOP command completes. It autoclears on next bus activity.

SC_I2CRXFIN [2] This bit is set when a byte is received. It autoclears on next bus activity.

SC_I2CTXFIN [1] This bit is set when a byte is transmitted. It autoclears on next bus activity.

SC_I2CRXNAK [0] This bit is set when a NACK is received from the slave. It autoclears on next bus activity.

SC1_DMACTRL [0x4498]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	SC_TXDMARST	SC_RXDMARST	SC_TXLODB	SC_TXLODA	SC_RXLODB	SC_RXLODA
0-R	0-R	0-W	0-W	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC_TXDMARST [5] Setting this bit will reset the transmit DMA. The bit is autocleared.

SC_RXDMARST [4] Setting this bit will reset the receive DMA. This bit is autocleared.

SC_TXLODB [3] Setting this bit loads DMA transmit buffer B addresses and starts the DMA controller processing transmit buffer B. This bit is autocleared when DMA completes. Writing a zero to this bit will not have any effect. Reading this bit as one indicates DMA processing for buffer B is active or pending. Reading this bit as zero indicates DMA processing for buffer B is complete or idle.

SC_TXLODA [2] Setting this bit loads DMA transmit buffer A addresses and starts the DMA controller processing transmit buffer A. This bit is autocleared when DMA completes. Writing a zero to this bit will not have any effect. Reading this bit as one indicates DMA processing for buffer A is active or pending. Reading this bit as zero indicates DMA processing for buffer A is complete or idle.

SC_RXLODB [1] Setting this bit loads DMA receive buffer B addresses and starts the DMA controller processing receive buffer B. This bit is autocleared when DMA completes. Writing a zero to this bit will not have any effect. Reading this bit as one indicates DMA processing for buffer B is active or pending. Reading this bit as zero indicates DMA processing for buffer B is complete or idle.

SC_RXLODA [0] Setting this bit loads DMA receive buffer A addresses and starts the DMA controller processing receive buffer A. This bit is autocleared when DMA completes. Writing a zero to this bit will not have any effect. Reading this bit as one indicates DMA processing for buffer A is active or pending. Reading this bit as zero indicates DMA processing for buffer A is complete or idle.

SC1_DMASTAT [0x4496]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	SC1_RXFRMB	SC1_RXFRMA
SC1_RXPARB	SC1_RXPARA	SC_RXOVFB	SC_RXOVFA	SC_TXACTB	SC_TXACTA	SC_RXACTB	SC_RXACTA
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC1_RXFRMB [9] This bit is set when DMA receive buffer B was passed a frame error from the lower hardware FIFO. This bit is autocleared the next time buffer B is loaded or when the receive DMA is reset.

SC1_RXFRMA [8] This bit is set when DMA receive buffer A was passed a frame error from the lower hardware FIFO. This bit is autocleared the next time buffer A is loaded or when the receive DMA is reset.

SC1_RXPARB [7] This bit is set when DMA receive buffer B was passed a parity error from the lower hardware FIFO. This bit is autocleared the next time buffer B is loaded or when the receive DMA is reset.

SC1_RXPARA [6] This bit is set when DMA receive buffer A was passed a parity error from the lower hardware FIFO. This bit is autocleared the next time buffer A is loaded or when the receive DMA is reset.

SC_RXOVFB [5] This bit is set when DMA receive buffer B was passed an overrun error from the lower hardware FIFO. Neither receive buffers were capable of accepting any more bytes (unloaded), and the FIFO filled up. Buffer B was the next buffer to load, and when it drained the FIFO, the overrun error was passed up to the DMA and flagged with this bit. This bit is autocleared the next time buffer B is loaded or when the receive DMA is reset.

SC_RXOVFA [4] This bit is set when DMA receive buffer A was passed an overrun error from the lower hardware FIFO. Neither receive buffers were capable of accepting any more bytes (unloaded), and the FIFO filled up. Buffer A was the next buffer to load, and when it drained the FIFO the overrun error was passed up to the DMA and flagged with this bit. This bit is autocleared the next time buffer A is loaded or when the receive DMA is reset.

SC_TXACTB [3] This bit is set when DMA transmit buffer B is currently active.

SC_TXACTA [2] This bit is set when DMA transmit buffer A is currently active.

SC_RXACTB [1] This bit is set when DMA receive buffer B is currently active.

SC_RXACTA [0] This bit is set when DMA receive buffer A is currently active.

SC1_RXCNTA [0x4490]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	SC1_RXCNTA				
SC1_RXCNTA							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC1_RXCNTA [12:0] A byte offset (from 0) which points to the location in DMA receive buffer A where the next byte will be written. When the buffer fills and subsequently unloads, this register wraps around and holds the value zero (pointing back to the first location in the buffer).

SC1_RXCNTB [0x4492]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	SC1_RXCNTB				
SC1_RXCNTB							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC1_RXCNTB [12:0] A byte offset (from 0) which points to the location in DMA receive buffer B where the next byte will be written. When the buffer fills and subsequently unloads, this register wraps around and holds the value zero (pointing back to the first location in the buffer).

SC1_TXCNT [0x4494]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	SC1_TXCNT				
SC1_TXCNT							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC1_TXCNT [12:0] A byte offset (from 0) which points to the location in the active (loaded) DMA transmit buffer from which the next byte will be read. When the buffer empties and subsequently unloads, this register wraps around and holds the value zero (pointing back to the first location in the buffer).

SC1_RXBEGA [0x4480]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC1_RXBEGA				
SC1_RXBEGA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_RXBEGA [12:0] DMA Start address (byte aligned) for receive buffer A.

SC1_RXENDA [0x4482]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC1_RXENDA				
SC1_RXENDA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_RXENDA [12:0] DMA End address (byte aligned) for receive buffer A.

SC1_RXBEGB [0x4484]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC1_RXBEGB				
SC1_RXBEGB							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_RXBEGB [12:0] DMA Start address (byte aligned) for receive buffer B.

SC1_RXENDB [0x4486]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC1_RXENDB				
SC1_RXENDB							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_RXENDB [12:0] DMA End address (byte aligned) for receive buffer B.

SC1_TXBEGA [0x4488]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC1_TXBEGA				
SC1_TXBEGA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_TXBEGA [12:0] DMA Start address (byte aligned) for transmit buffer A.

SC1_TXENDA [0x448A]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC1_TXENDA				
SC1_TXENDA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_TXENDA [12:0] DMA End address (byte aligned) for transmit buffer A.

SC1_TXBEGB [0x448C]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC1_TXBEGB				
SC1_TXBEGB							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_TXBEGB [12:0] DMA Start address (byte aligned) for transmit buffer B.

SC1_TXENDB [0x448E]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC1_TXENDB				
SC1_TXENDB							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC1_TXENDB [12:0] DMA End address (byte aligned) for transmit buffer B.

SC1_RXERRA [0x449A]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	SC1_RXERRA				
SC1_RXERRA							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC1_RXERRA [12:0] A byte offset (from 0) which points to the location of the first error in the DMA receive buffer A. If there is no error, it will hold the value zero. This register will not be updated by subsequent errors arriving in the DMA. The next error will only be recorded if the buffer unloads and is reloaded or the receive DMA is reset.

SC1_RXERRB [0x449C]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	SC1_RXERRB				
SC1_RXERRB							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC1_RXERRB [12:0] A byte offset (from 0) which points to the location of the first error in the DMA receive buffer B. If there is no error, it will hold the value zero. This register will not be updated by subsequent errors arriving in the DMA. The next error will only be recorded if the buffer unloads and is reloaded or the receive DMA is reset.

INT_SC1CFG [0x4624]

15 0-R	14 0-RW	13 0-RW	12 0-RW	11 0-RW	10 0-RW	9 0-RW	8 0-RW
0	INT_ SC1PARERR	INT_ SC1FRMERR	INT_ SCTXULDB	INT_ SCTXULDA	INT_ SCRXULDB	INT_ SCRXULDA	INT_SCNAK
INT_ SCCMDFIN	INT_SCTXFIN	INT_SCRXFIN	INT_ SCTXUND	INT_SCRXOVF	INT_ SCTXIDLE	INT_ SCTXFREE	INT_SCRXVAL
0-RW 7	0-RW 6	0-RW 5	0-RW 4	0-RW 3	0-RW 2	0-RW 1	0-RW 0

INT_SC1PARERR [14] Parity error received (UART) interrupt enable.

INT_SC1FRMERR [13] Frame error received (UART) interrupt enable.

INT_SCTXULDB [12] DMA Tx buffer B unloaded interrupt enable.

INT_SCTXULDA [11] DMA Tx buffer A unloaded interrupt enable.

INT_SCRXULDB [10] DMA Rx buffer B unloaded interrupt enable.

INT_SCRXULDA [9] DMA Rx buffer A unloaded interrupt enable.

INT_SCNAK [8] Nack received (I^2C) interrupt enable.

INT_SCCMDFIN [7] START/STOP command complete (I^2C) interrupt enable.

INT_SCTXFIN [6] Transmit operation complete (I^2C) interrupt enable.

INT_SCRXFIN [5] Receive operation complete (I^2C) interrupt enable.

INT_SCTXUND [4] Transmit buffer underrun interrupt enable.

INT_SCRXOVF [3] Receive buffer overrun interrupt enable.

INT_SCTXIDLE [2] Transmitter idle interrupt enable.

INT_SCTXFREE [1] Transmit buffer free interrupt enable.

INT_SCRXVAL [0] Receive buffer has data interrupt enable.

INT_SC1FLAG [0x460C]

15	14	13	12	11	10	9	8
0-R	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
0	INT_ SC1PARERR	INT_ SC1FRMERR	INT_ SCTXULDB	INT_ SCTXULDA	INT_ SCRXULDB	INT_ SCRXULDA	INT_SCNAK
INT_ SCCMDFIN	INT_SCTXFIN	INT_SCRXFIN	INT_ SCTXUND	INT_SCRXOVF	INT_ SCTXIDLE	INT_ SCTXFREE	INT_SCRXVAL
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

INT_SC1PARERR [14] Parity error received (UART) interrupt pending.

INT_SC1FRMERR [13] Frame error received (UART) interrupt pending.

INT_SCTXULDB [12] DMA Tx buffer B unloaded interrupt pending.

INT_SCTXULDA [11] DMA Tx buffer A unloaded interrupt pending.

INT_SCRXULDB [10] DMA Rx buffer B unloaded interrupt pending.

INT_SCRXULDA [9] DMA Rx buffer A unloaded interrupt pending.

INT_SCNAK [8] Nack received (I^2C) interrupt pending.

INT_SCCMDFIN [7] START/STOP command complete (I^2C) interrupt pending.

INT_SCTXFIN [6] Transmit operation complete (I^2C) interrupt pending.

INT_SCRXFIN [5] Receive operation complete (I^2C) interrupt pending.

INT_SCTXUND [4] Transmit buffer underrun interrupt pending.

INT_SCRXOVF [3] Receive buffer overrun interrupt pending.

INT_SCTXIDLE [2] Transmitter idle interrupt pending.

INT_SCTXFREE [1] Transmit buffer free interrupt pending.

INT_SCRXVAL [0] Receive buffer has data interrupt pending.

5.3 Serial Controller SC2

The EM250 SC2 module provides synchronous (SPI or I^2C) serial communications. Figure 8 is a block diagram of the SC2 module.

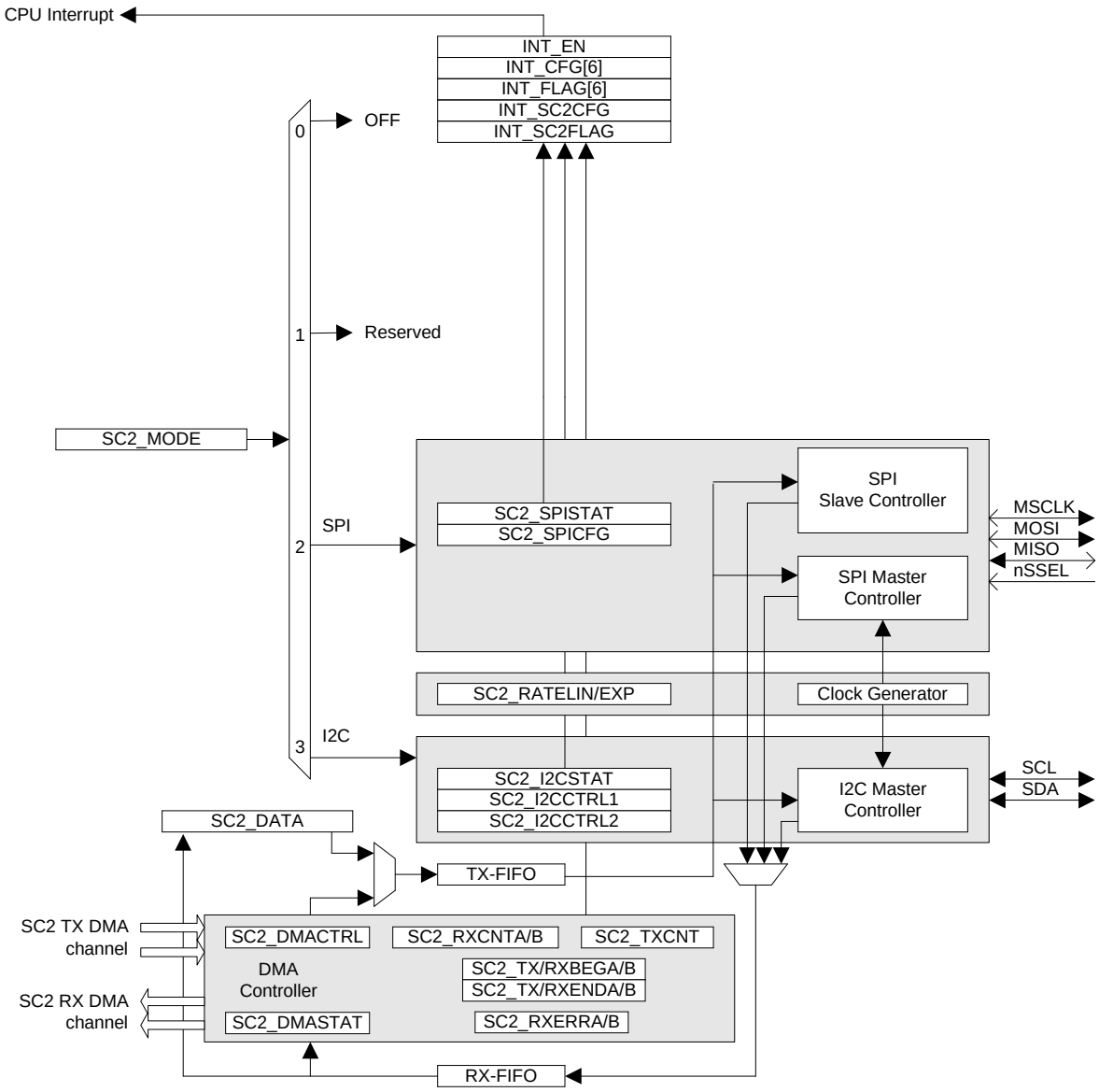


Figure 8. SC2 Block Diagram

The full-duplex interface of the SC2 module can be configured into one of these two communication modes, but it cannot run them simultaneously. To reduce the interrupt service requirements of the CPU, the SC2 module contains buffered data management schemes. A dedicated, buffered DMA controller is available to the SPI while a FIFO is available to both modes. In addition, a SC2 data register allows the software application direct access to the SC2 data. Finally, the SC2 routes the interface signals to GPIO pins. These are shared with other functions and are controlled by the GPIO_CFG register. For selecting alternate pin-functions, refer to Table 17 and Table 18.

5.3.1 SPI Modes

The SPI mode of the SC2 supports both master and slave modes. It has a fixed word length of 8 bits. The SC2 SPI controller is enabled with SC2_MODE set to 2.

The SC2 SPI mode has the following features:

- Master and slave modes
- Full duplex operation
- Programmable master mode clock frequency (12MHz max.)
- Slave mode up to 5MHz bit rate
- Programmable clock polarity and clock phase
- Selectable data shift direction (either LSB or MSB first)
- Optional slave select input

The following signals can be made available on the GPIO pins:

- MOSI (master out/slave in)
- MISO (master in/slave out)
- MSCLK (serial clock)
- nSSEL (slave select—only in slave mode)

5.3.1.1 SPI Master Mode

The SC2 SPI Master controller is enabled with the SC_SPIMST set in the SC2_SPICFG register.

The SC2 SPI module obtains its reference clock from a programmable clock generator. Clock rates are set by a clock division ratio from the 24MHz clock:

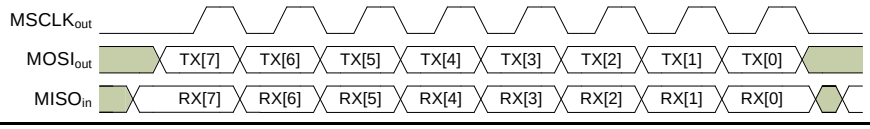
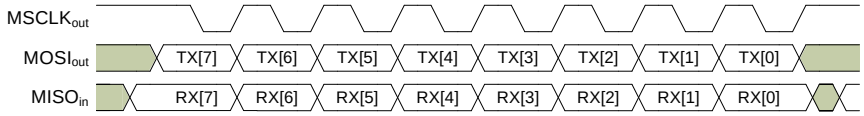
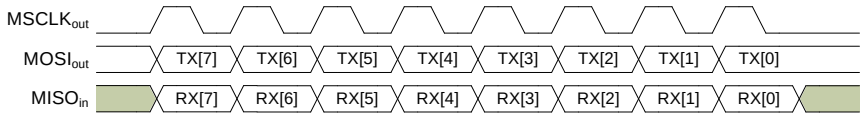
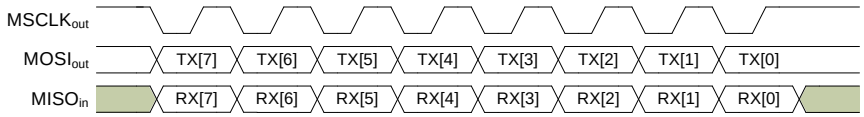
$$\text{rate} = 24\text{MHz} / (2 * (\text{LIN} + 1) * 2^{\text{EXP}})$$

EXP is written to the SC2_RATEEXP register and LIN to the SC2_RATELIN register. Since the range for both values is 0 to 15, the fastest data rate is 12Mbps and the slowest is 22.9bps.

The SC2 SPI Master supports various frame formats depending upon the clock polarity (SC_SPIPOL), clock phase (SC_SPIPHA), and direction of data (SC_SPIORD) (see Table 24). The register bits SC_SPIPOL, SC_SPIPHA, and SC_SPIORD are defined within the SC2_SPICFG register.

Note: Switching the SPI configuration from SC_SPIPOL=1 to SC_SPIPOL=0 without subsequently setting SC2_MODE=0 and reinitializing the SPI will cause an extra byte (0xFE) to be transmitted immediately before the first intended byte.

Table 24. SC2 SPI Master Mode Formats

SC2_MODE	SC2_SPICFG				GPIO_CFG[7:4]	Frame Format
	SC_SPIMST	SC_SPIORD	SC_SPIPHA	SC_SPIPOL		
2	1	0	0	0	SC2-3M mode	
2	1	0	0	1	SC2-3M mode	
2	1	0	1	0	SC2-3M mode	
2	1	0	1	1	SC2-3M mode	
2	1	1	-	-	SC2-3M mode	Same as above except LSB first instead of MSB first
2	1	-	-	-	SC2-4S mode	Illegal
2	1	-	-	-	SC2-2 mode	Illegal

Serialized SC2 SPI transmit data is driven to the output pin MOSI. SC2 SPI master data is received from the input pin MISO. To generate slave select signals to SPI slave devices, other GPIO pins have to be used and their assertion must be controlled by software.

