EM342

High-Performance, Integrated RF4CE System-on-Chip

Features

- 32-bit ARM® Cortex -M3 processor
- 2.4 GHz IEEE 802.15.4-2003 transceiver & lower MAC
- 128 kB flash, with optional read protection
- 12 kB RAM memory
- AES128 encryption accelerator
- UART/SPI serial communications
- 16 GPIOs

Industry-leading ARM® Cortex -M3 processor

- Leading 32-bit processing performance
- Highly efficient Thumb-2 instruction set
- Operation at 6, 12, or 24 MHz
- Flexible Nested Vectored Interrupt Controller

Low power consumption, advanced management

- RX Current (w/ CPU): 26 mA
- TX Current (w/ CPU, +3 dBm TX): 31 mA
- Low deep sleep current, with retained RAM and GPIO: 400 nA without/800 nA with sleep timer
- Low-frequency internal RC oscillator for low-power sleep timing
- High-frequency internal RC oscillator for fast (110 µs) processor start-up from sleep

Exceptional RF Performance

- Normal mode link budget up to 103 dB; configurable up to 110 dB
- –100 dBm normal RX sensitivity; configurable to –102 dBm (1% PER, 20 byte packet)
- Configurable up to +8 dBm
- Robust Wi-Fi and Bluetooth coexistence

Innovative network and processor debug

- Packet Trace Port for non-intrusive packet trace with Ember development tools
- Serial Wire/JTAG interface
- Standard ARM debug capabilities: Flash Patch & Breakpoint; Data Watchpoint & Trace; Instrumentation Trace Macrocell

Application Flexibility

- Single voltage operation: 2.1–3.6 V with internal 1.8 and 1.25 V regulators
- Optional 32.768 kHz crystal for higher timer accuracy
- Low external component count with single 24 MHz crystal
- Support for external power amplifier
- Small 7x7 mm 48-pin QFN package

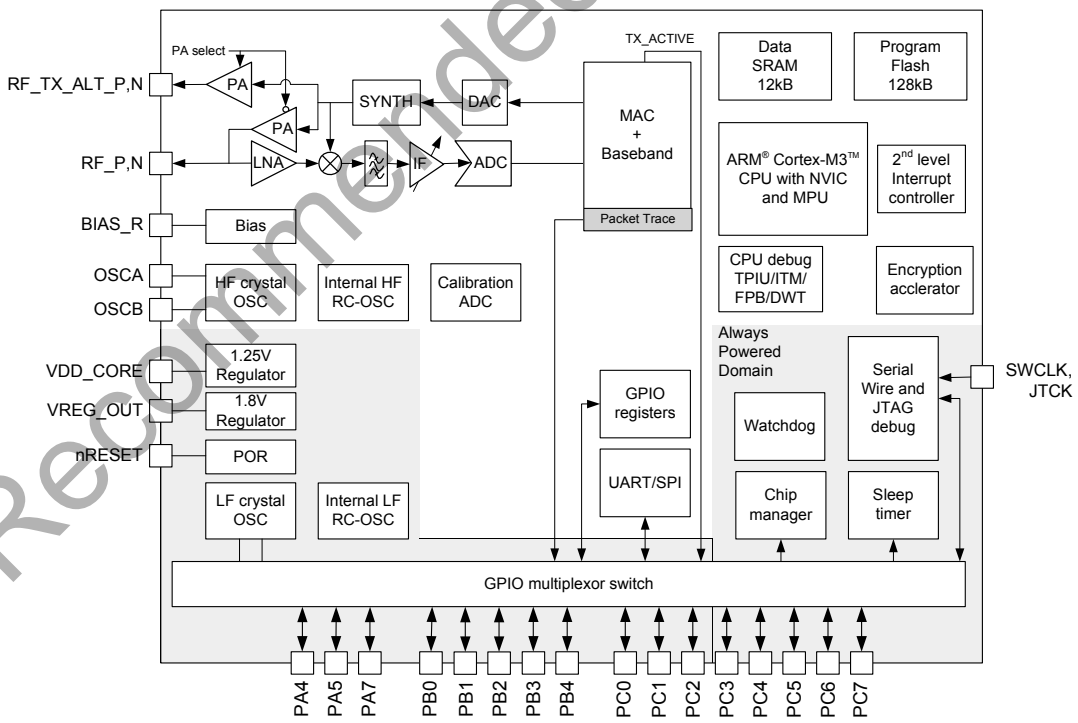


Table of Contents

1. Typical Application

Figure 1.1 illustrates the typical application circuit, and Table 1.1 contains an example bill of materials (BOM) for the off-chip components required by the EM342.

Note: The circuit shown in Figure 1.1 is for example purposes only, and the BOM is for budgetary quotes only. For a complete reference design, please download one of the latest Ember Hardware Reference Designs from the Silicon Labs website (www.silabs.com/zigbee-support).

The Balun provides an impedance transformation from the antenna to the EM342 for both TX and RX modes.

L1 tunes the impedance presented to the RF port for maximum transmit power and receive sensitivity.

The harmonic filter (L2, L3, C5, C6 and C9) provides additional suppression of the second harmonic, which increases the margin over the FCC limit.

The 24 MHz crystal Y1 with loading capacitors is required and provides the high-frequency crystal oscillator source for the EM342's main system clock. The 32.768 kHz crystal with loading capacitors generates a highly accurate low-frequency crystal oscillator for use with peripherals, but it is not mandatory as the low-frequency internal RC oscillator can be used.

Loading capacitance and ESR (C1 and R3) provides stability for the internal 1.8 V regulator.

Loading capacitance C2 provides stability for the internal 1.25 V regulator, no ESR is required because it is contained within the chip.

Resistor R1 reduces the operating voltage of the flash memory, this reduces current consumption and improves sensitivity by 1 dB when compared to not using it.

Various decoupling capacitors are required, these should be placed as close to their corresponding pins as possible. For values and locations see one of the latest reference designs.

An antenna matched to 50 Ω is required.

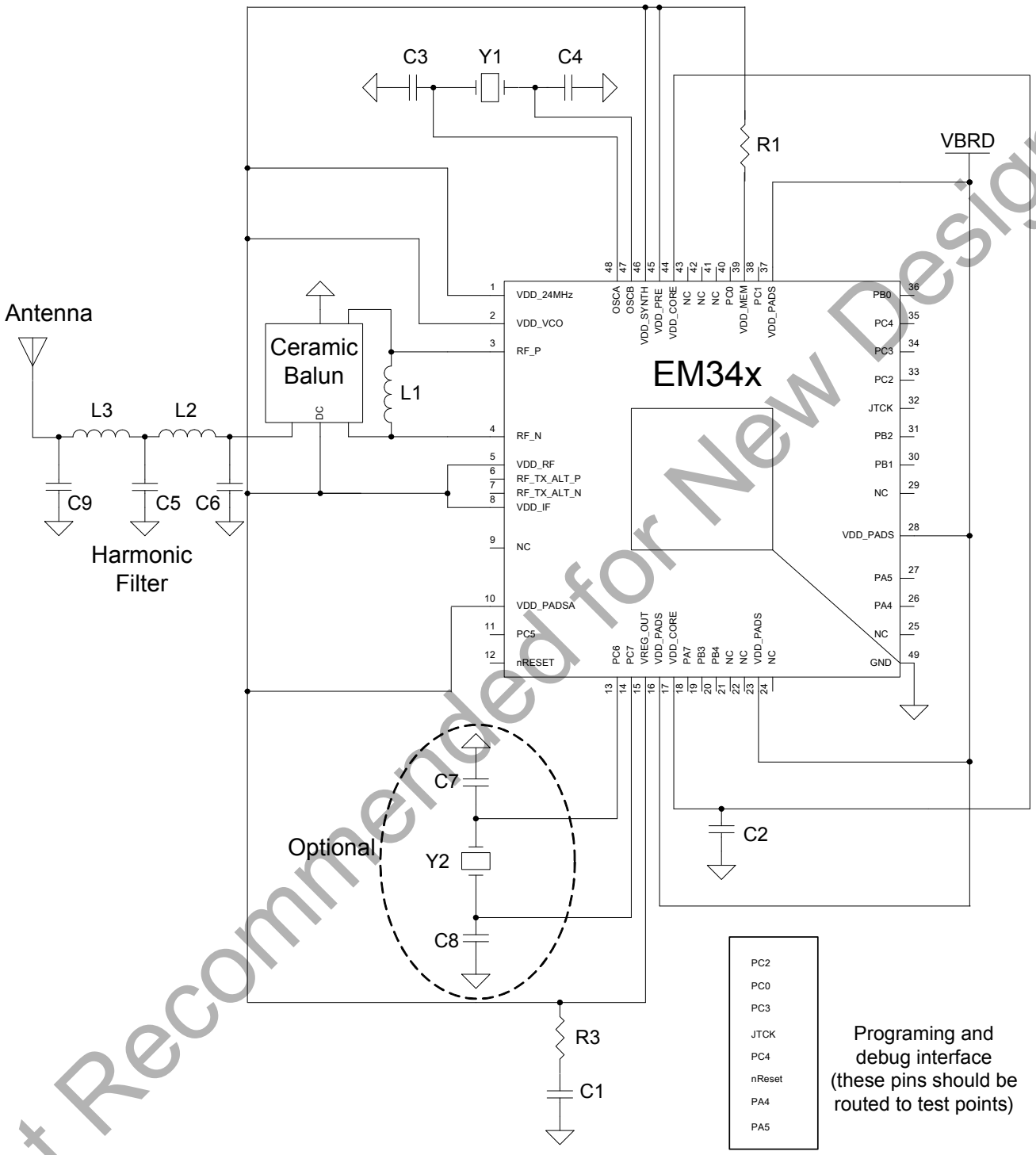


Figure 1.1. Typical Application Circuit

Table 1.1 contains a typical Bill of Materials for the application circuit shown in Figure 1.1. The information within this table should be used for a rough cost analysis. For a more detailed BOM, please refer to one of Ember EM357-based reference designs at the Silicon Labs website (www.silabs.com/zigbee-support).

Table 1.1. Bill of Materials for Typical Application Circuit

Item	Qty	Reference	Description	Manufacturer
1	1	C2	CAPACITOR, 1 µF, 6.3 V, X5R, 10%, 0402	<not specified>
2	1	C1	CAPACITOR, 2.2 µF, 10 V, X5R, 10%, 0603	<not specified>
3	1	C7	CAPACITOR, 22 pF, ±5%, 50 V, NPO, 0402	<not specified>
4	2	C3,C4	CAPACITOR, 18 pF, ±5%, 50 V, NPO, 0402	<not specified>
5	1	C8	CAPACITOR, 33 pF, ±5%, 50 V, NPO, 0402	<not specified>
6	2	C5, C9	CAPACITOR, 1 pF, ±0.25 pF, 50 V, 0402, NPO	<not specified>
7	1	C6	CAPACITOR, 1.8 pF, ±0.25 pF, 50 V, 0402, NPO	
8	1	L1	INDUCTOR, 5.1 nH, ±0.3 nH, 0402 MULTILAYER	Murata LQG15HS5N1
9	2	L2, L3	INDUCTOR, 2.7 nH, ±0.3 nH, 0402, MULTILAYER	Murata LQG15HS2N7
10	1	R1	RESISTOR, 10 Ω, 5%, 0402	<not specified>
11	1	R3	RESISTOR, 1 Ω, 5%, 0402	<not specified>
12	1	U1	EM342 SINGLE-CHIP ZIGBEE RF4CE SOLUTION	Ember EM342
13	1	Y1	CRYSTAL, 24.000 MHz, ± 25 ppm STABILITY OVER –40 to +85 ºC, 18 pF	ILSI, Abracon, KDS, Epson
14	1	Y2 (Optional)	CRYSTAL, 32.768 kHz, ±20 ppm INITIAL TOLERANCE AT +25ºC, 12.5 pF	Abracon, KDS, Epson
15	1	BLN1	BALUN, CERAMIC 50/100 Ω	Wurth 748421245 Johanson 2450BL15B100E Murata LDB212G4010C
16	1	ANT1	ANTENNA	Johanson 2450AT18B100E

2. Electrical Specifications

2.1. Absolute Maximum Ratings

Table 2.1 lists the absolute maximum ratings for the EM342.

Table 2.1. Absolute Maximum Ratings

Parameter	Test Condition	Min	Max	Unit
Regulator input voltage (VDD_PADS)		–0.3	+3.6	V
Analog, Memory and Core voltage (VDD_24MHZ, VDD_VCO, VDD_RF, VDD_IF, VDD_PADSA, VDD_MEM, 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 2.7)	RX signal into a lossless balun	—	+15	dBm
Voltage on any GPIO, SWCLK, nRESET, VREG_OUT		–0.3	VDD_PADS +0.3	V
Voltage on OSCA, OSCB, NC		–0.3	VDD_PADSA +0.3	V
Storage temperature		–40	+140	°C
Note: Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.				

2.2. Recommended Operating Conditions

Table 2.2 lists the rated operating conditions of the EM342.

Table 2.2. Operating Conditions

Parameter	Test Condition	Min	Typ	Max	Unit
Regulator input voltage (VDD_PADS)		2.1	—	3.6	V
Analog and memory input voltage (VDD_24MHZ, VDD_VCO, VDD_RF, VDD_IF, VDD_PADSA, VDD_MEM, VDD_PRE, VDD_SYNTH)		1.7	1.8	1.9	V
Core input voltage when supplied from internal regulator (VDD_CORE)		1.18	1.25	1.32	V
Core input voltage when supplied externally (VDD_CORE)		1.18	—	1.9	V
Operating temperature range, T_A		–40	—	+85	°C
Note: Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.					

2.3. Environmental Characteristics

Table 2.3 lists the rated environmental characteristics of the EM342.

Table 2.3. Environmental Characteristics

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
ESD (human body model)		On any pin	—	—	±2	kV
ESD (charged device model)		Non-RF pins	—	—	±400	V
ESD (charged device model)		RF pins	—	—	±225	V
Package Thermal Resistance*	θ_{JA}			27.1		°C/W
***Note:** Thermal resistance assumes multi-layer PCB with exposed pad soldered to a PCB board.						

2.4. DC Electrical Characteristics

Table 2.4 lists the dc electrical characteristics of the EM342.

Table 2.4. DC Characteristics

Measured on Silicon Labs' EM357 reference design with T_A = 25 °C and V_{DD} = 3 V, unless otherwise noted.

Parameter	Test Condition	Min	Typ	Max	Unit
Regulator input voltage (VDD_PADS)		2.1	—	3.6	V
Power supply range (VDD_MEM)	Regulator output or external input	1.7	1.8	1.9	V
Power supply range (VDD_CORE)	Regulator output	1.18	1.25	1.32	V
Deep Sleep Current					
Quiescent current, internal RC oscillator disabled	–40 °C, VDD_PADS=3.6 V	—	0.4	—	µA
	+25 °C, VDD_PADS=3.6 V	—	0.4	—	µA
	+85 °C, VDD_PADS=3.6 V	—	0.7	—	µA
Quiescent current, including internal RC oscillator	–40 °C, VDD_PADS=3.6 V	—	0.7	—	µA
	+25 °C, VDD_PADS=3.6 V	—	0.7	—	µA
	+85°C, VDD_PADS=3.6 V	—	1.1	—	µA
Quiescent current, including 32.768 kHz oscillator	–40 °C, VDD_PADS=3.6 V	—	0.8	—	µA
	+25 °C, VDD_PADS=3.6 V	—	1.0	—	µA
	+85 °C, VDD_PADS=3.6 V	—	1.5	—	µA
Quiescent current, including internal RC oscillator and 32.768 kHz oscillator	–40 °C, VDD_PADS=3.6 V	—	1.1	—	µA
	+25 °C, VDD_PADS=3.6 V	—	1.3	—	µA
	+85 °C, VDD_PADS=3.6 V	—	1.8	—	µA
Simulated deep sleep (debug mode) current	With no debugger activity	—	300	—	µA

Table 2.4. DC Characteristics (Continued)

Measured on Silicon Labs' EM357 reference design with T_A = 25 °C and V_{DD} = 3 V, unless otherwise noted.

Parameter	Test Condition	Min	Typ	Max	Unit
Reset Current					
Quiescent current, nRESET asserted	Typ at 25 °C/3.0 V Max at 85 °C/3.6 V	—	1.2	2.0	mA
Processor and Peripheral Currents					
ARM® Cortex™-M3, RAM, and flash memory	25 °C, 1.8 V memory and 1.25 V core ARM® Cortex™-M3 running at 12 MHz from crystal oscillator Radio and all peripherals off	—	6.5	—	mA
ARM® Cortex™-M3, RAM, and flash memory	25 °C, 1.8 V memory and 1.25 V core ARM® Cortex™-M3 running at 24 MHz from crystal oscillator Radio and all peripherals off	—	7.5	—	mA
ARM® Cortex™-M3, RAM, and flash memory sleep current	25 °C, 1.8 V memory and 1.25 V core ARM® Cortex™-M3 sleeping, CPU clock set to 12 MHz from the crystal oscillator Radio and all peripherals off	—	3.0	—	mA
ARM® Cortex™-M3, RAM, and flash memory sleep current	25 °C, 1.8 V memory and 1.25 V core ARM® Cortex™-M3 sleeping, CPU clock set to 6 MHz from the high frequency RC oscillator Radio and all peripherals off	—	2.0	—	mA
Serial controller current	For each controller at maximum data rate	—	0.2	—	mA
RX Current					
Radio receiver, MAC, and baseband	ARM® Cortex™-M3 sleeping, CPU clock set to 12 MHz	—	22.0	—	mA
Total RX current (= I_{Radio} receiver, MAC and baseband, CPU + IRAM, and Flash memory)	25 °C, VDD_PADS=3.0 V ARM® Cortex™-M3 running at 12 MHz	—	25.5	—	mA
	25 °C, VDD_PADS=3.0 V ARM® Cortex™-M3 running at 24 MHz	—	26.5	—	mA
Boost mode total RX current (= I_{Radio} receiver, MAC and baseband, CPU+ IRAM, and flash memory)	25 °C, VDD_PADS=3.0 V ARM® Cortex™-M3 running at 12 MHz	—	27.5	—	mA
	25 °C, VDD_PADS=3.0 V ARM® Cortex™-M3 running at 24 MHz	—	28.5	—	mA

Table 2.4. DC Characteristics (Continued)

Measured on Silicon Labs' EM357 reference design with T_A = 25 °C and V_{DD} = 3 V, unless otherwise noted.

Parameter	Test Condition	Min	Typ	Max	Unit
TX Current					
Radio transmitter, MAC, and baseband	25 °C and 1.8 V core; max. power out (+3 dBm typical) ARM® Cortex™-M3 sleeping, CPU clock set to 12 MHz	—	26.0	—	mA
Total TX current (= I_{Radio} transmitter, MAC and baseband, CPU + IRAM, and flash memory)	25 °C, VDD_PADS=3.0 V; maximum power setting (+8 dBm); ARM® Cortex™-M3 running at 12 MHz	—	42.5	—	mA
	25 °C, VDD_PADS=3.0 V; +3 dBm power setting; ARM® Cortex™-M3 running at 12 MHz	—	30.0	—	mA
	25 °C, VDD_PADS=3.0 V; 0 dBm power setting; ARM® Cortex™-M3 running at 12 MHz	—	27.5	—	mA
	25 °C, VDD_PADS=3.0 V; minimum power setting; ARM® Cortex™-M3 running at 12 MHz	—	21.5	—	mA
	25 °C, VDD_PADS=3.0 V; maximum power setting (+8 dBm); ARM® Cortex™-M3 running at 24 MHz	—	43.5	—	mA
	25 °C, VDD_PADS=3.0 V; +3 dBm power setting; ARM® Cortex™-M3 running at 24 MHz	—	31.0	—	mA
	25 °C, VDD_PADS=3.0 V; 0 dBm power setting; ARM® Cortex™-M3 running at 24 MHz	—	28.5	—	mA
	25 °C, VDD_PADS=3.0 V; minimum power setting; ARM® Cortex™-M3 running at 24 MHz	—	22.5	—	mA

Figure 2.1 shows the variation of current in transmit mode (with the ARM® Cortex™-M3 running at 12 MHz).

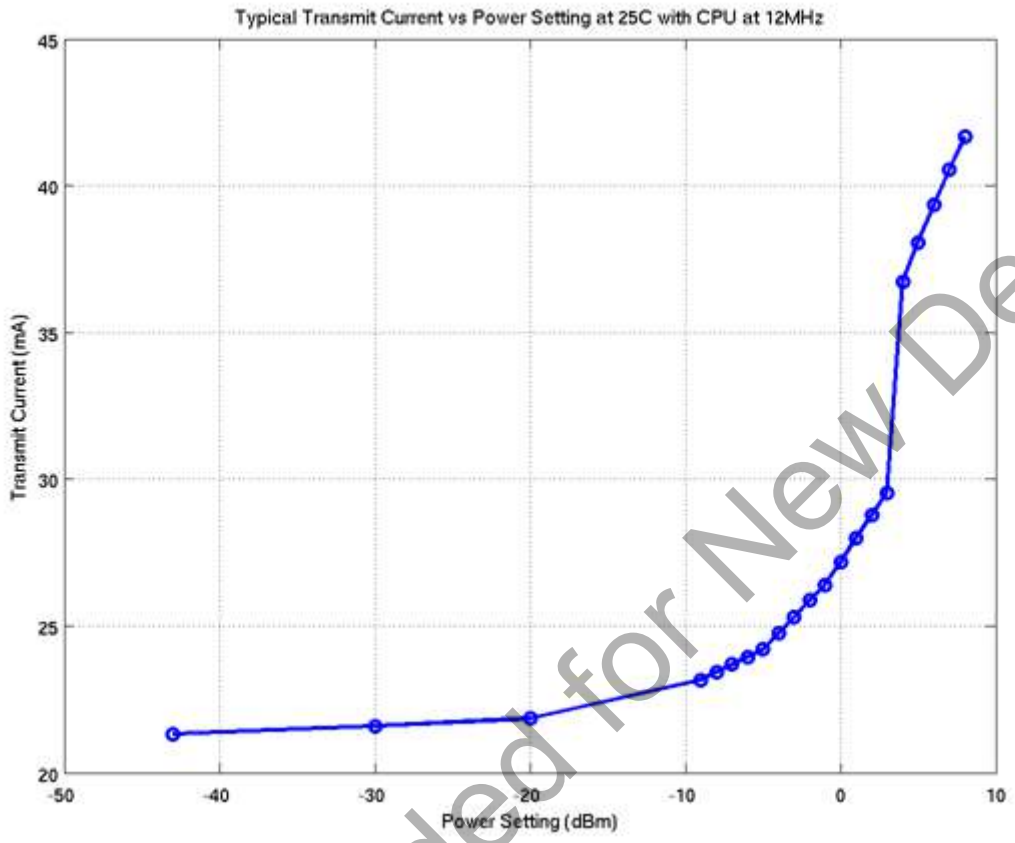


Figure 2.1. Transmit Power Consumption

Figure 2.2 shows typical output power against power setting on the Silicon Labs reference design.

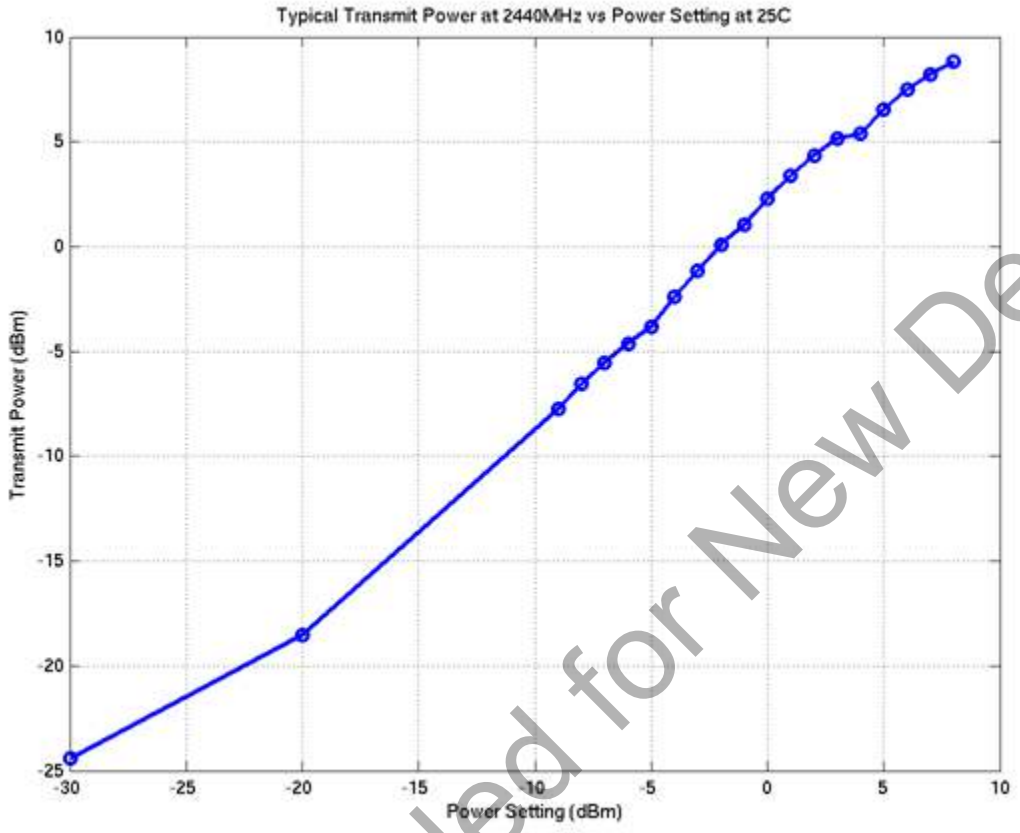


Figure 2.2. Transmit Output Power

2.5. Digital I/O Specifications

Table 2.5 lists the digital I/O specifications for the EM342. The digital I/O power (named VDD_PADS) comes from three dedicated pins (pins 23, 28 and 37). The voltage applied to these pins sets the I/O voltage.

Table 2.5. Digital I/O Specifications

Parameter	Test Condition	Min	Typ	Max	Unit
Voltage supply (regulator input voltage)		2.1	—	3.6	V
Low Schmitt switching threshold	V_{SWIL} Schmitt input threshold going from high to low	0.42 x VDD_PADS	—	0.50 x VDD_PADS	V
High Schmitt switching threshold	V_{SWIH} Schmitt input threshold going from low to high	0.62 x VDD_PADS	—	0.80 x 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}	24	29	34	kΩ
Input pull-down resistor value	R_{IPD}	24	29	34	kΩ
Output voltage for logic 0	V_{OL} (I_{OL} = 4 mA for standard pads, 8 mA for high current pads)	0	—	0.18 x VDD_PADS	V
Output voltage for logic 1	V_{OH} (I_{OH} = 4 mA for standard pads, 8 mA for high current pads)	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: PA7, PC0	I_{OHH}	—	—	8	mA
Output sink current high current pad: PA7, PC0	I_{OLH}	—	—	8	mA
Total output current (for I/O Pads)	$I_{OH} + I_{OL}$	—	—	40	mA

Table 2.6 lists the nRESET pin specifications for the EM342. The digital I/O power (named VDD_PADS) comes from three dedicated pins (pins 23, 28 and 37). The voltage applied to these pins sets the I/O voltage.

Table 2.6. nReset Pin Specifications

Parameter	Test Condition	Min	Typ	Max	Unit
Low Schmitt switching threshold	V_{SWIL} Schmitt input threshold going from high to low	0.42 x VDD_PADS	—	0.50 x VDD_PADS	V
High Schmitt switching threshold	V_{SWIH} Schmitt input threshold going from low to high	0.62 x VDD_PADS	—	0.80 x VDD_PADS	V
Input current for logic 1	I_{IH}	—	—	+0.5	µA
Input pull-up resistor value	R_{IPU} Pull-up value while the chip is not reset	24	29	34	kΩ
Input pull-up resistor value	$R_{IPURESET}$ Pull-up value while the chip is reset	12	14.5	17	kΩ

2.6. Non-RF System Electrical Characteristics

Table 2.7 lists the non-RF system level characteristics for the EM342.

Table 2.7. Non-RF System Electrical Characteristics

Measured on Silicon Labs' EM357 reference design with T_A = 25 °C and V_{DD} = 3 V, unless otherwise noted.

Parameter	Test Condition	Min	Typ	Max	Unit
System wake time from deep sleep	From wakeup event to first ARM® Cortex™-M3 instruction running from 6 MHz internal RC clock Includes supply ramp time and oscillator startup time	—	110	—	µs
Shutdown time going into deep sleep	From last ARM® Cortex™-M3 instruction to deep sleep mode	—	5	—	µs

2.7. RF Electrical Characteristics

2.7.1. Receive

Table 2.8 lists the key parameters of the integrated IEEE 802.15.4-2003 receiver on the EM342.

Receive measurements were collected with the Silicon Labs EM357 Ceramic Balun Reference Design (Version A0) at 2440 MHz. 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.

Table 2.8. Receive Characteristics

Parameter	Test Condition	Min	Typ	Max	Unit
Frequency range		2400	—	2500	MHz
Sensitivity (boost mode)	1% PER, 20 byte packet defined by IEEE 802.15.4-2003;	—	–102	–96	dBm
Sensitivity	1% PER, 20 byte packet defined by IEEE 802.15.4-2003;	—	–100	–94	dBm
High-side adjacent channel rejection	IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	35	—	dB
Low-side adjacent channel rejection	IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	35	—	dB
2nd high-side adjacent channel rejection	IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	46	—	dB
2nd low-side adjacent channel rejection	IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	46	—	dB
High-side adjacent channel rejection	Filtered IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	39	—	dB
Low-side adjacent channel rejection	Filtered IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	47	—	dB
2nd high-side adjacent channel rejection	Filtered IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	49	—	dB
2nd low-side adjacent channel rejection	Filtered IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	49	—	dB
High-side adjacent channel rejection	CW interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	44	—	dB
Low-side adjacent channel rejection	CW interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	47	—	dB

Table 2.8. Receive Characteristics (Continued)

Parameter	Test Condition	Min	Typ	Max	Unit
2nd high-side adjacent channel rejection	CW interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	59	—	dB
2nd low-side adjacent channel rejection	CW interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	59	—	dB
Channel rejection for all other channels	IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	40	—	dB
802.11g rejection centered at +12 MHz or –13 MHz	IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	36	—	dB
Maximum input signal level for correct operation		0	—	—	dBm
Co-channel rejection	IEEE 802.15.4-2003 interferer signal, wanted IEEE 802.15.4-2003 signal at –82 dBm	—	–6	—	dBc
Relative frequency error (50% greater than the 2x40 ppm required by IEEE 802.15.4-2003)		–120	—	+120	ppm
Relative timing error (50% greater than the 2x40 ppm required by IEEE 802.15.4-2003)		–120	—	+120	ppm
Linear RSSI range	As defined by IEEE 802.15.4-2003	40	—	—	dB
RSSI Range		–90	—	–40	dBm

Figure 2.3 shows the variation of receive sensitivity with temperature for boost mode and normal mode for a typical chip.

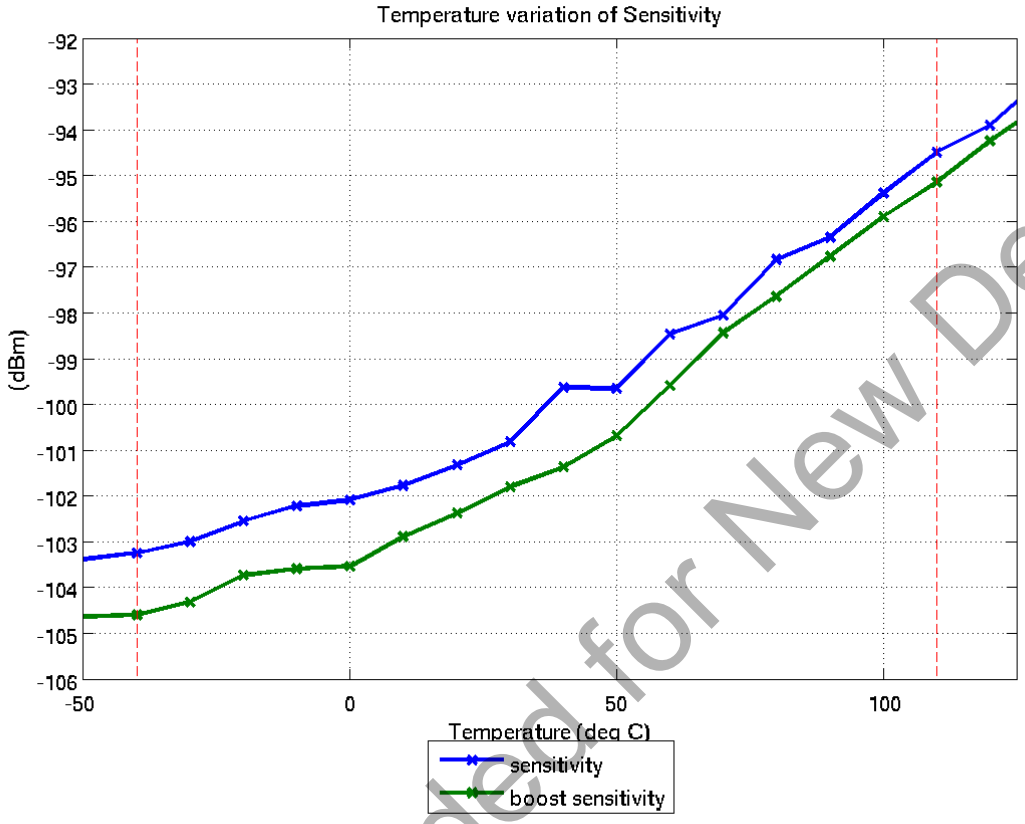


Figure 2.3. Receive Sensitivity vs. Temperature

2.7.2. Transmit

Table 2.9 lists the key parameters of the integrated IEEE 802.15.4-2003 transmitter on the EM342.

Transmit measurements were collected with the Silicon Labs EM342 Ceramic Balun Reference Design (Version A0) at 2440 MHz. 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. In terms of impedance, this reference design presents a 3n3 inductor in parallel with a 100:50 Ω balun to the RF pins.

Table 2.9. Transmit Characteristics

Parameter	Test Condition	Min	Typ	Max	Unit
Maximum output power (boost mode)	At highest boost mode power setting (+8)	—	8	—	dBm
Maximum output power	At highest normal mode power setting (+3)	1	5	—	dBm
Minimum output power	At lowest power setting	—	–55	—	dBm
Error vector magnitude (Offset-EVM)	As defined by IEEE 802.15.4-2003, which sets a 35% maximum	—	5	15	%
Carrier frequency error		–40	—	+40	ppm
PSD mask relative	3.5 MHz away	–20	—	—	dB
PSD mask absolute	3.5 MHz away	–30	—	—	dBm

Figure 2.4 shows the variation of transmit power with temperature for maximum boost mode power, and normal mode for a typical chip.

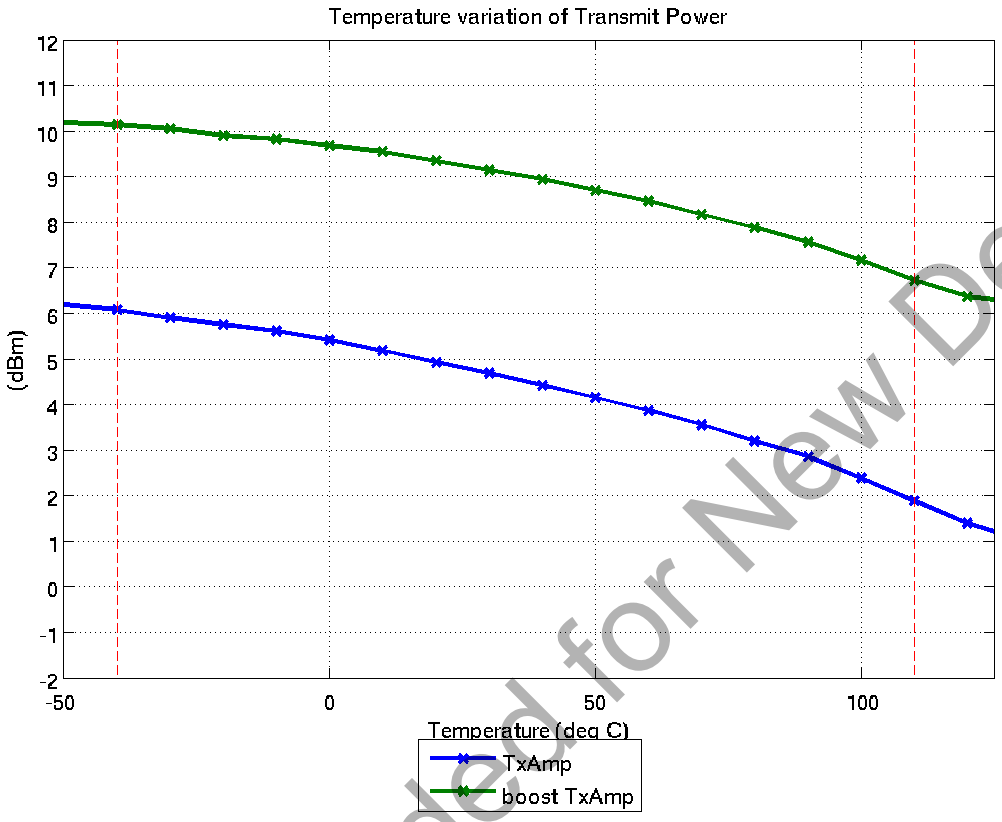


Figure 2.4. Transmit Power vs. Temperature

2.7.3. Synthesizer

Table 2.10 lists the key parameters of the integrated synthesizer on the EM342.

Table 2.10. Synthesizer Characteristics

Measured on Silicon Labs' EM357 reference design with T_A = 25 °C and V_{DD} = 3 V, unless otherwise noted.

Parameter	Test Condition	Min	Typ	Max	Unit
Frequency range		2400	—	2500	MHz
Frequency resolution		—	11.7	—	kHz
Lock time	From off	—	—	100	µs
Relock time	Channel change or RX/TX turnaround (IEEE 802.15.4-2003 defines 192 µs turnaround time)	—	—	100	µs
Phase noise at 100 kHz offset		—	–75	—	dBc/Hz
Phase noise at 1 MHz offset		—	–100	—	dBc/Hz
Phase noise at 4 MHz offset		—	–108	—	dBc/Hz
Phase noise at 10 MHz offset		—	–114	—	dBc/Hz

3. Functional Description

The EM342 is a fully integrated system-on-chip that integrates a 2.4 GHz, IEEE 802.15.4-2003-compliant transceiver, 32-bit ARM® Cortex-M3 microprocessor, flash and RAM memory, and peripherals of use to designers of ZigBee-based RF4CE systems.

The transceiver uses an efficient architecture that exceeds the dynamic range requirements imposed by the IEEE 802.15.4-2003 standard by over 15 dB. The integrated receive channel filtering allows for robust co-existence with other communication standards in the 2.4 GHz spectrum, such as IEEE 802.11-2007 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.

The integrated 32-bit ARM® Cortex™-M3 microprocessor is highly optimized for high performance, low power consumption, and efficient memory utilization. Including an integrated MPU, it supports two different modes of operation—privileged mode and user mode. This architecture could allow for separation of the networking stack from the application code, and prevents unwanted modification of restricted areas of memory and registers resulting in increased stability and reliability of deployed solutions.

The EM342 has 128 kB of embedded flash memory. It has 12 kB of integrated RAM for data and program storage. The Ember software for the EM342 employs an effective wear-leveling algorithm that optimizes the lifetime of the embedded flash.

To maintain the strict timing requirements imposed by the ZigBee RF4CE and IEEE 802.15.4-2003 standards, the EM342 integrates a number of MAC functions, AES128 encryption accelerator, and automatic CRC handling 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. The Ember Packet Trace Interface is also integrated with the MAC, allowing complete, non-intrusive capture of all packets to and from the EM342 with Ember development tools.

The EM342 offers a number of advanced power management features that enable long battery life. A high-frequency internal RC oscillator allows the processor core to begin code execution quickly upon waking. Various deep sleep modes are available with less than 1 µA power consumption while retaining RAM contents. To support user-defined applications, on-chip peripherals include UART, SPI (slave only), as well as up to 16 GPIOs. Additionally, an integrated voltage regulator, power-on-reset circuit, and sleep timer are available.

Finally, the EM342 utilizes standard Serial Wire and JTAG interfaces for powerful software debugging and programming of the ARM Cortex™-M3 core. The EM342 integrates the standard ARM system debug components: Flash Patch and Breakpoint (FPB), Data Watchpoint and Trace (DWT), and Instrumentation Trace Macrocell (ITM).

Target applications for the EM342 are ZigBee RF4CE target devices using the ZigBee Remote Control profile.

The technical data sheet details the EM342 features available to customers using it with Ember software.

Figure 3.1 shows a detailed block diagram of the EM342.

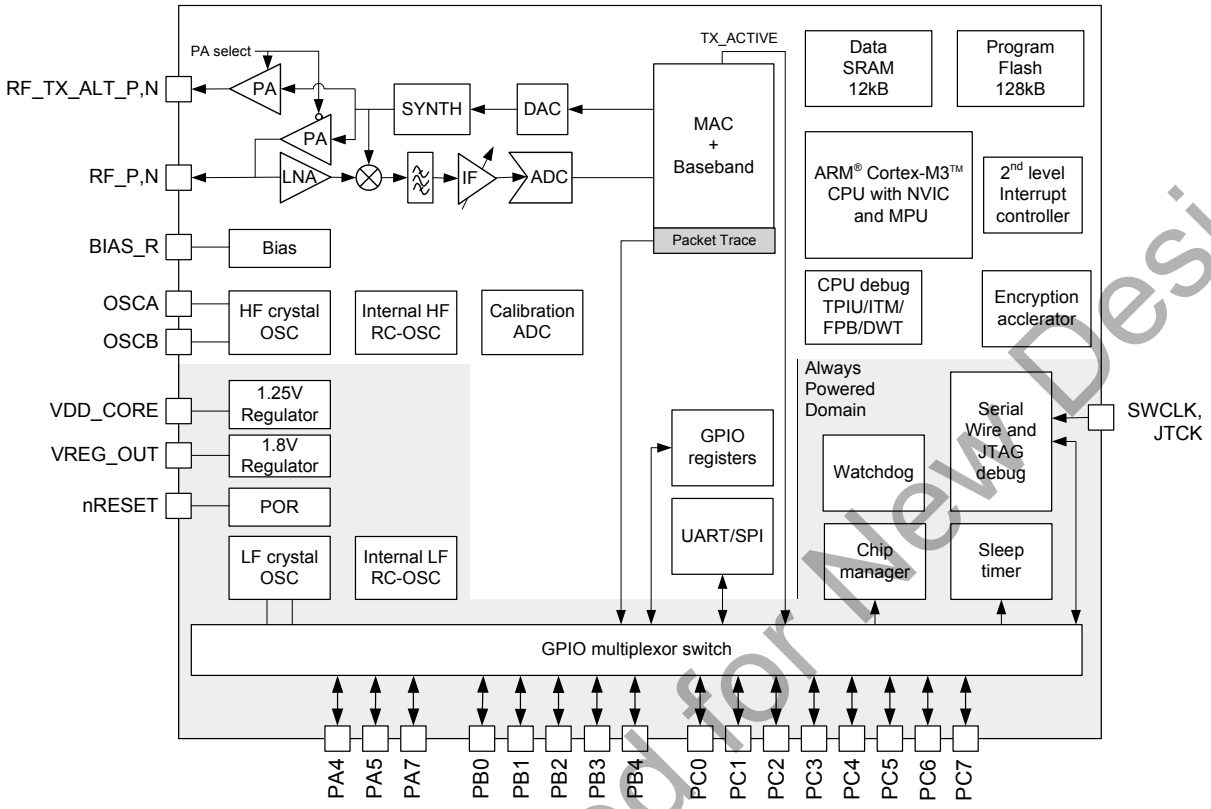


Figure 3.1. EM342 Block Diagram

The EM342 radio receiver is a low-IF, super-heterodyne receiver. The architecture has been chosen to optimize co-existence with other devices in the 2.4 GHz band (namely Wi-Fi and Bluetooth), and to minimize power consumption. The receiver uses differential signal paths to reduce sensitivity to noise interference. Following RF amplification, the signal is downconverted by an image-rejecting mixer, filtered, and then digitized by an ADC.

The digital section of the receiver uses a coherent demodulator to generate symbols for the hardware-based MAC. The digital receiver also contains the analog radio calibration routines, and controls the gain within the receiver path.

The radio transmitter uses an efficient architecture in which the data stream directly modulates the VCO frequency. An integrated PA provides the output power. Digital logic controls TX path and output power calibration. If the EM342 is to be used with an external PA, use the TX_ACTIVE or nTX_ACTIVE signal to control the timing of the external switching logic.

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

The MAC interfaces the on-chip RAM to the RX and TX baseband modules. The MAC provides hardware-based IEEE 802.15.4-2003 packet-level filtering. It supplies an accurate symbol time base that minimizes the synchronization effort of the Ember software and meets the protocol timing requirements. In addition, it provides timer and synchronization assistance for the IEEE 802.15.4-2003 CSMA-CA algorithm.

The EM342 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 development environment, and provides advanced network debug capability when used with the Ember Debug Adapter (ISA3).

The EM342 integrates an ARM® Cortex™-M3 microprocessor, revision r1p1. This industry-leading core provides 32-bit performance and is very power-efficient. It has excellent code density using the ARM® Thumb-2 instruction

set. The processor can be operated at 12 or 24 MHz when using the high-frequency crystal oscillator, or at 6 MHz or 12 MHz when using the high-frequency internal RC oscillator.

The EM342 has 128 kB of flash memory. The chip has 12 kB of RAM on-chip, and the ARM configurable memory protection unit (MPU).

The EM342 implements both the ARM Serial Wire and JTAG debug interfaces. These interfaces provide real time, non-intrusive programming and debugging capabilities. Serial Wire and JTAG provide the same functionality, but are mutually exclusive. The Serial Wire interface uses two pins; the JTAG interface uses five. Serial Wire is preferred, since it uses fewer pins.

The EM342 contains 16 GPIO pins shared with other peripheral or alternate functions. The integrated serial controller SC1 can be configured for SPI (slave) or UART operation.

The EM342 contains four oscillators: a high-frequency 24 MHz external crystal oscillator, a high-frequency 12 MHz internal RC oscillator, an optional low-frequency 32.768 kHz external crystal oscillator, and a low-frequency 10 kHz internal RC oscillator.

The EM342 has an ultra low power, deep sleep state with a choice of clocking modes. The sleep timer can be clocked with either the external 32.768 kHz crystal oscillator or with a 1 kHz clock derived from the internal 10 kHz RC oscillator. Alternatively, all clocks can be disabled for the lowest power mode. In the lowest power mode, only external events on GPIO pins will wake up the chip. The EM342 has a fast startup time (typically 110 µs) from deep sleep to the execution of the first ARM® Cortex™-M3 instruction.

The EM342 contains three power domains. The always-on high voltage supply powers the GPIO pads and critical chip functions. Regulated low voltage supplies power the rest of the chip. The low voltage supplies are disabled during deep sleep to reduce power consumption. Integrated voltage regulators generate regulated 1.25 V and 1.8 V voltages from an unregulated supply voltage. The 1.8 V regulator output is decoupled and routed externally to supply analog blocks, RAM, and flash memories. The 1.25 V regulator output is decoupled externally and supplies the core logic.

4. Radio Module

The radio module consists of an analog front end and digital baseband as shown in Figure 3.1 on page 22.

4.1. Receive (RX) Path

The RX path uses a low-IF, super-heterodyne receiver that rejects the image frequency using complex mixing and polyphase filtering. In the analog domain, the input RF signal from the antenna is first amplified and mixed down to a 4 MHz IF frequency. The mixers' output is filtered, combined, and amplified before being sampled by a 12 MSPS ADC. The digitized signal is then demodulated in the digital baseband. The filtering within the RX path improves the EM342's co-existence with other 2.4 GHz transceivers such as Zigbee/ 802.15.4-2003, IEEE 802.11-2007, and Bluetooth radios. The digital baseband also provides gain control of the RX path, both to enable the reception of small and large wanted signals and to tolerate large interferers.

4.1.1. RX Baseband

The EM342 RX digital baseband 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-defined preamble. An automatic gain control (AGC) module adjusts the analog gain continuously every ¼ symbol until the preamble is detected. Once detected, the gain is fixed for the remainder of the packet. The baseband despreads the demodulated data into 4-bit symbols. These symbols are buffered and passed to the hardware-based MAC module for packet assembly and 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 Ember software includes calibration algorithms that use this interface to reduce the effects of silicon process and temperature variation.

4.1.2. RSSI and CCA

The EM342 calculates the RSSI over every 8-symbol period as well as at the end of a received packet. The linear range of RSSI is specified to be at least 40 dB over temperature. At room temperature, the linear range is approximately 60 dB (–90 dBm to –30 dBm input signal).

The EM342 RX baseband provides support for the IEEE 802.15.4-2003 RSSI CCA method. Clear channel reports busy medium if RSSI exceeds its threshold.

4.2. Transmit (TX) Path

The EM342 TX path produces an O-QPSK-modulated signal using the analog front end and digital baseband. The area- and power-efficient TX architecture uses a two-point modulation scheme to modulate the RF signal generated by the synthesizer. The modulated RF signal is fed to the integrated PA and then out of the EM342.

4.2.1. TX Baseband

The EM342 TX baseband in the digital domain spreads the 4-bit symbol into its IEEE 802.15.4-2003-defined 32-chip sequence. It also provides the interface for the Ember software to calibrate the TX module to reduce silicon process, temperature, and voltage variations.

4.2.2. TX_ACTIVE and nTX_ACTIVE Signals

For applications requiring an external PA, two signals are provided called TX_ACTIVE and nTX_ACTIVE. These signals are the inverse of each other. They can be used for external PA power management and RF switching logic. In transmit mode the TX baseband drives TX_ACTIVE high, as described in Table 7.4 on page 54. In receive mode the TX_ACTIVE signal is low. TX_ACTIVE is the alternate function of PC5, and nTX_ACTIVE is the alternate function of PC6. See "7. GPIO (General Purpose Input/Output)" on page 48 for details of the alternate GPIO functions. The digital I/O that provide these signals have a 4 mA output sink and source capability.

4.3. Calibration

The Ember software calibrates the radio using dedicated hardware resources.

4.4. Integrated MAC Module

The EM342 integrates most of the IEEE 802.15.4-2003 MAC requirements in hardware. This allows the ARM® Cortex™-M3 CPU to provide greater bandwidth to application and network operations. In addition, the hardware acts as a first-line filter for unwanted packets. The EM342 MAC uses a DMA interface to RAM to further reduce the overall ARM® Cortex™-M3 CPU interaction when transmitting or receiving packets.

When a packet is ready for transmission, the Ember software configures the TX MAC DMA by indicating the packet buffer RAM location. The MAC waits for the backoff period, then switches 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 is in RX mode most of the time. In RX mode various format and address filters keep unwanted packets from using excessive RAM buffers, and prevent the CPU from being unnecessarily interrupted. When the reception of a packet begins, the MAC reads 4-bit symbols from the baseband and calculates the CRC. It then assembles the received data for storage in a RAM buffer. RX MAC DMA provides direct access to RAM. Once the packet has been received additional data, which provides statistical information on the packet to the Ember software, is appended to the end of the packet in the RAM buffer space.

The primary features of the MAC are as follows:

- CRC generation, appending, and checking
- Hardware timers and interrupts to achieve the MAC symbol timing
- Automatic preamble and SFD pre-pending on TX packets
- Address recognition and packet filtering on RX packets
- Automatic acknowledgment transmission
- Automatic transmission of packets from memory
- Automatic transmission after backoff time if channel is clear (CCA)
- Automatic acknowledgment checking
- Time stamping received and transmitted messages
- Attaching packet information to received packets (LQI, RSSI, gain, time stamp, and packet status)
- IEEE 802.15.4-2003 timing and slotted/unslotted timing

4.5. Packet Trace Interface (PTI)

The EM342 integrates a true PHY-level PTI for effective network-level debugging. It monitors all the PHY TX and RX packets between the MAC and baseband modules without affecting their normal operation. It cannot be used to inject packets into the PHY/MAC interface. This 500 kbps asynchronous interface comprises the frame signal (PTI_EN, PA4) and the data signal (PTI_DATA, PA5). PTI is supported by the Ember development tools.

4.6. Random Number Generator

Thermal noise in the analog circuitry is digitized to provide entropy for a true random number generator (TRNG). The TRNG produces 16-bit uniformly distributed numbers. The Ember software uses the TRNG to seed a pseudo random number generator (PRNG). The TRNG is also used directly for cryptographic key generation.

5. ARM® Cortex™-M3 and Memory Modules

This chapter discusses the ARM® Cortex™-M3 Microprocessor, and reviews the EM342's flash and RAM memory modules as well as the Memory Protection Unit (MPU).

5.1. ARM® Cortex™-M3 Microprocessor

The EM342 integrates the ARM® Cortex™-M3 microprocessor, revision r1p1, developed by ARM Ltd., making the EM342 a true System-on-Chip solution. The ARM® Cortex™-M3 is an advanced 32-bit modified Harvard architecture processor that has separate internal program and data buses, but presents a unified program and data address space to software. The word width is 32 bits for both the program and data sides. The ARM® Cortex™-M3 allows unaligned word and half-word data accesses to support efficiently-packed data structures.

The ARM® Cortex™-M3 in the EM342 has also been enhanced to support two separate memory protection levels. Basic protection is available without using the MPU, but normal operation uses the MPU. The MPU allows for protecting unimplemented areas of the memory map to prevent common software bugs from interfering with software operation. The architecture could also allow for separation of the networking stack from the application code using a fine granularity RAM protection module. Errant writes are captured and details are reported to the developer to assist in tracking down and fixing issues.

5.2. Embedded Memory

Figure 5.1 shows the EM342 ARM® Cortex™-M3 memory map.

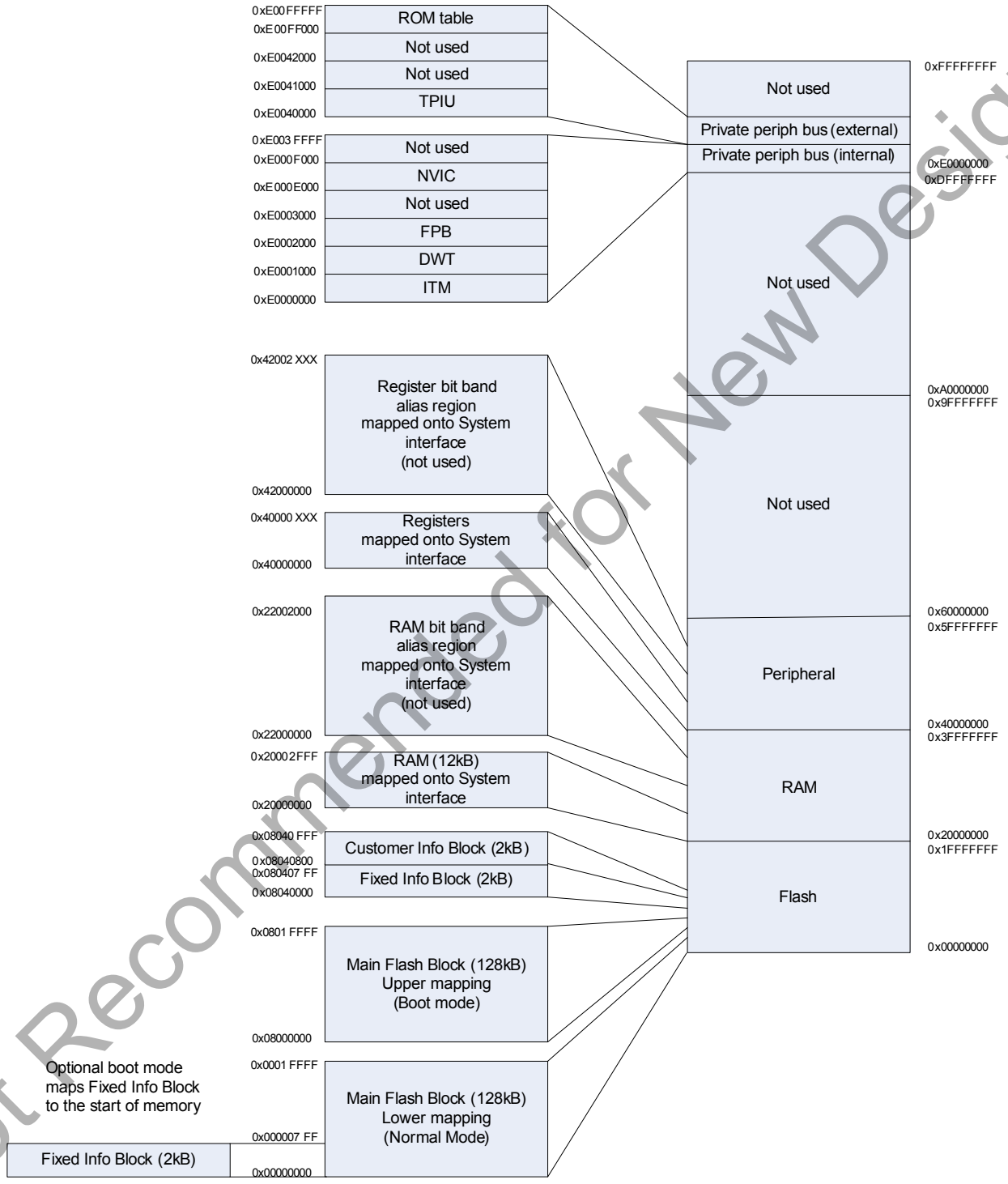


Figure 5.1. EM342 ARM® Cortex™-M3 Memory Map

5.2.1. Flash Memory

5.2.1.1. Flash Overview

The EM342 provides a total of 128 kB of flash memory. The flash memory is provided in three separate blocks:

- Main Flash Block (MFB)
- Fixed Information Block (FIB)
- Customer Information Block (CIB)

The MFB is divided into 2048-byte pages. The EM342 has 64 pages. The CIB is a single 2048-byte page. The FIB is a single 2048-byte page. The smallest erasable unit is one page and the smallest writable unit is an aligned 16-bit half-word. The flash is rated to have a guaranteed 20,000 write/erase cycles. The flash cell has been qualified for a data retention time of >100 years at room temperature.

Flash may be programmed either through the Serial Wire/JTAG interface or through bootloader software. Programming flash through Serial Wire/JTAG requires the assistance of RAM-based utility code. Programming through a bootloader requires Ember software for over-the-air loading or serial link loading.

5.2.1.2. Main Flash Block

The start of the MFB is mapped to both address 0x00000000 and address 0x08000000 in normal boot mode, but is mapped only to address 0x08000000 in FIB monitor mode (see also "7.5. Boot Configuration" on page 51). Consequently, it is recommended that software intended to execute from the MFB is designed to operate from the upper address, 0x08000000, since this address mapping is always available in all modes.

The MFB stores all program instructions and constant data. A small portion of the MFB is devoted to non-volatile token storage using the Ember Simulated EEPROM system.

5.2.1.3. Fixed Information Block

The 2 kB FIB is used to store fixed manufacturing data including serial numbers and calibration values. The start of the FIB is mapped to address 0x08040000. This block can only be programmed during production by Silicon Labs.

The FIB also contains a monitor program, which is a serial-link-only way of performing low-level memory accesses. In FIB monitor mode (see "7.5. Boot Configuration" on page 51), the start of the FIB is mapped to both address 0x00000000 and address 0x08040000 so the monitor may be executed out of reset.

5.2.1.4. Customer Information Block

The 2048 byte CIB can be used to store customer data. The start of the CIB is mapped to address 0x08040800. The CIB cannot be executed.