Characters transmitted and received are passed through transmit and receive FIFOs. The transmit and receive FIFOs are 4 bytes deep. These FIFOs are accessed under software control by accessing the SC2_DATA data register or under hardware control using a DMA controller.

When a transmit character is written to the (empty) transmit FIFO, the register bit SC_SPITXIDLE in the SC2_SPISTAT register clears and indicates that not all characters are transmitted yet. Further transmit characters can be written to the transmit FIFO until it is full, which causes the register bit SC_SPITXFREE in the SC2_SPISTAT register to clear. After shifting out one transmit character to the MOSI pin, space for one transmit character becomes available in the transmit FIFO. This causes the register bit SC_SPITXFREE in the SC2_SPISTAT register to get set. After all characters are shifted out, the transmit FIFO empties, which causes the register bit SC_SPITXIDLE in the SC2_SPISTAT register to get set also.

Any character received is stored in the (empty) receive FIFO. The register bit SC_SPIRXVAL in the SC2_SPISTAT register is set to indicate that not all received characters are read out from receive FIFO. If software or DMA is not reading from the receive FIFO, the receive FIFO will store up to 4 characters. Any further reception is dropped and the register bit SC_SPIRXOVF in the SC2_SPISTAT register is set. The RX FIFO hardware generates the INT_SCRXOVF interrupt, but the DMA register will not indicate the error condition until the RX FIFO is drained. Once the DMA marks a RX error, there are two conditions that will clear

the error indication: setting the appropriate SC_TX/RXDMARST bit in the SC2_DMACTRL register, or loading the appropriate DMA buffer after it has unloaded.

Receiving a character always requires transmitting a character. In a case when a long stream of receive characters is expected, a long sequence of (dummy) transmit characters must be generated. To avoid software or transmit DMA initiating these transfers (and consuming unnecessary bandwidth), the SPI serializer can be instructed to retransmit the last transmitted character or to transmit a busy token (0xFF), which is determined by the register bit SC_SPIRPT in the SC2_SPICFG register. This functionality can only be enabled (or disabled) when the transmit FIFO is empty and the transmit serializer is idle, as indicated by a cleared SC_SPITXIDLE register bit in the SC2_SPISTAT register.

Every time an automatic character transmission is started, a transmit underrun is detected (as there is no data in transmit FIFO) and the register bit INT_SCTXUND in the INT_SC2FLAG register is set. Note that after disabling the automatic character transmission, the reception of new characters stops and the receive FIFO holds characters just received.

Note: The event Receive DMA complete does not automatically mean receive FIFO empty.

Interrupts are generated by one of the following events:

- Transmit FIFO empty and last character shifted out (0 to 1 transition of SC_SPITXIDLE)
- Transmit FIFO changed from full to not full (0 to 1 transition of SC_SPITXFREE)
- Receive FIFO changed from empty to not empty (0 to 1 transition of SC_SPIRXVAL)
- Transmit DMA buffer A/B complete (1 to 0 transition of SC_TXACTA/B)
- Receive DMA buffer A/B complete (1 to 0 transition of SC_RXACTA/B)
- Received and lost character while receive FIFO was full (Receive overrun error)
- Transmitted character while transmit FIFO was empty (Transmit underrun error)

To generate interrupts to the CPU, the interrupt masks in the INT_SC2CFG and INT_CFG register must be enabled.

5.3.1.2 SPI Slave Mode

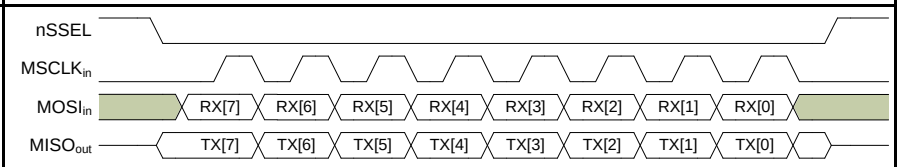
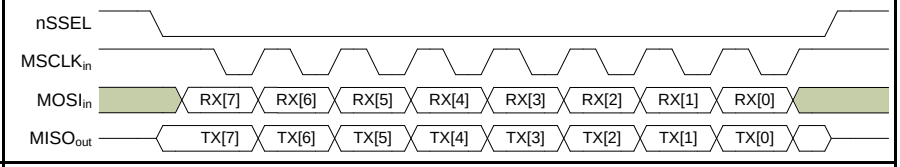
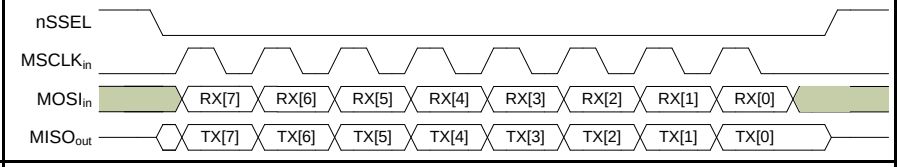
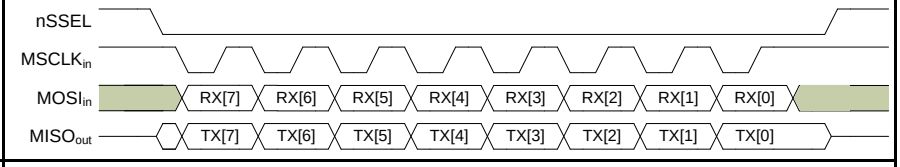
The SC2 SPI Slave controller is enabled with the SC_SPIMST cleared in the SC2_SPICFG register.

The SC2 SPI Slave controller receives its clock from an external SPI master device and supports rates up to 5Mbps.

The SC2 SPI Slave supports various frame formats depending upon the clock polarity (SC_SPIPOL), clock phase (SC_SPIPHA), and direction of data (SC_SPIORD) (see Table 25). The register bits SC_SPIPOL, SC_SPIPHA, and SC_SPIORD are defined within the SC2_SPICFG registers.

Note: Switching the SPI configuration from SC_SPIPOL=1 to SC_SPIPOL=0 without subsequently setting SC2_MODE=0 and reinitializing the SPI will cause an extra byte (0xFE) to be transmitted immediately before the first intended byte.

Table 25. SC2 SPI Slave Formats

SC2_MODE	SC2_SPICFG				GPIO_CFG[7:4]	Frame Format
	SC_SPIMST	SC_SPIORD	SC_SPIPHA	SC_SPIPOL		
2	0	0	0	0	SC2-4S mode	
2	0	0	0	1	SC2-4S mode	
2	0	0	1	0	SC2-4S mode	
2	0	0	1	1	SC2-4S mode	
2	0	1	-	-	SC2-4S mode	Same as above except LSB first instead of MSB first
2	0	-	-	-	SC2-3M mode	Illegal
2	0	-	-	-	SC2-2 mode	Illegal

When the slave select (nSSEL) signal is asserted (by the Master), SC2 SPI transmit data is driven to the output pin MISO and SC2 SPI data is received from the input pin MOSI. When the slave select (nSSEL) signal is deasserted (by the Master), no data is transferred on the MISO or MOSI pins and the output pin MISO is tri-stated. The slave select signal nSSEL is used to enable driving the serialized data output signal MISO. It is also used to reset the SC2 SPI slave shift register.

Characters received and transmitted are passed through receive and transmit FIFOs. The transmit and receive FIFOs are 4 bytes deep. These FIFOs are accessed under software control by accessing the SC2_DATA data register or under hardware control using a DMA controller.

Any character received is stored in the (empty) receive FIFO. The register bit SC_SPIRXVAL in the SC2_SPISTAT register is set to indicate that not all received characters are read out from receive FIFO. If software or DMA is not reading from the receive FIFO, the receive FIFO will store up to 4 characters. Any further reception is dropped, and the register bit SC_SPIRXOVF in the SC2_SPISTAT register is set. The RX FIFO hardware generates the INT_SCRXOVF interrupt, but the DMA register will not indicate the error condition until the RX FIFO is drained. Once the DMA marks a RX error, there are two conditions that will clear the error indication: setting the appropriate SC_TX/RXDMARST bit in the SC2_DMACTRL register, or loading the appropriate DMA buffer after it has unloaded.

Receiving a character always causes a serialization of a transmit character pulled from the transmit FIFO. When the transmit FIFO is empty, a transmit underrun is detected (no data in transmit FIFO) and the register bit INT_SCTXUND in the INT_SC2FLAG register is set. Because there is no character available for serialization, the SPI serializer retransmits the last transmitted character or a busy token (0xFF), which is determined by the register bit SC_SPIRPT in the SC2_SPICFG register.

Note: Even during a transmit underrun, the register bit SC_SPITXIDLE in the SC2_SPISTAT register will clear when the SPI master begins to clock data out of the MISO pin, indicating the transmitter is not idle. After a complete byte has been clocked out, the bit SC_SPITXIDLE will be set and the register bit INT_SCTXIDLE in the INT_SC2FLAG interrupt register will be set. The bits SC_SPITXIDLE and INT_SCTXIDLE will toggle in this manner for every byte that is transmitted as an underrun.

When a transmit character is written to the (empty) transmit FIFO, the SC2_SPISTAT register and the INT_SC2FLAG register do not change. Further transmit characters can be written to the transmit FIFO until it is full, which causes the register bit SC_SPITXFREE in the SC2_SPISTAT register to clear. When the SPI master begins to clock data out of the MISO pin, the register bit SC_SPITXIDLE in the SC2_SPISTAT register clears (after the first bit is clocked out) and indicates that not all characters are transmitted yet. After shifting one full transmit character to the MISO pin, space for one transmit character becomes available in the transmit FIFO. This causes the register bit SC_SPITXFREE in the SC2_SPISTAT register to be set. After all characters are shifted out, the transmit FIFO is empty, which causes the register bit SC_SPITXIDLE in the SC2_SPISTAT register to be set.

The SPI slave controller must guarantee that there is time to move new transmit data from the transmit FIFO into the hardware serializer. To provide sufficient time, the SPI slave controller inserts a byte of padding, 0xFF, onto the start of transmit data. This byte of padding is only inserted when slave select is deasserted, the FIFO is empty, and the transmitter (serializer) is idle. An idle transmitter is indicated by the SC_SPITXIDLE bit in the SC2_SPISTAT register. Subsequent transmissions while slave select remains asserted do not include a byte of padding. But, if any new data is written to the transmit FIFO while slave select is deasserted, the first byte immediately goes into the transmit serializer without waiting for clocks from the external SPI master. The transmit serializer will still hold this data until there are clocks from the master. Because the data goes directly into the serializer, there is a race condition between when the data enters the serializer and when the SPI master attempts to clock out the data. If the data enters the serializer after the SPI master begins clocking data, then a byte of padding is transmitted as described above. If the data enters the serializer before the SPI master begins clocking data, then the first byte of data is transmitted without a byte of padding. Because of this race condition and the inability of the SPI master to know the current, internal state of the EM250, it is best to design a protocol around SPI slave interaction that handles this race condition and avoids potential issues. Some possible protocol solutions are:

- SPI slave does not place data into the transmit FIFO until the SPI status indicates that the SPI master has begun clocking data. One possible indication of this is the SC_SPIRXVAL bit.
- The communications between the SPI master and SPI slave use an interrogation-response scheme where the SPI slave only queues up data for transmission after receiving data from the master, without slave select deasserting between the two events.
- SPI slave begins all transmissions with its own byte of padding, such that the master always receives one or two bytes of padding.
- The data from SPI slave includes a data integrity scheme where the information received by the master can be validated as accurate.

Interrupts are generated by one of the following events:

- Transmit FIFO empty and last character shifted out (0 to 1 transition of SC_SPITXIDLE)
- Transmit FIFO changed from full to not full (0 to 1 transition of SC_SPITXFREE)
- Receive FIFO changed from empty to not empty (0 to 1 transition of SC_SPIRXVAL)
- Transmit DMA buffer A/B complete (1 to 0 transition of SC_TXACTA/B)
- Receive DMA buffer A/B complete (1 to 0 transition of SC_RXACTA/B)
- Received and lost character while receive FIFO was full (Receive overrun error)

- Transmitted character while transmit FIFO was empty (Transmit underrun error)

To generate interrupts to the CPU, the interrupt masks in the INT_SC2CFG and INT_CFG register must be enabled.

5.3.2 I²C Master Mode

The SC2 I²C controller is only available in master mode. The SC2 I²C controller is enabled with SC2_MODE set to 3. The I²C Master controller supports Standard (100kbps) and Fast (400kbps) I²C modes. Address arbitration is not implemented, so multiple master applications are not supported. The I²C signals are pure open-collector signals, and external pull-up resistors are required.

The SC2 I²C mode has the following features:

- Programmable clock frequency (400kHz max.)
- 7- and 10-bit addressing

The following signals can be made available on the GPIO pins:

- SDA (serial data)
- SCL (serial clock)

The I²C Master controller obtains its reference clock from a programmable clock generator. Clock rates are set by a clock division ratio from the 24MHz clock:

$$\text{Nominal Rate} = 24\text{MHz} / (2 * (\text{LIN} + 1) * 2^{\text{EXP}})$$

EXP is written to the SC2_RATEEXP register and LIN to the SC2_RATELIN register. Table 26 shows the rate settings for Standard I²C (100kbps) and Fast I²C (400kbps) operation.

Table 26. I²C Nominal Rate Programming

Nominal Rate	SC2_RATELIN	SC2_RATEEXP
100kbps	14	3
375kbps	15	1
400kbps	14	1

Note that, at 400kbps, the I²C specification requires the minimum low period of SCL to be 1.3µs. To be strictly I²C compliant, the rate needs to be lowered to 375kbps.

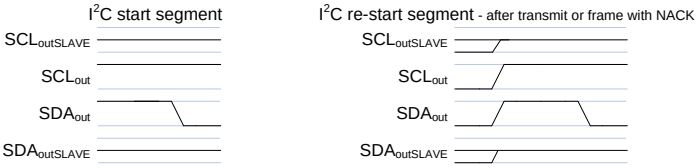
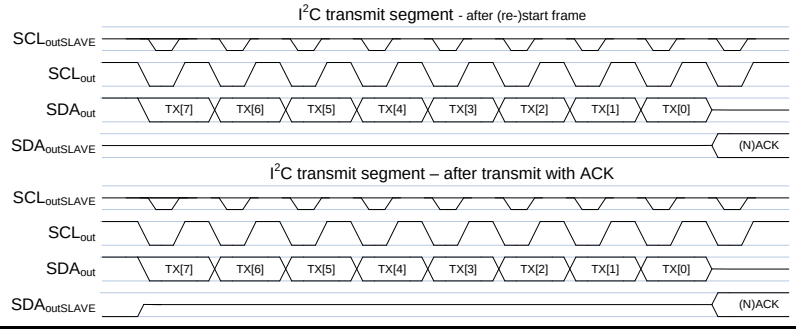
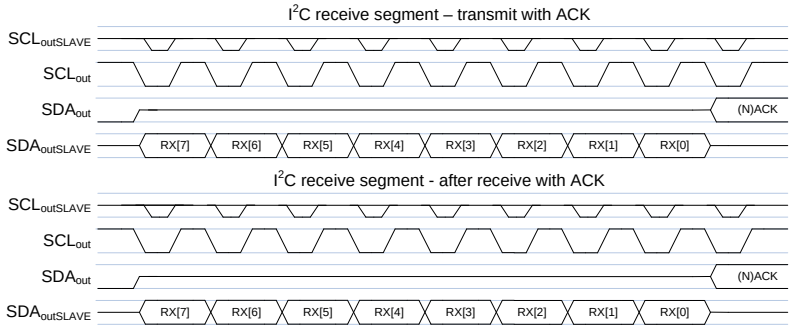
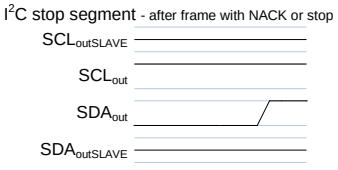
The I²C Master controller supports generation of various frame segments defined by the register bits SC_I2CSTART, SC_I2CSTOP, SC_I2CSEND, and SC_I2CRECV within the SC2_I2CCTRL1 register. Table 27 summarizes these frames.

Full I²C frames have to be constructed under software control by generating individual I²C segments. All necessary segment transitions are shown in Figure 7. ACK or NACK generation of an I²C receive frame segment is determined with the register bit SC_I2CACK in the SC2_I2CCTRL2 register.

Generation of a 7-bit address is accomplished with one transmit segment. The upper 7 bits of the transmitted character contain the 7-bit address. The remaining lower bit contains the command type (“read” or “write”).

Generation of a 10-bit address is accomplished with two transmit segments. The upper 5 bits of the first transmit character must be set to 0x1E. The next 2 bits are for the 2 most significant bits of the 10-bit address. The remaining lower bit contains the command type (“read” or “write”). The second transmit segment is for the remaining 8 bits of the 10-bit address.

Table 27. I²C Master Segment Formats

SC2_MODE	SC2_I2CCTRL1				GPIO_CFG[7:4]	Frame Segments
	SC_I2CSTART	SC_I2CSEND	SC_I2CRECV	SC_I2CSTOP		
3	1	0	0	0	SC2-2 mode	
3	0	1	0	0	SC2-2 mode	
3	0	0	1	0	SC2-2 mode	
3	0	0	0	1	SC2-2 mode	
3	0	0	0	0	SC2-2 mode	No pending frame segment
3	1 - - 1	1 1 - -	- 1 1 -	- - 1 1	SC2-2 mode	Illegal
3	-	-	-	-	SC2-4M mode	Illegal
3	-	-	-	-	SC2-4A mode	Illegal

Characters received and transmitted are passed through receive and transmit FIFOs. The SC2 I²C master transmit and receive FIFOs are 1 byte deep. These FIFOs are accessed under software control.

(Re)start and stop segments are initiated by setting the register bits SC_I2CSTART or SC_I2CSTOP in the SC2_I2CCTRL1 register, followed by waiting until they have cleared. Alternatively, the register bit SC_I2CCMDFIN in the SC2_I2CSTAT can be used for waiting.

For initiating a transmit segment, the data has to be written to the SC2_DATA data register, followed by setting the register bit SC_I2CSEND in the SC2_I2CCTRL1 register, and completed by waiting until it clears. Alternatively, the register bit SC_I2CTXFIN in the SC2_I2CSTAT can be used for waiting.

A receive segment is initiated by setting the register bit SC_I2CRECV in the SC2_I2CCTRL1 register, waiting until it clears, and then reading from the SC2_DATA data register. Alternatively, the register bit SC_I2CRXFIN in the SC2_I2CSTAT can be used for waiting. Now the register bit SC_I2CRXNAK in the SC2_I2CSTAT register indicates if a NACK or ACK was received from an I^2C slave device.

Interrupts are generated on the following events:

- Bus command (SC_I2CSTART/SC_I2CSTOP) completed (0 to 1 transition of SC_I2CCMDFIN)
- Character transmitted and slave device responded with NACK
- Character transmitted (0 to 1 transition of SC_I2CTXFIN)
- Character received (0 to 1 transition of SC_I2CRXFIN)
- Received and lost character while receive FIFO was full (Receive overrun error)
- Transmitted character while transmit FIFO was empty (Transmit underrun error)

To generate interrupts to the CPU, the interrupt masks in the INT_SC2CFG and INT_CFG register must be enabled.

5.3.3 Registers

SC2_MODE [0x442A]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	SC2_MODE	
0-R	0-R	0-R	0-R	0-R	0-R	0-RW	0-RW
7	6	5	4	3	2	1	0

SC2_MODE	[1:0]	SC2 Mode: 0 = disabled; 1 = disabled; 2 = SPI mode; 3 = I2C mode. Note: To change between modes, the previous mode must be disabled first.

SC2_DATA [0x441E]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
SC2_DATA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC2_DATA	[7:0]	Transmit and receive data register. Writing to this register pushes a byte onto the transmit FIFO. Reading from this register pulls a byte from the receive FIFO.

SC2_RATELIN [0x4430]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	SC2_RATELIN			
0-R	0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

NOTE: Changing the SC2_RATELIN register is only allowed when the serial controller is disabled, when SC2_MODE is 0.

SC2_RATELIN [3:0] The linear component (LIN) of the clock rate as seen in the equation: rate = 24MHz / (2 * (LIN + 1) * (2^EXP))

SC2_RATEEXP [0x4432]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	SC2_RATEEXP			
0-R	0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

NOTE: Changing the SC2_RATEEXP register is only allowed when the serial controller is disabled, when SC2_MODE is 0.

SC2_RATEEXP [3:0] The exponential component (EXP) of the clock rate as seen in the equation: rate = 24MHz / (2 * (LIN + 1) * (2^EXP))

SC2_SPICFG [0x442C]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	SC_SPIRXDRV	SC_SPIMST	SC_SPIRPT	SC_SPIORD	SC_SPIPHA	SC_SPIPOL
0-R	0-R	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

NOTE: Changing the SC2_SPICFG register is only allowed when the serial controller is disabled, when SC2_MODE is 0.

SC_SPIRXDRV [5] Receiver-driven mode selection bit (SPI master mode only). Clearing this bit will initiate transactions when transmit data is available. Setting this bit will initiate transactions when the receive buffer (FIFO or DMA) has space.

SC_SPIMST [4] Setting this bit will put the SPI in master mode while clearing this bit will put the SPI in slave mode.

SC_SPIRPT [3] This bit controls behavior on a transmit buffer underrun condition in slave mode. Clearing this bit will send the BUSY token (0xFF) and setting this bit will repeat the last byte. Changing this bit will only take effect when the transmit FIFO is empty and the transmit serializer is idle.

SC_SPIORD [2] Clearing this bit will result in the Most Significant Bit being transmitted first while setting this bit will result in the Least Significant Bit being transmitted first.

SC_SPIPHA [1] Clock phase configuration is selected with clearing this bit for sampling on the leading (first edge) and setting this bit for sampling on second edge.

SC_SPIPOL [0] Clock polarity configuration is selected with clearing this bit for a rising leading edge and setting this bit for a falling leading edge.

SC2_SPISTAT [0x4420]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	SC_SPITXIDLE	SC_SPITXFREE	SC_SPIRXVAL	SC_SPIRXOVF
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC_SPITXIDLE [3] This bit is set when the transmit FIFO is empty and the transmitter is idle.

SC_SPITXFREE [2] This bit is set when the transmit FIFO is ready to accept at least one byte.

SC_SPIRXVAL [1] This bit is set when the receive FIFO contains at least one byte.

SC_SPIRXOVF [0] This bit is set when the receive FIFO has been overrun. This bit clears when the data register (SC2_DATA) is read.

SC2_I2CCTRL1 [0x4426]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	SC_I2CSTOP	SC_I2CSTART	SC_I2CSEND	SC_I2CRECV
0-R	0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC_I2CSTOP [3] Setting this bit sends the STOP command. It autoclears when the command completes.

SC_I2CSTART [2] Setting this bit sends the START or repeated START command. It autoclears when the command completes.

SC_I2CSEND [1] Setting this bit transmits a byte. It autoclears when the command completes.

SC_I2CRECV [0] Setting this bit receives a byte. It autoclears when the command completes.

SC2_I2CCTRL2 [0x4428]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	SC_I2CACK
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
7	6	5	4	3	2	1	0

SC_I2CACK [0] Setting this bit will signal ACK after a received byte. Clearing this bit will signal NACK after a received byte.

SC2_I2CSTAT [0x4422]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	SC_I2CCMDFIN	SC_I2CRXFIN	SC_I2CTXFIN	SC_I2CRXNAK
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC_I2CCMDFIN [3] This bit is set when a START or STOP command completes. It autoclears on next bus activity.

SC_I2CRXFIN [2] This bit is set when a byte is received. It autoclears on next bus activity.

SC_I2CTXFIN [1] This bit is set when a byte is transmitted. It autoclears on next bus activity.

SC_I2CRXNAK [0] This bit is set when a NACK is received from the slave. It autoclears on next bus activity.

SC2_DMACTRL [0x4418]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	SC_TXDMARST	SC_RXDMARST	SC_TXLODB	SC_TXLODA	SC_RXLODB	SC_RXLODA
0-R	0-R	0-W	0-W	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC_TXDMARST [5] Setting this bit will reset the transmit DMA. The bit is autocleared.

SC_RXDMARST [4] Setting this bit will reset the receive DMA. This bit is autocleared.

SC_TXLODB [3] Setting this bit loads DMA transmit buffer B addresses and starts the DMA controller processing transmit buffer B. This bit is autocleared when DMA completes. Writing a zero to this bit will not have any effect. Reading this bit as one indicates DMA processing for buffer B is active or pending. Reading this bit as zero indicates DMA processing for buffer B is complete or idle.

SC_TXLODA [2] Setting this bit loads DMA transmit buffer A addresses and starts the DMA controller processing transmit buffer A. This bit is autocleared when DMA completes. Writing a zero to this bit will not have any effect. Reading this bit as one indicates DMA processing for buffer A is active or pending. Reading this bit as zero indicates DMA processing for buffer A is complete or idle.

SC_RXLODB [1] Setting this bit loads DMA receive buffer B addresses and starts the DMA controller processing receive buffer B. This bit is autocleared when DMA completes. Writing a zero to this bit will not have any effect. Reading this bit as one indicates DMA processing for buffer B is active or pending. Reading this bit as zero indicates DMA processing for buffer B is complete or idle.

SC_RXLODA [0] Setting this bit loads DMA receive buffer A addresses and starts the DMA controller processing receive buffer A. This bit is autocleared when DMA completes. Writing a zero to this bit will not have any effect. Reading this bit as one indicates DMA processing for buffer A is active or pending. Reading this bit as zero indicates DMA processing for buffer A is complete or idle.

SC2_DMASTAT [0x4416]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	SC_RXOVFB	SC_RXOVFA	SC_TXACTB	SC_TXACTA	SC_RXACTB	SC_RXACTA
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC_RXOVFB [5] This bit is set when DMA receive buffer B was passed an overrun error from the lower hardware FIFO. Neither receive buffers were capable of accepting any more bytes (unloaded), and the FIFO filled up. Buffer B was the next buffer to load, and when it drained the FIFO the overrun error was passed up to the DMA and flagged with this bit. This bit is autocleared the next time buffer B is loaded or when the receive DMA is reset.

SC_RXOVFA [4] This bit is set when DMA receive buffer A was passed an overrun error from the lower hardware FIFO. Neither receive buffers were capable of accepting any more bytes (unloaded), and the FIFO filled up. Buffer A was the next buffer to load, and when it drained the FIFO the overrun error was passed up to the DMA and flagged with this bit. This bit is autocleared the next time buffer A is loaded or when the receive DMA is reset.

SC_TXACTB [3] This bit is set when DMA transmit buffer B is currently active.

SC_TXACTA [2] This bit is set when DMA transmit buffer A is currently active.

SC_RXACTB [1] This bit is set when DMA receive buffer B is currently active.

SC_RXACTA [0] This bit is set when DMA receive buffer A is currently active.

SC2_RXCNTA [0x4410]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	SC2_RXCNTA				
SC2_RXCNTA							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC2_RXCNTA [12:0] A byte offset (from 0) which points to the location in DMA receive buffer A where the next byte will be written. When the buffer fills and subsequently unloads, this register wraps around and holds the value zero (pointing back to the first location in the buffer).

SC2_RXCNTB [0x4412]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	SC2_RXCNTB				
SC2_RXCNTB							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC2_RXCNTB [12:0] A byte offset (from 0) which points to the location in DMA receive buffer B where the next byte will be written. When the buffer fills and subsequently unloads, this register wraps around and holds the value zero (pointing back to the first location in the buffer).

SC2_TXCNT [0x4414]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	SC2_TXCNT				
SC2_TXCNT							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC2_TXCNT [12:0] A byte offset (from 0) which points to the location in the active (loaded) DMA transmit buffer from which the next byte will be read. When the buffer empties and subsequently unloads, this register wraps around and holds the value zero (pointing back to the first location in the buffer).

SC2_RXBEGA [0x4400]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC2_RXBEGA				
SC2_RXBEGA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC2_RXBEGA [12:0] DMA Start address (byte aligned) for receive buffer A.

SC2_RXENDA [0x4402]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC2_RXENDA				
SC2_RXENDA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC2_RXENDA [12:0] DMA End address (byte aligned) for receive buffer A.

SC2_RXBEGB [0x4404]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC2_RXBEGB				
SC2_RXBEGB							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC2_RXBEGB [12:0] DMA Start address (byte aligned) for receive buffer B.

SC2_RXENDB [0x4406]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC2_RXENDB				
SC2_RXENDB							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC2_RXENDB [12:0] DMA End address (byte aligned) for receive buffer B.

SC2_TXBEGA [0x4408]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC2_TXBEGA				
SC2_TXBEGA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC2_TXBEGA [12:0] DMA Start address (byte aligned) for transmit buffer A.

SC2_TXENDA [0x440A]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC2_TXENDA				
SC2_TXENDA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC2_TXENDA [12:0] DMA End address (byte aligned) for transmit buffer A.

SC2_TXBEGB [0x440C]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC2_TXBEGB				
SC2_TXBEGB							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC2_TXBEGB [12:0] DMA Start address (byte aligned) for transmit buffer B.

SC2_TXENDB [0x440E]

15	14	13	12	11	10	9	8
0-R	1-R	1-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	1	1	SC2_TXENDB				
SC2_TXENDB							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

SC2_TXENDB [12:0] DMA End address (byte aligned) for transmit buffer B.

SC2_RXERRA [0x441A]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	SC2_RXERRA				
SC2_RXERRA							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC2_RXERRA [12:0] A byte offset (from 0) which points to the location of the first error in the DMA receive buffer A. If there is no error, it will hold the value zero. This register will not be updated by subsequent errors arriving in the DMA. The next error will only be recorded if the buffer unloads and is reloaded or the receive DMA is reset.

SC2_RXERRB [0x441C]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	SC2_RXERRB				
SC2_RXERRB							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

SC2_RXERRB [12:0] A byte offset (from 0) which points to the location of the first error in the DMA receive buffer B. If there is no error, it will hold the value zero. This register will not be updated by subsequent errors arriving in the DMA. The next error will only be recorded if the buffer unloads and is reloaded or the receive DMA is reset.

INT_SC2CFG [0x4626]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	0	0	INT_ SCTXULDB	INT_ SCTXULDA	INT_ SCRXULDB	INT_ SCRXULDA	INT_SCNAK
INT_ SCCMDFIN	INT_SCTXFIN	INT_SCRXFIN	INT_ SCTXUND	INT_SCRXOVF	INT_ SCTXIDLE	INT_ SCTXFREE	INT_SCRXVAL
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

INT_SCTXULDB [12] DMA Tx buffer B unloaded interrupt enable.

INT_SCTXULDA [11] DMA Tx buffer A unloaded interrupt enable.

INT_SCRXULDB [10] DMA Rx buffer B unloaded interrupt enable.

INT_SCRXULDA [9] DMA Rx buffer A unloaded interrupt enable.

INT_SCNAK [8] Nack received (I^2C) interrupt enable.

INT_SCCMDFIN [7] START/STOP command complete (I^2C) interrupt enable.

INT_SCTXFIN [6] Transmit operation complete (I^2C) interrupt enable.

INT_SCRXFIN [5] Receive operation complete (I^2C) interrupt enable.

INT_SCTXUND [4] Transmit buffer underrun interrupt enable.

INT_SCRXOVF [3] Receive buffer overrun interrupt enable.

INT_SCTXIDLE [2] Transmitter idle interrupt enable.

INT_SCTXFREE [1] Transmit buffer free interrupt enable.

INT_SCRXVAL [0] Receive buffer has data interrupt enable.

INT_SC2FLAG [0x460E]

15 0-R	14 0-R	13 0-R	12 0-RW	11 0-RW	10 0-RW	9 0-RW	8 0-RW
0	0	0	INT_ SCTXULDB	INT_ SCTXULDA	INT_ SCRXULDB	INT_ SCRXULDA	INT_SCNAK
INT_ SCCMDFIN	INT_SCTXFIN	INT_SCRXFIN	INT_ SCTXUND	INT_SCRXOVF	INT_ SCTXIDLE	INT_ SCTXFREE	INT_SCRXVAL
0-RW 7	0-RW 6	0-RW 5	0-RW 4	0-RW 3	0-RW 2	0-RW 1	0-RW 0

INT_SCTXULDB	[12]	DMA Tx buffer B unloaded interrupt pending.
INT_SCTXULDA	[11]	DMA Tx buffer A unloaded interrupt pending.
INT_SCRXULDB	[10]	DMA Rx buffer B unloaded interrupt pending.
INT_SCRXULDA	[9]	DMA Rx buffer A unloaded interrupt pending.
INT_SCNAK	[8]	Nack received (I2C) interrupt pending.
INT_SCCMDFIN	[7]	START/STOP command complete (I2C) interrupt pending.
INT_SCTXFIN	[6]	Transmit operation complete (I2C) interrupt pending.
INT_SCRXFIN	[5]	Receive operation complete (I2C) interrupt pending.
INT_SCTXUND	[4]	Transmit buffer underrun interrupt pending.
INT_SCRXOVF	[3]	Receive buffer overrun interrupt pending.
INT_SCTXIDLE	[2]	Transmitter idle interrupt pending.
INT_SCTXFREE	[1]	Transmit buffer free interrupt pending.
INT_SCRXVAL	[0]	Receive buffer has data interrupt pending.

5.4 General Purpose Timers

The EM250 integrates two general-purpose, 16-bit timers—TMR1 and TMR2. Each of the two timers contains the following features:

- Configurable clock source
- Counter load
- Two output compare registers
- Two input capture registers
- Can be configured to do PWM
- Up/down counting (for PWM motor drive phase correction)
- Single shot operation mode (timer stops at zero or threshold)

Figure 9 is a block diagram of the Timer TMR1 module. Timer TMR2 is identical.

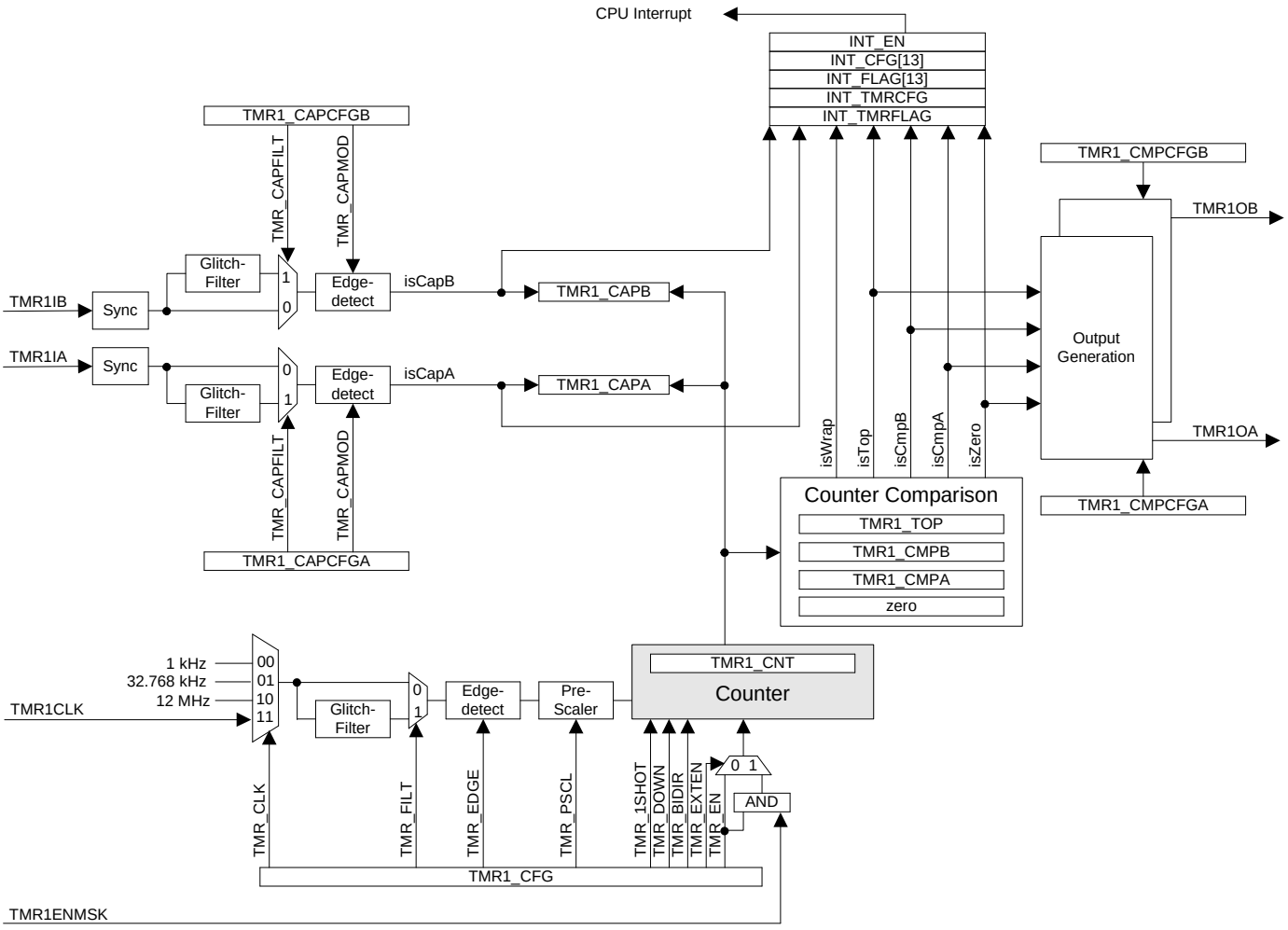


Figure 9. Timer TMR1 Block Diagram

5.4.1 Clock Sources

The clock source for each timer can be chosen from the main 12MHz clock, 32.768kHz clock, 1kHz RC-Clock, or from an external source (up to 100kHz) through TMR1CLK or TMR2CLK. After choosing the clock source (see Table 28), the frequency can be further divided to generate the final timer cycle provided to the timer controller (see Table 29). In addition, the clock edge (either rising or falling) for this timer clock can be selected (see Table 30).

Table 28. TMR1 and TMR2 Clock Source Settings

TMR_CLK[1:0]	Clock Source
0	1 kHz RC clock
1	32.768kHz clock
2	12 MHz clock
3	GPIO clock input

Table 29. Clock Source Divider Settings

TMR_PSCL[3:0]	Clock Source Prescale Factor
N = 0..10	2^N
N = 11..15	2^{10}

Table 30. Clock Edge Setting

TMR_EDGE	Clock Source
0	Rising
1	Falling

Note: All configuration changes do not take effect until the next edge of the timer's clock source.