The first eight half-words of the CIB are dedicated to special storage called option bytes. An option byte is a 16 bit quantity of flash where the lower 8 bits contain the data and the upper 8 contain the inverse of the lower 8 bits. The upper 8 bits are automatically generated by hardware and cannot be written to by the user, see Table 5.1.

The option byte hardware also verifies the inverse of each option byte when exiting from reset and generates an error, which prevents the CPU from executing code, if a discrepancy is found. All of this is transparent to the user.

Table 5.1. Option Byte Storage

Address	bits [15:8]	bits [7:0]	Notes
0x08040800	Inverse Option Byte 0	Option Byte 0	Configures flash read protection
0x08040802	Inverse Option Byte 1	Option Byte 1	Reserved
0x08040804	Inverse Option Byte 2	Option Byte 2	Available for customer use[1]
0x08040806	Inverse Option Byte 3	Option Byte 3	Available for customer use[1]
0x08040808	Inverse Option Byte 4	Option Byte 4	Configures flash write protection
0x0804080A	Inverse Option Byte 5	Option byte 5	Configures flash write protection
0x0804080C	Inverse Option Byte 6	Option Byte 6	Reserved
0x0804080E	Inverse Option Byte 7	Option Byte 7	Reserved

Notes:
1. Option bytes 2 and 3 do not link to any specific hardware functionality other than the option byte loader. Therefore, they are best used for storing data that requires a hardware verification of the data integrity.

Table 5.2 shows the mapping of the option bytes that are used for read and write protection of the flash. Each bit of the flash write protection option bytes protects a 4 page region of the main flash block. The EM342 has 16 regions and therefore option bytes 4 and 5 control flash write protection.These write protection bits are active low, and therefore the erased state of 0xFF disables write protection. Like read protection, write protection only takes effect after a reset. Write protection not only prevents a write to the region, but also prevents page erasure.

Option byte 0 controls flash read protection. When option byte 0 is set to 0xA5, read protection is disabled. All other values, including the erased state 0xFF, enable read protection when coming out of reset. The internal state of read protection (active versus disabled) can only be changed by applying a full chip reset. If a debugger is connected to the EM342, the intrusion state is latched. Read protection is combined with this latched intrusion signal. When both read protection and intrusion are set, all flash is disconnected from the internal bus. As a side effect, the CPU cannot execute code since all flash is disconnected from the bus. This functionality prevents a debug tool from being able to read the contents of any flash. The only means of clearing the intrusion signal is to disconnect the debugger and reset the entire chip using the nRESET pin. By requiring a chip reset, a debugger cannot install or execute malicious code that could allow the contents of the flash to be read.

The only way to disable read protection is to program option byte 0 with the value 0xA5. Option byte 0 must be erased before it can be programmed. Erasing option byte 0 while read protection is active automatically mass-erases the main flash block. By automatically erasing main flash, a debugger cannot disable read protection and readout the contents of main flash without destroying its contents.

When read protection is active, the bottom four flash pages, addresses 0x08000000 to 0x08001FFF, are automatically write-protected. Write protecting the bottom four flash pages of main flash prevents an attacker from reprogramming the reset vector and executing arbitrary code.

In general, if read protection is active then write protection should also be active. This prevents an attacker from reprogramming flash with malicious code that could readout the flash after the debugger is disconnected. Even though read protection automatically protects the reset vector, the same technique of reprogramming flash could be performed at an address outside the bottom four flash pages. To obtain fully protected flash, both read protection and write protection should be active.

Table 5.2. Option Byte Write Protection Bit Map

Option Byte	Bit	Notes
Option Byte 0	bit [7:0]	Read protection of all flash (MFB, FIB, CIB)
Option Byte 1	bit [7:0]	Reserved for Silicon Labs use
Option Byte 2	bit [7:0]	Available for customer use
Option Byte 3	bit [7:0]	Available for customer use
Option Byte 4	bit [0]	Write protection of address range 0x08000000 – 0x08001FFF
	bit [1]	Write protection of address range 0x08002000 – 0x08003FFF
	bit [2]	Write protection of address range 0x08004000 – 0x08005FFF
	bit [3]	Write protection of address range 0x08006000 – 0x08007FFF
	bit [4]	Write protection of address range 0x08008000 – 0x08009FFF
	bit [5]	Write protection of address range 0x0800A000 – 0x0800BFFF
	bit [6]	Write protection of address range 0x0800C000 – 0x0800DFFF
	bit [7]	Write protection of address range 0x0800E000 – 0x0800FFFF
Option Byte 5	bit [0]	Write protection of address range 0x08010000 – 0x08011FFF
	bit [1]	Write protection of address range 0x08012000 – 0x08013FFF
	bit [2]	Write protection of address range 0x08014000 – 0x08015FFF
	bit [3]	Write protection of address range 0x08016000 – 0x08017FFF
	bit [4]	Write protection of address range 0x08018000 – 0x08019FFF
	bit [5]	Write protection of address range 0x0801A000 – 0x0801BFFF
	bit [6]	Write protection of address range 0x0801C000 – 0x0801DFFF
	bit [7]	Write protection of address range 0x0801E000 – 0x0801FFFF
Option Byte 6	bit[7:0]	Reserved for Silicon Labs' use
Option Byte 7	bit [7:0]	Reserved for Silicon Labs use

5.2.1.5. Simulated EEPROM

Ember software reserves 8 kB of the main flash block as a simulated EEPROM storage area for stack and customer tokens. The simulated EEPROM storage area implements a wear-leveling algorithm to extend the number of simulated EEPROM write cycles beyond the physical limit of 20,000 write cycles for which each flash cell is qualified.

5.2.2. RAM

5.2.2.1. RAM Overview

The EM342 has 12 kB of static RAM on-chip. The start of RAM is mapped to address 0x20000000. Although the ARM® Cortex™-M3 allows bit band accesses to this address region, the standard MPU configuration does not permit use of the bit-band feature.

The RAM is physically connected to the AHB System bus and is therefore accessible to both the ARM® Cortex™-M3 microprocessor and the debugger. The RAM can be accessed for both instruction and data fetches as bytes, half words, or words. The standard MPU configuration does not permit execution from the RAM, but for special purposes the MPU may be disabled. To the bus, the RAM appears as 32-bit wide memory and in most situations has zero wait state read or write access. In the higher CPU clock mode the RAM requires two wait states. This is handled by hardware transparent to the user application with no configuration required.

5.2.2.2. Direct Memory Access (DMA) to RAM

Several of the peripherals are equipped with DMA controllers allowing them to transfer data into and out of RAM autonomously. This applies to the radio (802.15.4-2003 MAC) and serial controller. In the case of the serial controller, the DMA is full duplex so that a read and a write to RAM may be requested at the same time. Thus there are six DMA channels in total. See "8.5. DMA Channels" on page 86 and "10.1.4. DMA" on page 173 for a description of how to configure the serial controller for DMA operation. The DMA channels do not use AHB system bus bandwidth as they access the RAM directly.

The EM342 integrates a DMA arbiter that ensures fair access to the microprocessor as well as the peripherals through a fixed priority scheme appropriate to the memory bandwidth requirements of each master. The priority scheme is as follows, with the top peripheral being the highest priority:

1. MAC
2. Serial Controller 1 Receive
3. Serial Controller 1 Transmit

5.2.2.3. RAM Memory Protection

The EM342 integrates two memory protection mechanisms. The first memory protection mechanism is through the ARM® Cortex™-M3 Memory Protection Unit (MPU) described in “5.3. Memory Protection Unit”. The MPU may be used to protect any area of memory. MPU configuration is normally handled by Ember software. The second memory protection mechanism is through a fine granularity RAM protection module. This allows segmentation of the RAM into 32-byte blocks where any block can be marked as write protected. An attempt to write to a protected RAM block using a user mode write results in a bus error being signaled on the AHB System bus. A privileged mode write is allowed at any time and reads are allowed in either mode. The main purpose of this fine granularity RAM protection module is to notify the software of erroneous writes to system areas of memory. RAM protection is configured using a group of registers that provide a bit map. Each bit in the map represents a 32-byte block of RAM. When the bit is set the block is write-protected.

The fine granularity RAM memory protection mechanism is also available to the peripheral DMA controllers. A register bit enables protection from DMA writes to protected memory. If a DMA write is made to a protected location in RAM, a management interrupt is generated. At the same time the faulting address and the identification of the peripheral is captured for later debugging. Note that only peripherals capable of writing data to RAM, such as received packet data or a received serial port character, can generate this interrupt.

5.2.3. Registers

Appendix A, Register Address Table provides a short description of all application-accessible registers within the EM342. Complete descriptions are provided at the end of each applicable peripheral's description. The registers are mapped to the system address space starting at address 0x40000000. These registers allow for the control and configuration of the various peripherals and modules. The CPU only performs word-aligned accesses on the system bus. The CPU performs a word aligned read-modify-write for all byte, half-word, and unaligned writes and a word-aligned read for all reads. Silicon Labs recommends accessing all peripheral registers using word-aligned addressing.

As with the RAM, the peripheral registers fall within an address range that allows for bit-band access by the ARM® Cortex™-M3, but the standard MPU configuration does not allow access to this alias address range.

5.3. Memory Protection Unit

The EM342 includes the ARM® Cortex™-M3 Memory Protection Unit, or MPU. The MPU controls access rights and characteristics of up to eight address regions, each of which may be divided into eight equal sub-regions. Refer to the ARM® Cortex™-M3 Technical Reference Manual (DDI 0337A) for a detailed description of the MPU.

Ember software configures the MPU in a standard configuration and application software should not modify it. The configuration is designed for optimal detection of illegal instruction or data accesses. If an illegal access is attempted, the MPU captures information about the access type, the address being accessed, and the location of the offending software. This simplifies software debugging and increases the reliability of deployed devices. As a consequence of this MPU configuration, accessing RAM and register bit-band address alias regions is not permitted, and generates a bus fault if attempted.

6. System Modules

System modules encompass power domains, resets, clocks, system timers, power management, and encryption. Figure 6.1 shows these modules and how they interact.

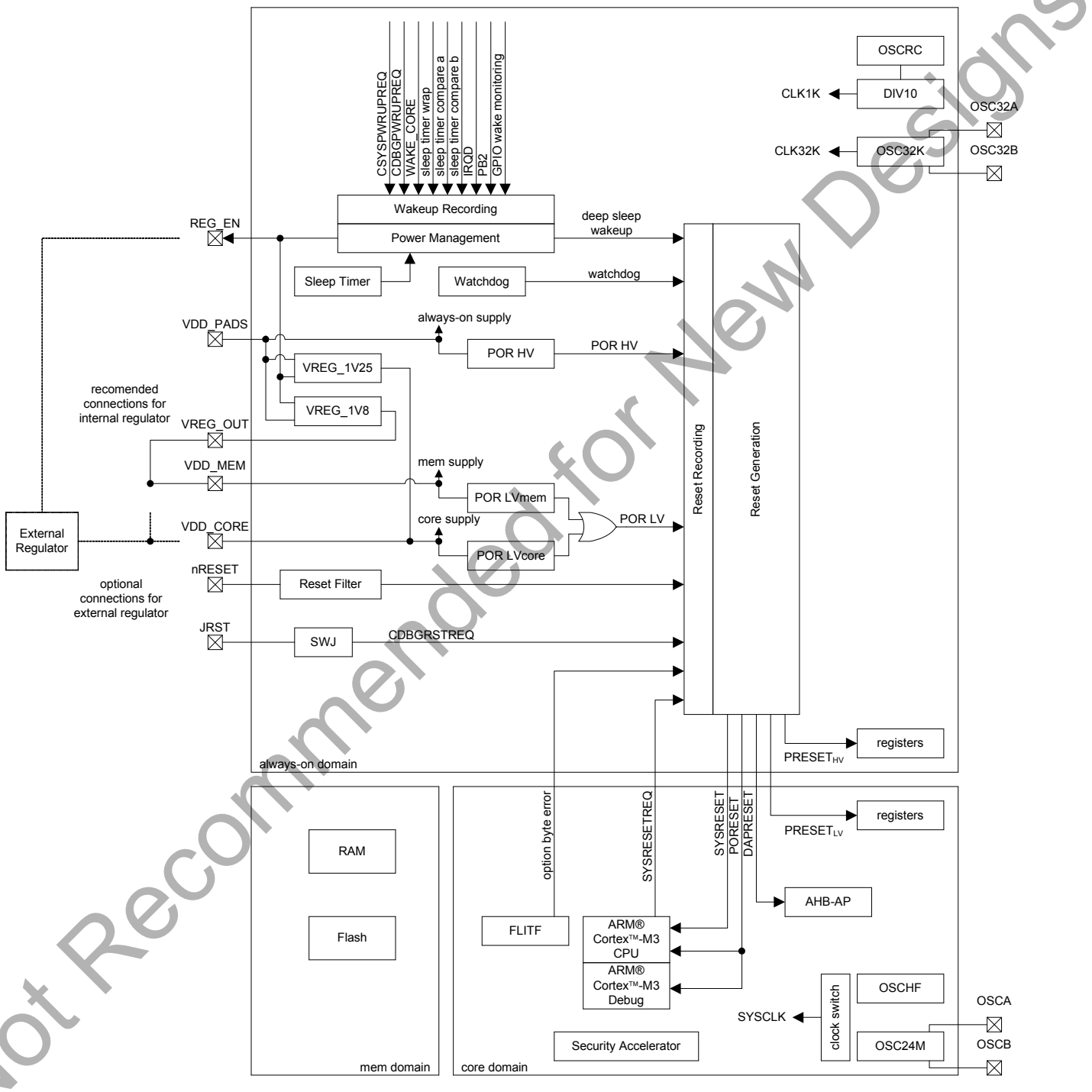


Figure 6.1. System Module Block Diagram

6.1. Power Domains

The EM342 contains three power domains:

- An “always-on domain” containing all logic and analog cells required to manage the EM342’s power modes, including the GPIO controller and sleep timer. This domain must remain powered.
- A “core domain” containing the CPU, Nested Vectored Interrupt Controller (NVIC), and peripherals. To save power, this domain can be powered down using a mode called deep sleep.
- A “memory domain” containing the RAM and flash memories. This domain is managed by the power management controller. When in deep sleep, the RAM portion of this domain is powered from the always-on domain supply to retain the RAM contents while the regulators are disabled. During deep sleep the flash portion is completely powered down.

6.1.1. Internally Regulated Power

The preferred and recommended power configuration is to use the internal regulated power supplies to provide power to the core and memory domains. The internal regulators (VREG_1V25 and VREG_1V8) generate nominal 1.25 and 1.8 V supplies. The 1.25 V supply is internally routed to the core domain and to an external pin. The 1.8 V supply is routed to an external pin where it can be externally routed back into the chip to supply the memory domain. The internal regulators are described in "14. Integrated Voltage Regulator" on page 117.

When using the internal regulators, the always-on domain must be powered between 2.1 and 3.6 V at all four VDD_PADS pins.

When using the internal regulators, the VREG_1V8 regulator output pin (VREG_OUT) must be connected to the VDD_MEM, VDD_PADSA, VDD_VCO, VDD_RF, VDD_IF, VDD_PRE, and VDD_SYNTH pins.

When using the internal regulators, the VREG_1V25 regulator output and supply requires a connection between both VDD_CORE pins.

6.1.2. Externally Regulated Power

Optionally, the on-chip regulators may be left unused, and the core and memory domains may instead be powered from external supplies. For simplicity, the voltage for the core domain can be raised to nominal 1.8 V, requiring only one external regulator, or the core domain can be powered from the on-chip regulators while the other domains are powered externally. Note that if the core domain is powered at a higher voltage (1.8 V instead of 1.25 V) then power consumption increases. A regulator enable signal, REG_EN, is provided for control of external regulators. This is an open-drain signal that requires an external pull-up resistor. If REG_EN is not required to control external regulators it can be disabled (see "7.3. Forced Functions" on page 50).

Using an external regulator requires the always-on domain to be powered between 2.1 and 3.6 V at all four VDD_PADS pins.

When using an external regulator, the VREG_1V8 regulator output pin (VREG_OUT) must be left unconnected.

When using an external regulator, this external nominal 1.8 V supply has to be connected to both VDD_CORE pins and to the VDD_MEM, VDD_PADSA, VDD_VCO, VDD_RF, VDD_IF, VDD_PRE and VDD_SYNTH pins.

6.2. Resets

The EM342 resets are generated from a number of sources. Each of these reset sources feeds into central reset detection logic that causes various parts of the system to be reset depending on the state of the system and the nature of the reset event.

6.2.1. Reset Sources

6.2.1.1. Power-On-Resets (POR HV and POR LV)

The EM342 measures the voltage levels supplied to the three power domains. If a supply voltage drops below a low threshold, then a reset is applied. The reset is released if the supply voltage rises above a high threshold. There are three detection circuits for power-on-reset as follows:

- POR HV monitors the always-on domain supply voltage. Thresholds are given in Table 6.1.
- POR LV core monitors the core domain supply voltage. Thresholds are given in Table 6.2.
- POR LV mem monitors the memory supply voltage. Thresholds are given in Table 6.3.

Table 6.1. POR HV Thresholds

Parameter	Test conditions	Min	Typ	Max	Unit
Always-on domain release		0.62	0.95	1.20	V
Always-on domain assert		0.45	0.65	0.85	V
Supply rise time	From 0.5 V to 1.7 V			250	µs

Table 6.2. POR LVcore Thresholds

Parameter	Test conditions	Min	Typ	Max	Unit
1.25 V domain release		0.9	1.0	1.1	V
1.25 V domain assert		0.8	0.9	1.0	V

Table 6.3. POR LVmem Thresholds

Parameter	Test conditions	Min	Typ	Max	Unit
1.8 V domain release		1.35	1.5	1.65	V
1.8 V domain assert		1.26	1.4	1.54	V

The POR LVcore and POR LVmem reset sources are merged to provide a single reset source, POR LV, to the Reset Generation module, since the detection of either event needs to reset the same system modules.

6.2.1.2. nRESET Pin

A single active low pin, nRESET, is provided to reset the system. This pin has a Schmitt triggered input.

To afford good noise immunity and resistance to switch bounce, the pin is filtered with the Reset Filter module and generates the pin reset source, nRESET, to the Reset Generation module. Table 6.4 contains the specification for the filter.

Table 6.4. Reset Filter Specification for nRESET

Parameter	Min	Typ	Max	Unit
Reset filter time constant	2.1	12.0	16.0	µs
Reset pulse width to guarantee a reset	26.0	—	—	µs
Reset pulse width guaranteed not to cause a reset	0	—	1.0	µs

6.2.1.3. Watchdog Reset

The EM342 contains a watchdog timer (see also "6.4.1. Watchdog Timer" on page 43) that is clocked by the internal 1 kHz timing reference. When the timer expires it generates the reset source WATCHDOG_RESET to the Reset Generation module.

6.2.1.4. Software Reset

The ARM® Cortex™-M3 CPU can initiate a reset under software control. This is indicated with the reset source SYSRESETREQ to the Reset Generation module.

6.2.1.5. Option Byte Error

The flash memory controller contains a state machine that reads configuration information from the information blocks in the flash at system start time. An error check is performed on the option bytes that are read from flash and, if the check fails, an error is signaled that provides the reset source OPT_BYTE_ERROR to the Reset Generation module.

If an option byte error is detected, the system restarts and the read and check process is repeated. If the error is detected again the process is repeated but stops on the 3rd failure. The system is then placed into an emulated deep sleep where recovery is possible. In this state, flash memory readout protection is forced active to prevent secure applications from being compromised.

6.2.1.6. Debug Reset

The Serial Wire/JTAG Interface (SWJ) provides access to the SWJ Debug Port (SWJ-DP) registers. By setting the register bit CDBGRSTREQ in the SWJ-DP, the reset source CDBGRSTREQ is provided to the Reset Generation module.

6.2.1.7. JRST

One of the EM342's pins can function as the JTAG reset, conforming to the requirements of the JTAG standard. This input acts independently of all other reset sources and, when asserted, does not reset any on-chip hardware except for the JTAG TAP. If the EM342 is in the Serial Wire mode or if the SWJ is disabled, this input has no effect.

6.2.1.8. Deep Sleep Reset

The Power Management module informs the Reset Generation module of entry into and exit from the deep sleep states. The deep sleep reset is applied in the following states: before entry into deep sleep, while removing power from the memory and core domain, while in deep sleep, while waking from deep sleep, and while reapplying power until reliable power levels have been detect by POR LV.

The Power Management module allows a special emulated deep sleep state that retains memory and core domain power while in deep sleep.

6.2.2. Reset Recording

The EM342 records the last reset condition that generated a restart to the system. The reset conditions recorded are as follows:

- POR HV — always-on domain power supply failure
- POR LV — core domain (POR LVcore) or memory domain (POR LVmem) power supply failure
- nRESET — pin reset asserted
- watchdog — watchdog timer expired
- SYSRESETREQ — software reset by SYSERSETREQ from ARM® Cortex™-M3 CPU
- deep sleep wakeup — wake-up from deep sleep
- option byte error — error check failed when reading option bytes from flash

Note: While CPU Lockup is shown as a reset condition in software, CPU Lockup is not specifically a reset event. CPU Lockup is set to indicate that the CPU entered an unrecoverable exception. Execution stops but a reset is not applied. This is so that a debugger can interpret the cause of the error. Silicon Labs recommends that in a live application (in other words, no debugger attached) the watchdog be enabled by default so that the EM342 can be restarted.

6.2.3. Reset Generation Module

The Reset Generation module responds to reset sources and generates the following reset signals:

- PORESET — Reset of the ARM® Cortex™-M3 CPU and ARM® Cortex™-M3 System Debug components (Flash Patch and Breakpoint, Data Watchpoint and Trace, Instrumentation Trace Macrocell, Nested Vectored Interrupt Controller). ARM defines PORESET as the region that is reset when power is applied.
- SYSRESET — Reset of the ARM® Cortex™-M3 CPU without resetting the Core Debug and System Debug components, so that a live system can be reset without disturbing the debug configuration.
- DAPRESET — Reset to the SWJ's AHB Access Port (AHB-AP)
- $PRESET_{HV}$ — Peripheral reset for always-on power domain, for peripherals that are required to retain their configuration across a deep sleep cycle
- $PRESET_{LV}$ — Peripheral reset for core power domain, for peripherals that are not required to retain their configuration across a deep sleep cycle

Table 6.5 shows which reset sources generate certain resets.

Table 6.5. Generated Resets

Reset Source	Reset Generation Module Output				
	PORESET	SYSRESET	DAPRESET	$PRESET_{HV}$	$PRESET_{LV}$
POR HV	X	X	X	X	X
POR LV (due to waking from normal deep sleep)	X	X	X		X
POR LV (not due to waking from normal deep sleep)	X	X	X	X	X
nRESET	X	X		X	X
Watchdog		X		X	X
SYSRESETREQ		X		X	X
Option byte error	X	X			X
Normal deep sleep	X	X	X		X
Emulated deep sleep		X			X
Debug reset		X			

6.3. Clocks

The EM342 integrates four oscillators:

- 12 MHz RC oscillator
- 24 MHz crystal oscillator
- 10 kHz RC oscillator
- 32.768 kHz crystal oscillator

Figure 6.2 shows a block diagram of the clocks in the EM342. This simplified view shows all the clock sources and the general areas of the chip to which they are routed.

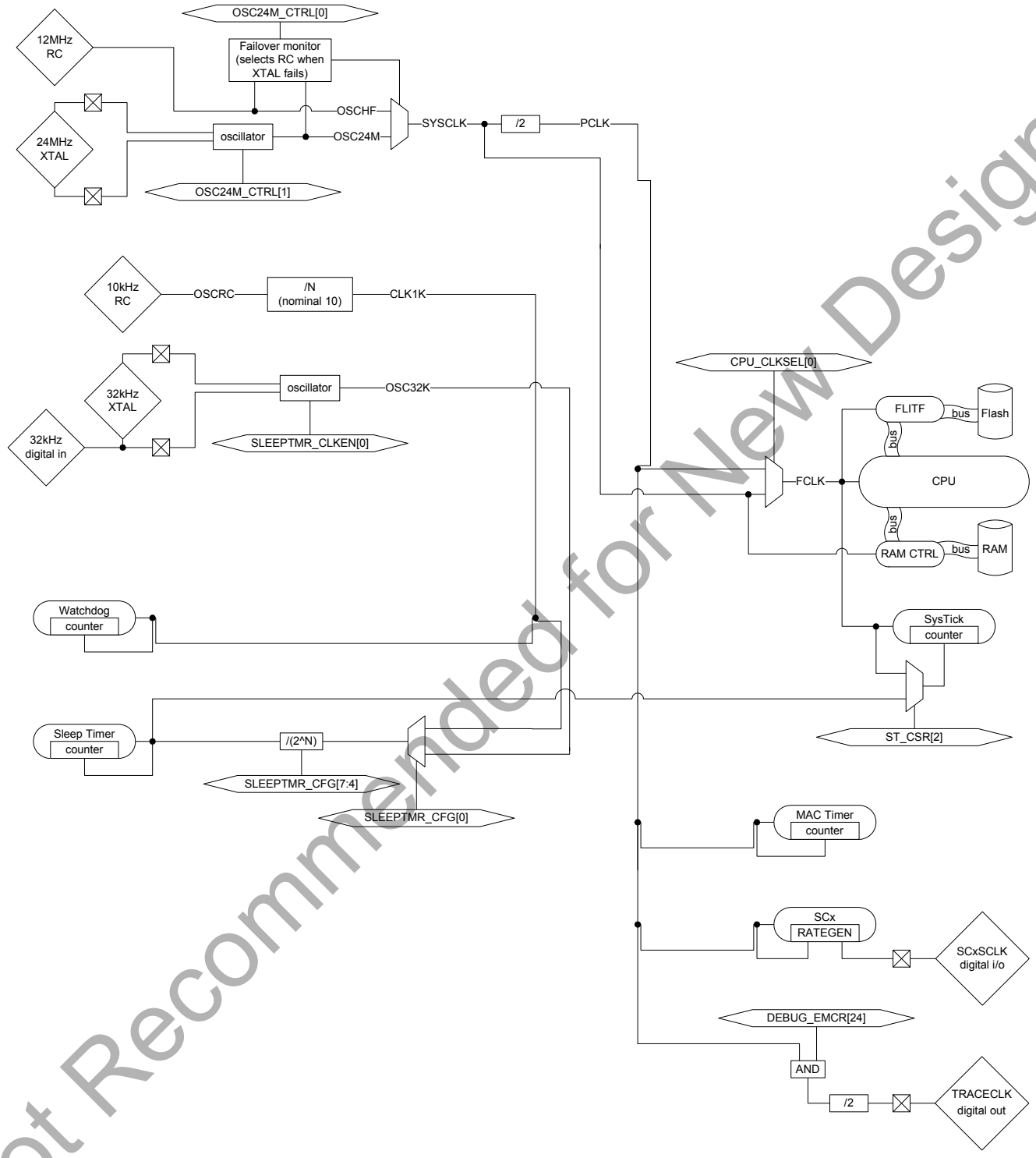


Figure 6.2. Clocks Block Diagram

6.3.1. High-Frequency Internal RC Oscillator (OSCHF)

The high-frequency RC oscillator (OSCHF) is used as the default system clock source when power is applied to the core domain. The nominal frequency coming out of reset is 12 MHz and Ember software calibrates this clock to 12 MHz. Table 6.6 contains the specification for the high frequency RC oscillator.

Most peripherals, excluding the radio peripheral, are fully functional using the OSCHF clock source. Application software must be aware that peripherals are clocked at different speeds depending on whether OSCHF or OSC24M is being used. Since the frequency step of OSCHF is 0.3 MHz and the high-frequency crystal oscillator is used for calibration, the calibrated accuracy of OSCHF is ±150 kHz ±40 ppm. The UART peripheral may not be usable due to the lower accuracy of the OSCHF frequency.

Table 6.6. High-Frequency RC Oscillator Specification

Parameter	Test Conditions	Min	Typ	Max	Unit
Frequency at reset		6	12	20	MHz
Frequency Steps		—	0.3	—	MHz
Duty cycle		40	—	60	%
Supply dependence	Change in supply = 0.1 V Test at supply changes: 1.8 to 1.7 V	—	—	5	%

6.3.2. High-Frequency Crystal Oscillator (OSC24M)

The high-frequency crystal oscillator (OSC24M) requires an external 24 MHz crystal with an accuracy of ±40 ppm. Based upon the application's bill of materials and current consumption requirements, the external crystal may cover a range of ESR requirements. Table 6.7 contains the specification for the high frequency crystal oscillator.

The crystal oscillator has a software-programmable bias circuit to minimize current consumption. Ember software configures the bias circuit for minimum current consumption.

All peripherals including the radio peripheral are fully functional using the OSC24M clock source. Application software must be aware that peripherals are clocked at different speeds depending on whether OSCHF or OSC24M is being used.

If the 24 MHz crystal fails, a hardware failover mechanism forces the system to switch back to the high-frequency RC oscillator as the main clock source, and a non-maskable interrupt (NMI) is signaled to the ARM® Cortex™-M3 NVIC.

Table 6.7. High-Frequency Crystal Oscillator Specification

Parameter	Test Conditions	Min	Typ	Max	Unit
Frequency		—	24	—	MHz
Accuracy		–40	—	+40	ppm
Duty cycle		40	—	60	%
Start-up time at max bias		—	—	1	ms
Start up time at optimal bias		—	—	2	ms
Current consumption		—	200	300	µA
Current consumption at max bias		—	—	1	mA
Crystal with high ESR		—	—	100	Ω
Load capacitance		—	—	10	pF
Crystal capacitance		—	—	7	pF
Crystal power dissipation		—	—	200	µW
Crystal with low ESR		—	—	60	Ω
Load capacitance		—	—	18	pF
Crystal capacitance		—	—	7	pF
Crystal power dissipation		—	—	1	mW

6.3.3. Low-Frequency Internal RC Oscillator (OSCRC)

A low-frequency RC oscillator (OSCRC) is provided as an internal timing reference. The nominal frequency coming out of reset is 10 kHz, and Ember software calibrates this clock to 10 kHz. From the tuned 10 kHz oscillator (OSCRC) Ember software calibrates a fractional-N divider to produce a 1 kHz reference clock, CLK1K. Table 6.8 contains the specification for the low frequency RC oscillator.

Table 6.8. Low-Frequency RC Oscillator Specification

Parameter	Test Condition	Min	Typ	Max	Unit
Nominal frequency	After trimming	9	10	11	kHz
Analog trim step size		—	0.5	—	kHz
Supply dependence	For a voltage drop from 3.6 V to 3.1 V or 2.6 V to 2.1 V (without re-calibration)	—	1	—	%
Temperature dependence	Frequency variation with temperature for a change from –40 to +85 ^{o}C (without re-calibration)	—	2	—	%

6.3.4. Low-Frequency Crystal Oscillator (OSC32K)

A low-frequency 32.768 kHz crystal oscillator (OSC32K) is provided as an optional timing reference for on-chip timers. This oscillator is designed for use with an external watch crystal. When using the 32.768 kHz crystal, you must connect it to GPIO PC6 and PC7 and must configure these two GPIOs for analog input. Alternatively, when PC7 is configured as a digital input, PC7 can accept an external digital clock input instead of a 32.786 kHz crystal. The digital clock input signal must be a 1 V peak-to-peak sine wave with a dc bias of 0.5 V. Refer to "7. GPIO (General Purpose Input/Output)" on page 48 for GPIO configuration details. Using the low-frequency oscillator, crystal or digital clock, is enabled through Ember software.

Table 6.9 contains the specification for the low frequency crystal oscillator.

Table 6.9. Low-Frequency Crystal Oscillator Specification

Parameter	Test conditions	Min	Typ	Max	Unit
Frequency		—	32.768	—	kHz
Accuracy	At 25 ºC	–20	—	+20	ppm
Load capacitance OSC32A		—	27	—	pF
Load capacitance OSC32B		—	18	—	pF
Crystal ESR		—	—	100	kΩ
Start-up time		—	—	2	s
Current consumption	At 25 °C, VDD_PADS=3.0 V	—	—	0.5	µA

6.3.5. Clock Switching

The EM342 has two switching mechanisms for the main system clock, providing four clock modes. Table 6.10 shows these clock modes and how they affect the internal clocks.

The register bit OSC24M_CTRL_OSC24M_SEL in the OSC24M_CTRL register switches between the high-frequency RC oscillator (OSCHF) and the high-frequency crystal oscillator (OSC24M) as the main system clock (SYSCLK). The peripheral clock (PCLK) is always half the frequency of SYSCLK.

The register bit CPU_CLKSEL_FIELD in the CPU_CLKSEL register switches between PCLK and SYSCLK to produce the ARM® Cortex™-M3 CPU clock (FCLK). The default and preferred mode of operation is to run the CPU at the higher PCLK frequency, 24 MHz, to give higher processing performance for all applications and improved duty cycling for applications using sleep modes.

In addition to these modes, further automatic control is invoked by hardware when flash programming is enabled. To ensure accuracy of the flash controller's timers, the FCLK frequency is forced to 12 MHz during flash programming and erase operations.

Table 6.10. System Clock Modes

OSC24M_CTRL_OSC24M_SEL	CPU_CLKSEL_FIELD	SYSCLK	PCLK	FCLK	
				Flash Program/ Erase Inactive	Flash Program/ Erase Active
0 (OSCHF)	0 (Normal CPU)	12 MHz	6 MHz	6 MHz	12 MHz
0 (OSCHF)	1 (Fast CPU)	12 MHz	6 MHz	12 MHz	12 MHz
1 (OSC24M)	0 (Normal CPU)	24 MHz	12 MHz	12 MHz	12 MHz
1 (OSC24M)	1 (Fast CPU)	24 MHz	12 MHz	24 MHz	12 MHz

6.4. System Timers

6.4.1. Watchdog Timer

The EM342 integrates a watchdog timer which can be enabled to provide protection against software crashes and ARM® Cortex™-M3 CPU lockup. By default, it is disabled at power up of the always-on power domain. The watchdog timer uses the calibrated 1 kHz clock (CLK1K) as its reference and provides a nominal 2.048 s timeout. A low water mark interrupt occurs at 1.792 s and triggers an NMI to the ARM® Cortex™-M3 NVIC as an early warning. When the watchdog is enabled, the timer must be periodically reset before it expires. The watchdog timer is paused when the debugger halts the ARM® Cortex™-M3. Additionally, the Ember software that implements deep sleep functionality disables the watchdog when entering deep sleep and restores the watchdog, if it was enabled, when exiting deep sleep.

Ember software provides an API for enabling, resetting, and disabling the watchdog timer.

6.4.2. Sleep Timer

The EM342 integrates a 32-bit timer dedicated to system timing and waking from sleep at specific times. The sleep timer can use either the calibrated 1 kHz reference (CLK1K), or the 32 kHz crystal clock (CLK32K). The default clock source is the internal 1 kHz clock.

The sleep timer has a prescaler, a divider of the form 2^N, where N can be programmed from 1 to 2^15. This divider allows for very long periods of sleep to be timed. Ember software's default configuration is to use the prescaler to always produce a 1024 Hz sleep timer tick. The timer provides two compare outputs and wrap detection, all of which can be used to generate an interrupt or a wake up event.

While it is possible to do so, by default the sleep timer is not paused when the debugger halts the ARM® Cortex™-M3. Silicon Labs does not advise pausing the sleep timer when the debugger halts the CPU.

To save current during deep sleep, the low-frequency internal RC oscillator (OSCRC) can be turned off. If OSCRC is turned off during deep sleep and a low-frequency 32.768 kHz crystal oscillator is not being used, then the sleep timer will not operate during deep sleep and sleep timer wake events cannot be used to wake up the EM342.

Ember software provides the system timer software API for interacting with the sleep timer as well as using the sleep timer and RC oscillator during deep sleep.

Note: Because the system timer software module handles all interactions with the sleep timer, the module will return the correct value in all situations. In the situation where the chip performs a deep sleep that maintains the system time and is woken up from an external event (that is, not a sleep timer event), the deep sleep module in the Ember software delays until the next sleep timer clock tick (up to 1 ms) to guarantee that the sleep timer updates correctly.

6.4.3. Event Timer

The SysTick timer is an ARM® standard system timer in the NVIC. The SysTick timer can be clocked from either the FCLK (the clock going into the CPU) or the Sleep Timer clock. FCLK is either the SYSCLK or PCLK as selected by CPU_CLKSEL register (see "6.3.5. Clock Switching").

6.5. Power Management

The EM342's power management system is designed to achieve the lowest deep sleep current consumption possible while still providing flexible wakeup sources, timer activity, and debugger operation. The EM342 has four main sleep modes:

- Idle Sleep: Puts the CPU into an idle state where execution is suspended until any interrupt occurs. All power domains remain fully powered and nothing is reset.
- Deep Sleep 1: The primary deep sleep state. In this state, the core power domain is fully powered down and the sleep timer is active.
- Deep Sleep 2: The same as Deep Sleep 1 except that the sleep timer is inactive to save power. In this mode the sleep timer cannot wake up the EM342.
- Deep Sleep 0 (also known as Emulated Deep Sleep): The chip emulates a true deep sleep without powering down the core domain. Instead, the core domain remains powered and all peripherals except the system debug components (ITM, DWT, FPB, NVIC) are held in reset. The purpose of this sleep state is to allow EM342 software to perform a deep sleep cycle while maintaining debug configuration such as breakpoints.
  CSYSPWRUPREQ, CDBGPWRUPREQ, and the corresponding CSYSPWRUPACK and CDBGPWRUPACK are bits in the debug port's CTRL/STAT register in the SWJ. For further information on these bits and the operation of the SWJ-DP please refer to the ARM Debug Interface v5 Architecture Specification (ARM IHI 0031A).

For further power savings when not in deep sleep, the Serial Controller1 peripheral can be individually disabled through the PERIPHERAL_DISABLE register. Disabling a peripheral saves power by stopping the clock feeding that peripheral. A peripheral should only be disabled through the PERIPHERAL_DISABLE register when the peripheral is idle and disabled through the peripheral's own configuration registers, otherwise undefined behavior may occur. When a peripheral is disabled through the PERIPHERAL_DISABLE register, all registers associated with that peripheral ignore all subsequent writes, and subsequent reads return the value seen in the register at the moment the peripheral is disabled.

6.5.1. Wake Sources

When in deep sleep the EM342 can be returned to the running state in a number of ways, and the wake sources are split depending on deep sleep 1 or deep sleep 2.

The following wake sources are available in both deep sleep 1 and 2.

- Wake on GPIO activity: Wake due to change of state on any GPIO.
- Wake on serial controller 1: Wake due to a change of state on GPIO Pin PB2.
- Wake on IRQD: Wake due to a change of state on IRQD. Since IRQD can be configured to point to any GPIO, this wake source is another means of waking on any GPIO activity.
- Wake on setting of CDBGPWRUPREQ: Wake due to setting the CDBGPWRUPREQ bit in the debug port in the SWJ.
- Wake on setting of CSYSPWRUPREQ: Wake due to setting the CSYSPWRUPREQ bit in the debug port in the SWJ.

The following sources are only available in deep sleep 1 since the sleep timer is not active in deep sleep 2.

- Wake on sleep timer compare A.
- Wake on sleep timer compare B.
- Wake on sleep timer wrap.

The following source is only available in deep sleep 0 since the SWJ is required to write a memory mapped register to set this wake source and the SWJ only has access to some registers in deep sleep 0.

- Wake on write to the WAKE_CORE register bit.

The Wakeup Recording module monitors all possible wakeup sources. More than one wakeup source may be recorded because events are continually being recorded (not just in deep-sleep) and another event may happen between the first wake event and when the EM342 wakes up.

6.5.2. Basic Sleep Modes

The power management state diagram in Figure 6.3 shows the basic operation of the power management controller.

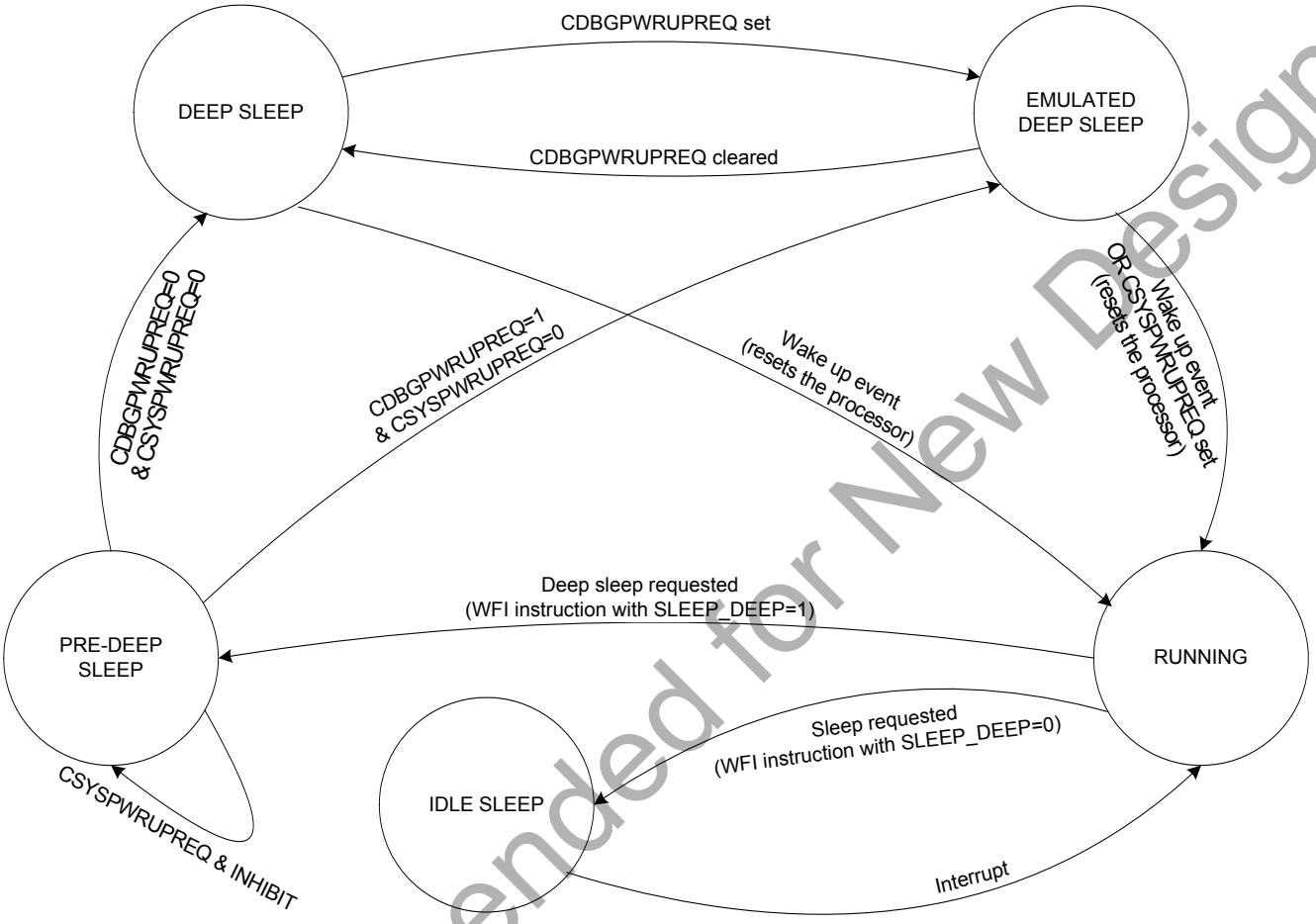


Figure 6.3. Power Management State Diagram

In normal operation an application may request one of two low power modes through program execution:

- Idle Sleep is achieved by executing a WFI instruction while the SLEEPDEEP bit in the Cortex System Control register (SCS_SCR) is clear. This puts the CPU into an idle state where execution is suspended until an interrupt occurs. This is indicated by the state at the bottom of the diagram. Power is maintained to the core logic of the EM342 during the Idle Sleeping state.
- Deep sleep is achieved by executing a WFI instruction with the SLEEPDEEP bit in SCS_SCR set. This triggers the state transitions around the main loop of the diagram, resulting in powering down the EM342's core logic, and leaving only the always-on domain powered. Wake up is triggered when one of the pre-determined events occurs.

If a deep sleep is requested the EM342 first enters a pre-deep sleep state. This state prevents any section of the chip from being powered off or reset until the SWJ goes idle (by clearing CSYSPWRUPREQ). This pre-deep sleep state ensures debug operations are not interrupted.

In the deep sleep state the EM342 waits for a wake up event which will return it to the running state. In powering up the core logic the ARM® Cortex™-M3 is put through a reset cycle and Ember software restores the stack and application state to the point where deep sleep was invoked.

6.5.3. Further Options for Deep Sleep

By default the low-frequency internal RC oscillator (OSCRC) is running during deep sleep (known as deep sleep 1).

To conserver power, OSCRC can be turned of during deep sleep. This mode is known as deep sleep 2. Since the OSCRC is disabled, the sleep timer and watchdog timer do not function and cannot wake the chip unless the low-frequency 32.768 kHz crystal oscillator is used. Non-timer based wake sources continue to function. Once a wake event does occur, OSCRC is restarted and comes back up.

6.5.4. Use of Debugger with Sleep Modes

The debugger communicates with the EM342 using the SWJ.

When the debugger is logically connected, the CDBGPWRUPREQ bit in the debug port in the SWJ is set, and the EM342 will only enter deep sleep 0 (the Emulated Deep Sleep state). The CDBGPWRUPREQ bit indicates that a debug tool is logically connected to the chip and therefore debug state may be in the system debug components. To maintain the debug state in the system debug components only deep sleep 0 may be used, since deep sleep 0 will not cause a power cycle or reset of the core domain. The CSYSPWRUPREQ bit in the debug port in the SWJ indicates that a debugger wants to access memory actively in the EM342. Therefore, whenever the CSYSPWRUPREQ bit is set while the EM342 is awake, the EM342 cannot enter deep sleep until this bit is cleared. This ensures the EM342 does not disrupt debug communication into memory.

Clearing both CSYSPWRUPREQ and CDBGPWRUPREQ allows the EM342 to achieve a true deep sleep state (deep sleep 1 or 2). Both of these signals also operate as wake sources, so that when a debugger logically connects to the EM342 and begins accessing the chip, the EM342 automatically comes out of deep sleep. When the debugger initiates access while the EM342 is in deep sleep, the SWJ intelligently holds off the debugger for a brief period of time until the EM342 is properly powered and ready.

Note: The SWJ-DP signals CSYSPWRUPREQ and CDBGPWRUPREQ are only reset by a power-on-reset or a debugger. Physically connecting or disconnecting a debugger from the chip will not alter the state of these signals. A debugger must logically communicate with the SWJ-DP to set or clear these two signals.

For more information regarding the SWJ and the interaction of debuggers with deep sleep, contact customer support for Application Notes and ARM® CoreSight™ documentation.

6.5.5. Registers

Register 6.1. PERIPHERAL_DISABLE

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	0	0	PERIDIS_RSVD	0	0	0	PERIDIS_SC1	0

Address: 0x40004038 Reset: 0x0

Bitname	Bitfield	Access	Description
PERIDIS_RSVD	[5]	RW	Reserved: This bit can change during normal operation. When writing to PERIPHERAL_DISABLE, the value of this bit must be preserved.
PERIDIS_SC1	[1]	RW	Disable the clock to the SC1 peripheral.

6.6. Security Accelerator

The EM342 contains a hardware AES encryption engine accessible from the ARM® Cortex™-M3. 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.

7. GPIO (General Purpose Input/Output)

The EM342 has 16 GPIO pins, which may be individually configured as follows:

- General purpose output
- General purpose open-drain output
- Alternate output controlled by a peripheral device
- Alternate open-drain output controlled by a peripheral device
- Analog
- General purpose input
- General purpose input with pull-up or pull-down resistor

The basic structure of a single GPIO is illustrated in GPIO Block DiagramFigure 7.1.

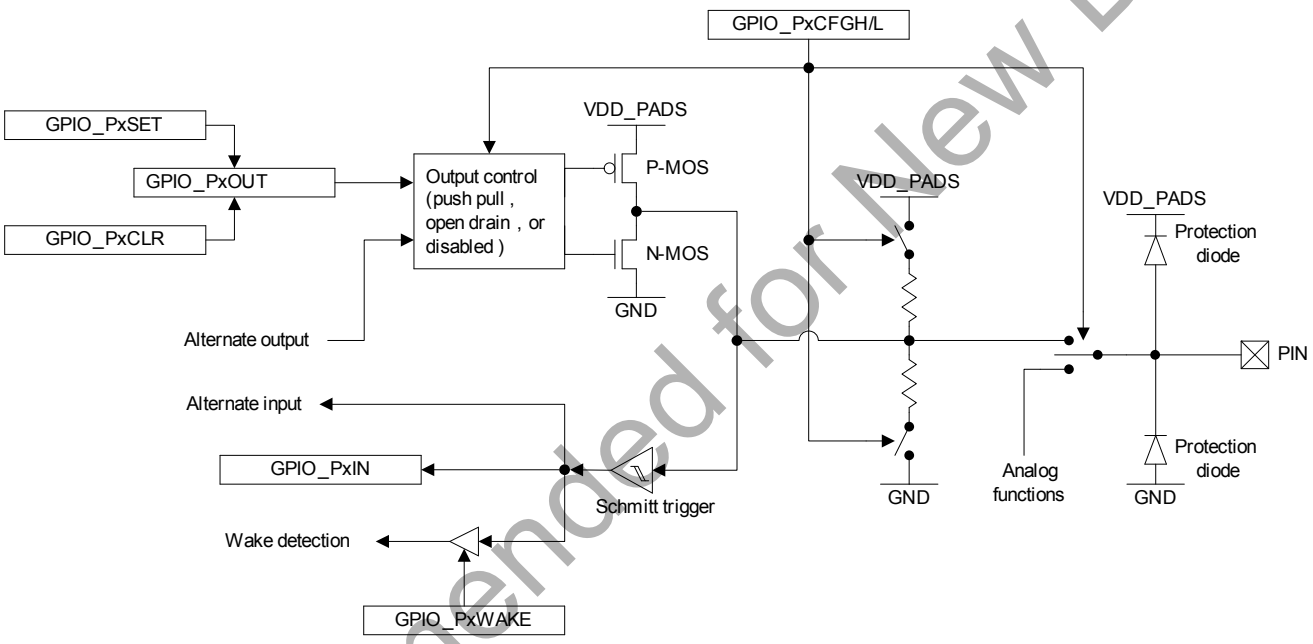


Figure 7.1. GPIO Block Diagram

A Schmitt trigger converts the GPIO pin voltage to a digital input value. The digital input signal is then always routed to the GPIO_PxIN register; to the alternate inputs of associated peripheral devices; to wake detection logic if wake detection is enabled; and, for certain pins, to interrupt generation logic. Configuring a pin in analog mode disconnects the digital input from the pin and applies a high logic level to the input of the Schmitt trigger.

Only one device at a time can control a GPIO output. The output is controlled in normal output mode by the GPIO_PxOUT register and in alternate output mode by a peripheral device. When in input mode or analog mode, digital output is disabled.

7.1. GPIO Ports

The 16 GPIO pins are grouped into three ports: PA, PB, and PC. Individual GPIOs within a port are numbered 0 to 7 according to their bit positions within the GPIO registers.

Note: Because GPIO port registers' functions are identical, the notation Px is used here to refer to PA, PB, or PC. For example, GPIO_PxIN refers to the registers GPIO_PAIN, GPIO_PBIN, and GPIO_PCIN.

Each of the three GPIO ports has the following registers whose low-order eight bits correspond to the port's eight GPIO pins:

- GPIO_PxIN (input data register) returns the pin level (unless in analog mode).
- GPIO_PxOUT (output data register) controls the output level in normal output mode.
- GPIO_PxCLR (clear output data register) clears bits in GPIO_PxOUT.
- GPIO_PxSET (set output data register) sets bits in GPIO_PxOUT.
- GPIO_PxWAKE (wake monitor register) specifies the pins that can wake the EM342.

In addition to these registers, each port has a pair of configuration registers, GPIO_PxCFGH and GPIO_PxCFGL. These registers specify the basic operating mode for the port's pins. GPIO_PxCFGL configures the pins Px[3:0] and GPIO_PxCFGH configures the pins Px[7:4]. For brevity, the notation GPIO_PxCFGH/L refers to the pair of configuration registers.

Two GPIO pins (PA7 and PC0) can sink and source higher current than standard GPIO outputs. Refer to Table 2.5 Digital I/O Specifications in "2. Electrical Specifications" on page 7 for more information.

7.2. Configuration

Each pin has a 4-bit configuration value in the GPIO_PxCFGH/L register. The various GPIO modes and their 4-bit configuration values are shown in Table 7.1.

Table 7.1. GPIO Configuration Modes

GPIO Mode	GPIO_PxCFGH/L	Description
Analog	0x0	Analog input or output. When in analog mode, the digital input (GPIO_PxIN) always reads 1.
Input (floating)	0x4	Digital input without an internal pull up or pull down. Output is disabled.
Input (pull-up or pull-down)	0x8	Digital input with an internal pull up or pull down. A set bit in GPIO_PxOUT selects pull up and a cleared bit selects pull down. Output is disabled.
Output (push-pull)	0x1	Push-pull output. GPIO_PxOUT controls the output.
Output (open-drain)	0x5	Open-drain output. GPIO_PxOUT controls the output. If a pull up is required, it must be external.
Alternate Output (push-pull)	0x9	Push-pull output. An onboard peripheral controls the output.
Alternate Output (open-drain)	0xD	Open-drain output. An onboard peripheral controls the output. If a pull up is required, it must be external.

If a GPIO has two peripherals that can be the source of alternate output mode data, then other registers in addition to GPIO_PxCFGH/L determine which peripheral controls the output.

If a GPIO does not have an associated peripheral in alternate output mode, its output is set to 0.

For outputs assigned to the serial controllers, the serial interface mode registers (SCx_MODE) determine how the GPIO pins are used. The alternate outputs of PA4 and PA5 can either provide packet trace data (PTI_EN and PTI_DATA) or synchronous CPU trace data (TRACEDATA2 and TRACEDATA3). The selection of packet trace or CPU trace is made through the Ember software.

7.3. Forced Functions

For some GPIOs, the GPIO_PxCFGH/L configuration will be overridden. These functions are forced when the EM342 is reset and remain forced until software overrides the forced functions. Table 7.2 shows the GPIOs that have different functions forced on them regardless of the GPIO_PxCFGH/L registers.

Table 7.2. GPIO Forced Functions

GPIO	Forced Mode	Forced Signal
PA7	Open-drain output	REG_EN
PC0	Input with pull up	JRST
PC2	Push-pull output	JTDO
PC3	Input with pull up	JDTI
PC4*	Input with pull up	JTMS
PC4*	Bidirectional (push-pull output or floating input) controlled by debugger interface	SWDIO

***Note:** The choice of PC4's forced signal is controlled by an external debug tool. JTMS is forced when the SWJ is in JTAG mode, and SWDIO is forced when the SWJ is in Serial Wire mode.

PA7 is forced to be the regulator enable signal, REG_EN. If an external regulator is used and controlled through REG_EN, PA7's forced functionality must not be overridden. If an external regulator is not used, REG_EN may be disabled and PA7 may be reclaimed as a normal GPIO. Disabling REG_EN is done by clearing the bit GPIO_EXTREGEN in the GPIO_DBGCFG register.

PC0, PC2, PC3, and PC4 are forced to be the Serial Wire and JTAG (SWJ) Interface. When the EM342 resets, these four GPIOs are forced to operate in JTAG mode. Switching the debug interface between JTAG mode and Serial Wire mode can only be accomplished by the external debug tool and cannot be affected by software executing on the EM342. Due to the fact that Serial Wire mode can only be invoked by an external debug tool and JTAG mode is forced when the EM342 resets, a designer must treat all four debug GPIOs as working in unison even though the Serial Wire interface only uses one of the GPIO, PC4.

Note: An application must disable all debug SWJ debug functionality to reclaim any of the four GPIOs: PC0, PC2, PC3, and PC4. Disabling SWJ debug functionality prevents external debug tools from operating, including flash programming and high-level debug tools.

Disabling the SWJ debugger interface is accomplished by setting the GPIO_DEBUGDIS bit in the GPIO_DBGCFG register. When this bit is set, all debugger-related pins (PC0, PC2, PC3, PC4) behave as standard GPIOs. If the SWJ debugger interface is already active, the bit GPIO_DEBUGDIS cannot be set. When GPIO_DEBUGDIS is set, the SWJ debugger interface can be reclaimed by activating the SWJ while the EM342 is held in reset. If the SWJ debugger interface is forced active in this manner, the bit GPIO_FORCEDBG is set in the GPIO_DBGSTAT register. The SWJ debugger interface is defined as active when the CDBGPWRUPREQ signal, a bit in the debug port's CRTL/STAT register in the SWJ, is set high by an external debug tool.

7.4. Reset

A full chip reset is one due to power on (low or high voltage), the nRESET pin, the watchdog, or the SYSRESETREQ bit. A full chip reset affects the GPIO configuration as follows:

- The GPIO_PxCFGH/L configurations of all pins are configured as floating inputs.
- The GPIO_EXTREGEN bit is set in the GPIO_DBGCFG register, which overrides the normal configuration for PA7.
- The GPIO_DEBUGDIS bit in the GPIO_DBGCFG register is cleared, allowing Serial Wire/JTAG access to override the normal configuration of PC0, PC2, PC3, and PC4.

7.5. Boot Configuration

nBOOTMODE is a special alternate function of PA5 that is active only during a pin reset (nRESET) or a power-on-reset of the always-powered domain (POR HV). If nBOOTMODE is asserted (pulled or driven low) when coming out of reset, the processor starts executing an embedded serial-link-only monitor instead of its normal program.

While in reset and during the subsequent power-on-reset startup delay (512 OSCHF clocks), PA5 is automatically configured as an input with a pull-up resistor. At the end of this time, the EM342 samples nBOOTMODE: a high level selects normal boot mode, and a low level selects the embedded monitor. Figure 7.2 shows the timing parameters for invoking monitor mode from a pin (nRESET) reset. Because OSCHF is running uncalibrated during the reset sequence, the time for 512 OSCHF clocks may vary as indicated.

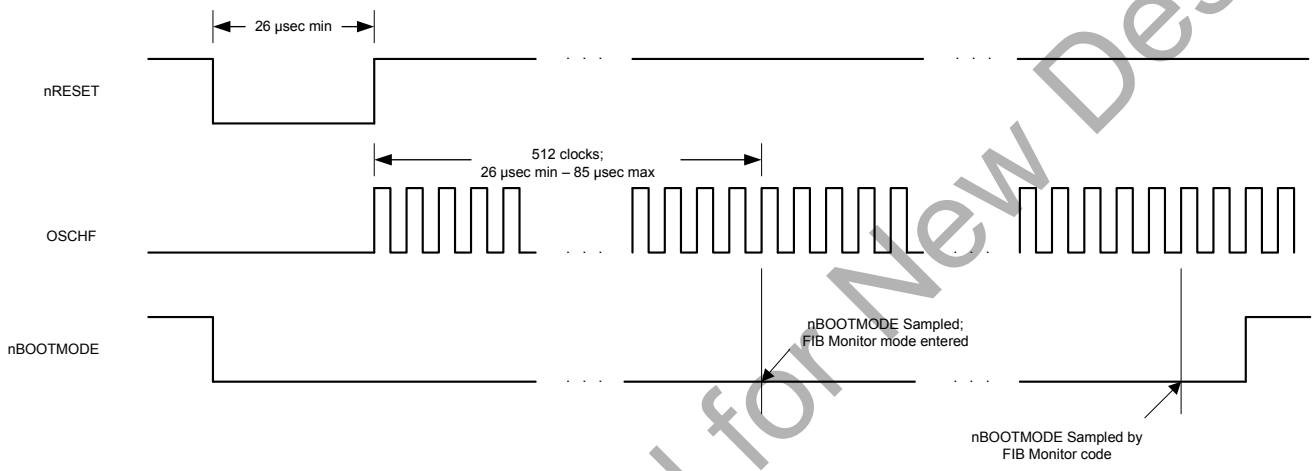


Figure 7.2. nBOOTMODE and nRESET Timing

Timing for a power-on-reset is similar except that OSCHF does not begin oscillating until up to 70 µsec after both core and HV supplies are valid. Combined with the maximum 250 µsec allowed for HV to ramp from 0.5 V to 1.7 V, an additional 320 µsec may be added to the 512 OSCHF clocks until nBOOTMODE is sampled.