These functions are separately controlled for TMR1 and TMR2 by setting the bits TMR_CLK, TMR_FILT, TMR_EDGE, and TMR_PSCL in the timer registers TMR1_CFG and TMR2_CFG, respectively.

5.4.2 Timer Functionality (Counting)

Each timer supports three counting modes: increasing, decreasing, or alternating (where the counting will increase, then decrease, then increase). These modes are controlled by setting the TMR_DOWN and TMR_BIDIR bits within the TMR1_CFG or TMR2_CFG registers.

Upward counting continues until the counter value reaches the threshold value stored in the TMR1_TOP or TMR2_TOP register. Downward counting continues until the counter value reaches the value zero. When the alternating counting mode is enabled, a triangular-shaped waveform of the count-value can be created. Figure 10 through Figure 13 illustrate the different counting modes available from the timers.

Counting can be enabled and disabled with the register bit TMR_EN in the TMR1_CFG or TMR2_CFG registers. When the timer is disabled, the counter stops counting and maintains its count value. Enabling can be masked with the pin TMR1ENMSK or TMR2ENMSK, depending on register bit TMR_EXTEN in the TMR1_CFG or TMR2_CFG registers.

By default, the counting operation is repetitive. It can be restricted to single counting enabled with the register bit TMR_1SHOT located in the TMR1_CFG or TMR2_CFG registers.

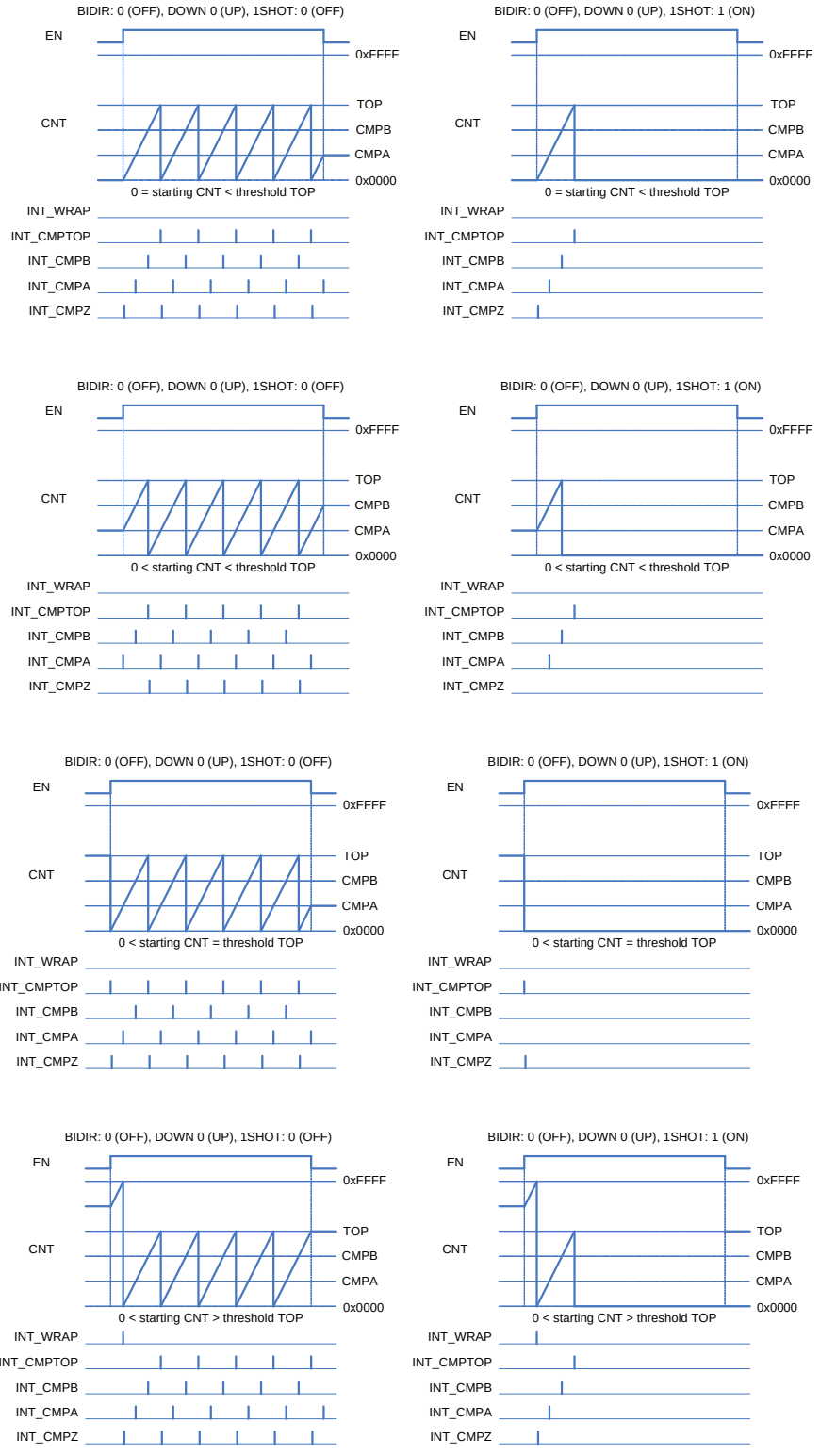


Figure 10. Timer Counting Mode—Saw Tooth, Up

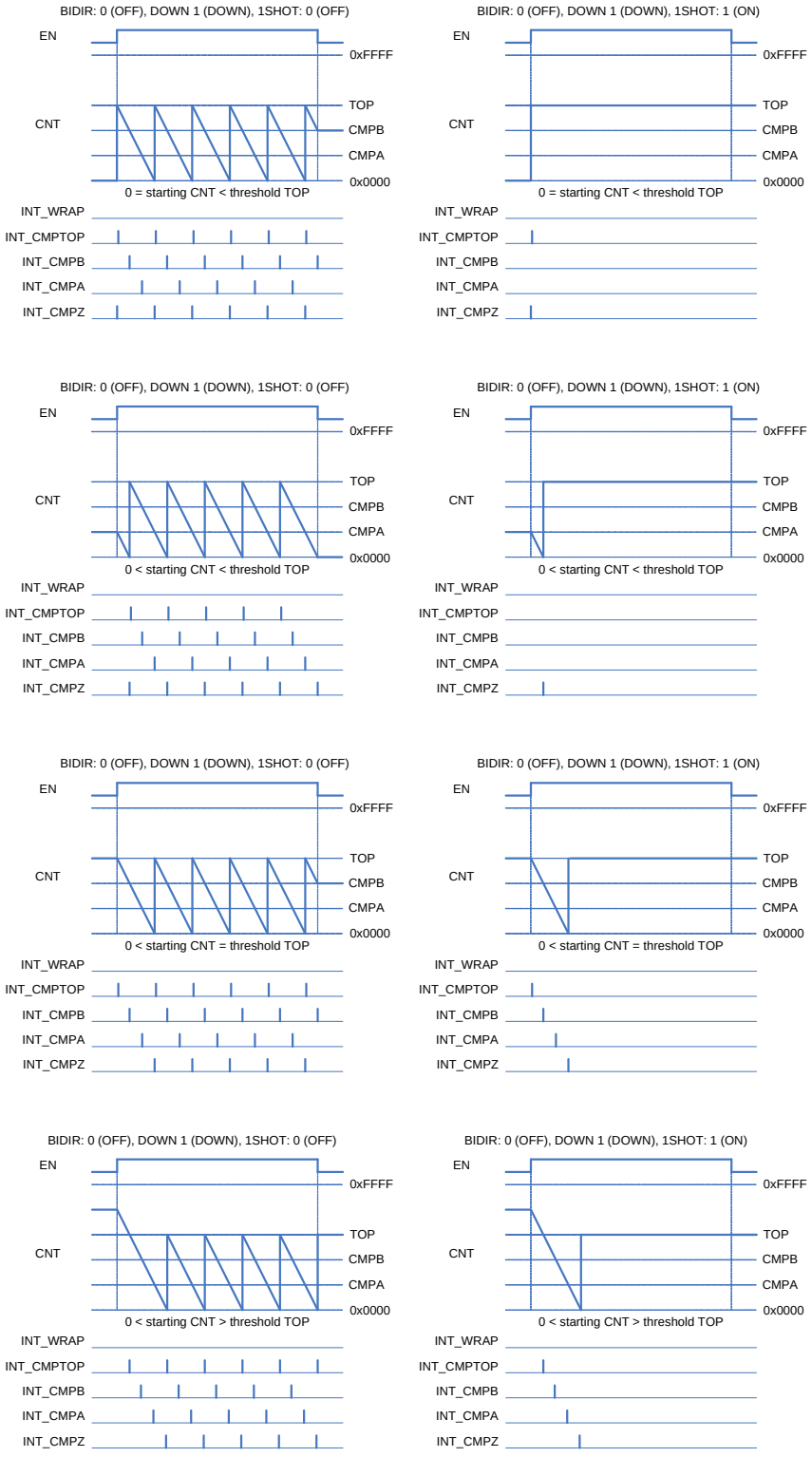


Figure 11. Timer Counting Mode—Saw Tooth, Down

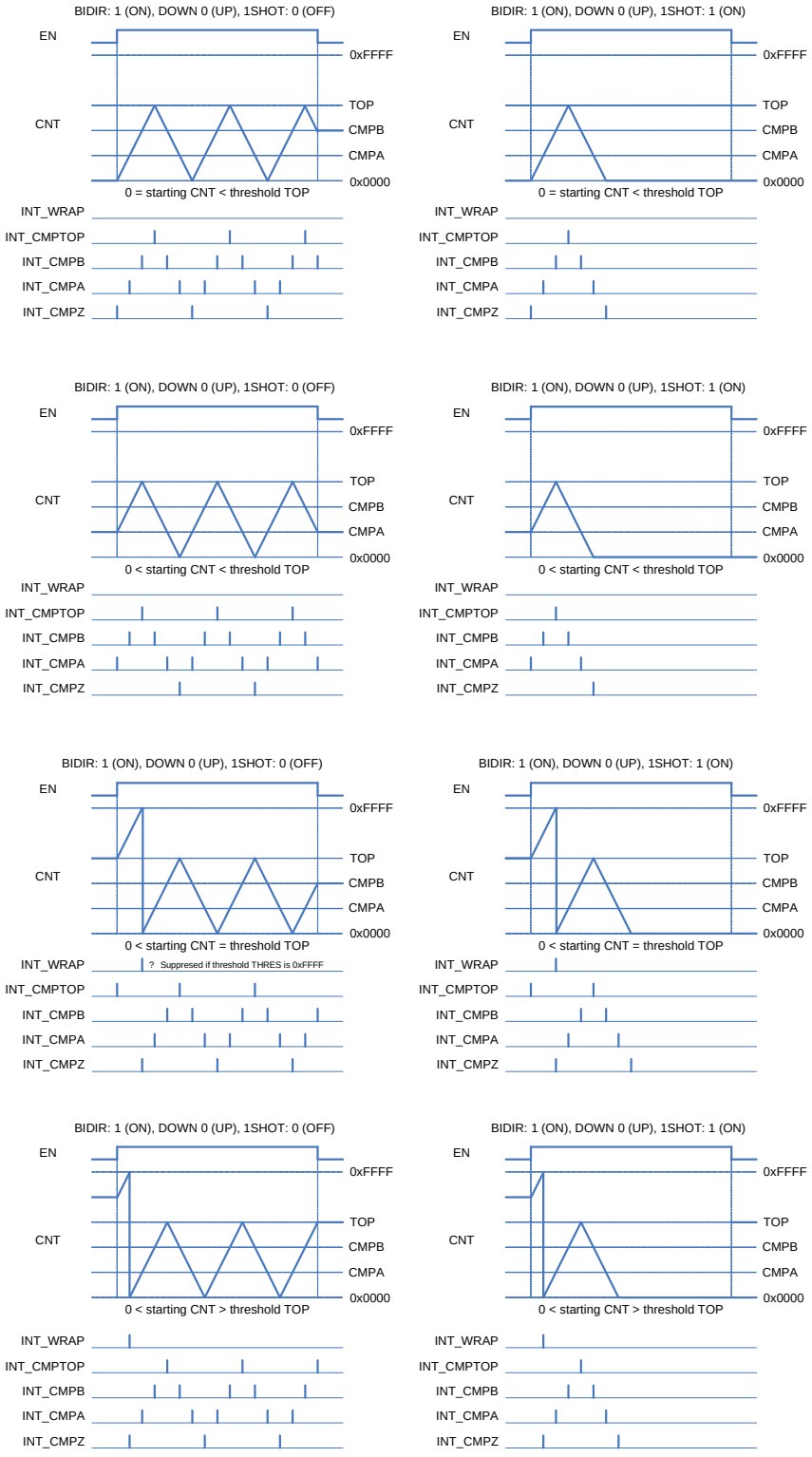


Figure 12. Timer Counting Mode—Alternating, Initially Up

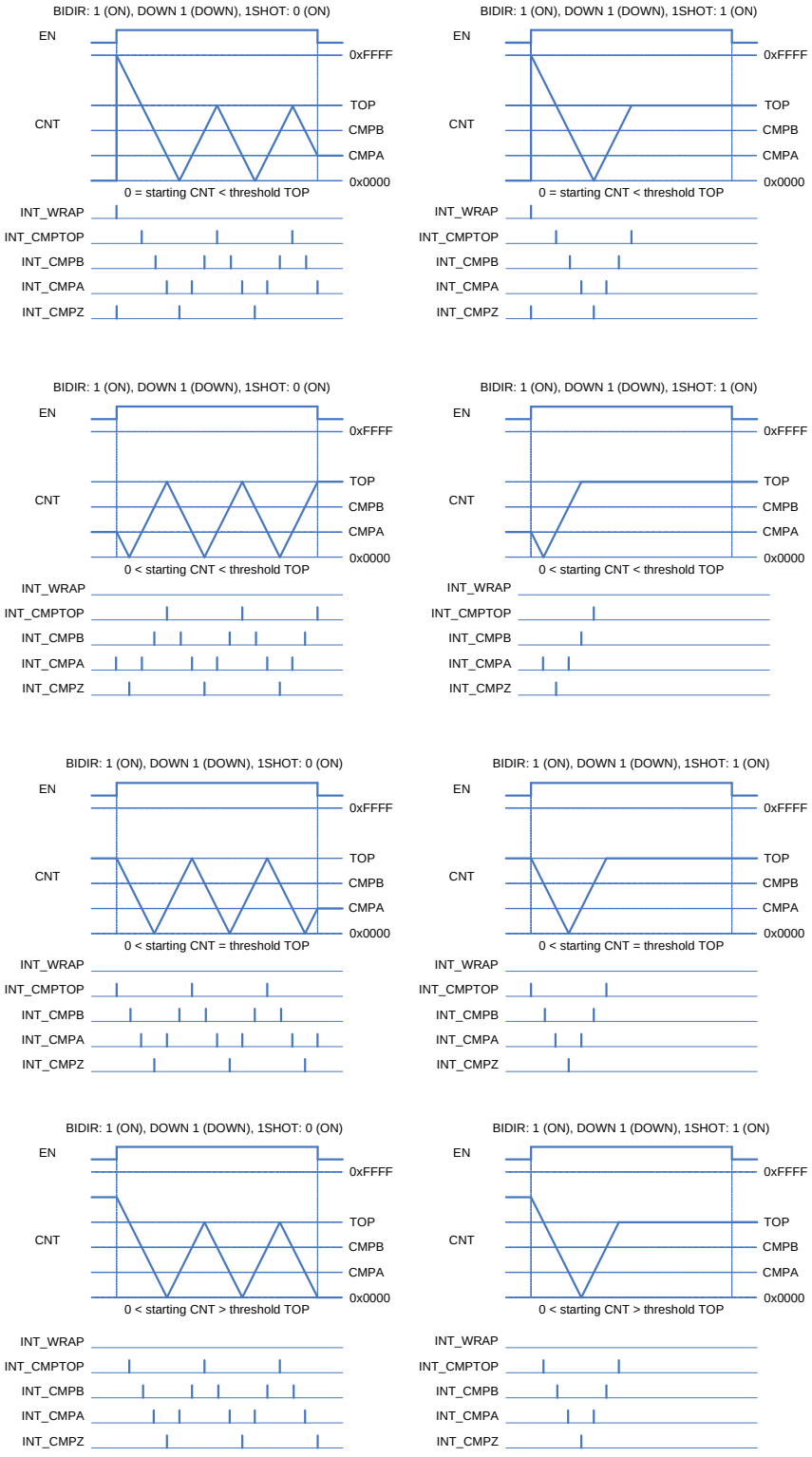


Figure 13. Timer Counting Mode—Alternating, Initially Down

5.4.3 Timer Functionality (Output Compare)

There are two output signals from each timer to generate application-specific waveforms. These waveforms are generated or altered by comparison results with the timer count value.

There are four comparison results:

- Counter value reaches zero.
- Counter value reaches threshold value of TMR1_TOP or TMR2_TOP register.
- Counter value reaches comparison value of TMR1_CMPA or TMR2_CMPA register.
- Counter value reaches comparison value of TMR1_CMPB or TMR2_CMPB register.

The output waveform generation from each timer is controlled with the register bits (TMR_CMPMOD or inverted with TMR_CMPINV) in the TMR1_CMPCFGA, TMR1_CMPCFGB, TMR2_CMPCFGA, and TMR2_CMPCFGB registers. Table 31 summarizes the output waveform generation modes.

Table 31. Output Waveform Settings

TMR_CMPMOD[3:0]	Output Waveform Generation Mode
0	Disable alteration
1	Toggle on count = TOP
2	Set on count = TOP, clear on count = CMPA
3	Set on count = TOP, clear on count = CMPB
4	Set to 1
5	Set on count = CMPA, clear on count = TOP
6	Toggle on count = CMPA
7	Set on count = CMPA, clear on count = CMPB
8	Clear to 0
9	Set on count = CMPB, clear on count = TOP
10	Set on count = CMPB, clear on count = CMPA
11	Toggle on count = CMPB
12	Toggle on count = ZERO
13	Set on count = ZERO, clear on count = TOP
14	Set on count = ZERO, clear on count = CMPA
15	Set on count = ZERO, clear on count = CMPB

The output signals TMR1OA and TMR1OB from Timer 1, and TMR2OA and TMR2OB from Timer 2, are available on GPIO. For selecting alternate pin functions, refer to Table 17 and Table 18.

Figure 14 and Figure 15 show examples of all timer output generation modes.

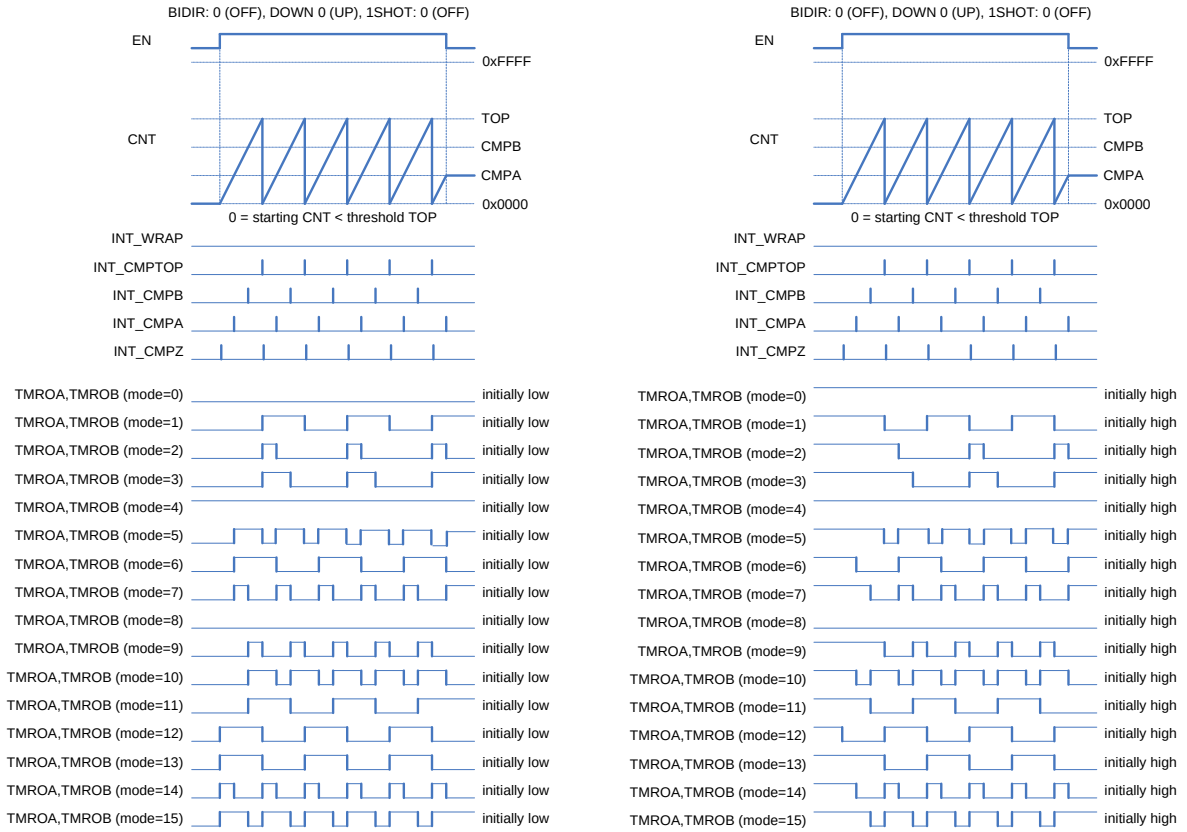


Figure 14. Timer Output Generation Mode Example—Saw Tooth, Non-inverting

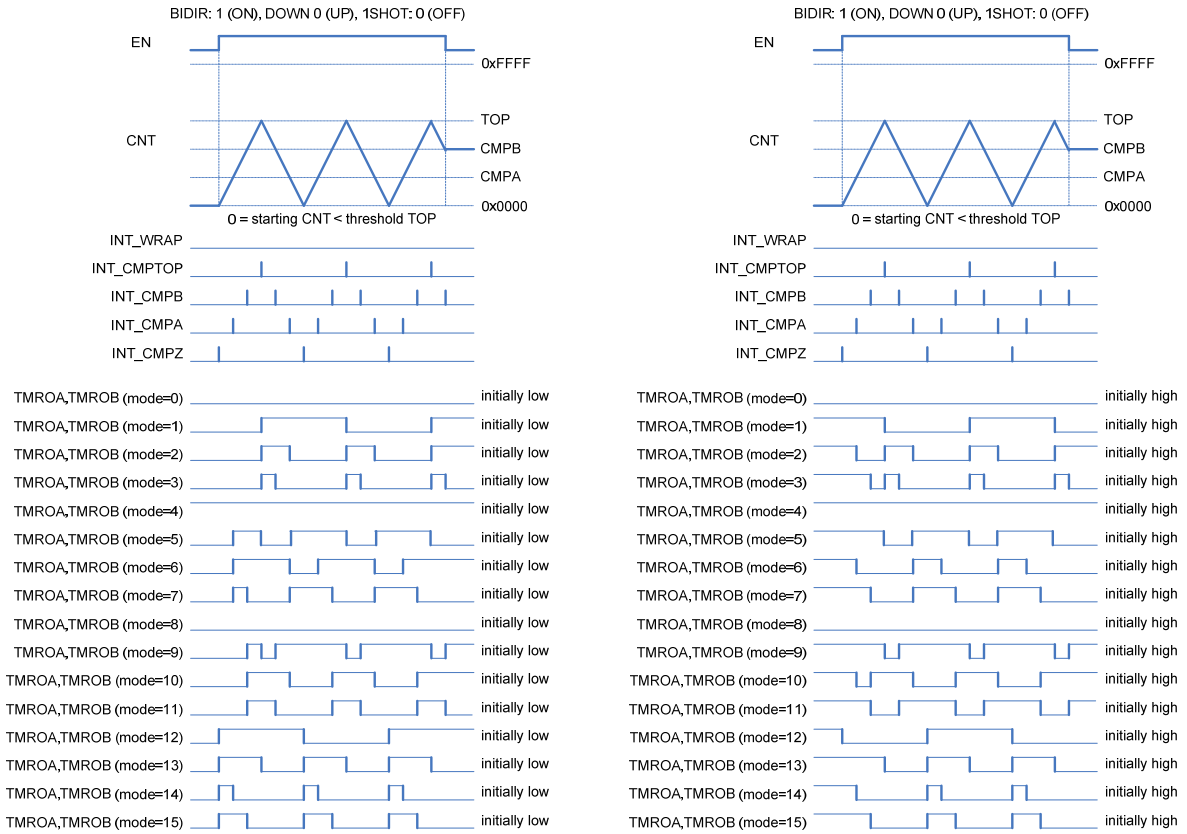


Figure 15. Timer Output Generation Mode Example—Alternating, Non-inverting

5.4.4 Timer Functionality (Input Capture)

There are two capture registers that store the timer count value on a trigger condition from GPIO signals. The timer trigger signals TMR1IA and TMR1IB for Timer 1, and TMR2IA and TMR2IB for Timer 2 are provided by external signals routed to the GPIO pins.

These timer trigger signals are synchronized to the main 12MHz clock, passed to an optional glitch filter, and followed by an edge detection circuitry.

These functions are controlled by software with the register bits `TMR_CAPMOD[1:0]`, and `TMR_CAPFILT` in the `TMR1_CAPCFGA`, `TMR1_CAPCFGB`, `TMR2_CAPCFGA`, and `TMR2_CAPCFGB` registers.

Table 32. GPIO/Timer Trigger Conditioning

TMR_CAPMOD[1:0]	Detection mode
0	Disabled
1	Rising Edge
2	Falling Edge
3	Either Edge

All glitch filters consist of a flip-flop-driven, 4-bit shift register clocked with the main 12MHz clock.

5.4.5 Timer Interrupt Sources

Each timer supports a number of interrupts sources:

- On overflow during up-count from all 1s to zero.
- On counter reaching output compare values stored in the TMR1_CMPA, TMR1_CMPB or TMR2_CMPA, and TMR2_CMPB registers.
- On counter reaching zero, TMR1_TOP, or TMR2_TOP.
- On capturing events from GPIO.

To generate interrupts to the CPU, the interrupt masks in the INT_TMRCFG and INT_CFG registers must be enabled.

5.4.6 Registers

TMR1_CFG [0x450C]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	0	0	TMR_EXTEN	TMR_EN	TMR_BIDIR	TMR_DOWN	TMR_1SHOT
TMR_PSCL				TMR_FILT	TMR_EDGE	TMR_CLK	
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR_EXTEN	[12]	Control bit for the external enable mask on a pin. When this bit is clear, do not check status of the TMR1ENMSK pin. When this bit is set, check status of the TMR1ENMSK pin.
TMR_EN	[11]	Set this bit to enable counting. To change other register bits, this bit must be cleared.
TMR_BIDIR	[10]	Set this bit to enable bi-directional alternation mode.
TMR_DOWN	[9]	Initial count direction after enabling the timer. Clear this bit to count up; set this bit to count down.
TMR_1SHOT	[8]	Clear this bit for auto repetition mode. Set this bit for a single shot.
TMR_PSCL	[7:4]	Clock divider setting (N). The possible clock divisors are: 0 - 2^N (N=0..10).
TMR_FILT	[3]	Set this bit to enable clock source glitch filtering.
TMR_EDGE	[2]	Clock source edge selection. Clear this bit for rising edge; set this bit for falling edge.
TMR_CLK	[1:0]	Clock source selection: 0 = calibrated RC oscillator (default); 1 = 32kHz; 2 = 12MHz; 3 = External (GPIO).

TMR1_CNT [0x4500]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
TMR1_CNT							
TMR1_CNT							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR1_CNT [15:0] Current Timer 1 counter value. When read, returns the current timer counter. When written, overwrites the timer counter and restarts wrap detection.

TMR1_TOP [0x4506]

15	14	13	12	11	10	9	8
1-RW	1-RW	1-RW	1-RW	1-RW	1-RW	1-RW	1-RW
TMR2_TOP							
TMR2_TOP							
1-RW	1-RW	1-RW	1-RW	1-RW	1-RW	1-RW	1-RW
7	6	5	4	3	2	1	0

TMR2_TOP [15:0] Timer 1 threshold value.

TMR1_CMPCFGA [0x450E]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
TMR_CMPEN	0	0	0	0	0	0	0
0	0	0	TMR_CMPINV	TMR_CMPMOD			
0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR_CMPEN [15] Set this bit to enable output A.

TMR_CMPINV [4] Set this bit to invert output A.

TMR_CMPMOD [3:0] Output mode selection bits. Refer to Table 31 for the modes.

TMR1_CMPCFGB [0x4510]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
TMR_CMPEN	0	0	0	0	0	0	0
0	0	0	TMR_CMPINV	TMR_CMPMOD			
0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR_CMPEN [15] Set this bit to enable output B.

TMR_CMPINV [4] Set this bit to invert output B.

TMR_CMPMOD [3:0] Output mode selection bits. Refer to Table 31 for the modes.

TMR1_CMPA [0x4508]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
TMR_CMPEN	0	0	0	0	0	0	0
0	0	0	TMR_CMPINV	TMR_CMPMOD			
0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR1_CMPA [15:0] Timer 1 compare A value.

TMR1_CMPB [0x450A]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
TMR1_CMPB							
TMR1_CMPB							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR1_CMPB [15:0] Timer 1 compare B value.

TMR1_CAPCFGA [0x4512]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
0	0	0	0	0	0	0	TMR_CAPFILT
0	TMR_CAPMOD		0	0	0	0	0
0-R	0-RW	0-RW	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

TMR_CAPFILT [8] Set this bit to enable the input A filter.

TMR_CAPMOD [6:5] Input edge triggering selection: 0 = disabled; 1 = rising; 2 = falling; 3 = both edges.

TMR1_CAPCFGB [0x4514]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
0	0	0	0	0	0	0	TMR_CAPFILT
0	TMR_CAPMOD		0	0	0	0	0
0-R	0-RW	0-RW	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

TMR_CAPFILT [8] Set this bit to enable the input A filter.

TMR_CAPMOD [6:5] Input edge triggering selection: 0 = disabled; 1 = rising; 2 = falling; 3 = both edges.

TMR1_CAPA [0x4502]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
TMR1_CAPA							
TMR1_CAPA							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

TMR1_CAPA [15:0] Timer 1 capture A value.

TMR1_CAPB [0x4504]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
TMR1_CAPB							
TMR1_CAPB							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

TMR1_CAPB [15:0] Timer 1 capture B value.

TMR2_CFG [0x458C]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW	0-RW
0	0	0	TMR_EXTEN	TMR_EN	TMR_BIDIR	TMR_DOWN	TMR_1SHOT
TMR_PSCL				TMR_FILT	TMR_EDGE	TMR_CLK	
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR_EXTEN [12] Control bit for the external enable mask on a pin. When this bit is clear, do not check status of the TMR2ENMSK pin. When this bit is set, check status of the TMR2ENMSK pin.

TMR_EN [11] Set this bit to enable counting. To change other register bits, this bit must be cleared.

TMR_BIDIR [10] Set this bit to enable bi-directional alternation mode.

TMR_DOWN [9] Initial count direction after enabling the timer. Clear this bit to count up; set this bit to count down.

TMR_1SHOT [8] Clear this bit for auto repetition mode. Set this bit for a single shot.

TMR_PSCL [7:4] Clock divider setting (N). The possible clock divisors are: 0 - 2^N (N=0..10).

TMR_FILT [3] Set this bit to enable clock source glitch filtering.

TMR_EDGE [2] Clock source edge selection. Clear this bit for rising edge; set this bit for falling edge.

TMR_CLK [1:0] Clock source selection: 0 = calibrated RC oscillator (default); 1 = 32kHz; 2 = 12MHz; 3 = External (GPIO).

TMR2_CNT [0x4580]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
TMR2_CNT							
TMR2_CNT							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR2_CNT [15:0] Current Timer 2 counter value. When read, returns the current timer counter. When written, overwrites the timer counter and restarts wrap detection.

TMR2_TOP [0x4586]

15	14	13	12	11	10	9	8
1-RW	1-RW	1-RW	1-RW	1-RW	1-RW	1-RW	1-RW
TMR2_TOP							
TMR2_TOP							
1-RW	1-RW	1-RW	1-RW	1-RW	1-RW	1-RW	1-RW
7	6	5	4	3	2	1	0

TMR2_TOP [15:0] Timer 2 threshold value.

TMR2_CMPCFGA [0x458E]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
TMR_CMPEN	0	0	0	0	0	0	0
0	0	0	TMR_CMPINV	TMR_CMPMOD			
0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR_CMPEN [15] Set this bit to enable output A.

TMR_CMPINV [4] Set this bit to invert output A.

TMR_CMPMOD [3:0] Output mode selection bits. Refer to Table 31 for the modes.

TMR2_CMPCFGB [0x4590]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
TMR_CMPEN	0	0	0	0	0	0	0
0	0	0	TMR_CMPINV	TMR_CMPMOD			
0-R	0-R	0-R	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR_CMPEN [15] Set this bit to enable output B.

TMR_CMPINV [4] Set this bit to invert output B.

TMR_CMPMOD [3:0] Output mode selection bits. Refer to Table 31 for the modes.

TMR2_CMPA [0x4588]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
TMR2_CMPA							
TMR2_CMPA							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR2_CMPA [15:0] Timer 2 compare A value.

TMR2_CMPB [0x458A]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
TMR2_CMPB							
TMR2_CMPB							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

TMR2_CMPB [15:0] Timer 2 compare B value.

TMR2_CAPCFGA [0x4592]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
0	0	0	0	0	0	0	TMR_CAPFILT
0	TMR_CAPMOD		0	0	0	0	0
0-R	0-RW	0-RW	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

TMR_CAPFILT [8] Set this bit to enable the input A filter.

TMR_CAPMOD [6:5] Input edge triggering selection: 0 = disabled; 1 = rising; 2 = falling; 3 = both edges.

TMR2_CAPCFGB [0x4594]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
0	0	0	0	0	0	0	TMR_CAPFILT
0	TMR_CAPMOD		0	0	0	0	0
0-R	0-RW	0-RW	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

TMR_CAPFILT [8] Set this bit to enable the input A filter.

TMR_CAPMOD [6:5] Input edge triggering selection: 0 = disabled; 1 = rising; 2 = falling; 3 = both edges.

TMR2_CAPA [0x4582]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
TMR2_CAPA							
TMR2_CAPA							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

TMR2_CAPA [15:0] Timer 2 capture value.

TMR2_CAPB [0x4584]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
TMR2_CAPB							
TMR2_CAPB							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

TMR2_CAPB [15:0] Timer 2 capture value.

INT_TMRCFG [0x462C]

15	14	13	12	11	10	9	8
0-R	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
0	INT_ TMR2CAPB	INT_ TMR2CAPA	INT_ TMR2CMPTOP	INT_ TMR2CMPZ	INT_ TMR2CMPB	INT_ TMR2CMPA	INT_ TMR2WRAP
0	INT_ TMR1CAPB	INT_ TMR1CAPA	INT_ TMR1CMPTOP	INT_ TMR1CMPZ	INT_ TMR1CMPB	INT_ TMR1CMPA	INT_ TMR1WRAP
0-R	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

INT_TMR2CAPB [14] Timer 2 capture B interrupt enable.

INT_TMR2CAPA [13] Timer 2 capture A interrupt enable.

INT_TMR2CMPTOP [12] Timer 2 compare Top interrupt enable.

INT_TMR2CMPZ [11] Timer 2 compare Zero interrupt enable.

INT_TMR2CMPB [10] Timer 2 compare B interrupt enable.

INT_TMR2CMPA [9] Timer 2 compare A interrupt enable.

INT_TMR2WRAP [8] Timer 2 overflow interrupt enable.

INT_TMR1CAPB [6] Timer 1 capture B interrupt enable.

INT_TMR1CAPA [5] Timer 1 capture A interrupt enable.

INT_TMR1CMPTOP [4] Timer 1 compare Top interrupt enable.

INT_TMR1CMPZ [3] Timer 1 compare Zero interrupt enable.

INT_TMR1CMPB [2] Timer 1 compare B interrupt enable.

INT_TMR1CMPA [1] Timer 1 compare A interrupt enable.

INT_TMR1WRAP [0] Timer 1 overflow interrupt enable.

INT_TMRFLAG [0x4614]

15	14	13	12	11	10	9	8
0-R	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
0	INT_ TMR2CAPB	INT_ TMR2CAPA	INT_ TMR2CMPTOP	INT_ TMR2CMPZ	INT_ TMR2CMPB	INT_ TMR2CMPA	INT_ TMR2WRAP
0	INT_ TMR1CAPB	INT_ TMR1CAPA	INT_ TMR1CMPTOP	INT_ TMR1CMPZ	INT_ TMR1CMPB	INT_ TMR1CMPA	INT_ TMR1WRAP
0-R	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

INT_TMR2CAPB [14] Timer 2 capture B interrupt pending.

INT_TMR2CAPA [13] Timer 2 capture A interrupt pending.

INT_TMR2CMPTOP [12] Timer 2 compare Top interrupt pending.

INT_TMR2CMPZ [11] Timer 2 compare Zero interrupt pending.

INT_TMR2CMPB [10] Timer 2 compare B interrupt pending.

INT_TMR2CMPA [9] Timer 2 compare A interrupt pending.

INT_TMR2WRAP [8] Timer 2 overflow interrupt pending.

INT_TMR1CAPB [6] Timer 1 capture B interrupt pending.

INT_TMR1CAPA [5] Timer 1 capture A interrupt pending.

INT_TMR1CMPTOP [4] Timer 1 compare Top interrupt pending.

INT_TMR1CMPZ [3] Timer 1 compare Zero interrupt pending.

INT_TMR1CMPB [2] Timer 1 compare B interrupt pending.

INT_TMR1CMPA [1] Timer 1 compare A interrupt pending.

INT_TMR1WRAP [0] Timer 1 overflow interrupt pending.

5.5 ADC Module

The ADC is a first-order sigma-delta converter sampling at 1MHz with programmable resolution and conversion rate. Table 33 describes the key ADC Module parameter measured at 25°C and VDD_PADS at 3.0V. The single-ended measurements were done at f_{input} = 7.7% $f_{Nyquist}$; 0dBFS level (where full-scale is a 1.2Vp-p swing). The differential measurements were done at f_{input} = 7.7% $f_{Nyquist}$; -6dBFS level (where full-scale is a 2.4Vp-p swing).