If the monitor mode is selected (nBOOTMODE is low after 512 clocks), the FIB monitor software begins execution. In order to filter out inadvertent jumps into the monitor, the FIB monitor re-samples the nBOOTMODE signal after a 3 ms delay. If the signal is still low, then the device stays in monitor mode. If the signal is high, then monitor mode is exited and the normal program begins execution. In summary, the nBOOTMODE signal must be held low for 4 ms in order to properly invoke the FIB monitor.

After nBOOTMODE has been sampled, PA5 is configured as a floating input like the other GPIO configurations. The GPIO_BOOTMODE bit in the GPIO_DBGSTAT register captures the state of nBOOTMODE so that software may act on this signal if required.

Note: To avoid inadvertently asserting nBOOTMODE, PA5's capacitive load may not exceed 250 pF.

7.6. GPIO Modes

7.6.1. Analog Mode

Analog mode enables analog functions, and disconnects a pin from the digital input and output logic. Only the following GPIO pins have analog functions:

- PC6 and PC7 can connect to an optional 32.768 kHz crystal.

Note: When an external timing source is required, a 32.768 kHz crystal is commonly connected to PC6 and PC7. Alternatively, when PC7 is configured as a digital input, PC7 can accept a digital external clock input.

When configured in analog mode:

- The output drivers are disabled.
- The internal pull-up and pull-down resistors are disabled.
- The Schmitt trigger input is connected to a high logic level.

- Reading GPIO_PxIN returns a constant 1.

7.6.2. Input Mode

Input mode is used both for general purpose input and for on-chip peripheral inputs. Input floating mode disables the internal pull-up and pull-down resistors, leaving the pin in a high-impedance state. Input pull-up or pull-down mode enables either an internal pull-up or pull-down resistor based on the GPIO_PxOUT register. Setting a bit to 0 in GPIO_PxOUT enables the pull-down and setting a bit to 1 enables the pull up.

When configured in input mode:

- The output drivers are disabled.
- An internal pull-up or pull-down resistor may be activated depending on GPIO_PxCFGH/L and GPIO_PxOUT.
- The Schmitt trigger input is connected to the pin.
- Reading GPIO_PxIN returns the input at the pin.
- The input is also available to on-chip peripherals.

7.6.3. Output Mode

Output mode provides a general purpose output under direct software control. Regardless of whether an output is configured as push-pull or open-drain, the GPIO's bit in the GPIO_PxOUT register controls the output. The GPIO_PxSET and GPIO_PxCLR registers can atomically set and clear bits within GPIO_PxOUT register. These set and clear registers simplify software using the output port because they eliminate the need to disable interrupts to perform an atomic read-modify-write operation of GPIO_PxOUT.

When configured in output mode:

- The output drivers are enabled and are controlled by the value written to GPIO_PxOUT:
  - In open-drain mode: 0 activates the N-MOS current sink; 1 tri-states the pin.
  - In push-pull mode: 0 activates the N-MOS current sink; 1 activates the P-MOS current source.
- The internal pull-up and pull-down resistors are disabled.
- The Schmitt trigger input is connected to the pin.
- Reading GPIO_PxIN returns the input at the pin.

Note: Reading GPIO_PxOUT returns the last value written to the register.
Depending on configuration and usage, GPIO_PxOUT and GPIO_PxIN may not have the same value.

7.6.4. Alternate Output Mode

In this mode, the output is controlled by an on-chip peripheral instead of GPIO_PxOUT and may be configured as either push-pull or open-drain. Most peripherals require a particular output type, but since using a peripheral does not by itself configure a pin, the GPIO_PxCFGH/L registers must be configured properly for a peripheral's particular needs. As described in "7.2. Configuration" on page 49, when more than one peripheral can be the source of output data, registers in addition to GPIO_PxCFGH/L determine which to use.

When configured in alternate output mode:

- The output drivers are enabled and are controlled by the output of an on-chip peripheral:
  - In open-drain mode: 0 activates the N-MOS current sink; 1 tri-states the pin.
  - In push-pull mode: 0 activates the N-MOS current sink; 1 activates the P-MOS current source.
- The internal pull-up and pull-down resistors are disabled.
- The Schmitt trigger input is connected to the pin.

Note: Reading GPIO_PxIN returns the input to the pin.
Depending on configuration and usage, GPIO_PxOUT and GPIO_PxIN may not have the same value.

7.7. Wake Monitoring

The GPIO_PxWAKE registers specify which GPIOs are monitored to wake the processor. If a GPIO's wake enable bit is set in GPIO_PxWAKE, then a change in the logic value of that GPIO causes the EM342 to wake from deep sleep. The logic values of all GPIOs are captured by hardware upon entering sleep. If any GPIO's logic value changes while in sleep and that GPIO's GPIO_PxWAKE bit is set, then the EM342 wakes from deep sleep. (There is no mechanism for selecting a specific rising-edge, falling-edge, or level on a GPIO: any change in logic value

triggers a wake event.) Hardware records the fact that GPIO activity caused a wake event, but not which specific GPIO was responsible. Instead, the Ember software reads the state of the GPIOs on waking to determine this.

The register GPIO_WAKEFILT contains bits to enable digital filtering of the external wakeup event sources: the GPIO pins, SC1 activity, and IRQD. The digital filter operates by taking samples based on the (nominal) 10 kHz RC oscillator. If three samples in a row all have the same logic value, and this sampled logic value is different from the logic value seen upon entering sleep, the filter outputs a wakeup event.

In order to use GPIO pins to wake the EM342 from deep sleep, the GPIO_WAKE bit in the WAKE_SEL register must be set. Waking up from GPIO activity does not work with pins configured for analog mode since the digital logic input is always set to 1 when in analog mode. Refer to "6. System Modules" on page 33 for information on the EM342's power management and sleep modes.

7.8. External Interrupts

The EM342can use up to three external interrupt sources (IRQA, IRQC, and IRQD), each with its own top-level NVIC interrupt vector. Since these external interrupt sources connect to the standard GPIO input path, an external interrupt pin may simultaneously be used by a peripheral device or even configured as an output. Analog mode is the only GPIO configuration that is not compatible with using a pin as an external interrupt.

External interrupts have individual triggering and filtering options selected using the registers GPIO_INTCFGA, GPIO_INTCFGB, GPIO_INTCFGC, and GPIO_INTCFGD. The bit field GPIO_INTMOD of the GPIO_INTCFGx register enables IRQx's second-level interrupt and selects the triggering mode: 0 is disabled; 1 for rising edge; 2 for falling edge; 3 for both edges; 4 for active high level; 5 for active low level. The minimum width needed to latch an unfiltered external interrupt in both level- and edge-triggered mode is 80 ns. With the digital filter enabled (the GPIO_INTFILT bit in the GPIO_INTCFGx register is set), the minimum width needed is 450 ns.

The register INT_GPIOFLAG is the second-level interrupt flag register that indicates pending external interrupts. Writing 1 to a bit in the INT_GPIOFLAG register clears the flag while writing 0 has no effect. If the interrupt is level-triggered, the flag bit is set again immediately after being cleared if its input is still in the active state.

IRQA has a fixed pin assignment. The other two external interrupts, IRQC and IRQD, can use any GPIO pin. The GPIO_IRQCSEL and GPIO_IRQDSEL registers specify the GPIO pins assigned to IRQC and IRQD, respectively. Table 7.3 shows how the GPIO_IRQCSEL and GPIO_IRQDSEL register values select the GPIO pin used for the external interrupt.

Table 7.3. IRQC/D GPIO Selection

GPIO_IRQxSEL	GPIO		GPIO_IRQxSEL	GPIO		GPIO_IRQxSEL	GPIO
4	PA4		8	PB0		16	PC0
5	PA5		9	PB1		17	PC1
7	PA7		10	PB2		18	PC2
			11	PB3		19	PC3
			12	PB4		20	PC4
						21	PC5
						22	PC6
						23	PC7

In some cases, it may be useful to assign IRQC or IRQD to an input also in use by a peripheral, for example to generate an interrupt from the slave select signal (nSSEL) in an SPI slave mode interface.

Refer to "9. Interrupt System" on page 100 for further information regarding the EM342 interrupt system.

7.9. Debug Control and Status

Two GPIO registers are largely concerned with debugger functions. GPIO_DBGCFG can disable debugger

operation, but has other miscellaneous control bits as well. GPIO_DBGSTAT, a read-only register, returns status related to debugger activity (GPIO_FORCEDBG and GPIO_SWEN), as well a flag (GPIO_BOOTMODE) indicating whether nBOOTMODE was asserted at the last power-on or nRESET-based reset.

7.10. GPIO Signal Assignment Summary

The GPIO signal assignments are shown in Table 7.4.

Table 7.4. GPIO Signal Assignments

GPIO	Analog	Alternate Output	Input	Output Current Drive
PA4		PTI_EN, TRACEDATA2		Standard
PA5		PTI_DATA, TRACEDATA3	nBOOTMODE[1]	Standard
PA7		REG_EN[2]		High
PB0		TRACECLK	IRQA	Standard
PB1		SC1TXD, SC1MOSI, SC1MISO, SC1SDA	SC1SDA	Standard
PB2		SC1SCLK	SC1MISO, SC1MOSI, SC1SCL, SC1RXD	Standard
PB3		SC1SCLK	SC1SCLK, SC1nCTS	Standard
PB4		SC1nRTS	SC1nSSEL	Standard
PC0		TRACEDATA1	JRST[3]	High
PC1		TRACEDATA0, SWO		Standard
PC2		JTDO[4], SWO		Standard
PC3			JTDI[3]	Standard
PC4		SWDIO[5]	SWDIO[5], JTMS[5]	Standard
PC5		TX_ACTIVE		Standard
PC6	OSC32B	nTX_ACTIVE		Standard
PC7	OSC32A			Standard

Notes:
1. Overrides during reset as an input with pull up.
2. Overrides after reset as an open-drain output.
3. Overrides in JTAG mode as a input with pull up.
4. Overrides in JTAG mode as a push-pull output.
5. Overrides in Serial Wire mode as either a push-pull output, or a floating input, controlled by the debugger.

7.11. Registers

Note: Substitute “A”, “B”, or “C” for “x” in the following detailed descriptions.

Register 7.1. GPIO_PxCFGL
GPIO_PACFGL: Port A Configuration Register (Low)
GPIO_PBCFGL: Port B Configuration Register (Low)
GPIO_PCCFGL: Port C Configuration Register (Low)

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	Px3_CFG				Px2_CFG			
Bit	7	6	5	4	3	2	1	0
Name	Px1_CFG				Px0_CFG			

GPIO_PACFGL: Address: 0x4000B000 Reset: 0x4444
GPIO_PBCFGL: Address: 0x4000B400 Reset: 0x4444
GPIO_PCCFGL: Address: 0x4000B800 Reset: 0x4444

Bitname	Bitfield	Access	Description
Px3_CFG	[15:12]	RW	GPIO configuration control. 0x0: Analog, input or output (GPIO_PxIN always reads 1). 0x1: Output, push-pull (GPIO_PxOUT controls the output). 0x4: Input, floating. 0x5: Output, open-drain (GPIO_PxOUT controls the output). 0x8: Input, pulled up or down (selected by GPIO_PxOUT: 0 = pull-down, 1 = pull-up). 0x9: Alternate output, push-pull (peripheral controls the output). 0xD: Alternate output, open-drain (peripheral controls the output).
Px2_CFG	[11:8]	RW	GPIO configuration control: see Px3_CFG above.
Px1_CFG	[7:4]	RW	GPIO configuration control: see Px3_CFG above.
Px0_CFG	[3:0]	RW	GPIO configuration control: see Px3_CFG above.

Register 7.2. GPIO_PxCFGH
GPIO_PACFGH: Port A Configuration Register (High)
GPIO_PBCFGH: Port B Configuration Register (High)
GPIO_PCCFGH: Port C Configuration Register (High)

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	Px7_CFG				Px6_CFG			
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Px5_CFG				Px4_CFG			

GPIO_PACFGH: Address: 0x4000B004 Reset: 0x4444
GPIO_PBCFGH: Address: 0x4000B404 Reset: 0x4444
GPIO_PCCFGH: Address: 0x4000B804 Reset: 0x4444

Bitname	Bitfield	Access	Description
Px7_CFG	[15:12]	RW	GPIO configuration control. 0x0: Analog, input or output (GPIO_PxIN always reads 1). 0x1: Output, push-pull (GPIO_PxOUT controls the output). 0x4: Input, floating. 0x5: Output, open-drain (GPIO_PxOUT controls the output). 0x8: Input, pulled up or down (selected by GPIO_PxOUT: 0 = pull-down, 1 = pull-up). 0x9: Alternate output, push-pull (peripheral controls the output). 0xD: Alternate output, open-drain (peripheral controls the output).
Px6_CFG	[11:8]	RW	GPIO configuration control: see Px7_CFG above.
Px5_CFG	[7:4]	RW	GPIO configuration control: see Px7_CFG above.
Px4_CFG	[3:0]	RW	GPIO configuration control: see Px7_CFG above.

Register 7.3. GPIO_PxIN
GPIO_PAIN: Port A Input Data Register
GPIO_PBIN: Port B Input Data Register
GPIO_PCIN: Port C Input Data Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0

GPIO_PAIN: Address: 0x4000B008 Reset: 0x0
GPIO_PBIN: Address: 0x4000B408 Reset: 0x0
GPIO_PCIN: Address: 0x4000B808 Reset: 0x0

Bitname	Bitfield	Access	Description
Px7	[7]	RW	Input level at pin Px7.
Px6	[6]	RW	Input level at pin Px6.
Px5	[5]	RW	Input level at pin Px5.
Px4	[4]	RW	Input level at pin Px4.
Px3	[3]	RW	Input level at pin Px3.
Px2	[2]	RW	Input level at pin Px2.
Px1	[1]	RW	Input level at pin Px1.
Px0	[0]	RW	Input level at pin Px0.

Register 7.4. GPIO_PxOUT
GPIO_PAOUT: Port A Output Data Register
GPIO_PBOUT: Port B Output Data Register
GPIO_PCOUT: Port C Output Data Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0

GPIO_PAOUT: Address: 0x4000B00C Reset: 0x0
GPIO_PBOUT: Address: 0x4000B40C Reset: 0x0
GPIO_PCOUT: Address: 0x4000B80C Reset: 0x0

Bitname	Bitfield	Access	Description
Px7	[7]	RW	Output data for Px7.
Px6	[6]	RW	Output data for Px6.
Px5	[5]	RW	Output data for Px5.
Px4	[4]	RW	Output data for Px4.
Px3	[3]	RW	Output data for Px3.
Px2	[2]	RW	Output data for Px2.
Px1	[1]	RW	Output data for Px1.
Px0	[0]	RW	Output data for Px0.

Register 7.5. GPIO_PxCLR
GPIO_PACLR: Port A Output Clear Register
GPIO_PBCLR: Port B Output Clear Register
GPIO_PCCLR: Port C Output Clear Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0

GPIO_PACLR: Address: 0x4000B014 Reset: 0x0
GPIO_PBCLR: Address: 0x4000B414 Reset: 0x0
GPIO_PCCLR: Address: 0x4000B814 Reset: 0x0

Bitname	Bitfield	Access	Description
Px7	[7]	W	Write 1 to clear the output data bit for Px7 (writing 0 has no effect).
Px6	[6]	W	Write 1 to clear the output data bit for Px6 (writing 0 has no effect).
Px5	[5]	W	Write 1 to clear the output data bit for Px5 (writing 0 has no effect).
Px4	[4]	W	Write 1 to clear the output data bit for Px4 (writing 0 has no effect).
Px3	[3]	W	Write 1 to clear the output data bit for Px3 (writing 0 has no effect).
Px2	[2]	W	Write 1 to clear the output data bit for Px2 (writing 0 has no effect).
Px1	[1]	W	Write 1 to clear the output data bit for Px1 (writing 0 has no effect).
Px0	[0]	W	Write 1 to clear the output data bit for Px0 (writing 0 has no effect).

Register 7.6. GPIO_PxSET
GPIO_PASET: Port A Output Set Register
GPIO_PBSET: Port B Output Set Register
GPIO_PCSET: Port C Output Set Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	GPIO_PXSETRSVD							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0

GPIO_PASET: Address: 0x4000B010 Reset: 0x0
GPIO_PBSET: Address: 0x4000B410 Reset: 0x0
GPIO_PCSET: Address: 0x4000B810 Reset: 0x0

Bitname	Bitfield	Access	Description
GPIO_PXSETRSVD	[15:8]	W	Reserved: these bits must be set to 0.
Px7	[7]	W	Write 1 to set the output data bit for Px7 (writing 0 has no effect).
Px6	[6]	W	Write 1 to set the output data bit for Px6 (writing 0 has no effect).
Px5	[5]	W	Write 1 to set the output data bit for Px5 (writing 0 has no effect).
Px4	[4]	W	Write 1 to set the output data bit for Px4 (writing 0 has no effect).
Px3	[3]	W	Write 1 to set the output data bit for Px3 (writing 0 has no effect).
Px2	[2]	W	Write 1 to set the output data bit for Px2 (writing 0 has no effect).
Px1	[1]	W	Write 1 to set the output data bit for Px1 (writing 0 has no effect).
Px0	[0]	W	Write 1 to set the output data bit for Px0 (writing 0 has no effect).

Register 7.7. GPIO_PxWAKE
GPIO_PAWAKE: Port A Wakeup Monitor Register
GPIO_PBWAKE: Port B Wakeup Monitor Register
GPIO_PCWAKE: Port C Wakeup Monitor Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0

GPIO_PAWAKE: Address: 0x4000BC08 Reset: 0x0
GPIO_PBWAKE: Address: 0x4000BC0C Reset: 0x0
GPIO_PCWAKE: Address: 0x4000BC10 Reset: 0x0

Bitname	Bitfield	Access	Description
Px7	[7]	RW	Write 1 to enable wakeup monitoring of Px7.
Px6	[6]	RW	Write 1 to enable wakeup monitoring of Px6.
Px5	[5]	RW	Write 1 to enable wakeup monitoring of Px5.
Px4	[4]	RW	Write 1 to enable wakeup monitoring of Px4.
Px3	[3]	RW	Write 1 to enable wakeup monitoring of Px3.
Px2	[2]	RW	Write 1 to enable wakeup monitoring of Px2.
Px1	[1]	RW	Write 1 to enable wakeup monitoring of Px1.
Px0	[0]	RW	Write 1 to enable wakeup monitoring of Px0.

Register 7.8. GPIO_WAKEFILT: GPIO Wakeup Filtering Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	0	0	0	0	IRQD_WAKE_ FILTER	0	SC1_WAKE_ FILTER	GPIO_WAKE_ FILTER

Address: 0x4000BC1C Reset: 0x0

Bitname	Bitfield	Access	Description
IRQD_WAKE_FILTER	[3]	RW	Enable filter on GPIO wakeup source IRQD.
SC1_WAKE_FILTER	[1]	RW	Enable filter on GPIO wakeup source SC1 (PB2).
GPIO_WAKE_FILTER	[0]	RW	Enable filter on GPIO wakeup sources enabled by the GPIO_PnWAKE registers.

Note: Substitute “C” or “D” for “x” in the following detailed description.

Register 7.9. GPIO_IRQxSEL
GPIO_IRQCSEL: Interrupt C Select Register
GPIO_IRQDSEL: Interrupt D Select Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	0	0	0	SEL_GPIO				

GPIO_IRQCSEL: Address: 0x4000BC14 Reset: 0xF
GPIO_IRQDSEL: Address: 0x4000BC18 Reset: 0x10

Bitname	Bitfield	Access	Description
SEL_GPIO	[4:0]	RW	Pin assigned to IRQx. 0x04: PA4 0x05: PA5 0x07: PA7 0x08: PB0 0x09: PB1 0x0A: PB2 0x0B: PB3 0x0C: PB4 0x10: PC0 0x11: PC1 0x12: PC2 0x13: PC3 0x14: PC4 0x15: PC5 0x16: PC6 0x17: PC7 0x18–0x1F: Reserved

Note: Substitute “A”, “B”, “C”, or “D” for “x” in the following detailed description.

Register 7.10. GPIO_INTCFGx
GPIO_INTCFGA: GPIO Interrupt A Configuration Register
GPIO_INTCFGB: GPIO Interrupt B Configuration Register
GPIO_INTCFGC: GPIO Interrupt C Configuration Register
GPIO_INTCFGD: GPIO Interrupt D Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	GPIO_INTFILT
Bit	7	6	5	4	3	2	1	0
Name	GPIO_INTMOD			0	0	0	0	0

GPIO_INTCFGA: Address: 0x4000A860 Reset: 0x0
GPIO_INTCFGB: Address: 0x4000A864 Reset: 0x0
GPIO_INTCFGC: Address: 0x4000A868 Reset: 0x0
GPIO_INTCFGD: Address: 0x4000A86C Reset: 0x0

Bitname	Bitfield	Access	Description
GPIO_INTFILT	[8]	RW	Set this bit to enable digital filtering on IRQx.
GPIO_INTMOD	[7:5]	RW	IRQx triggering mode. 0x0: Disabled. 0x1: Rising edge triggered. 0x2: Falling edge triggered. 0x3: Rising and falling edge triggered. 0x4: Active high level triggered. 0x5: Active low level triggered. 0x6, 0x7: Reserved.

Register 7.11. INT_GPIOFLAG: GPIO Interrupt Flag Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	INT_IRQDFLAG	INT_IRQCFLAG	0	INT_IRQAFLAG

Address: 0x4000A814 Reset: 0x0

Bitname	Bitfield	Access	Description
INT_IRQDFLAG	[3]	RW	IRQD interrupt pending. Write 1 to clear IRQD interrupt (writing 0 has no effect).
INT_IRQCFLAG	[2]	RW	IRQC interrupt pending. Write 1 to clear IRQC interrupt (writing 0 has no effect).
INT_IRQAFLAG	[0]	RW	IRQA interrupt pending. Write 1 to clear IRQA interrupt (writing 0 has no effect).

Register 7.12. GPIO_DBGCFG: GPIO Debug Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	GPIO_DEBUGDIS	GPIO_EXTREGEN	GPIO_DBGCFGRSVD	0	0	0

Address: 0x4000BC00 Reset: 0x10

Bitname	Bitfield	Access	Description
GPIO_DEBUGDIS	[5]	RW	Disable debug interface override of normal GPIO configuration. 0: Permit debug interface to be active. 1: Disable debug interface (if it is not already active).
GPIO_EXTREGEN	[4]	RW	Enable REG_EN override of PA7's normal GPIO configuration. 0: Disable override. 1: Enable override.
GPIO_DBGCFGRSVD	[3]	RW	Reserved: this bit can change during normal operation. When writing to GPIO_DBGCFG, the value of this bit must be preserved.

Register 7.13. GPIO_DBGSTAT: GPIO Debug Status Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	GPIO_BOOTMODE	0	GPIO_FORCEDBG	GPIO_SWEN

Address: 0x4000BC04 Reset: 0x0

Bitname	Bitfield	Access	Description
GPIO_BOOTMODE	[3]	R	The state of the nBOOTMODE signal sampled at the end of reset. 0: nBOOTMODE was not asserted (it read high). 1: nBOOTMODE was asserted (it read low).
GPIO_FORCEDBG	[1]	R	Status of debugger interface. 0: Debugger interface not forced active. 1: Debugger interface forced active by debugger cable.
GPIO_SWEN	[0]	R	Status of Serial Wire interface. 0: Not enabled by SWJ-DP. 1: Enabled by SWJ-DP.

8. Serial Controllers

8.1. Overview

The EM342 has one serial controller, SC1, which provides several options for full-duplex synchronous and asynchronous serial communications.

- SPI (Serial Peripheral Interface), slave only
- UART (Universal Asynchronous Receiver/Transmitter)
- Receive and transmit FIFOs and DMA channels, SPI and UART modes

Receive and transmit FIFOs allow faster data speeds using byte-at-a-time interrupts. For the highest SPI and UART speeds, dedicated receive and transmit DMA channels reduce CPU loading and extend the allowable time to service a serial controller interrupt. Polled operation is also possible using direct access to the serial data registers. Figure 8.1 shows the components of the serial controllers.

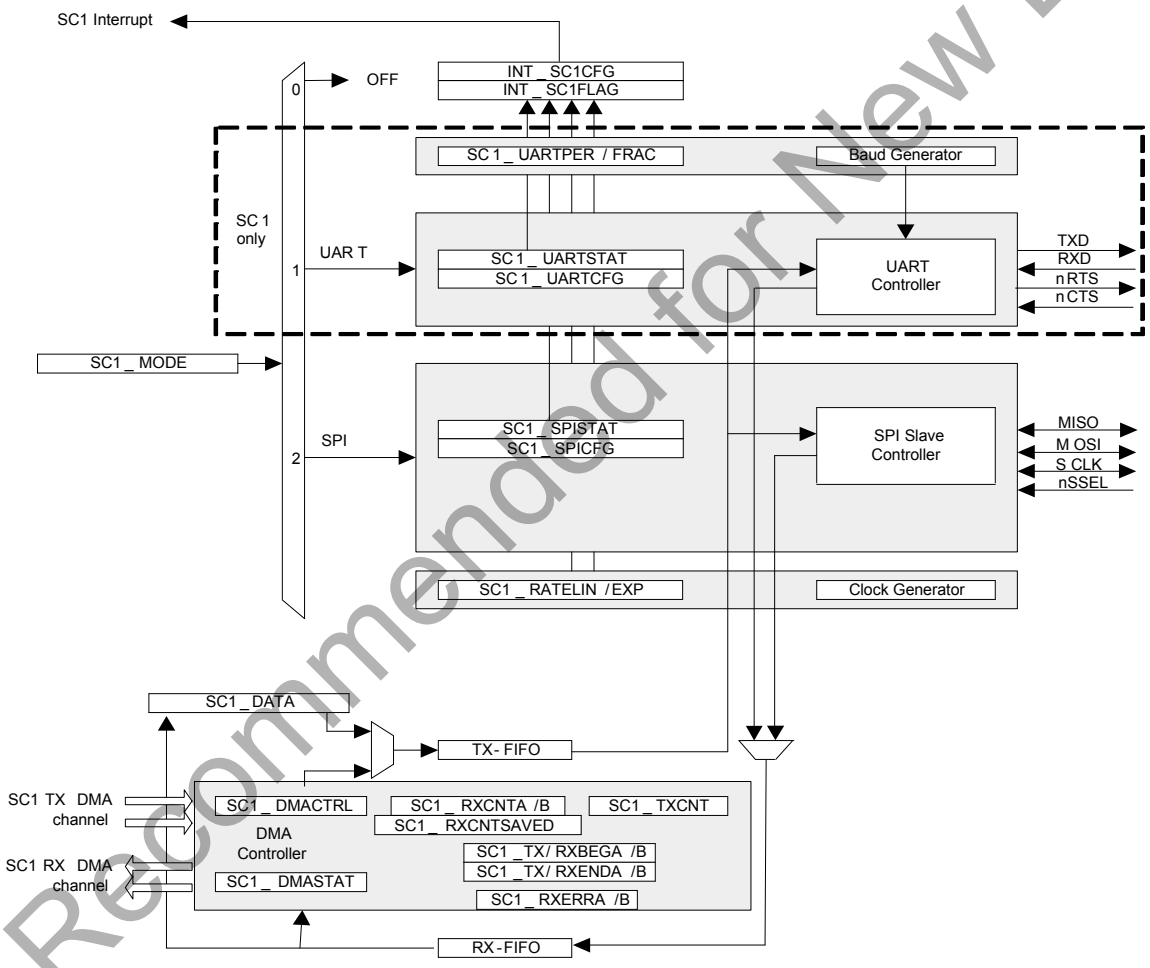


Figure 8.1. Serial Controller Block Diagram

8.2. Configuration

Before using a serial controller, configure and initialize it as follows:

1. Set up the parameters specific to the operating mode (slave for SPI, baud rate for UART, etc.).
2. Configure the GPIO pins used by the serial controller as shown in Tables 8.1. "7.2. Configuration" on page 49 shows how to configure GPIO pins.
3. If using DMA, set up the DMA and buffers. This is described fully in "8.5. DMA Channels" on page 86.
4. If using interrupts, select edge- or level-triggered interrupts with the SCx_INTMODE register, enable the desired second-level interrupt sources in the INT_SCxCFG register, and finally enable the top-level SCx interrupt in the NVIC.
5. Write the serial interface operating mode (SPI or UART) to the SCx_MODE register.

Table 8.1. SC1 GPIO Usage and Configuration

	PB1	PB2	PB3	PB4
SPI - Slave	SC1MISO Alternate Output (push-pull)	SC1MOSI Input	SC1SCLK Input	SC1nSSEL Input
UART	TXD Alternate Output (push-pull)	RXD Input	nCTS Input[1]	nRTS Alternate Output (push-pull)*

***Note:** used if RTS/CTS hardware flow control is enabled.

8.2.1. Registers

Register 8.1. SC1_MODE: Serial Mode Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	0	0	0	0	0	0	SC_MODE	

SC1_MODE: Address: 0x4000C854 Reset: 0x0

Bitname	Bitfield	Access	Description
SC_MODE	[1:0]	RW	Serial controller mode. 0: Disabled. 1: UART mode . 2: SPI mode.