Table 33. ADC Module Key Parameters

Parameter	Performance							
ADC_RATE[2:0]	0	1	2	3	4	5	6	7
Conversion Time (µs)	32	64	128	256	512	1024	2048	4096
Nyquist Freq (Hz)	15.6k	7.8k	3.9k	1.9k	980	490	240	120
3dB Cut-off (Hz)	9.4k	4.7k	2.3k	1.1k	590	290	140	72
INL (codes peak)	0.04	0.04	0.05	0.09	0.2	0.3	0.6	1.2
INL (codes RMS)	0.02	0.02	0.02	0.03	0.05	0.1	0.2	0.4
DNL (codes peak)	0.04	0.03	0.04	0.04	0.04	0.09	0.12	0.1
DNL (codes RMS)	0.02	0.01	0.01	0.01	0.01	0.03	0.03	0.03
ENOB (from single-cycle test)	5.5	7.0	7.5	10.0	11.5	12.0	12.5	12.5
SNR (dB)								
Single-Ended	35	44	53	62	69	74	75	74
Differential	36	45	54	62	68	71	73	73
SINAD (dB)								
Single-Ended	35	44	53	61	66	67	68	67
Differential	35	44	53	62	68	71	73	72
SDFR (dB)								
Single-Ended	63	69	70	70	70	69	70	70
Differential	61	70	75	74	78	80	84	89
THD (dB)								
Single-Ended	-55	-63	-67	-69	-69	-69	-69	-69
Differential	-57	-64	-74	-82	-87	-93	-94	-93
ENOB (from SNR)								
Single-Ended	5.8	7.3	8.8	10.3	11.5	12.3	12.5	12.3
Differential	7.0	8.5	10.0	11.3	12.3	12.8	13.1	13.1
ENOB (from SINAD)								
Single-Ended	5.8	7.3	8.8	10.1	11.0	11.1	11.3	11.1
Differential	7.0	8.3	9.8	11.3	12.3	12.8	13.1	13.0
Equivalent ADC Bits	5	6	7	8	9	10	11	12
ADC_DATA[x:y]	[15:11]	[15:10]	[15:9]	[15:8]	[15:7]	[15:6]	[15:5]	[15:4]

Note: INL and DNL are referenced to a LSB of the Equivalent ADC Bits shown in the last row of Table 33. ENOB (effective number of bits) can be calculated from either SNR (signal to non-harmonic noise ratio) or SINAD (signal-to-noise and distortion ratio).

The conversion rate is programmed by setting the ADC_RATE bits in the ADC_CFG register. The analog input of the ADC can be chosen from various sources and is configured with the ADC_SEL bits in the ADC_CFG register. As described in Table 34, the ADC inputs can be single-ended (routed individually to ADC0, ADC1, ADC2, or ADC3) or differential (routed to pairs ADC0-ADC1 and ADC2-ADC3). For selecting alternate pin functions, refer to Table 17 and Table 18.

Table 34. ADC Inputs

ADC_SEL[3:0]	Analog Source of ADC	GPIO Pin	Purpose
0	ADC0	4	Single-ended
1	ADC1	5	Single-ended
2	ADC2	6	Single-ended
3	ADC3	7	Single-ended
4	(1/4) * VDD_PADS (2.1–3.6V pad supply)		Supply monitoring
5	(1/2) * VDD (1.8V core supply)		Supply monitoring
6	RESERVED		
7	VSS (0V)		Calibration
8	VREF	8	Calibration
9	ADC0–ADC1	4–5	Differential
10	ADC2–ADC3	6–7	Differential

Setting the ADC_EN bit in the ADC_CFG register will cause the ADC to immediately begin conversions. The ADC will continually generate conversions until the ADC_EN bit is cleared. When each conversion completes, an INT_ADC interrupt is generated. In order for this to interrupt the CPU the interrupt mask INT_ADC must be enabled in the INT_CFG register. The INT_ADC interrupt is the only means for determining when a conversion completes. After each INT_ADC interrupt, the INT_ADC interrupt bit must be cleared to detect completion of the next conversion.

To ensure the pipelined digital filter in the ADC is flushed, ADC_EN should be cleared before changes are made to ADC_SEL or ADC_RATE. Discard the first sample after ADC_EN is set.

The ADC uses an internal reference, VREF, which may be routed out to the alternate pin function of GPIO8, VREF_OUT. VREF_OUT is only enabled when the ADC_EN bit in the ADC_CFG register is set. VREF is trimmed as close to 1.2V as possible by the EmberZNet PRO software, using the regulated supply (VDD) as reference. VREF is able to source modest current (see Table 36) and is stable under capacitive loads. The ADC cannot accept an external VREF input. For selecting alternate pin functions, refer to Table 17 and Table 18.

While the ADC Module supports both single-ended and differential inputs, the ADC input stage is differential. Single-ended operation is provided by internally connecting one of the differential inputs to VREF/2 while fully differential operation uses two external signals. The full-scale differential input range spans -VREF to +VREF and the single-ended input range spans 0 to VREF.

Sampling of internal connections VSS and VREF allow for offset and gain calibration of the ADC in applications where absolute accuracy is important. Measurement of the unregulated supply VDD_PADS, 2.1-3.6V pad supply, allows battery voltage to be monitored. Measurement of the regulated supply VDD, 1.8V core supply, provides an accurate means of calibrating the ADC as the regulator is factory trimmed to 1.72V.

Offset and gain correction using VREF or VDD reduces both ADC gain errors and reference errors but it is limited by the absolute accuracy of the supply. Correction using VREF is recommended because VREF is calibrated by the EmberZNet PRO software against VDD, which is factory trimmed to 1.72V. Table 35 shows the equations used.

Table 35. Offset and Gain Correction

Calculation Type	Corrected Sample	Absolute Voltage
Offset corrected	$N = (N_X - N_{VSS})$	
Offset and gain corrected using VREF, normalized to VREF	$N = \frac{(N_X - N_{VSS}) << 16}{(N_{VREF} - N_{VSS})}$	$V = \frac{(N \times VREF)}{2^{16}}$
Offset and gain corrected using VDD, normalized to VDD	$N = \frac{(N_X - N_{VSS}) << 16}{2 \times (N_{VDD} - N_{VSS})}$	$V = \frac{(N \times VDD)}{2^{16}}$

Equation notes

- All N are 16-bit numbers.
- N_X is a sampling of the desired analog source.
- N_{VSS} is a sampling of ground. Due to the nature of the ADC's internal design, ground does not yield 0x0000 in the ADC_DATA register. Instead, ground yields a value closer to 1/3 of the range—for example, 0x5200.
- N_{VREF} is a sampling of VREF. Due to the nature of the ADC's internal design, VREF does not yield 0xFFFF in the ADC_DATA register. Instead, VREF yields a value closer to 2/3 of the range—for example, 0xA800.
- N_{VDD} is a sampling of the regulated supply, VDD/2.
- <<16 indicates a bit shift left by 16 bits.
- When calculating the voltage of VDD_PADS (ADC_SEL = 4), V = (1/4) * VDD_PADS
- When calculating the voltage of VDD (ADC_SEL = 5), V = (1/2) * VDD

Table 36 lists other specifications for the ADC Module not covered in Table 33.

Table 36. ADC Specifications

Parameter	Min.	Typ.	Max.	Unit
VREF	1.19	1.2	1.21	V
VREF output current			1	mA
VREF load capacitance			10	nF
Minimum input voltage	0			V
Maximum input voltage			VREF	V
Single-ended signal range	0		VREF	V
Differential signal range	- VREF		+ VREF	V
Common mode range	0		VREF	V
Input referred ADC offset	- 10		10	mV
Input Impedance When taking a Sample When not taking a Sample	 1 10			M Ohm

Note: The signal-ended ADC measurements are limited in their range and only guaranteed for accuracy in the range 0 to VREF. The nature of the ADC's internal design allows for measurements outside of this range (+/- 200mV), but the accuracy of such measurements are not guaranteed. The maximum input voltage is of more interest to the differential sampling where a differential measurement might be small, but a common mode can push the actual input voltage on one of the signals towards VDD.

5.5.1 Registers

ADC_CFG [0x4902]

15	14	13	12	11	10	9	8
0-R	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
0	ADC_RATE			ADC_SEL			
0	0	0	0	0	0	ADC_DITH	ADC_EN
0-R	0-R	0-R	0-R	0-R	0-R	0-RW	0-RW
7	6	5	4	3	2	1	0

ADC_RATE [14:12] ADC conversion rate selection. Refer to Table 33 for details.

ADC_SEL [11:8] ADC input selection. Refer to Table 34 for details.

ADC_DITH [1] Set this bit to disable dither.

ADC_EN [0] Set this bit to enable the ADC.

ADC_DATA [0x4900]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
ADC_DATA							
ADC_DATA							
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
7	6	5	4	3	2	1	0

ADC_DATA [15:0] ADC sample value. Refer to Table 33 and Table 35 for details.

5.6 Event Manager

The XAP2b core supports one IRQ and one wake-up input; however, the EM250 contains an advanced Event Manager that takes IRQ and WAKE_UP signals from a variety of internal and external sources and provides them to the XAP2b. The Event Manager allows for each event to be separately masked and cleared by the CPU, and ensures that all events are serviced properly and promptly.

IRQ event sources include:

- Timer events
- GPIO events
- SC1 and SC2 events
- ADC
- System-mode sources (MAC, Watchdog, etc.)

WAKE_UP event sources include:

- Timer events
- GPIO events
- SC1 and SC2 events
- System-mode sources (MAC, Watchdog, etc.)

All interrupt source signals (except level-triggered GPIO interrupt signals) are momentary pulses that are guaranteed to be a single cycle of the main 12MHz clock. They will synchronously set the corresponding interrupt source bit(s) within a set of hierarchically organized interrupt source register(s). The interrupt controller merges these hierarchical interrupt sources into the single interrupt input to the CPU. Table 37 illustrates the enable and configuration status of each event within the EM250.

Table 37. Event Enable and Configuration Status

Event	Configuration
Interrupt pin to CPU	INT_EN
Top: INT_FLAG	INT_CFG
2nd: INT_periphFLAG	INT_periphCFG

The hierarchy has two levels of interrupt source and associated mask registers for fine control of interrupt processing. The top-level `INT_FLAG` and `INT_CFG` registers have one bit per major functional module of the EM250. The second level is a set of `INT_periphFLAG` and `INT_periphCFG` registers that each have one bit per sub-function within their respective module. Some modules, like ADC, have no second level. For a top-level event to actually interrupt the CPU, it must be enabled in the top-level `INT_CFG` register. Second-level events must additionally be enabled in their respective second-level `INT_periphCFG` registers.

To clear (acknowledge) an interrupt, software must write a 1 into the corresponding bit of the interrupt's lowest level `INT_periphFLAG` register. For example, to acknowledge an ADC interrupt, which has no second level, software must write a 1 into the `INT_ADC` bit of the top-level `INT_FLAG` register. To acknowledge a SC1 RXVALID second-level interrupt, software must write a 1 into the `INT_SCRXVAL` bit of the second-level `INT_SC1FLAG` register. If there were other enabled SC1 interrupts pending, the top-level `INT_SC1` bit in the `INT_FLAG` register would remain set, representing the “or” of all second-level-enabled SC1 interrupt events. The interrupt source register bits are designed to remain set if the event reoccurs at the same moment the bit is being cleared to acknowledge a prior occurrence.

If another enabled interrupt of the same type occurs before being acknowledged by the software ISR, it will be lost because no counting or queuing is used. However, this condition is detected and stored in the top-level `INT_MISS` register to facilitate software detection of such problems. The `INT_MISS` register is

"acknowledged" in the same way as the INT_FLAG register—by writing a 1 into the corresponding bit to be cleared.

If another enabled interrupt occurs after being acknowledged but while interrupts remain disabled, the CPU will be re-interrupted to service it when the software ISR returns and interrupts are re-enabled.

Applications only have write access to certain bits in the top-level INT_FLAG, INT_CFG, and INT_MISS registers that pertain to application peripherals. They have full access to second-level INT_periphFLAG and INT_periphCFG registers for application peripherals. System peripheral events and masking are protected from application interference.

Applications can also trigger a software interrupt by writing into the INT_SWCTRL register. System software is responsible for processing and acknowledging this interrupt.

The EM250 also provides a global INT_EN enable bit to enable or disable all interrupts into the CPU. This bit can be used to easily protect brief critical sections in application or system software.

5.6.1 Registers

INT_EN [0x4618]

15	14	13	12	11	10	9	8
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-R
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	INT_EN
0-R	0-R	0-R	0-R	0-R	0-R	0-R	0-RW
7	6	5	4	3	2	1	0

INT_EN [0] IRQ enable to CPU.

INT_CFG [0x461A]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
INT_WDOG	INT_FAULT	INT_TMR	INT_GPIO	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR
INT_SEC	INT_SC2	INT_SC1	INT_SLEEP	INT_BB	INT_SIF	INT_SW	0
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-R
7	6	5	4	3	2	1	0

INT_WDOG	[15]	Watchdog low watermark interrupt enable. Write is ignored in Application Mode.
INT_FAULT	[14]	Memory protection fault interrupt enable. Write is ignored in Application Mode.
INT_TMR	[13]	Timer interrupt enable.
INT_GPIO	[12]	GPIO interrupt enable.
INT_ADC	[11]	ADC interrupt enable.
INT_MACRX	[10]	MAC receive interrupt enable. Write is ignored in Application Mode.
INT_MACTX	[9]	MAC transmit interrupt enable. Write is ignored in Application Mode.
INT_MACTMR	[8]	MAC timer interrupt enable. Write is ignored in Application Mode
INT_SEC	[7]	Security interrupt enable. Write is ignored in Application Mode.
INT_SC2	[6]	SC2 interrupt enable.
INT_SC1	[5]	SC1 interrupt enable.
INT_SLEEP	[4]	Sleep Timer interrupt enable. Write is ignored in Application Mode.
INT_BB	[3]	Baseband interrupt enable. Write is ignored in Application Mode.
INT_SIF	[2]	SIF interrupt enable. Write is ignored in Application Mode.
INT_SW	[1]	Software interrupt enable. Write is ignored in Application Mode.

INT_FLAG [0x4600]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-R	0-R	0-RW	0-R	0-R	0-R
INT_WDOG	INT_FAULT	INT_TMR	INT_GPIO	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR
INT_SEC	INT_SC2	INT_SC1	INT_SLEEP	INT_BB	INT_SIF	INT_SW	0
0-R	0-R	0-R	0-R	0-R	0-RW	0-RW	0-R
7	6	5	4	3	2	1	0

INT_WDOG	[15]	Watchdog low watermark interrupt pending. Write is ignored in Application Mode.
INT_FAULT	[14]	Memory protection fault interrupt pending. Write is ignored in Application Mode.
INT_TMR	[13]	Timer interrupt pending.
INT_GPIO	[12]	GPIO interrupt pending.
INT_ADC	[11]	ADC interrupt pending.
INT_MACRX	[10]	MAC receive interrupt pending. Write is ignored in Application Mode.
INT_MACTX	[9]	MAC transmit interrupt pending. Write is ignored in Application Mode.
INT_MACTMR	[8]	MAC timer interrupt pending. Write is ignored in Application Mode.
INT_SEC	[7]	Security interrupt pending. Write is ignored in Application Mode.
INT_SC2	[6]	SC2 interrupt pending.
INT_SC1	[5]	SC1 interrupt pending.
INT_SLEEP	[4]	Sleep Timer interrupt pending. Write is ignored in Application Mode.
INT_BB	[3]	Baseband interrupt pending. Write is ignored in Application Mode.
INT_SIF	[2]	SIF interrupt pending. Write is ignored in Application Mode.
INT_SW	[1]	Software interrupt pending. Write is ignored in Application Mode.

INT_MISS [0x4602]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
INT_WDOG	INT_FAULT	INT_TMR	INT_GPIO	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR
INT_SEC	INT_SC2	INT_SC1	INT_SLEEP	INT_BB	INT_SIF	INT_SW	0
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-R
7	6	5	4	3	2	1	0

INT_WDOG [15] Watchdog low watermark interrupt miss. Write is ignored in Application Mode.

INT_FAULT [14] Memory protection fault miss. Write is ignored in Application Mode.

INT_TMR [13] Timer interrupt miss.

INT_GPIO [12] GPIO interrupt miss.

INT_ADC [11] ADC interrupt miss.

INT_MACRX [10] MAC receive interrupt miss. Write is ignored in Application Mode.

INT_MACTX [9] MAC transmit interrupt miss. Write is ignored in Application Mode.

INT_MACTMR [8] MAC timer interrupt miss. Write is ignored in Application Mode.

INT_SEC [7] Security interrupt miss. Write is ignored in Application Mode.

INT_SC2 [6] SC2 interrupt miss.

INT_SC1 [5] SC1 interrupt miss.

INT_SLEEP [4] Sleep Timer interrupt miss. Write is ignored in Application Mode.

INT_BB [3] Baseband interrupt miss. Write is ignored in Application Mode.

INT_SIF [2] SIF interrupt miss. Write is ignored in Application Mode.

INT_SW [1] Software interrupt miss. Write is ignored in Application Mode.

INT_SWCTRL [0x4638]

15	14	13	12	11	10	9	8
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
INT_SWCTRL							
INT_SWCTRL							
0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW	0-RW
7	6	5	4	3	2	1	0

INT_SWCTRL [15:0] Writing to this register generates software interrupt. Possible values to be written are explained and controlled in the EmberZNet PRO software stack.

5.7 Integrated Voltage Regulator

The EM250 integrates a low dropout regulator to provide an accurate core voltage at a low quiescent current. Table 38 lists the specifications for the integrated voltage regulator. With the regulator enabled, the pads supply voltage VDD_PADS is stepped down to the 1.8V regulator output VREG_OUT. The VREG_OUT signal must be externally decoupled and routed to the 1.8V core supply pins VDD_24MHZ, VDD_VCO, VDD_RF, VDD_IF, VDD_PRE, VDD_SYNTH, VDD_PADSA, VDD_CORE, and VDD_FLASH.

In addition, the regulator can be operated with several configurations of external load capacitors and decoupling capacitors. The Silicon Labs *EM250 Reference Design* details the different configurations recommended by Silicon Labs.

Table 38. Integrated Voltage Regulator Specifications

Spec Point	Min.	Typ.	Max.	Units	Comments
Supply range for regulator	2.1		3.6	V	VDD_PADS
Regulated output	1.7	1.8	1.9	V	
PSRR			- 40	dB	@100KHz
Supplied current	0		50	mA	
Current		200		μA	No load current (bandgap, regulator, feedback)
Quiescent current		10		nA	

6 Programming and Debug Interface (SIF Module)

SIF is a synchronous serial interface developed by Cambridge Consultants Ltd. It is the primary programming and debug interface of the EM250. The SIF module allows external devices to read and write memory-mapped registers in real-time without changing the functionality or timing of the XAP2b core. See document AN695, *PCB Design with an EM250*, for the PCB-level design details regarding the implementation of the SIF interface.

The EM250 pins involved in the SIF Interface:

- nSIF_LOAD
- SIF_CLK
- SIF_MOSI
- SIF_MISO
- nRESET

In addition, the VDD_PADS and Ground Net are required for external voltage translation and buffering of the SIF Signals.

The SIF interface provides the following:

- PCB production test interface via Virtual UART and a Debug Adapter (ISA)
- Programming and debug interface during EmberZNet PRO Application Development

In order to achieve the deep sleep currents specified in Table 5, a pull-down resistor must be connected to the SIF_MOSI pad. In addition, Silicon Labs recommends a pull-up resistor to be placed on the nSIF_LOAD net in order to prevent noise from coupling onto the Signal. Both of these recommendations are documented within the EM250 Reference designs.

When developing application-specific manufacturing test procedures, Silicon Labs recommends the designer refer to document AN700, *Manufacturing Test Guidelines*.

This document provides more detail regarding importance of designing the proper SIF interface as well as different tools offered by Silicon Labs for Production Programming on the MFG Production Line.