Register 8.2. INT_SC1FLAG: Serial Controller 1 Interrupt Flag Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	INT_ SC1PARERR	INT_ SC1FRMERR	INT_ SCTXULDB	INT_ SCTXULDA	INT_ SCRXULDB	INT_ SCRXULDA	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	INT_ SCTXUND	INT_ SCRXOVF	INT_ SCTXIDLE	INT_ SCTXFREE	INT_SCRXVAL

INT_SC1FLAG: Address: 0x4000A808 Reset: 0x0

Bitname	Bitfield	Access	Description
INT_SC1PARERR	[14]	RW	Parity error received (UART) interrupt pending.
INT_SC1FRMERR	[13]	RW	Frame error received (UART) interrupt pending.
INT_SCTXULDB	[12]	RW	DMA transmit buffer B unloaded interrupt pending.
INT_SCTXULDA	[11]	RW	DMA transmit buffer A unloaded interrupt pending.
INT_SCRXULDB	[10]	RW	DMA receive buffer B unloaded interrupt pending.
INT_SCRXULDA	[9]	RW	DMA receive buffer A unloaded interrupt pending.
INT_SCTXUND	[4]	RW	Transmit buffer underrun interrupt pending.
INT_SCRXOVF	[3]	RW	Receive buffer overrun interrupt pending.
INT_SCTXIDLE	[2]	RW	Transmitter idle interrupt pending.
INT_SCTXFREE	[1]	RW	Transmit buffer free interrupt pending.
INT_SCRXVAL	[0]	RW	Receive buffer has data interrupt pending.

Register 8.3. INT_SC1CFG: Serial Controller 1 Interrupt Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	INT_ SC1PARERR	INT_ SC1FRMERR	INT_ SCTXULDB	INT_ SCTXULDA	INT_ SCRXULDB	INT_ SCRXULDA	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	INT_ SCTXUND	INT_ SCRXOVF	INT_ SCTXIDLE	INT_ SCTXFREE	INT_ SCRXVAL

INT_SC1CFG: Address: 0x4000A848 Reset: 0x0

Bitname	Bitfield	Access	Description
INT_SC1PARERR	[14]	RW	Parity error received (UART) interrupt enable.
INT_SC1FRMERR	[13]	RW	Frame error received (UART) interrupt enable.
INT_SCTXULDB	[12]	RW	DMA transmit buffer B unloaded interrupt enable.
INT_SCTXULDA	[11]	RW	DMA transmit buffer A unloaded interrupt enable.
INT_SCRXULDB	[10]	RW	DMA receive buffer B unloaded interrupt enable.
INT_SCRXULDA	[9]	RW	DMA receive buffer A unloaded interrupt enable.
INT_SCTXUND	[4]	RW	Transmit buffer underrun interrupt enable.
INT_SCRXOVF	[3]	RW	Receive buffer overrun interrupt enable.
INT_SCTXIDLE	[2]	RW	Transmitter idle interrupt enable.
INT_SCTXFREE	[1]	RW	Transmit buffer free interrupt enable.
INT_SCRXVAL	[0]	RW	Receive buffer has data interrupt enable.

Register 8.4. SC1_INTMODE: Serial Controller 1 Interrupt Mode Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	0	SC_TXIDLELEVEL	SC_TXFREELEVEL	SC_RXVALLEVEL

SC1_INTMODE: Address: 0x4000A854 Reset: 0x0

Bitname	Bitfield	Access	Description
SC_TXIDLELEVEL	[2]	RW	Transmitter idle interrupt mode - 0: edge triggered, 1: level triggered.
SC_TXFREELEVEL	[1]	RW	Transmit buffer free interrupt mode - 0: edge triggered, 1: level triggered.
SC_RXVALLEVEL	[0]	RW	Receive buffer has data interrupt mode - 0: edge triggered, 1: level triggered.

8.3. SPI—Slave Mode

The SC1 controller includes an SPI slave controller with these features:

- Full duplex operation
- Up to 5 Mbps data transfer rate
- Programmable clock polarity and clock phase
- Selectable data shift direction (either LSB or MSB first)
- Slave select input

8.3.1. GPIO Usage

The SPI slave controller uses four signals:

- MOSI (Master Out, Slave In) - inputs serial data from the master
- MISO (Master In, Slave Out) - outputs serial data to the master
- SCLK (Serial Clock) - clocks data transfers on MOSI and MISO
- nSSEL (Slave Select) - enables serial communication with the slave

Note: The SPI slave controller does not tri-state the MISO signal when slave select is deasserted.

The GPIO pins that can be assigned to these signals are shown in Table 8.2.

Table 8.2. SPI Slave GPIO Usage

	MOSI	MISO	SCLK	nSSEL
Direction	Input	Output	Input	Input
GPIO Configuration	Input	Alternate Output (push-pull)	Input	Input
SC1 pin	PB2	PB1	PB3	PB4

8.3.2. Set Up and Configuration

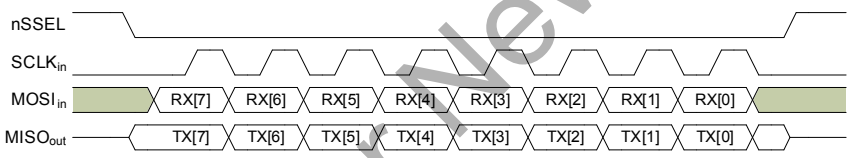
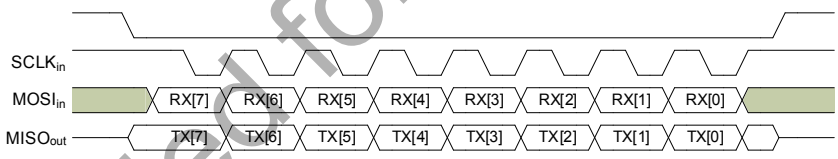
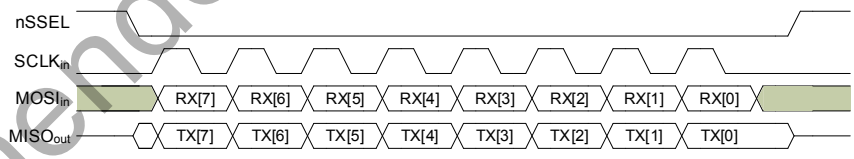
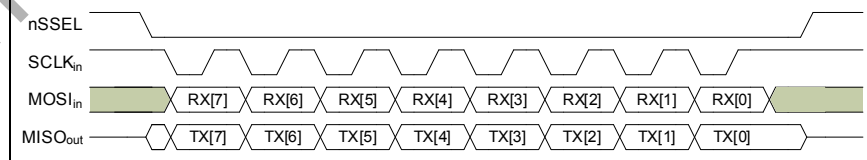
The serial controller, SC1, supports SPI slave mode. SPI slave mode is enabled by the following register settings:

- The serial controller mode register, SCx_MODE, is 2
- The SC_SPIMST bit in the SPI configuration register, SCx_SPICFG, is 0

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

The SPI slave controller supports various frame formats depending upon the clock polarity (SC_SPIPOL), clock phase (SC_SPIPHA), and direction of data (SC_SPIORD) (see Table 8.3). The SC_SPIPOL, SC_SPIPHA, and SC_SPIORD bits are defined within the SCx_SPICFG registers.

Table 8.3. SPI Slave Formats

SCx_SPICFG				Frame Format
SC_SPIxxx*				
MST	ORD	PHA	POL	
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	—	—	Same as above except LSB first instead of MSB first

***Note:** The notation "xxx" means that the corresponding column header below is inserted to form the field name.

8.3.3. Operation

When the slave select (nSSEL) signal is asserted by the master, SPI transmit data is driven to the output pin MISO, and SPI data is received from the input pin MOSI. The nSSEL pin has to be asserted to enable the transmit serializer to drive data to the output signal MISO. A falling edge on nSSEL resets the SPI slave shift registers.

Note: The SPI slave controller does not tri-state the MISO signal when slave select is deasserted.

Characters transmitted and received by the SPI slave controller are buffered in the transmit and receive FIFOs that are both four entries deep. When software writes a character to the SCx_DATA register, it is pushed onto the transmit FIFO. Similarly, when software reads from the SCx_DATA register, the character returned is pulled from the receive FIFO. If the transmit and receive DMA channels are used, the DMA channels also write to and read from the transmit and receive FIFOs.

Characters received are stored in the receive FIFO. Receiving characters sets the SC_SPIRXVAL bit in the SCx_SPISTAT register, to indicate that characters can be read from the receive FIFO. Characters received while the receive FIFO is full are dropped, and the SC_SPIRXOVF bit in the SCx_SPISTAT register is set. The receive FIFO hardware generates the INT_SCRXOVF interrupt, but the DMA register will not indicate the error condition until the receive FIFO is drained. Once the DMA marks a receive error, two conditions will clear the error indication: setting the appropriate SC_TX/RXDMARST bit in the SCx_DMACTRL register, or loading the appropriate DMA buffer after it has unloaded.

Receiving a character causes the serial transmission of a character pulled from the transmit FIFO. When the transmit FIFO is empty, a transmit underrun is detected (no data in transmit FIFO) and the INT_SCTXUND bit in the INT_SCxFLAG register is set. Because no character is available for serialization, the SPI serializer retransmits the last transmitted character or a busy token (0xFF), determined by the SC_SPIRPT bit in the SCx_SPICFG register. Refer to the register description of SCx_SPICFG for more detailed information about SC_SPIRPT.

When the transmit FIFO and the serializer are both empty, writing a character to the transmit FIFO clears the SC_SPITXIDLE bit in the SCx_SPISTAT register. This indicates that not all characters have been transmitted. If characters are written to the transmit FIFO until it is full, the SC_SPITXFREE bit in the SCx_SPISTAT register is cleared. Shifting out a transmit character to the MISO pin causes the SC_SPITXFREE bit in the SCx_SPISTAT register to get set. When the transmit FIFO empties and the last character has been shifted out, the SC_SPITXIDLE bit in the SCx_SPISTAT register is 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 at the start of every new string of transmit data defined by every time nSSEL is asserted. This byte is inserted as if this byte was placed there by software. The value of the byte of padding is always 0xFF.

8.3.4. DMA

The DMA Channels "8.5. DMA Channels" on page 86 describes how to configure and use the serial receive and transmit DMA channels.

When using the receive DMA channel and nSSEL transitions to the high (deasserted) state, the active buffer's receive DMA count register (SCx_RXCNTA/B) is saved in the SCx_RXCNTSAVED register. SCx_RXCNTSAVED is only written the first time nSSEL goes high after a buffer has been loaded. Subsequent rising edges set a status bit but are otherwise ignored. The 3-bit field SC_RXSSEL in the SCx_DMASTAT register records what, if anything, was saved to the SCx_RXCNTSAVED register, and whether or not another rising edge occurred on nSSEL.

8.3.5. Interrupts

SPI slave controller second-level interrupts are generated on the following events:

- Transmit FIFO empty and last character shifted out (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_SPITXIDLE)
- Transmit FIFO changed from full to not full (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_SPITXFREE)
- Receive FIFO changed from empty to not empty (depending on SCx_INTMODE, either the 0 to 1 transition or the high level 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 enable CPU interrupts, set desired interrupt bits in the second-level INT_SCxCFG register, and also enable the top-level SCx interrupt in the NVIC by writing the INT_SCx bit in the INT_CFGSET register.

8.4. UART—Universal Asynchronous Receiver/Transmitter

The SC1 UART is enabled by writing 1 to SC1_MODE. The UART supports the following features:

- Flexible baud rate clock (300 bps to 921.6 kbps)
- Data bits (7 or 8)
- Parity bits (none, odd, or even)
- Stop bits (1 or 2)
- False start bit and noise filtering
- Receive and transmit FIFOs
- Optional RTS/CTS flow control
- Receive and transmit DMA channels

8.4.1. GPIO Usage

The UART uses two signals to transmit and receive serial data:

- TXD (Transmitted Data) - serial data sent by the EM342
- RXD (Received Data) - serial data received by the EM342

If RTS/CTS flow control is enabled, these two signals are also used:

- nRTS (Request To Send) - indicates the EM342 is able to receive data
- nCTS (Clear To Send) - inhibits sending data from the EM342 if not asserted

The GPIO pins assigned to these signals are shown in Table 8.4.

Table 8.4. UART GPIO Usage

	TXD	RXD	nCTS[1]	nRTS*
Direction	Output	Input	Input	Output
GPIO Configuration	Alternate Output (push-pull)	Input	Input	Alternate Output (push-pull)
SC1 pin	PB1	PB2	PB3	PB4
***Note:** Only used if RTS/CTS hardware flow control is enabled.				

8.4.2. Set Up and Configuration

The UART baud rate clock is produced by a programmable baud generator starting from the 24 Hz clock:

$$\text{baud} = \frac{24\ \text{MHz}}{2N + F}$$

The integer portion of the divisor, N, is written to the SC1_UARTPER register and the fractional part, F, to the SC1_UARTFRAC register. Table 8.5 shows the values used to generate some common baud rates and their associated clock frequency error. The UART requires an internal clock that is at least eight times the baud rate clock, so the minimum allowable setting for SC1_UARTPER is 8.

Table 8.5. UART Baud Rate Divisors for Common Baud Rates

Baud Rate (bits/sec)	SC1_UARTPER	SC1_UARTFRAC	Baud Rate Error (%)
300	40000	0	0
2400	5000	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
230400	52	0	+ 0.16
460800	26	0	+ 0.16
921600	13	0	+ 0.16

The UART can miss bytes when the inter-byte gap is long or there is a baud rate mismatch between receiver and transmitter. The UART may detect a parity and/or framing error on the corrupted byte, but there will not necessarily be any error detected.

The UART is best operated in systems where the other side of the communication link also uses a crystal as its timing reference, and baud rates should be selected to minimize the baud rate mismatch to the crystal tolerance. Additionally, UART protocols should contain some form of error checking (for example CRC) at the packet level to detect, and retry in the event of errors. Since the probability of corruption is low, there would only be a small effect on UART throughput due to retries.

Errors may occur when:

$$T_{gap} \geq \frac{10^6}{\text{baud} \times \text{Ferror}}$$

Where:

T_{gap}= inter-byte gap in seconds

baud = baud rate in bps

Ferror = relative frequency error in ppm

For example, if the baud rate tolerance between receive and transmit is 200 ppm (reasonable if both sides are derived from a crystal), and the baud rate is 115200 bps, then errors will not occur until the inter-byte gap exceeds 43 ms. If the gap is exceeded then the chance of an error is essentially random, with a probability of approximately P = baud / 24e6. At 115200 bps, the probability of corruption is 0.5%.

The UART character frame format is determined by four bits in the SC1_UARTCFG register:

- SC_UART8BIT specifies the number of data bits in received and transmitted characters. If this bit is clear, characters have 7 data bits; if set, characters have 8 data bits.
- SC_UART2STP selects the number of stop bits in transmitted characters. (Only one stop bit is required in received characters.) If this bit is clear, characters are transmitted with one stop bit; if set, characters are transmitted with two stop bits.
- SC_UARTPAR controls whether or not received and transmitted characters include a parity bit. If SC_UARTPAR is clear, characters do not contain a parity bit, otherwise, characters do contain a parity bit.
- SC_UARTODD specifies whether transmitted and received parity bits contain odd or even parity. If this bit is clear, the parity bit is even, and if set, the parity bit is odd. Even parity is the exclusive-or of all of the data bits, and odd parity is the inverse of the even parity value. SC_UARTODD has no effect if SC_UARTPAR is clear.

A UART character frame contains, in sequence:

- The start bit
- The least significant data bit
- The remaining data bits
- If parity is enabled, the parity bit
- The stop bit, or bits, if 2 stop bits are selected.

Figure 8.2 shows the UART character frame format, with optional bits indicated. Depending on the options chosen for the character frame, the length of a character frame ranges from 9 to 12 bit times.

Note that asynchronous serial data may have arbitrarily long idle periods between characters. When idle, serial data (TXD or RXD) is held in the high state. Serial data transitions to the low state in the start bit at the beginning of a character frame.

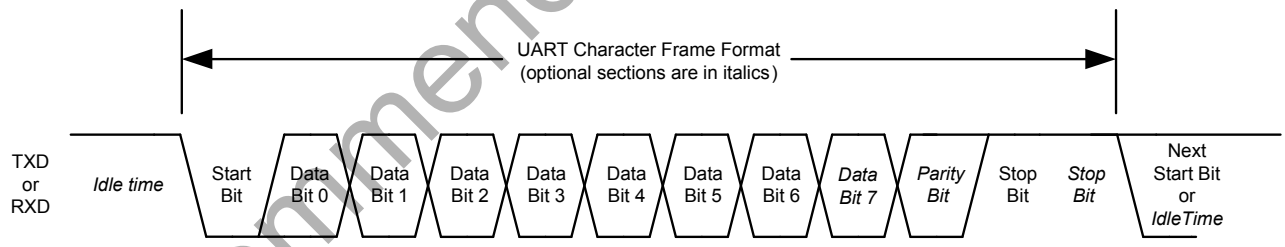


Figure 8.2. UART Character Frame Format

8.4.3. FIFOs

Characters transmitted and received by the UART are buffered in the transmit and receive FIFOs that are both 4 entries deep (see Figure 8.3). When software writes a character to the SC1_DATA register, it is pushed onto the transmit FIFO. Similarly, when software reads from the SC1_DATA register, the character returned is pulled from the receive FIFO. If the transmit and receive DMA channels are used, the DMA channels also write to and read from the transmit and receive FIFOs.

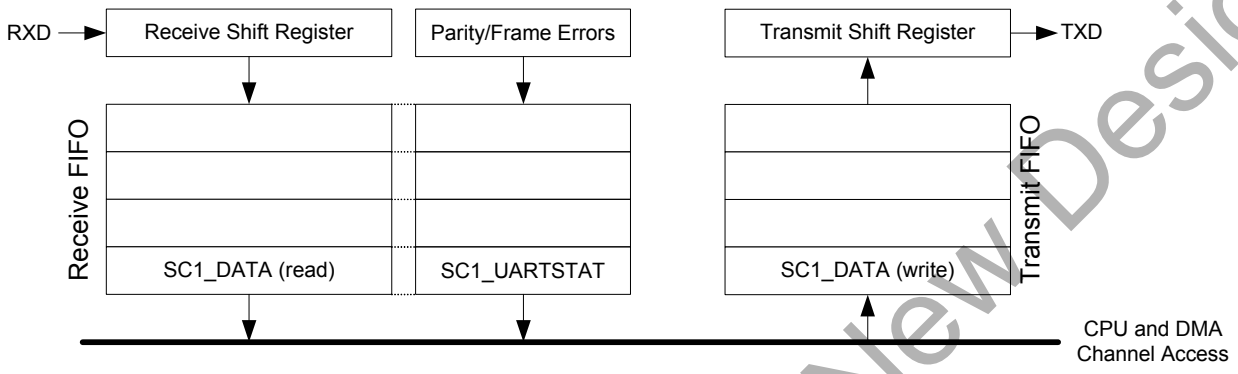


Figure 8.3. UART FIFOs

8.4.4. RTS/CTS Flow control

RTS/CTS flow control, also called hardware flow control, uses two signals (nRTS and nCTS) in addition to received and transmitted data (see Figure 8.4). Flow control is used by a data receiver to prevent buffer overflow, by signaling an external device when it is and is not allowed to transmit.

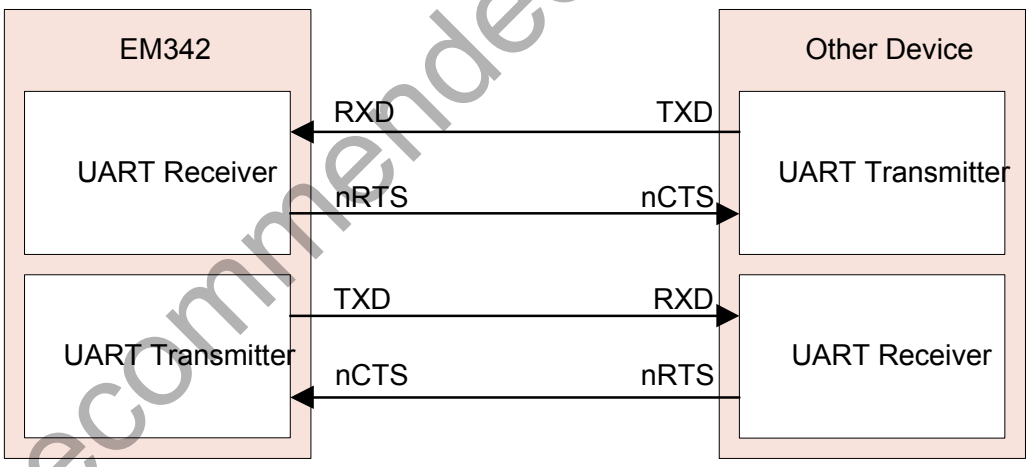


Figure 8.4. RTS/CTS Flow Control Connections

The UART RTS/CTS flow control options are selected by the SC_UARTFLOW and SC_UARTAUTO bits in the SC1_UARTCFG register (see Table 8.6). Whenever the SC_UARTFLOW bit is set, the UART will not start transmitting a character unless nCTS is low (asserted). If nCTS transitions to the high state (deasserts) while a character is being transmitted, transmission of that character continues until it is complete.

If the SC_UARTAUTO bit is set, nRTS is controlled automatically by hardware: nRTS is put into the low state (asserted) when the receive FIFO has room for at least two characters, otherwise is it in the high state (unasserted). If SC_UARTAUTO is clear, software controls the nRTS output by setting or clearing the SC_UARTRTS bit in the SC1_UARTCFG register. Software control of nRTS is useful if the external serial device cannot stop transmitting characters promptly when nRTS is set to the high state (deasserted).

Table 8.6. UART RTS/CTS Flow Control Configurations

SC1_UARTCFG			Pins Used	Operating Mode
SC_UARTxxx*				
FLOW	AUTO	RTS		
0	—	—	TXD, RXD	No RTS/CTS flow control
1	0	0/1	TXD, RXD, nCTS, nRTS	Flow control using RTS/CTS with software control of nRTS: nRTS controlled by SC_UARTRTS bit in SC1_UARTCFG register
1	1	—	TXD, RXD, nCTS, nRTS	Flow control using RTS/CTS with hardware control of nRTS: nRTS is asserted if room for at least 2 characters in receive FIFO

***Note:** The notation “xxx” means that the corresponding column header below is inserted to form the field name.

8.4.5. DMA

"8.5. DMA Channels" on page 86 describes how to configure and use the serial receive and transmit DMA channels.

The receive DMA channel has special provisions to record UART receive errors. When the DMA channel transfers a character from the receive FIFO to a buffer in memory, 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 a parity or frame error. Similarly if a receive overrun error occurs, the SC1_RXERRA/B registers mark the error offset. The receive FIFO hardware generates the INT_SCRXOVF interrupt and DMA status register indicates the error immediately, but in this case the error offset is 4 characters ahead of the actual overflow at the input to the receive FIFO. Two conditions will clear the error indication: setting the appropriate SC_RXDMARST bit in the SC1_DMACTRL register, or loading the appropriate DMA buffer after it has unloaded.

8.4.6. Interrupts

UART interrupts are generated on the following events:

- Transmit FIFO empty and last character shifted out (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_UARTTXIDLE)
- Transmit FIFO changed from full to not full (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_UARTTXFREE)
- Receive FIFO changed from empty to not empty (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_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
- Character received and lost when receive FIFO was full (receive overrun error)

To enable CPU interrupts, set the desired interrupt bits in the second-level INT_SCxCFG register, and enable the top-level SCx interrupt in the NVIC by writing the INT_SCx bit in the INT_CFGSET register.

8.4.7. Registers

Refer to " " on page 73 (in “ ”) for a description of the SCx_DATA register.

Register 8.9. SC1_UARTSTAT: UART Status Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	SC_ UARTTXIDLE	SC_ UARTPARERR	SC_ UARTFRMERR	SC_ UARTRXOVF	SC_ UARTTXFREE	SC_ UARTRXVAL	SC_ UARTCTS

SC1_UARTSTAT: Address: 0x4000C848 Reset: 0x40

Bitname	Bitfield	Access	Description
SC_UARTTXIDLE	[6]	R	This bit is set when both the transmit FIFO and the transmit serializer are empty.
SC_UARTPARERR	[5]	R	This bit is set when the byte in the data register was received with a parity error. This bit is updated when the data register is read, and is cleared if the receive FIFO is empty.
SC_UARTFRMERR	[4]	R	This bit is set when the byte in the data register was received with a frame error. This bit is updated when the data register is read, and is cleared if the receive FIFO is empty.
SC_UARTRXOVF	[3]	R	This bit is set when the receive FIFO has been overrun. This occurs if a byte is received when the receive FIFO is full. This bit is cleared by reading the data register.
SC_UARTTXFREE	[2]	R	This bit is set when the transmit FIFO has space for at least one byte.
SC_UARTRXVAL	[1]	R	This bit is set when the receive FIFO contains at least one byte.
SC_UARTCTS	[0]	R	This bit shows the logical state (not voltage level) of the nCTS input: 0: nCTS is deasserted (pin is high, 'XOFF', RS232 negative voltage); the UART is inhibited from starting to transmit a byte. 1: nCTS is asserted (pin is low, 'XON', RS232 positive voltage); the UART may transmit.

Register 8.10. SC1_UARTCFG: UART Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	SC_ UARTAUTO	SC_ UARTFLOW	SC_ UARTODD	SC_ UARTPAR	SC_ UART2STP	SC_ UART8BIT	SC_ UARTRTS

SC1_UARTCFG: Address: 0x4000C85C Reset: 0x0

Bitname	Bitfield	Access	Description
SC_UARTAUTO	[6]	RW	Set this bit to enable automatic nRTS control by hardware (SC_UARTFLOW must also be set). When automatic control is enabled, nRTS will be deasserted when the receive FIFO has space for only one more byte (inhibits transmission from the other device) and will be asserted if it has space for more than one byte (enables transmission from the other device). The SC_UARTRTS bit in this register has no effect if this bit is set.
SC_UARTFLOW	[5]	RW	Set this bit to enable using nRTS/nCTS flow control signals. Clear this bit to disable the signals. When this bit is clear, the UART transmitter will not be inhibited by nCTS.
SC_UARTODD	[4]	RW	If parity is enabled, specifies the kind of parity. 0: Even parity. 1: Odd parity.
SC_UARTPAR	[3]	RW	Specifies whether to use parity bits. 0: Don't use parity. 1: Use parity.
SC_UART2STP	[2]	RW	Number of stop bits transmitted. 0: 1 stop bit. 1: 2 stop bits.
SC_UART8BIT	[1]	RW	Number of data bits. 0: 7 data bits. 1: 8 data bits.
SC_UARTRTS	[0]	RW	nRTS is an output to control the flow of serial data sent to the EM342 from another device. This bit directly controls the output at the nRTS pin (SC_UARTFLOW must be set and SC_UARTAUTO must be cleared). When this bit is set, nRTS is asserted (pin is low, 'XON', RS232 positive voltage); the other device's transmission is enabled. When this bit is cleared, nRTS is deasserted (pin is high, 'XOFF', RS232 negative voltage), the other device's transmission is inhibited.

Register 8.11. SC1_UARTPER: UART Baud Rate Period Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	SC_UARTPER							
Bit	7	6	5	4	3	2	1	0
Name	SC_UARTPER							

SC1_UARTPER: Address: 0x4000C868 Reset: 0x0

Bitname	Bitfield	Access	Description
SC_UARTPER	[15:0]	RW	The integer part of baud rate period (N) in the equation: $$rate = \frac{24\ MHz}{((2 \times N) + F)}$$

Register 8.12. SC1_UARTFRAC: UART Baud Rate Fractional Period Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	0	0	0	SC_UARTFRAC

SC1_UARTFRAC: Address: 0x4000C86C Reset: 0x0

Bitname	Bitfield	Access	Description
SC_UARTFRAC	[0]	RW	The fractional part of the baud rate period (F) in the equation: $$rate = \frac{24\ MHz}{((2 \times N) + F)}$$

8.5. DMA Channels

The EM342 serial DMA channels enable efficient, high-speed operation of the SPI and UART controllers by reducing the load on the CPU as well as decreasing the frequency of interrupts that it must service. The transmit and receive DMA channels can transfer data between the transmit and receive FIFOs and the DMA buffers in main memory as quickly as it can be transmitted or received. Once software defines, configures, and activates the DMA, it only needs to handle an interrupt when a transmit buffer has been emptied or a receive buffer has been filled. The DMA channels each support two memory buffers, labeled A and B, and can alternate ("ping-pong") between them automatically to allow continuous communication without critical interrupt timing.

Note: DMA memory buffer terminology

- load - make a buffer available for the DMA channel to use
- pending - a buffer loaded but not yet active
- active - the buffer that will be used for the next DMA transfer
- unload - DMA channel action when it has finished with a buffer
- idle - a buffer that has not been loaded, or has been unloaded

To use a DMA channel, software should follow these steps:

1. Reset the DMA channel by setting the SC_TXDMARST (or SC_RXDMARST) bit in the SCx_DMACTRL register.
2. Set up the DMA buffers. The two DMA buffers, A and B, are defined by writing the start address to SCx_TXBEGA/B (or SCx_RXBEGA/B) and the (inclusive) end address to SCx_TXENDA/B (or SCx_RXENDA/B). Note that DMA buffers must be in RAM.
3. Configure and initialize SCx for the desired operating mode.
4. Enable second-level interrupts triggered when DMA buffers unload by setting the INT_SCTXULDA/B (or INT_SCRXULDA/B) bits in the INT_SCxFLAG register.
5. Enable top-level NVIC interrupts by setting the INT_SCx bit in the INT_CFGSET register.
6. Start the DMA by loading the DMA buffers by setting the SC_TXLODA/B (or SC_RXLODA/B) bits in the SCx_DMACTRL register.

A DMA buffer's end address, SCx_TXENDA/B (or SCx_RXENDA/B), can be written while the buffer is loaded or active. This is useful for receiving messages that contain an initial byte count, since it allows software to set the buffer end address at the last byte of the message.