7 Typical Application

Figure 16 illustrates the typical application circuit, and Table 39 contains an example Bill of Materials for the off-chip components required by the EM250.

Note: The circuit shown in Figure 16 is for illustrations purposes only as it does not contain the decoupling capacitors for the different VDD nets. For a complete reference design, please download one of the Silicon Labs Reference Designs from the Silicon Labs website (www.silabs.com/zigbee).

The Balun provides the impedance transformation from the antenna to the EM250 for both TX and RX modes. Silicon Labs has developed reference designs based upon two balun topologies, a monolithic (ceramic) balun and a LC Lattice Balun.

The harmonic filter (L2, C2 and C3) provides additional suppression of the second harmonic, which increases the margin over the FCC limit.

The 24MHz crystal with loading capacitors is required and provides the high frequency source for the EM250. The 32.768kHz crystal generates the clock source for the Sleep Timer, but it is not mandatory as the internal RC Oscillator can be used.

The RC debounce filter (R4 and C9) is suggested to improve the noise immunity of the RESET logic (Pin 13).

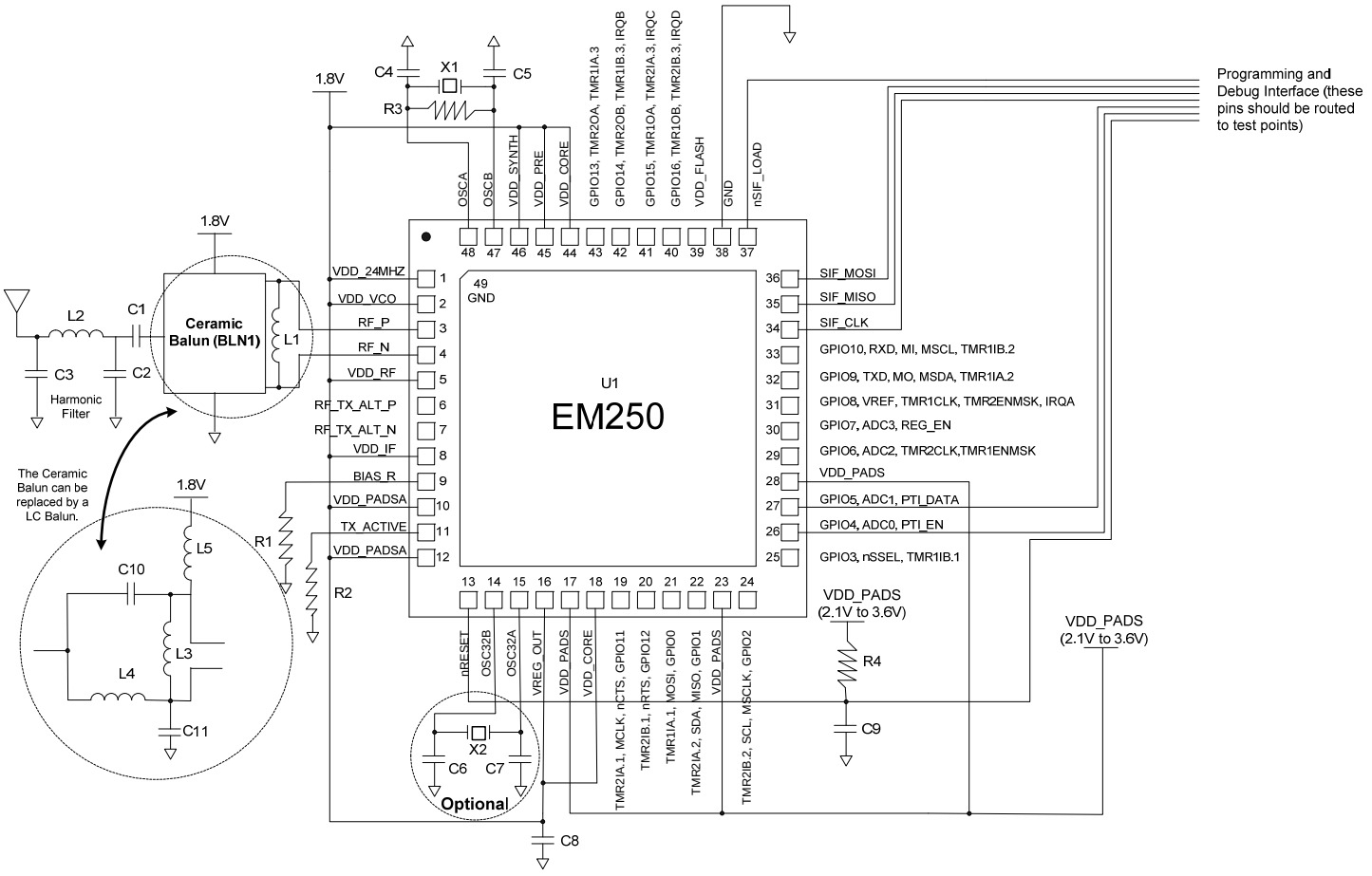


Figure 16. Typical Application Circuit

Table 39 contains a typical Bill of Materials for both the Ceramic and Lattice Balun Application circuits shown in Figure 16. The information within this table should be used for a rough cost analysis. It does not contain the decoupling capacitors. For a more detailed BOM, please refer to one of the Silicon Labs EM250-based reference designs.

Table 39. Bill of Materials for Figure 16

Item	Qty	Reference	Description	Manufacturer
1	1	C1	CAPACITOR, 8.2PF, 50V, NPO, 0402	<not specified>
2	2	C2,C3	CAPACITOR, 0.5PF, 50V, NPO, 0402	<not specified>
3	1	C7	CAPACITOR, 27PF, 50V, NPO, 0402	<not specified>
4	1	C6	CAPACITOR, 18PF, 50V, NPO, 0402	<not specified>
5	2	C4,C5	CAPACITOR, 27PF, 50V, NPO, 0402	<not specified>
6	1	C8	CAPACITOR, 10UF, 10V, TANTALUM, 3216 (SIZE A)	<not specified>
7	1	C9	CAPACITOR, 10PF, 5V, NPO, 0402	<not specified>
8	2	C10, C11	CAPACITOR, 0.75PF, ± 0.25pF, 50V, 0402, NPO	<not specified>
9	1	L1	INDUCTOR, 2.7NH, ± 0.1nH, 0603, MULTILAYER	MURATA LQG18HN2N7
10	1	L2	INDUCTOR, 3.3NH, ± 0.1nH, 0603, MULTILAYER	MURATA LQG18HN3N3
11	1	L3	INDUCTOR, 5.8NH, ± 0.1nH, 0402, WIREWOUND	MURATA LQW15AN5N8C00
12	2	L4, L5	INDUCTOR, 5.1NH, ± 0.1nH, 0603, MULTILAYER	MURATA LPQ15MN5N1B02
13	1	R1	RESISTOR, 169 KOHM, 1%, 0402	<not specified>
14	1	R2	RESISTOR, 100 KOHM, 5%, 0402	<not specified>
15	1	R3	RESISTOR, 3.3 KOHM, 5%, 0402	<not specified>
16	1	R4	RESISTOR, 10 KOHM, ± 5%, 0402	<not specified>
17	1	U1	EM250 SINGLE-CHIP ZIGBEE/802.15.4 SOLUTION	SILICON LABS EM250
18	1	X1	CRYSTAL, 24.000MHZ, ±10PPM TOLERANCE, ± 25PPM STABILITY, 18PF, - 40 TO + 85C	ILSI ILCX08-JG5F18-24.000MHZ
19	1	X2 (Optional)	CRYSTAL, 32.768KHZ, ± 20PPM TOLERANCE, 12.5PF, - 40 TO + 85C	ILSI IL3X-HX5-12.5-32.768KHZ
20	1	BLN1	BALUN, CERAMIC	TDK HHM1521

8 Mechanical Details

The EM250 package is a plastic 48-pin QFN that is 7 mm x 7 mm x 0.9 mm. Figure 17 and Figure 18 illustrate the Malaysia (SCM) and Taiwan (AESCL) package mechanical drawings, respectively.

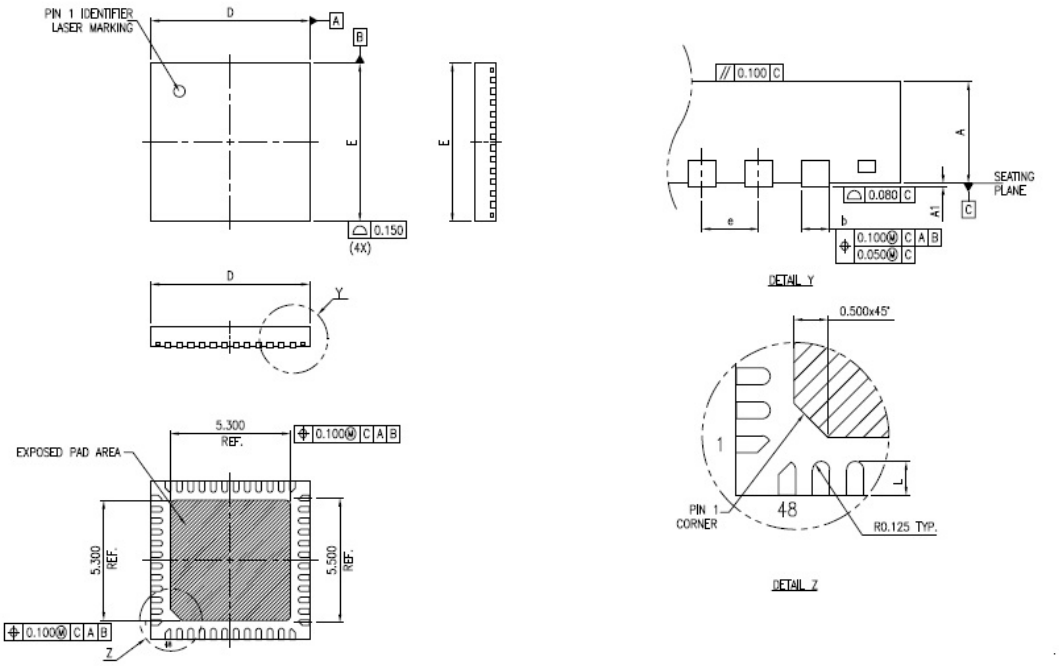
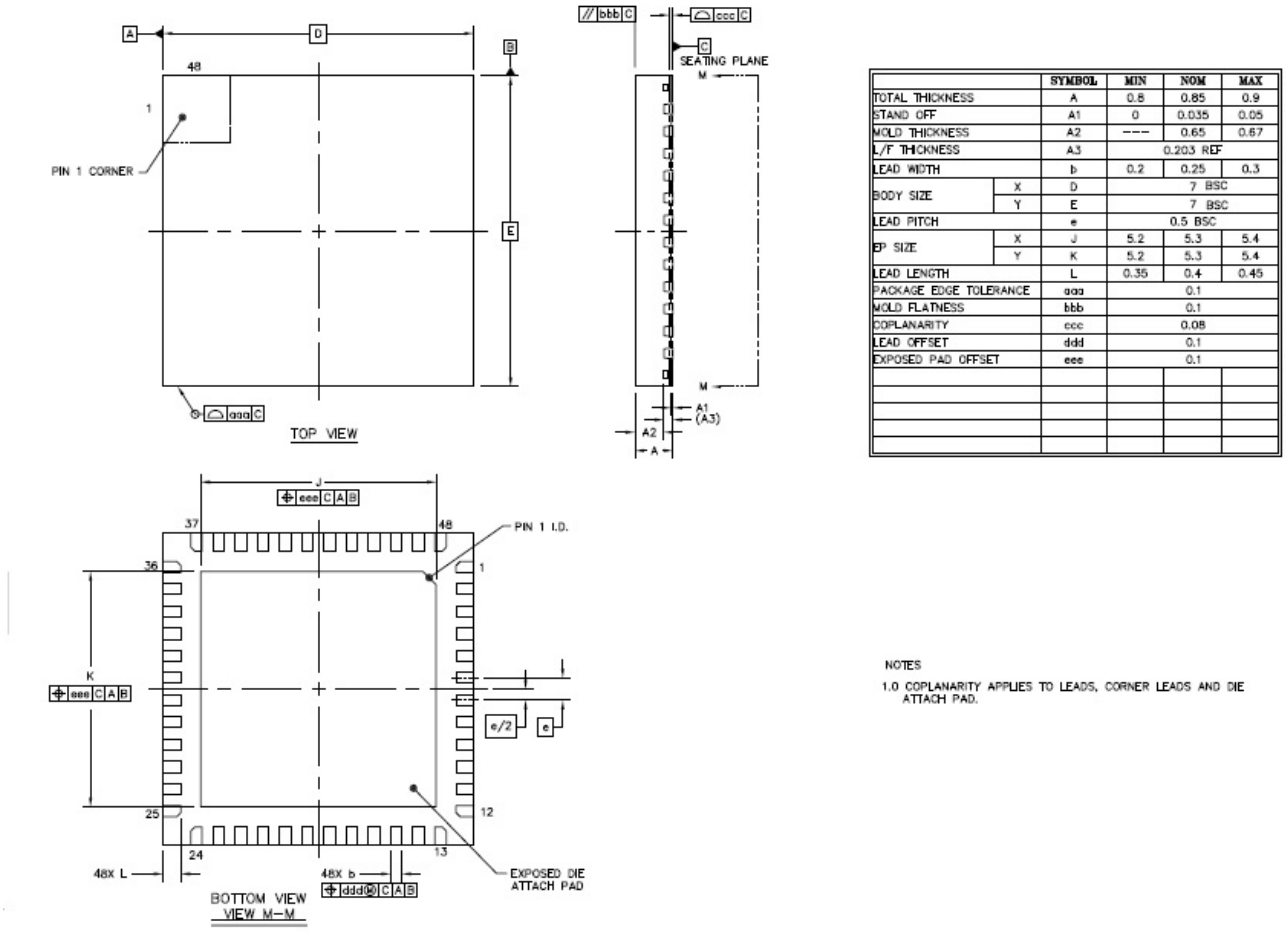


Figure 17: EM250 Malaysia (SCM) Package Drawing



		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	0.8	0.85	0.9
STAND OFF		A1	0	0.035	0.05
MOLD THICKNESS		A2	---	0.65	0.67
L/F THICKNESS		A3		0.203 REF	
LEAD WIDTH		b	0.2	0.25	0.3
BODY SIZE	X	D		7 BSC	
	Y	E		7 BSC	
LEAD PITCH		e		0.5 BSC	
EP SIZE	X	J	5.2	5.3	5.4
	Y	K	5.2	5.3	5.4
LEAD LENGTH		L	0.35	0.4	0.45
PACKAGE EDGE TOLERANCE		aaa		0.1	
MOLD FLATNESS		bbb		0.1	
COPLANARITY		ccc		0.08	
LEAD OFFSET		ddd		0.1	
EXPOSED PAD OFFSET		eee		0.1	

Figure 18. EM250 Taiwan (AESCL) Package Drawing

Table 40. Package Dimensions

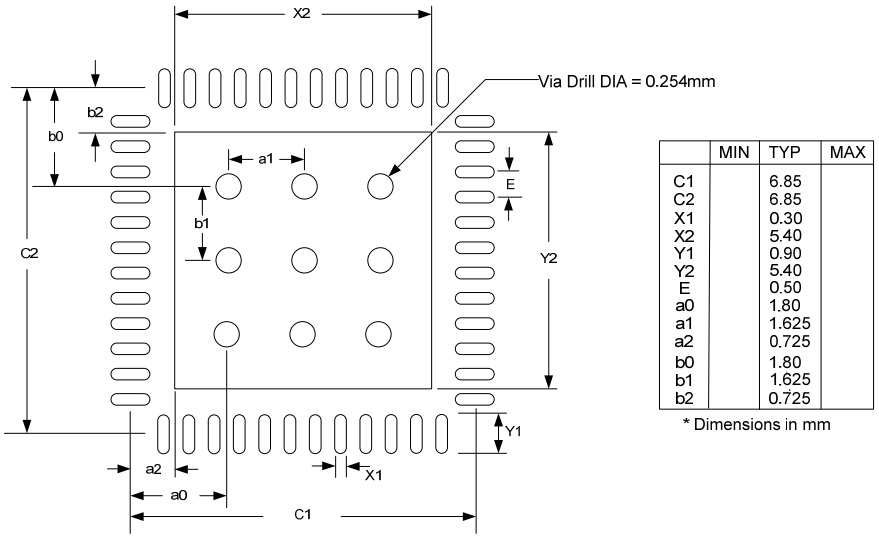
Package type	SCM 48QFN 7x7			ASECL 48QFN 7x7		
Dimensions	Minimum (mm)	Maximum (mm)	Nominal (mm)	Minimum (mm)	Maximum (mm)	Nominal (mm)
Package thickness	0.80	1.00	0.90	0.80	0.90	0.85
Lead Length	0.40	0.60	0.50	0.35	0.45	0.40

9 QFN48 Footprint Recommendations

Figure 19 demonstrates the IPC-7351 recommended PCB Footprint for the EM250 (QFN50P700X700X90-49N). A ground pad in the bottom center of the package forms a 49th pin.

A 3 x 3 array of non-thermal vias should connect the EM250 decal center shown in Figure 19 to the PCB ground plane through the ground pad. In order to properly solder the EM250 to the footprint, the Paste Mask layer should have a 3 x 3 array of circular openings at 1.015mm diameter spaced approximately 1.625mm (center to center) apart, as shown in Figure 20. This will cause an evenly distributed solder flow and coplanar attachment to the PCB. The solder mask layer (illustrated in Figure 21) should be the same as the copper layer for the EM250 footprint.

For more information on the package footprint, please refer to the Silicon Labs *EM250 Reference Design*.



	MIN	TYP	MAX
C1		6.85	
C2		6.85	
X1		0.30	
X2		5.40	
Y1		0.90	
Y2		5.40	
E		0.50	
a0		1.80	
a1		1.625	
a2		0.725	
b0		1.80	
b1		1.625	
b2		0.725	

* Dimensions in mm

Figure 19. PCB Footprint for the EM250

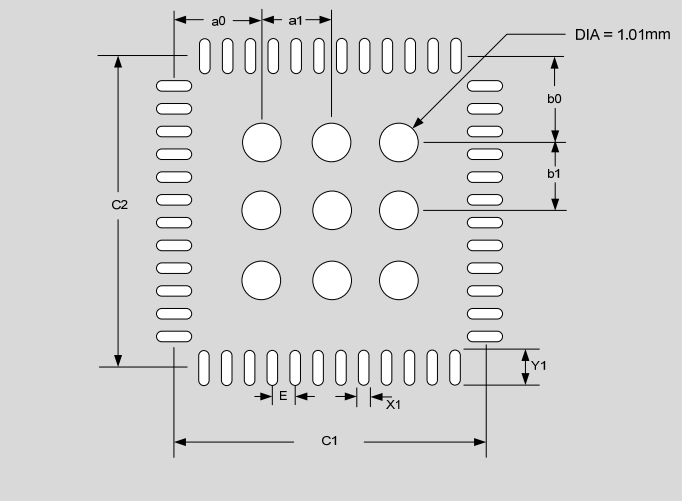


Figure 20. Paste Mask Dimensions

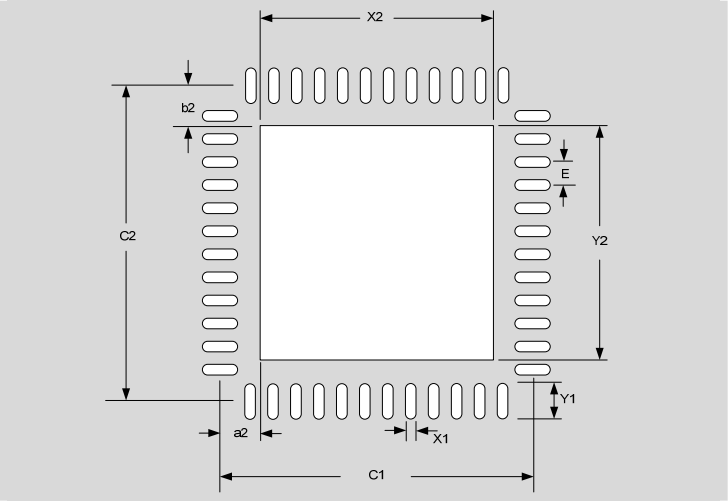


Figure 21. Solder Mask Dimensions

10 Part Marking

Figure 22 and Figure 23 show the part marking for the EM250. The circle in the top corner indicates pin 1. Pins are numbered counter-clockwise from Pin 1 with 12 pins per package edge.

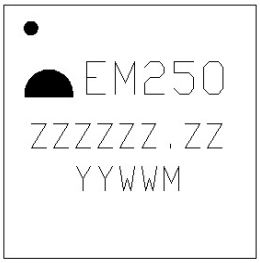


Figure 22. Stats Chippac Malaysia Part Marking for EM250

where:

- ZZZZZZ.ZZ defines the production lot code.
- YYWW defines the year and week assembled.
- M indicates Malaysia is the country of origin.

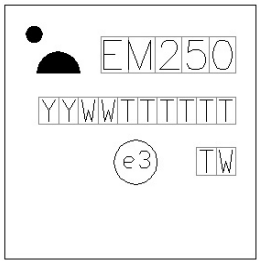


Figure 23. ASE Chung Li Part Marking for EM250

- YYWW defines the year and week assembled
- TTTTTT defines the manufacturing code
- e3 is the logo for 100% Sn (tin finish lead free) part
- TW indicates Taiwan is the country of origin

11 Ordering Information

Use the following part numbers to order the EM250:

- EM250-RTR Reel, RoHS – contains 2000 units/reel

EM250-RTR Reel conforms to EIA specification 481.

To order parts, contact Silicon Labs at 1+(877) 444-3032, or find a sales office or distributor on our website, www.silabs.com.

12 Shipping Box Label

Silicon Labs includes the following information on each tape and reel box label (EM250-RTR):

- Package
- Device Type
- Quantity (Bar coded)
- Box ID (Bar coded)
- Lot Number (Bar coded)
- Date Code (Bar coded)

Figure 24 depicts the label position on the box. As shown in this figure, there can be up to two date codes in a single tape and reel.

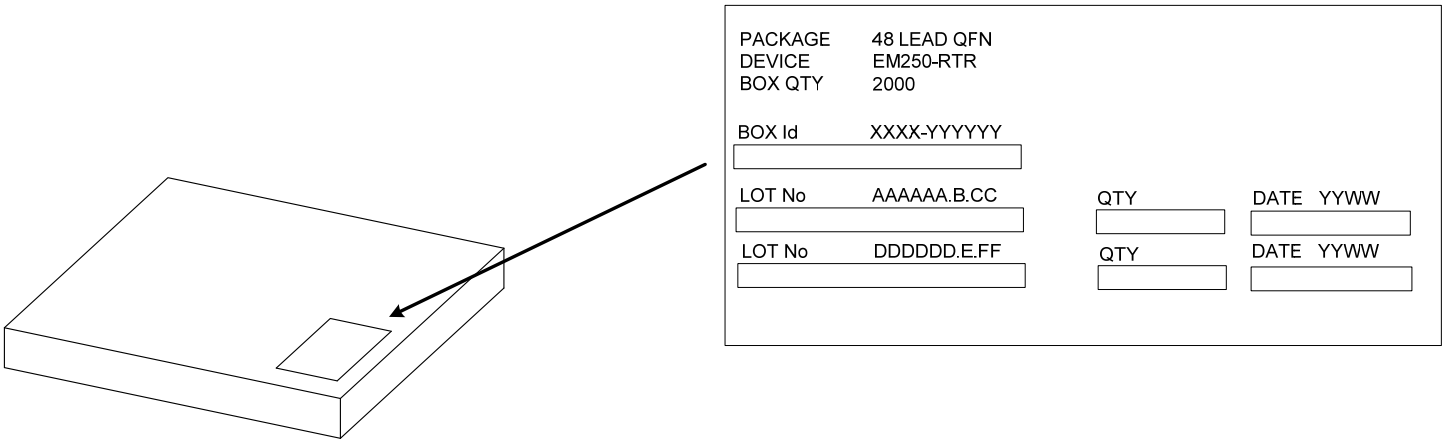


Figure 24. Contents Label

13 Register Address Table

Table 41 provides the address, reset value, and description of the registers in the EM250. These registers are accessible by the application (user).

Table 41. Register Addresses

Block:	SERIAL	4400–44B6 SC1 and SC2 control and status registers		
Address	**Name**	**Type**	**Reset**	
4400	SC2_RXBEGA	RW	6000	Rx DMA start address A
4402	SC2_RXENDA	RW	6000	Rx DMA end address A
4404	SC2_RXBEGB	RW	6000	Rx DMA start address B
4406	SC2_RXENDB	RW	6000	Rx DMA end address B
4408	SC2_TXBEGA	RW	6000	Tx DMA start address A
440A	SC2_TXENDA	RW	6000	Tx DMA end address A
440C	SC2_TXBEGB	RW	6000	Tx DMA start address B
440E	SC2_TXENDB	RW	6000	Tx DMA end address B
4410	SC2_RXCNTA	R	0000	Rx DMA Buffer A byte count
4412	SC2_RXCNTB	R	0000	Rx DMA Buffer B byte count
4414	SC2_TXCNT	R	0000	Tx DMA Buffer count
4416	SC2_DMASTAT	R	0000	DMA status
4418	SC2_DMACTRL	RW	0000	DMA control
441A	SC2_RXERRA	R	0000	Rx DMA Buffer A first error marker
441C	SC2_RXERRB	R	0000	Rx DMA Buffer B first error marker
441E	SC2_DATA	RW	0000	SC2 data
4420	SC2_SPISTAT	R	0000	SC2 SPI status
4422	SC2_I2CSTAT	R	0000	SC2 I^2C status
4426	SC2_I2CCTRL1	RW	0000	SC2 I^2C control 1
4428	SC2_I2CCTRL2	RW	0000	SC2 I^2C control 2
442A	SC2_MODE	RW	0000	SC2 Mode control
442C	SC2_SPICFG	RW	0000	SC2 SPI control
4430	SC2_RATELIN	RW	0000	SC2 Linear Component of Clock Rate
4432	SC2_RATEEXP	RW	0000	SC2 Exponential Component of Clock Rate
4480	SC1_RXBEGA	RW	6000	Rx DMA start address A
4482	SC1_RXENDA	RW	6000	Rx DMA end address A
4484	SC1_RXBEGB	RW	6000	Rx DMA start address B
4486	SC1_RXENDB	RW	6000	Rx DMA end address B
4488	SC1_TXBEGA	RW	6000	Tx DMA start address A
448A	SC1_TXENDA	RW	6000	Tx DMA end address A
448C	SC1_TXBEGB	RW	6000	Tx DMA start address B
448E	SC1_TXENDB	RW	6000	Tx DMA end address B

Block:	SERIAL	4400–44B6 SC1 and SC2 control and status registers		
Address	**Name**	**Type**	**Reset**	
4490	SC1_RXCNTA	R	0000	Rx DMA Buffer A byte count
4492	SC1_RXCNTB	R	0000	Rx DMA Buffer B byte count
4494	SC1_TXCNT	R	0000	Tx DMA Buffercount
4496	SC1_DMASTAT	R	0000	DMA status
4498	SC1_DMACTRL	RW	0000	DMA control
449A	SC1_RXERRA	R	0000	Rx DMA Buffer A first error marker
449C	SC1_RXERRB	R	0000	Rx DMA Buffer B first error marker
449E	SC1_DATA	RW	0000	SC1 data
44A0	SC1_SPISTAT	R	0000	SC1 SPI status
44A2	SC1_I2CSTAT	R	0000	SC1 I^2C status
44A4	SC1_UARTSTAT	R	0040	SC1 UART status
44A6	SC1_I2CCTRL1	RW	0000	SC1 I^2C control 1
44A8	SC1_I2CCTRL2	RW	0000	SC1 I^2C control 2
44AA	SC1_MODE	RW	0000	SC1 Mode control
44AC	SC1_SPICFG	RW	0000	SC1 SPI control
44AE	SC1_UARTCFG	RW	0000	SC1 UART control
44B0	SC1_RATELIN	RW	0000	SC1 Linear Component of Clock Rate
44B2	SC1_RATEEXP	RW	0000	SC1 Exponential Component of Clock Rate
44B4	SC1_UARTPER	RW	0000	SC1 Baud Rate Period
44B6	SC1_UARTFRAC	RW	0000	SC1 Baud Rate Fraction

Block:	TIMER1	4500-4514 Timer 1 control and status registers		
Address	**Name**	**Type**	**Reset**	
4500	TMR1_CNT	RW	0000	Timer 1 counter
4502	TMR1_CAPA	R	0000	Timer 1 capture A
4504	TMR1_CAPB	R	0000	Timer 1 capture B
4506	TMR1_TOP	RW	FFFF	Timer 1 threshold
4508	TMR1_CMPA	RW	0000	Timer 1 compare A
450A	TMR1_CMPB	RW	0000	Timer 1 compare B
450C	TMR1_CFG	RW	0000	Timer 1 config
450E	TMR1_CMPCFGA	RW	0000	Timer 1 output A config
4510	TMR1_CMPCFGB	RW	0000	Timer 1 output B config
4512	TMR1_CAPCFGA	RW	0000	Timer 1 input capture A config
4514	TMR1_CAPCFGB	RW	0000	Timer 1 input capture B config

Block:	TIMER2	4580–4594 Timer 2 control and status registers		
Address	**Name**	**Type**	**Reset**	
4580	TMR2_CNT	RW	0000	Timer 2 counter
4582	TMR2_CAPA	R	0000	Timer 2 capture A
4584	TMR2_CAPB	R	0000	Timer 2 capture B
4586	TMR2_TOP	RW	FFFF	Timer 2 threshold
4588	TMR2_CMPA	RW	0000	Timer 2 compare A.
458A	TMR2_CMPB	RW	0000	Timer 2 compare B
458C	TMR2_CFG	RW	0000	Timer 2 config
458E	TMR2_CMPCFGA	RW	0000	Timer 2 output A config
4590	TMR2_CMPCFGB	RW	0000	Timer 2 output B config
4592	TMR2_CAPCFGA	RW	0000	Timer 2 input capture A config
4594	TMR2_CAPCFGB	RW	0000	Timer 2 input capture B config

Block:	EVENT	4600-4638 Event control and status registers		
Address	**Name**	**Type**	**Reset**	
4600	INT_FLAG	RW	0000	Interrupt source
4602	INT_MISS	RW	0000	Interrupt event missed
460C	INT_SC1FLAG	RW	0000	SC1 Interrupt source
460E	INT_SC2FLAG	RW	0000	SC2 Interrupt source
4610	INT_GPIOFLAG	RW	0000	GPIO Interrupt source
4614	INT_TMRFLAG	RW	0000	Timer Interrupt source
4618	INT_EN	RW	0000	Interrupt Enable
461A	INT_CFG	RW	0000	Interrupt config
4624	INT_SC1CFG	RW	0000	SC1 Interrupt config
4626	INT_SC2CFG	RW	0000	SC2 Interrupt config
4628	INT_GPIOCFG	RW	0000	GPIO Interrupt config
462C	INT_TMRCFG	RW	0000	Timer Interrupt config
4630	GPIO_INTCFGA	RW	0000	GPIO Interrupt A config
4632	GPIO_INTCFGB	RW	0000	GPIO Interrupt B config
4634	GPIO_INTCFGC	RW	0000	GPIO Interrupt C config
4636	GPIO_INTCFGD	RW	0000	GPIO Interrupt D config
4638	INT_SWCTRL	RW	0000	Software interrupt

Block:	GPIO	4700–4728 General purpose IO control and data		
Address	**Name**	**Type**	**Reset**	
4700	GPIO_INH	R	0000	GPIO input data-upper bits
4702	GPIO_INL	R	0000	GPIO input data-lower bits
4704	GPIO_OUTH	RW	0000	GPIO output data-upper bits
4706	GPIO_OUTL	RW	0000	GPIO output data-lower bits
4708	GPIO_SETH	RW	0000	GPIO set output data-upper bits
470A	GPIO_SETL	W	0000	GPIO set output data-lower bits
470C	GPIO_CLRH	RW	0000	GPIO clear output data-upper bits
470E	GPIO_CLRL	W	0000	GPIO clear output data-lower bits
4710	GPIO_DBG	RW	0000	GPIO debug
4712	GPIO_CFG	RW	2000	GPIO config
4714	GPIO_DIRH	RW	0000	GPIO output enable-upper bits
4716	GPIO_DIRL	RW	0000	GPIO output enable-lower bits
4718	GPIO_DIRSETH	RW	0000	GPIO set enable-upper bits
471A	GPIO_DIRSETL	W	0000	GPIO set enable-lower bits
471C	GPIO_DIRCLRH	RW	0000	GPIO clear enable-upper bits
471E	GPIO_DIRCLRL	W	0000	GPIO clear enable-lower bits
4720	GPIO_PDH	RW	0000	GPIO pin pull-down enable-upper bits
4722	GPIO_PDL	RW	0000	GPIO pin pull-down enable-lower bits
4724	GPIO_PUH	RW	0000	GPIO pin pull-up enable-upper bits
4726	GPIO_PUL	RW	0000	GPIO pin pull-up enable-lower bits
4728	GPIO_WAKEL	RW	0000	GPIO wakeup monitor register

Block:	ADC	4900-4902 ADC control and status		
Address	**Name**	**Type**	**Reset**	
4900	ADC_DATA	R	0000	ADC data
4902	ADC_CFG	RW	0000	ADC config

14 Abbreviations and Acronyms

Acronym/Abbreviation	Meaning
ACR	Adjacent Channel Rejection
AES	Advanced Encryption Standard
AGC	Automatic gain control
CBC-MAC	Cipher Block Chaining—Message Authentication Code
CCA	Clear Channel Assessment
CCM	Counter with CBC-MAC Mode for AES encryption
CCM*	Improved Counter with CBC-MAC Mode for AES encryption
CSMA	Carrier Sense Multiple Access
CTR	Counter Mode
EEPROM	Electrically Erasable Programmable Read Only Memory
ESD	Electro Static Discharge
ESR	Equivalent Series Resistance
FFD	Full Function Device (ZigBee)
FIA	Flash Information Area
GPIO	General Purpose I/O (pins)
HF	High Frequency (24MHz)
I^2C	Inter-Integrated Circuit bus
IDE	Integrated Development Environment
IF	Intermediate Frequency
IP3	Third order Intermodulation Product
ISR	Interrupt Service Routine
kB	Kilobyte
kbps	kilobits/second
LF	Low Frequency
LNA	Low Noise Amplifier
LQI	Link Quality Indicator
MAC	Medium Access Control
MSL	Moisture Sensitivity Level
Msps	Mega samples per second
O-QPSK	Offset-Quadrature Phase Shift Keying
PA	Power Amplifier
PER	Packet Error Rate
PHY	Physical Layer
PLL	Phase-Locked Loop
POR	Power-On-Reset
PSD	Power Spectral Density

Acronym/Abbreviation	Meaning
PSRR	Power Supply Rejection Ratio
PTI	Packet Trace Interface
PWM	Pulse Width Modulation
RoHS	Restriction of Hazardous Substances
RSSI	Receive Signal Strength Indicator
SFD	Start Frame Delimiter
SIF	Serial Interface
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
VCO	Voltage Controlled Oscillator
VDD	Voltage Supply

15 References

- ZigBee Specification (www.zigbee.org; ZigBee Document 053474) (ZigBee Alliance membership required)
- ZigBee-PRO Stack Profile (www.zigbee.org; ZigBee Document 074855) (ZigBee Alliance membership required)
- ZigBee Stack Profile (www.zigbee.org; ZigBee Document 064321) (ZigBee Alliance membership required)
- Bluetooth Core Specification v2.1 (http://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=241363)
- IEEE 802.15.4-2003 (http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf)
- IEEE 802.11g (standards.ieee.org/getieee802/download/802.11g-2003.pdf)

16 Revision History

Revision	Location	Description of Change
1.1	Chapter 8 and Chapter 11 updated.	SCM package updated and AESCL package added.
V	Datasheet	Updated cross references to Silicon Labs documentation.
U	Chapter 12	Updated Figure 24.
T	Datasheet	Rebranding to Silicon Labs.
S	Figure 17. Package Drawing for EM250s assembled in Wales or China and Figure 19. PCB Footprint for the EM250	Corrected dimensions.
	Chapter 8 (new Figure 18. Package Drawing for EM250s assembled in Malaysia) and Chapter 11	Added specifications for Malaysia manufacture.
	SCn_SPICFG, SCn_RATEEXPO, and SCn_RATELIN register descriptions	Clarified the circumstances under which the registers can be changed.
	Section 5.3.1.2, SPI Slave Mode	Clarified state of MISO pin when nSSEL signal is deasserted; added information about transmit behavior.
	Chapter 16	Updated reference information.
R	Table 1. Pin Descriptions	Updated the following pin descriptions: 1, 2, 5, 8, 10, 12, 18, 30, 39, 44, 45, and 46.
	Section 5.7, Integrated Voltage Regulator	Updated section 5.7 to remove support for an external 1v8 regulator.
	Chapter 11, Ordering Information, **Error! Reference source not found.**	Removed Tray information and added Tube information according to PCN 121-1006-000.
Q	Figure 14. Timer Output Generation Mode Example—Saw Tooth, Non-inverting	Corrected timer mode 5.
	Figure 15. Timer Output Generation Mode Example—Alternating, Non-inverting	Corrected timer mode 5.
	Figure 19. PCB Footprint for the EM250	Corrected via dimensions.
	Figure 20. Paste Mask Dimensions	Corrected via dimensions.
P	Table 5. DC Characteristics	Added nRESET active current parameter.
	Section 4.1.1, RX Baseband	Updated AGC information.
	Section 5.5, ADC Module	Added ADC information.
	Table 33. ADC Module Key Parameters	Added new table.
	Chapter 9, QFN48 Footprint Recommendations	Added new chapter and figures.
	Chapter **Error! Reference source not found.**, **Error! Reference source not found.**	Added new chapter and figures.
	Table 39. Bill of Materials for Figure 16	Updated BOM information.
O	Figure 21. Contents Label	Added new figure.
N	Table 1. Pin Descriptions	Added more detail around the SIF Interface pins.
	Table 2. Absolute Maximum Ratings	Added a Maximum Input level in dBM.
	Table 7. Receive Characteristics	Modified the 802.11g rejection from 40 to 35dB.
	Section 5.1, GPIO	Added more detail around the GPIO output configuration.

Revision	Location	Description of Change
	Section 5.2.4, Registers	Corrected errors in the description of the SC1_TXCNT and SC2_TXCNT registers.
	Figure 19. PCB Footprint for the EM250	Corrected the PCB footprint dimensions.
	Chapter 9, Part Marking	Added new chapter and figure.
	Chapter 10, Ordering Information	Added tape and reel information.