As the DMA channel transfers data between the transmit or receive FIFO and a memory buffer, the DMA count register contains the byte offset from the start of the buffer to the address of the next byte that will be written or read. A transmit DMA channel has a single DMA count register (SCx_TXCNT) that applies to whichever transmit buffer is active, but a receive DMA channel has two DMA count registers (SCx_RXCNTA/B), one for each receive buffer. The DMA count register contents are preserved until the corresponding buffer, or either buffer in the case of the transmit DMA count, is loaded, or until the DMA is reset.

The receive DMA count register may be written while the corresponding buffer is loaded. If the buffer is not loaded, writing the DMA count register also loads the buffer while preserving the count value written. This feature can simplify handling UART receive errors.

The DMA channel stops using a buffer and unloads it when the following is true:

(DMA buffer start address + DMA buffer count) > DMA buffer end address

Typically a transmit buffer is unloaded after all its data has been sent, and a receive buffer is unloaded after it is filled with data, but writing to the buffer end address or buffer count registers can also cause a buffer to unload early.

Serial controller DMA channels include additional features specific to the SPI and UART operation and are described in those sections.

8.5.1. Registers

Register 8.13. SC1_DMACTRL: Serial DMA Control Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	SC_TXDMARST	SC_RXDMARST	SC_TXLODB	SC_TXLODA	SC_RXLODB	SC_RXLODA

SC1_DMACTRL: Address: 0x4000C830 Reset: 0x0

Bitname	Bitfield	Access	Description
SC_TXDMARST	[5]	W	Setting this bit resets the transmit DMA. The bit clears automatically.
SC_RXDMARST	[4]	W	Setting this bit resets the receive DMA. The bit clears automatically.
SC_TXLODB	[3]	RW	Setting this bit loads DMA transmit buffer B addresses and allows the DMA controller to start processing transmit buffer B. If both buffer A and B are loaded simultaneously, buffer A will be used first. This bit is cleared when DMA completes. Writing a zero to this bit has no effect. Reading this bit returns DMA buffer status: 0: DMA processing is complete or idle. 1: DMA processing is active or pending.
SC_TXLODA	[2]	RW	Setting this bit loads DMA transmit buffer A addresses and allows the DMA controller to start processing transmit buffer A. If both buffer A and B are loaded simultaneously, buffer A will be used first. This bit is cleared when DMA completes. Writing a zero to this bit has no effect. Reading this bit returns DMA buffer status: 0: DMA processing is complete or idle. 1: DMA processing is active or pending.
SC_RXLODB	[1]	RW	Setting this bit loads DMA receive buffer B addresses and allows the DMA controller to start processing receive buffer B. If both buffer A and B are loaded simultaneously, buffer A will be used first. This bit is cleared when DMA completes. Writing a zero to this bit has no effect. Reading this bit returns DMA buffer status: 0: DMA processing is complete or idle. 1: DMA processing is active or pending.
SC_RXLODA	[0]	RW	Setting this bit loads DMA receive buffer A addresses and allows the DMA controller to start processing receive buffer A. If both buffer A and B are loaded simultaneously, buffer A will be used first. This bit is cleared when DMA completes. Writing a zero to this bit has no effect. Reading this bit returns DMA buffer status: 0: DMA processing is complete or idle. 1: DMA processing is active or pending.

Register 8.14. SC1_DMASTAT: Serial DMA Status Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	SC_RXSSEL			SC_RXFRMB	SC_RXFRMA
Bit	7	6	5	4	3	2	1	0
Name	SC_RXPARB	SC_RXPARA	SC_RXOVFB	SC_RXOVFA	SC_TXACTB	SC_TXACTA	SC_RXACTB	SC_RXACTA

SC1_DMASTAT: Address: 0x4000C82C Reset: 0x0

Bitname	Bitfield	Access	Description
SC_RXSSEL	[12:10]	R	Status of the receive count saved in SCx_RXCNTSAVED (SPI slave mode) when nSSEL deasserts. Cleared when a receive buffer is loaded and when the receive DMA is reset. 0: No count was saved because nSSEL did not deassert. 2: Buffer A's count was saved, nSSEL deasserted once. 3: Buffer B's count was saved, nSSEL deasserted once. 6: Buffer A's count was saved, nSSEL deasserted more than once. 7: Buffer B's count was saved, nSSEL deasserted more than once. 1, 4, 5: Reserved.
SC_RXFRMB	[9]	R	This bit is set when DMA receive buffer B reads a byte with a frame error from the receive FIFO. It is cleared the next time buffer B is loaded or when the receive DMA is reset. (SC1 in UART mode only)
SC_RXFRMA	[8]	R	This bit is set when DMA receive buffer A reads a byte with a frame error from the receive FIFO. It is cleared the next time buffer A is loaded or when the receive DMA is reset. (SC1 in UART mode only)
SC_RXPARB	[7]	R	This bit is set when DMA receive buffer B reads a byte with a parity error from the receive FIFO. It is cleared the next time buffer B is loaded or when the receive DMA is reset. (SC1 in UART mode only)
SC_RXPARA	[6]	R	This bit is set when DMA receive buffer A reads a byte with a parity error from the receive FIFO. It is cleared the next time buffer A is loaded or when the receive DMA is reset. (SC1 in UART mode only)
SC_RXOVFB	[5]	R	This bit is set when DMA receive buffer B was passed an overrun error from the receive FIFO. Neither receive buffer was 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. Cleared the next time buffer B is loaded and when the receive DMA is reset.
SC_RXOVFA	[4]	R	This bit is set when DMA receive buffer A was passed an overrun error from the receive FIFO. Neither receive buffer was 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. Cleared the next time buffer A is loaded and when the receive DMA is reset.
SC_TXACTB	[3]	R	This bit is set when DMA transmit buffer B is active.
SC_TXACTA	[2]	R	This bit is set when DMA transmit buffer A is active.

SC_RXACTB	[1]	R	This bit is set when DMA receive buffer B is active.
SC_RXACTA	[0]	R	This bit is set when DMA receive buffer A is active.

Register 8.15. SC1_TXBEGA: Transmit DMA Begin Address Register A

Bit	31	30	29	28	27	26	25	24
Name	0	0	1	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	SC_TXBEGA					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SC_TXBEGA							

SC1_TXBEGA: Address: 0x4000C810 Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_TXBEGA	[13:0]	RW	DMA transmit buffer A start address.

Register 8.16. SC1_TXBEGB: Transmit DMA Begin Address Register B

Bit	31	30	29	28	27	26	25	24
Name	0	0	1	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	SC_TXBEGB					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SC_TXBEGB							

SC1_TXBEGB: Transmit DMA Begin Address Register B

Bitname	Bitfield	Access	Description
SC_TXBEGB	[13:0]	RW	DMA transmit buffer B start address.

Register 8.17. SC1_TXENDA: Transmit DMA End Address Register A

Bit	31	30	29	28	27	26	25	24
Name	0	0	1	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	SC_TXENDA					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SC_TXENDA							

SC1_TXENDA: Address: 0x4000C814 Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_TXENDA	[13:0]	RW	Address of the last byte that will be read from the DMA transmit buffer A.

Register 8.18. SC1_TXENDB: Transmit DMA End Address Register B

Bit	31	30	29	28	27	26	25	24
Name	0	0	1	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	SC_TXENDB					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SC_TXENDB							

SC1_TXENDB: Address: 0x4000C81C Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_TXENDB	[13:0]	RW	Address of the last byte that will be read from the DMA transmit buffer B.

Register 8.19. SC1_TXCNT: Transmit DMA Count Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	SC_TXCNT					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SC_TXCNT							

SC1_TXCNT: Address: 0x4000C828 Reset: 0x0

Bitname	Bitfield	Access	Description
SC_TXCNT	[13:0]	R	The offset from the start of the active DMA transmit buffer from which the next byte will be read. This register is set to zero when the buffer is loaded and when the DMA is reset.

Register 8.20. SC1_RXBEGA: Receive DMA Begin Address Register A

Bit	31	30	29	28	27	26	25	24
Name	0	0	1	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	SC_RXBEGA					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SC_RXBEGA							

SC1_RXBEGA: Address: 0x4000C800 Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_RXBEGA	[13:0]	RW	DMA receive buffer A start address.

Register 8.21. SC1_RXBEGB: Receive DMA Begin Address Register B

Bit	31	30	29	28	27	26	25	24
Name	0	0	1	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	SC_RXBEGB					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SC_RXBEGB							

SC1_RXBEGB: Address: 0x4000C808 Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_RXBEGB	[13:0]	RW	DMA receive buffer B start address.

Register 8.22. SC1_RXENDA: Receive DMA End Address Register A

Bit	31	30	29	28	27	26	25	24
Name	0	0	1	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	SC_RXENDA					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SC_RXENDA							

SC1_RXENDA: Address: 0x4000C804 Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_RXENDA	[13:0]	RW	Address of the last byte that will be written in the DMA receive buffer A.

Register 8.23. SC1_RXENDB: Receive DMA End Address Register B

Bit	31	30	29	28	27	26	25	24
Name	0	0	1	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	SC_RXENDB					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SC_RXENDB							

SC1_RXENDB: Address: 0x4000C80C Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_RXENDB	[13:0]	RW	Address of the last byte that will be written in the DMA receive buffer B.

Register 8.24. SC1_RXCNTA: Receive DMA Count Register A

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	SC_RXCNTA					
Bit	7	6	5	4	3	2	1	0
Name	SC_RXCNTA							

SC1_RXCNTA: Address: 0x4000C820 Reset: 0x0

Bitname	Bitfield	Access	Description
SC_RXCNTA	[13:0]	RW	The offset from the start of DMA receive buffer A at which the next byte will be written. This register is set to zero when the buffer is loaded and when the DMA is reset. If this register is written when the buffer is not loaded, the buffer is loaded.

Register 8.25. SC1_RXCNTB: Receive DMA Count Register B

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	SC_RXCNTB					
Bit	7	6	5	4	3	2	1	0
Name	SC_RXCNTB							

SC1_RXCNTB: Address: 0x4000C824 Reset: 0x0

Bitname	Bitfield	Access	Description
SC_RXCNTB	[13:0]	RW	The offset from the start of DMA receive buffer B at which the next byte will be written. This register is set to zero when the buffer is loaded and when the DMA is reset. If this register is written when the buffer is not loaded, the buffer is loaded.

Register 8.26. SC1_RXCNTSAVED: Saved Receive DMA Count Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	0	0	SC_RXCNTSAVED					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	SC_RXCNTSAVED							

SC1_RXCNTSAVED: Address: 0x4000C870 Reset: 0x0

Bitname	Bitfield	Access	Description
SC_RXCNTSAVED	[13:0]	R	Receive DMA count saved in SPI slave mode when nSSEL deasserts. The count is only saved the first time nSSEL deasserts.

Register 8.27. SC1_RXERRA: DMA First Receive Error Register A

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	SC_RXERRA					
Bit	7	6	5	4	3	2	1	0
Name	SC_RXERRA							

SC1_RXERRA: DMA First Receive Error Register A

Bitname	Bitfield	Access	Description
SC_RXERRA	[13:0]	R	The offset from the start of DMA receive buffer A of the first byte received with a parity, frame, or overflow error. Note that an overflow error occurs at the input to the receive FIFO, so this offset is 4 bytes before the overflow position. If there is no error, it reads zero. This register will not be updated by subsequent errors until the buffer unloads and is reloaded, or the receive DMA is reset.

Register 8.28. SC1_RXERRB: DMA First Receive Error Register B

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	SC_RXERRB					
Bit	7	6	5	4	3	2	1	0
Name	SC_RXERRB							

SC1_RXERRB: Address: 0x4000C838 Reset: 0x0

Bitname	Bitfield	Access	Description
SC_RXERRB	[13:0]	R	The offset from the start of DMA receive buffer B of the first byte received with a parity, frame, or overflow error. Note that an overflow error occurs at the input to the receive FIFO, so this offset is 4 bytes before the overflow position. If there is no error, it reads zero. This register will not be updated by subsequent errors until the buffer unloads and is reloaded, or the receive DMA is reset.

9. Interrupt System

The EM342's interrupt system is composed of two parts: a standard ARM® Cortex™-M3 Nested Vectored Interrupt Controller (NVIC) that provides top-level interrupts, and a proprietary Event Manager (EM) that provides second-level interrupts. The NVIC and EM provide a simple hierarchy. All second-level interrupts from the EM feed into top-level interrupts in the NVIC. This two-level hierarchy allows for both fine granular control of interrupt sources and coarse granular control over entire peripherals, while allowing peripherals to have their own interrupt vector.

"9.1. Nested Vectored Interrupt Controller (NVIC)" on page 100 provides a description of the NVIC and an overview of the exception table (ARM nomenclature refers to interrupts as exceptions). "9.2. Event Manager" on page 102 provides a more detailed description of the Event Manager including a table of all top-level peripheral interrupts and their second-level interrupt sources.

In practice, top-level peripheral interrupts are only used to enable or disable interrupts for an entire peripheral. Second-level interrupts originate from hardware sources, and therefore are the main focus of applications using interrupts.

9.1. Nested Vectored Interrupt Controller (NVIC)

The ARM® Cortex™-M3 Nested Vectored Interrupt Controller (NVIC) facilitates low-latency exception and interrupt handling. The NVIC and the processor core interface are closely coupled, which enables low-latency interrupt processing and efficient processing of late-arriving interrupts. The NVIC also maintains knowledge of the stacked (nested) interrupts to enable tail-chaining of interrupts.

The ARM® Cortex™-M3 NVIC contains 10 standard interrupts that are related to chip and CPU operation and management. In addition to the 10 standard interrupts, it contains 17 individually vectored peripheral interrupts specific to the EM342.

The NVIC defines a list of exceptions. These exceptions include not only traditional peripheral interrupts, but also more specialized events such as faults and CPU reset. In the ARM® Cortex™-M3 NVIC, a CPU reset event is considered an exception of the highest priority, and the stack pointer is loaded from the first position in the NVIC exception table. The NVIC exception table defines all exceptions and their position, including peripheral interrupts. The position of each exception is important since it directly translates to the location of a 32-bit interrupt vector for each interrupt, and defines the hardware priority of exceptions. Each exception in the table is a 32-bit address that is loaded into the program counter when that exception occurs. Table 9.1 lists the entire exception table. Exceptions 0 (stack pointer) through 15 (SysTick) are part of the standard ARM® Cortex™-M3 NVIC, while exceptions 18 (Management) through 32 (Debug) are the peripheral interrupts specific to the EM342 peripherals. The peripheral interrupts are listed in greater detail in Table 9.2.

Table 9.1. NVIC Exception Table

Exception	Position	Description
—	0	Stack top is loaded from first entry of vector table on reset.
Reset	1	Invoked on power up and warm reset. On first instruction, drops to lowest priority (Thread mode). Asynchronous.
NMI	2	Cannot be stopped or preempted by any exception but reset. Asynchronous.
Hard Fault	3	All classes of fault, when the fault cannot activate because of priority or the Configurable Fault handler has been disabled. Synchronous.
Memory Fault	4	MPU mismatch, including access violation and no match. Synchronous.
Bus Fault	5	Pre-fetch, memory access, and other address/memory-related faults. Synchronous when precise and asynchronous when imprecise.
Usage Fault	6	Usage fault, such as “undefined instruction executed” or “illegal state transition attempt”. Synchronous.
—	7–10	Reserved.
SVCall	11	System service call with SVC instruction. Synchronous.
Debug Monitor	12	Debug monitor, when not halting. Synchronous, but only active when enabled. It does not activate if lower priority than the current activation.
—	13	Reserved.
PendSV	14	Pendable request for system service. Asynchronous and only pended by software.
SysTick	15	System tick timer has fired. Asynchronous.
—	16	Reserved.
—	17	Reserved.
Management	18	Management peripheral interrupt.
Baseband	19	Baseband peripheral interrupt.
Sleep Timer	20	Sleep Timer peripheral interrupt.
Serial Controller 1	21	Serial Controller 1 peripheral interrupt.
—	22	Reserved.
Security	23	Security peripheral interrupt.
MAC Timer	24	MAC Timer peripheral interrupt.
MAC Transmit	25	MAC Transmit peripheral interrupt.
MAC Receive	26	MAC Receive peripheral interrupt.
—	27	Reserved.
IRQA	28	IRQA peripheral interrupt.
—	29	Reserved
IRQC	30	IRQC peripheral interrupt.
IRQD	31	IRQD peripheral interrupt.
Debug	32	Debug peripheral interrupt.

The NVIC also contains a software-configurable interrupt prioritization mechanism. The Reset, NMI, and Hard Fault exceptions, in that order, are always the highest priority, and are not software-configurable. All other exceptions can be assigned a 5-bit priority number, with low values representing higher priority. If any exceptions have the same software-configurable priority, then the NVIC uses the hardware-defined priority. The hardware-defined priority number is the same as the position of the exception in the exception table. For example, if IRQA and IRQC both fire at the same time and have the same software-defined priority, the NVIC handles IRQA, with priority number 28, first because it has a higher hardware priority than IRQC with priority number 30.

The top-level interrupts are controlled through five ARM® Cortex™-M3 NVIC registers: INT_CFGSET, INT_CFGCLR, INT_PENDSET, INT_PENDCLR, and INT_ACTIVE. Writing 0 into any bit in any of these five register is ineffective.

- INT_CFGSET—Writing 1 to a bit in INT_CFGSET enables that top-level interrupt.
- INT_CFGCLR—Writing 1 to a bit in INT_CFGCLR disables that top-level interrupt.
- INT_PENDSET—Writing 1 to a bit in INT_PENDSET triggers that top-level interrupt.
- INT_PENDCLR—Writing 1 to a bit in INT_PENDCLR clears that top-level interrupt.
- INT_ACTIVE cannot be written to and is used for indicating which interrupts are currently active.

INT_PENDSET and INT_PENDCLR set and clear a simple latch; INT_CFGSET and INT_CFGCLR set and clear a mask on the output of the latch. Interrupts may be pended and cleared at any time, but any pended interrupt will not be taken unless the corresponding mask (INT_CFGSET) is set, which allows that interrupt to propagate. If an INT_CFGSET bit is set and the corresponding INT_PENDSET bit is set, then the interrupt will propagate and be taken. If INT_CFGSET is set after INT_PENDSET is set, then the interrupt will also propagate and be taken. Interrupt flags (signals) from the top-level interrupts are level-sensitive.

The second-level interrupt registers, which provide control of the second-level Event Manager peripheral interrupts, are described in “9.2. Event Manager” .

For further information on the NVIC and ARM® Cortex™-M3 exceptions, refer to the ARM® Cortex™-M3 Technical Reference Manual and the ARM ARMv7-M Architecture Reference Manual.

9.2. Event Manager

While the standard ARM® Cortex™-M3 Nested Vectored Interrupt Controller provides top-level interrupts into the CPU, the proprietary Event Manager provides second-level interrupts. The Event Manager takes a large variety of hardware interrupt sources from the peripherals and merges them into a smaller group of interrupts in the NVIC. Effectively, all second-level interrupts from a peripheral are “ORd” together into a single interrupt in the NVIC. In addition, the Event Manager provides missed indicators for the top-level peripheral interrupts with the register INT_MISS.

The description of each peripheral's interrupt configuration and flag registers can be found in the chapters of this datasheet describing each peripheral. Figure 9.1 shows the Peripheral Interrupts Block Diagram.

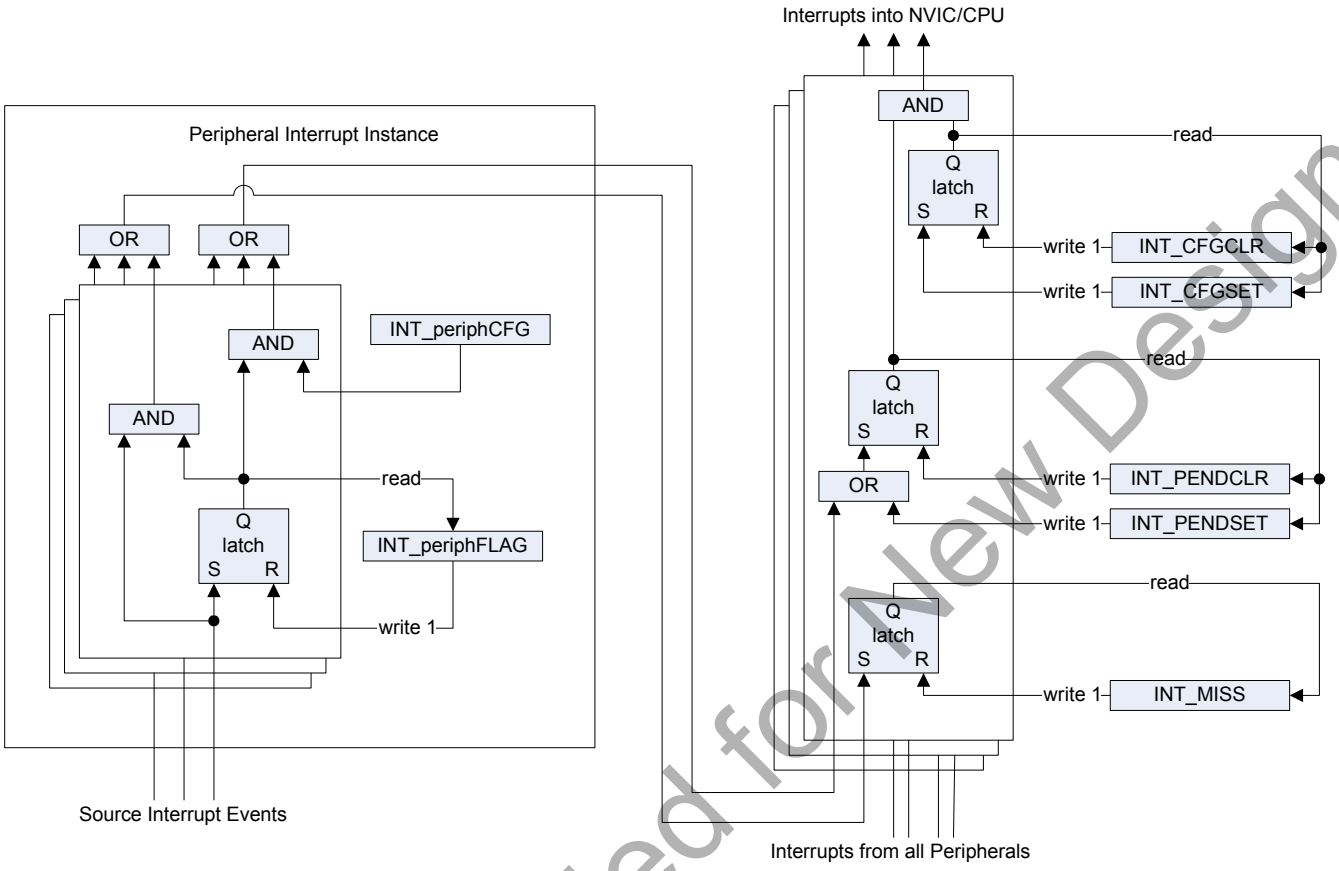


Figure 9.1. Peripheral Interrupts Block Diagram

Given a peripheral, “periph”, the Event Manager registers (INT_periphCFG and INT_periphFLAG) follow the form:

- INT_periphCFG enables and disables second-level interrupts. Writing 1 to a bit in the INT_periphCFG register enables the second-level interrupt. Writing 0 to a bit in the INT_periphCFG register disables it. The INT_periphCFG register behaves like a mask, and is responsible for allowing the INT_periphFLAG bits to propagate into the top-level NVIC interrupts.
- INT_periphFLAG indicates second-level interrupts that have occurred. Writing 1 to a bit in a INT_periphFLAG register clears the second-level interrupt. Writing 0 to any bit in the INT_periphFLAG register is ineffective. The INT_periphFLAG register is always active and may be set or cleared at any time, meaning if any second-level interrupt occurs, then the corresponding bit in the INT_periphFLAG register is set regardless of the state of INT_periphCFG.

If a bit in the INT_periphCFG register is set after the corresponding bit in the INT_periphFLAG register is set then the second-level interrupt propagates into the top-level interrupts. The interrupt flags (signals) from the second-level interrupts into the top-level interrupts are level-sensitive. If a top-level NVIC interrupt is driven by a second-level EM interrupt, then the top-level NVIC interrupt cannot be cleared until all second-level EM interrupts are cleared.

The INT_periphFLAG register bits are designed to remain set if the second-level interrupt event re-occurs at the same moment as the INT_periphFLAG register bit is being cleared. This ensures the re-occurring second-level interrupt event is not missed.

If another enabled second-level interrupt event of the same type occurs before the first interrupt event is cleared, the second interrupt event is 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_periphFLAG register-by writing a 1 into the corresponding bit to be cleared.

Table 9.2 provides a map of all peripheral interrupts. This map lists the top-level NVIC Interrupt bits and, if there is one, the corresponding second-level EM Interrupt register bits that feed the top-level interrupts.

Table 9.2. NVIC and EM Peripheral Interrupt Map

NVIC Interrupt (Top-Level)		EM Interrupt (Second-Level)	
16	INT_DEBUG		
15	INT_IRQD		
14	INT_IRQC		
13	Reserved		
12	INT_IRQA		
11	Reserved		
10	INT_MACRX		
9	INT_MACTX		
8	INT_MACTMR		
7	INT_SEC		
6	Reserved		
5	INT_SC1	INT_SC1FLAG register	
		14	INT_SC1PARERR
		13	INT_SC1FRMERR
		12	INT_SCTXULDB
		11	INT_SCTXULDA
		10	INT_SCRXULDB
		9	INT_SCRXULDA
		8	INT_SCNAK
		7	INT_SCCDMFIN
		6	INT_SCTXFIN
		5	INT_SCRXFIN
		4	INT_SCTXUND
		3	INT_SCRXOVF
		2	INT_SCTXIDLE
		1	INT_SCTXFREE
		0	INT_SCRXVAL

NVIC Interrupt (Top-Level)	EM Interrupt (Second-Level)
4	INT_SLEEPTMR
3	INT_BB
2	INT_MGMT
1	Reserved.
0	Reserved

9.3. Non-Maskable Interrupt (NMI)

The non-maskable interrupt (NMI) is a special case. Despite being one of the 10 standard ARM® CortexTM-M3 NVIC interrupts, it is sourced from the Event Manager like a peripheral interrupt. The NMI has two second-level sources; failure of the 24 MHz crystal and watchdog low water mark.

1. Failure of the 24 MHz crystal: If the EM342's main clock, SYSCLK, is operating from the 24 MHz crystal and the crystal fails, the EM342 detects the failure and automatically switches to the internal 12 MHz RC clock. When this failure detection and switch has occurred, the EM342 triggers the CLK24M_FAIL second-

level interrupt, which then triggers the NMI.

2. Watchdog low water mark: If the EM342's watchdog is active and the watchdog counter has not been reset for nominally 1.792 s, the watchdog triggers the WATCHDOG_INT second-level interrupt, which then triggers the NMI.

9.4. Faults

Four of the exceptions in the NVIC are faults: Hard Fault, Memory Fault, Bus Fault, and Usage Fault. Of these, three (Hard Fault, Memory Fault, and Usage Fault) are standard ARM® CortexTM-M3 exceptions.

The Bus Fault, though, is derived from EM342-specific sources. The Bus Fault sources are recorded in the SCS_AFSR register. Note that it is possible for one access to set multiple SCS_AFSR bits. Also note that MPU configurations could prevent most of these bus fault accesses from occurring, with the advantage that illegal writes are made precise faults. The four bus faults are:

- WRONGSIZE—Generated by an 8-bit or 16-bit read or write of an APB peripheral register. This fault can also result from an unaligned 32-bit access.
- PROTECTED—Generated by a user mode (unprivileged) write to a system APB or AHB peripheral or protected RAM (see "5.2.2.3. RAM Memory Protection" on page 31).
- RESERVED—Generated by a read or write to an address within an APB peripheral's 4 kB block range, but the address is above the last physical register in that block range. Also generated by a read or write to an address above the top of RAM or flash.
- MISSED—Generated by a second SCS_AFSR fault. In practice, this bit is not seen since a second fault also generates a hard fault, and the hard fault preempts the bus fault.

9.5. Registers

Register 9.1. INT_CFGSET: Top-Level Set Interrupts Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	INT_DEBUG
Bit	15	14	13	12	11	10	9	8
Name	INT_IRQD	INT_IRQC	Reserved	INT_IRQA	Reserved	INT_MACRX	INT_MACTX	INT_MACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_SEC	Reserved	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	Reserved	Reserved

Address: 0xE000E100; Reset: 0x0

Bit Name	Bit Field	Access	Description
INT_DEBUG	[16]	RW	Write 1 to enable debug interrupt. (Writing 0 has no effect.)
INT_IRQD	[15]	RW	Write 1 to enable IRQD interrupt. (Writing 0 has no effect.)
INT_IRQC	[14]	RW	Write 1 to enable IRQC interrupt. (Writing 0 has no effect.)
INT_IRQA	[12]	RW	Write 1 to enable IRQA interrupt. (Writing 0 has no effect.)
INT_MACRX	[10]	RW	Write 1 to enable MAC receive interrupt. (Writing 0 has no effect.)
INT_MACTX	[9]	RW	Write 1 to enable MAC transmit interrupt. (Writing 0 has no effect.)
INT_MACTMR	[8]	RW	Write 1 to enable MAC timer interrupt. (Writing 0 has no effect.)
INT_SEC	[7]	RW	Write 1 to enable security interrupt. (Writing 0 has no effect.)
INT_SC1	[5]	RW	Write 1 to enable serial controller 1 interrupt. (Writing 0 has no effect.)
INT_SLEEPTMR	[4]	RW	Write 1 to enable sleep timer interrupt. (Writing 0 has no effect.)
INT_BB	[3]	RW	Write 1 to enable baseband interrupt. (Writing 0 has no effect.)
INT_MGMT	[2]	RW	Write 1 to enable management interrupt. (Writing 0 has no effect.)

Register 9.2. INT_CFGCLR: Top-Level Clear Interrupts Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	INT_DEBUG
Bit	15	14	13	12	11	10	9	8
Name	INT_IRQD	INT_IRQC	Reserved	INT_IRQA	Reserved	INT_MACRX	INT_MACTX	INT_MACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_SEC	Reserved	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	Reserved	Reserved

Address: 0xE000E180 Reset: 0x0

Bitname	Bitfield	Access	Description
INT_DEBUG	[16]	RW	Write 1 to disable debug interrupt. (Writing 0 has no effect.)
INT_IRQD	[15]	RW	Write 1 to disable IRQD interrupt. (Writing 0 has no effect.)
INT_IRQC	[14]	RW	Write 1 to disable IRQC interrupt. (Writing 0 has no effect.)
INT_IRQA	[12]	RW	Write 1 to disable IRQA interrupt. (Writing 0 has no effect.)
INT_MACRX	[10]	RW	Write 1 to disable MAC receive interrupt. (Writing 0 has no effect.)
INT_MACTX	[9]	RW	Write 1 to disable MAC transmit interrupt. (Writing 0 has no effect.)
INT_MACTMR	[8]	RW	Write 1 to disable MAC timer interrupt. (Writing 0 has no effect.)
INT_SEC	[7]	RW	Write 1 to disable security interrupt. (Writing 0 has no effect.)
INT_SC1	[5]	RW	Write 1 to disable serial controller 1 interrupt. (Writing 0 has no effect.)
INT_SLEEPTMR	[4]	RW	Write 1 to disable sleep timer interrupt. (Writing 0 has no effect.)
INT_BB	[3]	RW	Write 1 to disable baseband interrupt. (Writing 0 has no effect.)
INT_MGMT	[2]	RW	Write 1 to disable management interrupt. (Writing 0 has no effect.)

Register 9.3. INT_PENDSET: Top-Level Set Interrupts Pending Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	INT_DEBUG
Bit	15	14	13	12	11	10	9	8
Name	INT_IRQD	INT_IRQC	Reserved	INT_IRQA	Reserved	INT_MACRX	INT_MACTX	INT_MACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_SEC	Reserved	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	Reserved	Reserved

Address: 0xE000E200 Reset: 0x0

Bitname	Bitfield	Access	Description
INT_DEBUG	[16]	RW	Write 1 to pend debug interrupt. (Writing 0 has no effect.)
INT_IRQD	[15]	RW	Write 1 to pend IRQD interrupt. (Writing 0 has no effect.)
INT_IRQC	[14]	RW	Write 1 to pend IRQC interrupt. (Writing 0 has no effect.).
INT_IRQA	[12]	RW	Write 1 to pend IRQA interrupt. (Writing 0 has no effect.)
INT_MACRX	[10]	RW	Write 1 to pend MAC receive interrupt. (Writing 0 has no effect.)
INT_MACTX	[9]	RW	Write 1 to pend MAC transmit interrupt. (Writing 0 has no effect.)
INT_MACTMR	[8]	RW	Write 1 to pend MAC timer interrupt. (Writing 0 has no effect.)
INT_SEC	[7]	RW	Write 1 to pend security interrupt. (Writing 0 has no effect.)
INT_SC1	[5]	RW	Write 1 to pend serial controller 1 interrupt. (Writing 0 has no effect.)
INT_SLEEPTMR	[4]	RW	Write 1 to pend sleep timer interrupt. (Writing 0 has no effect.)
INT_BB	[3]	RW	Write 1 to pend baseband interrupt. (Writing 0 has no effect.)
INT_MGMT	[2]	RW	Write 1 to pend management interrupt. (Writing 0 has no effect.)

Register 9.4. INT_PENDCLR: Top-Level Clear Interrupts Pending Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	INT_DEBUG
Bit	15	14	13	12	11	10	9	8
Name	INT_IRQD	INT_IRQC	Reserved	INT_IRQA	Reserved	INT_MACRX	INT_MACTX	INT_MACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_SEC	Reserved	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	Reserved	Reserved

Address: 0xE000E280 Reset: 0x0

Bitname	Bitfield	Access	Description
INT_DEBUG	[16]	RW	Write 1 to unpend debug interrupt. (Writing 0 has no effect.)
INT_IRQD	[15]	RW	Write 1 to unpend IRQD interrupt. (Writing 0 has no effect.)
INT_IRQC	[14]	RW	Write 1 to unpend IRQC interrupt. (Writing 0 has no effect.)
INT_IRQA	[12]	RW	Write 1 to unpend IRQA interrupt. (Writing 0 has no effect.)
INT_MACRX	[10]	RW	Write 1 to unpend MAC receive interrupt. (Writing 0 has no effect.)
INT_MACTX	[9]	RW	Write 1 to unpend MAC transmit interrupt. (Writing 0 has no effect.)
INT_MACTMR	[8]	RW	Write 1 to unpend MAC timer interrupt. (Writing 0 has no effect.)
INT_SEC	[7]	RW	Write 1 to unpend security interrupt. (Writing 0 has no effect.)
INT_SC1	[5]	RW	Write 1 to unpend serial controller 1 interrupt. (Writing 0 has no effect.)
INT_SLEEPTMR	[4]	RW	Write 1 to unpend sleep timer interrupt. (Writing 0 has no effect.)
INT_BB	[3]	RW	Write 1 to unpend baseband interrupt. (Writing 0 has no effect.)
INT_MGMT	[2]	RW	Write 1 to unpend management interrupt. (Writing 0 has no effect.)

Register 9.5. INT_ACTIVE: Top-Level Active Interrupts Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	INT_DEBUG
Bit	15	14	13	12	11	10	9	8
Name	INT_IRQD	INT_IRQC	Reserved	INT_IRQA	Reserved	INT_MACRX	INT_MACTX	INT_MACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_SEC	Reserved	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	Reserved	Reserved

Address: 0xE000E300 Reset: 0x0

Bitname	Bitfield	Access	Description
INT_DEBUG	[16]	R	Debug interrupt active.
INT_IRQD	[15]	R	IRQD interrupt active.
INT_IRQC	[14]	R	IRQC interrupt active.
INT_IRQA	[12]	R	IRQA interrupt active.
INT_MACRX	[10]	R	MAC receive interrupt active.
INT_MACTX	[9]	R	MAC transmit interrupt active.
INT_MACTMR	[8]	R	MAC timer interrupt active.
INT_SEC	[7]	R	Security interrupt active.
INT_SC1	[5]	R	Serial controller 1 interrupt active.
INT_SLEEPTMR	[4]	R	Sleep timer interrupt active.
INT_BB	[3]	R	Baseband interrupt active.
INT_MGMT	[2]	R	Management interrupt active.

Register 9.6. INT_MISS: Top-Level Missed Interrupts Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	INT_ MISSIRQD	INT_ MISSIRQC	Reserved	INT_ MISSIRQA	Reserved	INT_ MISSMACRX	INT_ MISSMACTX	INT_ MISSMACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_ MISSSEC	Reserved	INT_ MISSSC1	INT_ MISSSLEEP	INT_ MISSBB	INT_ MISSMGMT	0	0

Address: 0x4000A820 Reset: 0x0

Bitname	Bitfield	Access	Description
INT_MISSIRQD	[15]	RW	IRQD interrupt missed.
INT_MISSIRQC	[14]	RW	IRQC interrupt missed.
INT_MISSIRQA	[12]	RW	IRQA interrupt missed.
INT_MISSMACRX	[10]	RW	MAC receive interrupt missed.
INT_MISSMACTX	[9]	RW	MAC transmit interrupt missed.
INT_MISSMACTMR	[8]	RW	MAC Timer interrupt missed.
INT_MISSSEC	[7]	RW	Security interrupt missed.
INT_MISSSC1	[5]	RW	Serial controller 1 interrupt missed.
INT_MISSSLEEP	[4]	RW	Sleep timer interrupt missed.
INT_MISSBB	[3]	RW	Baseband interrupt missed.
INT_MISSMGMT	[2]	RW	Management interrupt missed.

Register 9.7. SCS_AFSR: Auxiliary Fault Status Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	WRONGSIZE	PROTECTED	RESERVED	MISSED

Address: 0xE000ED3C Reset: 0x0

Bitname	Bitfield	Access	Description
WRONGSIZE	[3]	RW	A bus fault resulted from an 8-bit or 16-bit read or write of an APB peripheral register. This fault can also result from an unaligned 32-bit access.
PROTECTED	[2]	RW	A bus fault resulted from a user mode (unprivileged) write to a system APB or AHB peripheral or protected RAM.
RESERVED	[1]	RW	A bus fault resulted from a read or write to an address within an APB peripheral's 4 kB block range, but above the last physical register in that block. Can also result from a read or write to an address above the top of RAM or flash.
MISSED	[0]	RW	A bus fault occurred when a bit was already set in this register.

10. Trace Port Interface Unit (TPIU)

The EM342 integrates the standard ARM® Trace Port Interface Unit (TPIU). The TPIU receives a data stream from the on-chip trace data generated by the standard ARM® Instrument Trace Macrocell (ITM), buffers the data in a FIFO, formats the data, and serializes the data to be sent off-chip through alternate functions of the GPIO. Since the primary function of the TPIU is to provide a bridge between on-chip ARM system debug components and external GPIO, the TPIU itself does not generate data. Figure 10.1 illustrates the three primary components of the TPIU.

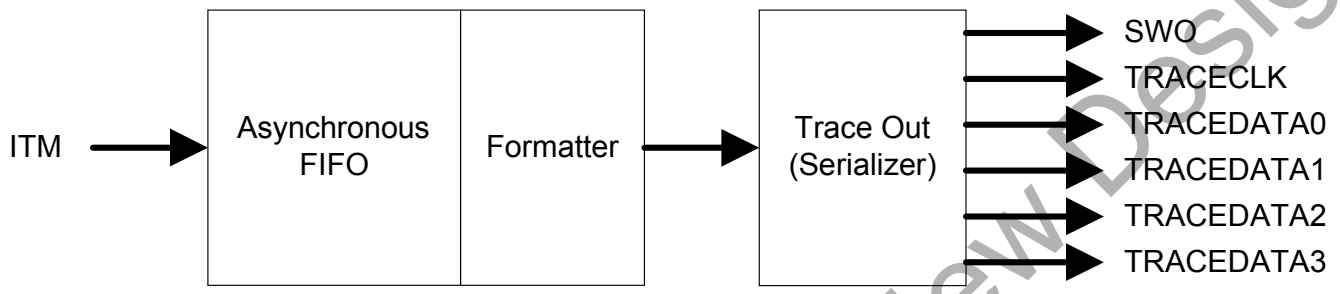


Figure 10.1. TPIU Block Diagram

The TPIU is composed of:

- Asynchronous FIFO: The asynchronous FIFO receives a data stream generated by the ITM and enables the trace data to be sent off-chip at a speed that is not dependent on the speed of the data source.
- Formatter: The formatter inserts source ID signals into the data packet stream so that trace data can be re-associated with its trace source. Since the EM342 has only one trace source, the ITM, it is not necessary to use the formatter, and, therefore, the formatter only adds overhead into the data stream. Since certain modes of the TPIU automatically enable the formatter, these modes should be avoided whenever possible.
- Trace Out: The trace out block serializes the data and sends it off-chip by the proper alternate output GPIO functions.

The six pins available to the TPIU are:

- SWO
- TRACECLK
- TRACEDATA0
- TRACEDATA1
- TRACEDATA2
- TRACEDATA3

Since these pins are alternate outputs of GPIO, refer to "17. Pin Definitions" on page 121 and "7. GPIO (General Purpose Input/Output)" on page 48 for complete pin descriptions and configurations.

Notes:

1. The SWO alternate output is mirrored on GPIO PC1 and PC2.
2. GPIO PC1 shares both the SWO and TRACEDATA0 alternate outputs. This is possible because SWO and TRACEDATA0 are mutually exclusive, and only one may be selected at a time in the trace-out block.

The Ember software utilizes the TPIU to efficiently output debug data. Altering the TPIU configuration may conflict with Ember debug output.

For further information on the TPIU, contact Silicon Labs support for the ARM® Cortex™-M3 Technical Reference Manual, the ARM® CoreSight™ Components Technical Reference Manual, the ARM® v7-M Architecture Reference Manual, and the ARM® v7-M Architecture Application Level Reference Manual.

11. Instrumentation Trace Macrocell (ITM)

The EM342 integrates the standard ARM® Instrumentation Trace Macrocell (ITM). The ITM is an application-driven trace source that supports printf style debugging to trace software events and emits diagnostic system information from the ARM® Data Watchpoint and Trace (DWT). Software using the ITM generates Software Instrumentation Trace (SWIT). In addition, the ITM provides coarse-grained timestamp functionality. The ITM emits trace information as packets, and these packets are sent to the Trace Port Interface Unit (TPIU). Three sources can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which the packets are output. The three sources, in decreasing order of priority, are:

- Software trace. Software can write directly to ITM stimulus registers, emitting packets.
- Hardware trace. The DWT generates packets that the ITM emits.
- Time stamping. Timestamps are emitted relative to packets, and the ITM contains a 21-bit counter to generate the timestamps.

The Ember software utilizes the ITM for efficiently generating debug data. Altering the ITM configuration may conflict with Ember debug output.

For further information on the ITM, contact Silicon Labs support for the ARM® CortexTM-M3 Technical Reference Manual, the ARM® CoreSight™ Components Technical Reference Manual, the ARM® v7-M Architecture Reference Manual, and the ARM® v7-M Architecture Application Level Reference Manual.

12. Data Watchpoint and Trace (DWT)

The EM342 integrates the standard ARM® Data Watchpoint and Trace (DWT). The DWT provides hardware support for profiling and debugging functionality. The DWT offers the following features:

- PC sampling
- Comparators to support:
  - Watchpoints - enters debug state
  - Data tracing
  - Cycle count matched PC sampling
- Exception trace support
- Instruction cycle count calculation support

Apart from exception tracing, DWT functionality is counter- or comparator-based. Watchpoint and data trace support use a set of compare, mask, and function registers. DWT-generated events result in one of two actions:

- Generation of a hardware event packet. Packets are generated and combined with software events and timestamp packets for transmission through the ITM/TPIU.
- A core halt - entry to debug state.

When exception tracing is enabled, the DWT emits an exception trace packet under the following conditions:

- Exception entry (from thread mode or pre-emption of a thread or handler).
- Exception exit when exiting a handler.
- Exception return when reentering a preempted thread or handler code sequence.

The DWT is designed for use with advanced profiling and debug tools, available from multiple vendors. Altering DWT configuration may conflict with the operation of advanced profiling and debug tools.

For further information on the DWT, contact Silicon Labs support for the ARM® CortexTM-M3 Technical Reference Manual, the ARM® CoreSight™ Components Technical Reference Manual, the ARM® v7-M Architecture Reference Manual, and the ARM® v7-M Architecture Application Level Reference Manual.

13. Flash Patch and Breakpoint (FPB)

The EM342 integrates the standard ARM® Flash Patch and Breakpoint (FPB). The FPB implements hardware breakpoints. The FPB also provides support for remapping of specific instruction or literal locations from flash memory to an address in RAM memory. The FPB contains:

- Two literal comparators for matching against literal loads from flash space and remapping to a corresponding RAM space.
- Six instruction comparators for matching against instruction fetches from flash space and remapping to a corresponding RAM space. Alternatively, the comparators can be individually configured to return a breakpoint instruction to the processor core on a match, implementing hardware breakpoint capability.

The FPB contains a global enable, but also individual enables for the eight comparators. If the comparison for an entry matches, the address is remapped to the address defined in the remap register plus and offset corresponding to the comparator that matched. Alternately, the address is remapped to a breakpoint instruction. The comparison happens on the fly, but the result of the comparison occurs too late to stop the original instruction fetch or literal load taking place from the flash space. The processor ignores this transaction, however, and only the remapped transaction is used.

Memory Protection Unit (MPU) lookups are performed for the original address, not the remapped address.

Unaligned literal accesses are not remapped. The original access to the bus takes place in this case.

The FPB is designed for use with advanced debug tools, available from multiple vendors. Altering the FPB configuration may conflict with the operation of advanced debug tools.

For further information on the FPB, contact Silicon Labs support for the ARM® CortexTM-M3 Technical Reference Manual, the ARM® CoreSight™ Components Technical Reference Manual, the ARM® v7-M Architecture Reference Manual, and the ARM® v7-M Architecture Application Level Reference Manual.

14. Integrated Voltage Regulator

The EM342 integrates two low dropout regulators to provide 1.8 V and 1.25 V power supplies as detailed in Table 14.1. The 1V8 regulator supplies the analog and memories, and the 1V25 regulator supplies the digital core. In deep sleep, the voltage regulators are disabled.

When enabled, the 1V8 regulator steps down the pads supply voltage (VDD_PADS) from a nominal 3.0 V to 1.8 V. The regulator output pin (VREG_OUT) must be decoupled externally with a suitable capacitor. VREG_OUT should be connected to the 1.8 V supply pins VDDA, VDD_RF, VDD_VCO, VDD_SYNTH, VDD_IF, and VDD_MEM. The 1V8 regulator can supply a maximum of 50 mA.

When enabled, the 1V25 regulator steps down VDD_PADS to 1.25 V. The regulator output pin (VDD_CORE, Pin 17) must be decoupled externally with a suitable capacitor. It should connect to the other VDD_CORE pin (Pin 44). The 1V25 regulator can supply a maximum of 10 mA.

The regulators are controlled by the digital portion of the chip as described in "6. System Modules" on page 33.

An example of decoupling capacitors and PCB layout can be found in the application notes (see the various Ember EM35x reference design documentation).

Table 14.1. Integrated Voltage Regulator Specifications

Spec Point	Min	Typ	Max	Units	Comments
Supply range for regulator	2.1		3.6	V	VDD_PADS
1V8 regulator output	–5%	1.8	+5%	V	Regulator output after initialization
1V8 regulator output after reset	–5%	1.75	+5%		Regulator output after reset
1V25 regulator output	–5%	1.25	+5%	V	Regulator output after initialization
1V25 regulator output after reset	–5%	1.45	+5%		Regulator output after reset
1V8 regulator capacitor		2.2		µF	Low ESR tantalum capacitor ESR greater than 2 Ω ESR less than 10 Ω de-coupling less than 100 nF ceramic
1V25 regulator capacitor		1.0		µF	Ceramic capacitor (0603)
1V8 regulator output current	0		50	mA	Regulator output current
1V25 regulator output current	0		10	mA	Regulator output current
No load current		600		µA	No load current (bandgap and regulators)
1V8 regulator current limit		200		mA	Short circuit current limit
1V25 regulator current limit		25		mA	Short circuit current limit
1V8 regulator start-up time		50		µs	0 V to POR threshold 2.2 µF capacitor
1V25 regulator start-up time		50		µs	0 V to POR threshold 1.0 µF capacitor

An external 1.8 V regulator may replace both internal regulators. The EM342 can control external regulators during deep sleep using open-drain GPIO PA7, as described in "7. GPIO (General Purpose Input/Output)" on page 48. The EM342 drives PA7 low during deep sleep to disable the external regulator, and an external pull-up is required to release this signal to indicate that supply voltage should be provided. Current consumption increases approximately 2 mA when using an external regulator. When using an external regulator, the internal regulators should be disabled through Ember software. The always-on domain needs to be minimally powered at 2.1 V and cannot be powered from the external 1.8 V regulator.

15. Serial Wire and JTAG (SWJ) Interface

The EM342 includes a standard Serial Wire and JTAG (SWJ) Interface. The SWJ is the primary debug and programming interface of the EM342. The SWJ gives debug tools access to the internal buses of the EM342 and allows for non-intrusive memory and register access as well as CPU halt-step style debugging. Therefore, any design implementing the EM342 should make the SWJ signals readily available.

Serial Wire is an ARM® standard, bidirectional, two-wire protocol designed to replace JTAG and provides all the normal JTAG debug and test functionality. JTAG is a standard five-wire protocol providing debug and test functionality. In addition, the two Serial Wire signals (SWDIO and SWCLK) are overlaid on two of the JTAG signals (JTMS and JTCK). This keeps the design compact and allows debug tools to switch between Serial Wire and JTAG as needed, without changing pin connections.

While Serial Wire and JTAG offer the same debug and test functionality, Silicon Labs recommends Serial Wire. Serial Wire uses only two pins instead of five, and offers a simple communication protocol, high performance data rates, low power, built-in error detection, and protection from glitches.

The ARM CoreSight™ Debug Access Port (DAP) comprises the Serial Wire and JTAG Interface (SWJ). As illustrated in Figure 15.1, the DAP includes two primary components: a debug port (the SWJ-DP) and an access port (the AHB-AP). The SWJ-DP provides external debug access while the AHB-AP provides internal bus access. An external debug tool connected to the EM342's debug pins communicates with the SWJ-DP. The SWJ-DP then communicates with the AHB-AP. Finally, the AHB-AP communicates on the internal bus.

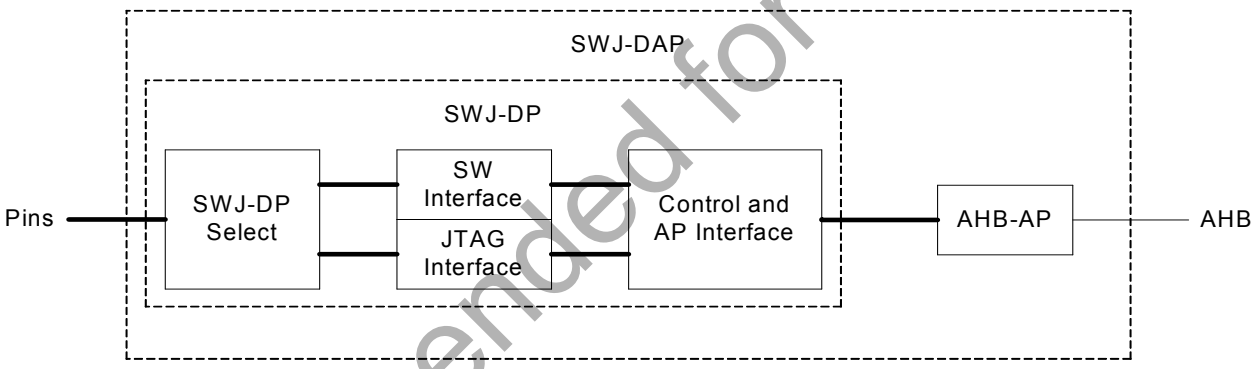


Figure 15.1. SWJ Block Diagram

Serial Wire and JTAG share five pins:

- JRST
- JTDO
- JTDI
- SWDIO/JTMS
- SWCLK/JTCK

Note: The SWJ pins are forced functions, and their corresponding GPIO_PxCFGH/L configurations are overridden when the EM342 resets. An application must disable all debug SWJ debug functionality to reclaim any of the four SWJ GPIOs: PC0, PC2, PC3, and PC4.

Since these pins can be repurposed, refer to "17. Pin Definitions" on page 121 and "7.3. Forced Functions" on page 50 for complete pin descriptions and configurations.

For further information on the SWJ, contact customer support for application notes and ARM® CoreSight™ documentation.

16. Ordering Information

Use the following part number to order the EM342:

Part Number	Part	Packaging Material	Configuration
EM342-RTR	EM342	2000 unit reel	Standard

The EM300 Series package is RoHS-compliant. It conforms to the European Court of Justice decision regarding the Deca-BDE exemption of the RoHS Directive. It is PFOS-compliant in accordance with European Directive 2006/122/EC*1 released in December 2006. The EM342 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.

17. Pin Definitions

17.1. Pin Definitions

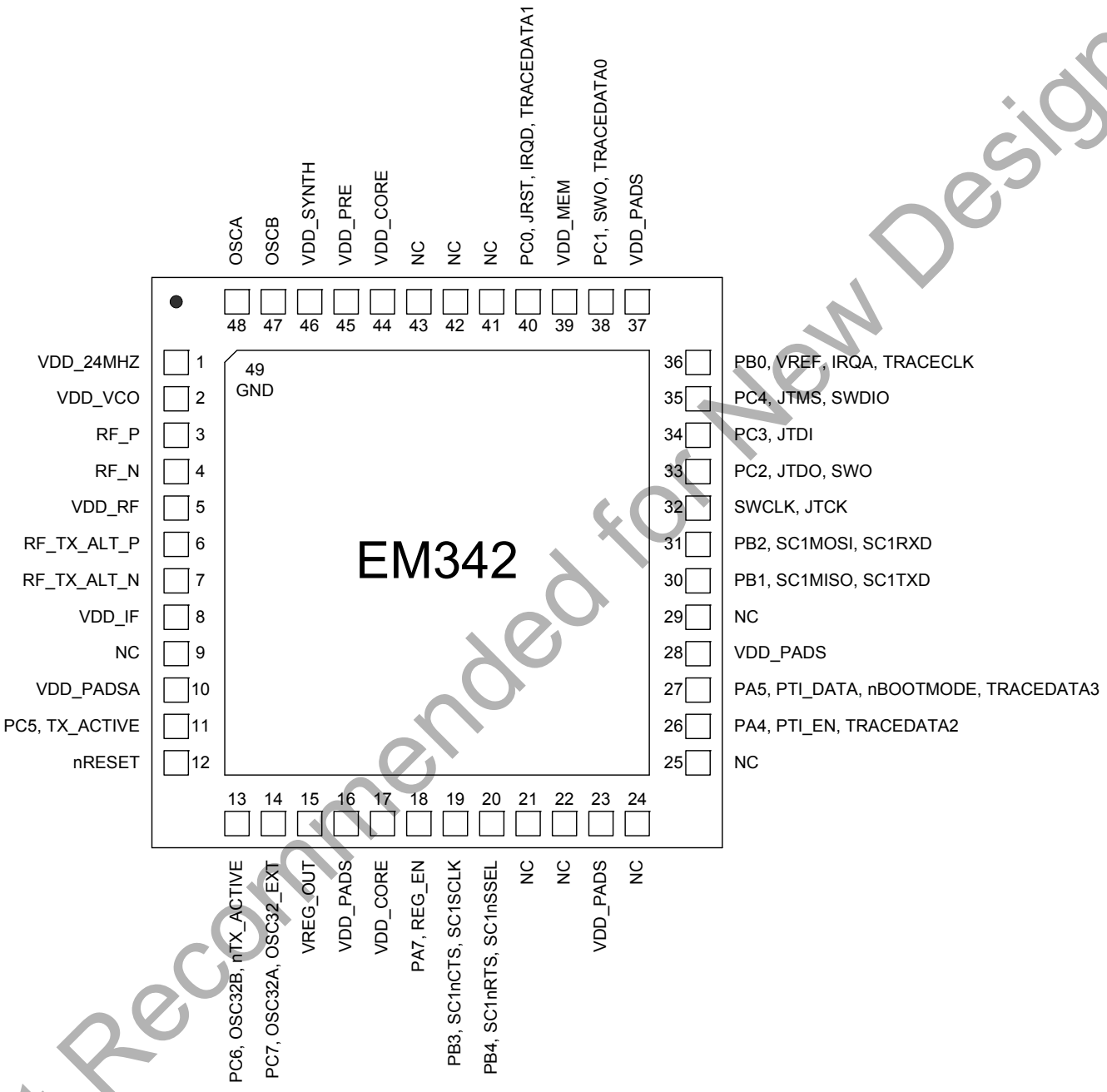


Figure 17.1. EM342 Pin Definitions

Refer to "7. GPIO (General Purpose Input/Output)" on page 48 for details about selecting GPIO pin functions.

Table 17.1. EM342 Pin Descriptions

Pin #	Signal	Direction	Description
1	VDD_24MHZ	Power	1.8 V high-frequency oscillator supply
2	VDD_VCO	Power	1.8 V VCO supply
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	1.8 V RF supply (LNA and PA)
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	1.8 V IF supply (mixers and filters)
9	NC		Do not connect
10	VDD_PADSA	Power	Analog pad supply (1.8 V)
11	PC5	I/O	Digital I/O
	TX_ACTIVE	O	Logic-level control for external RX/TX switch. The EM342 baseband controls TX_ACTIVE and drives it high (VDD_PADS) when in TX mode. Select alternate output function with GPIO_PCCFGH[7:4]
12	nRESET	I	Active low chip reset (internal pull-up)
13	PC6	I/O	Digital I/O
	OSC32B	I/O	32.768 kHz crystal oscillator Select analog function with GPIO_PCCFGH[11:8]
	nTX_ACTIVE	O	Inverted TX_ACTIVE signal (see PC5) Select alternate output function with GPIO_PCCFGH[11:8]
14	PC7	I/O	Digital I/O
	OSC32A	I/O	32.768 kHz crystal oscillator Select analog function with GPIO_PCCFGH[15:12]
	OSC32_EXT	I	Digital 32.768 kHz clock input source
15	VREG_OUT	Power	Regulator output (1.8 V while awake, 0 V during deep sleep)
16	VDD_PADS	Power	Pads supply (2.1–3.6 V)
17	VDD_CORE	Power	1.25 V digital core supply decoupling
18	PA7	I/O High current	Digital I/O Disable REG_EN with GPIO_DBGCFG[4]
	REG_EN	O	External regulator open drain output Enabled after reset

Table 17.1. EM342 Pin Descriptions (Continued)

Pin #	Signal	Direction	Description
19	PB3	I/O	Digital I/O
	SC1nCTS	I	UART CTS handshake of Serial Controller 1 Enable with SC1_UARTCFG[5] Select UART with SC1_MODE
	SC1SCLK	I	SPI slave clock of Serial Controller 1 Enable slave with SC1_SPICFG[4] Select SPI with SC1_MODE
20	PB4	I/O	Digital I/O
	SC1nRTS	O	UART RTS handshake of Serial Controller 1 Enable with SC1_UARTCFG[5] Select UART with SC1_MODE Select alternate output function with GPIO_PBCFGH[3:0]
	SC1nSSEL	I	SPI slave select of Serial Controller 1 Enable slave with SC1_SPICFG[4] Select SPI with SC1_MODE
21	NC		Do not connect
22	NC		Do not connect
23	VDD_PADS	Power	Pads supply (2.1–3.6 V)
24	NC		Do not connect
25	NC		Do not connect
26	PA4	I/O	Digital I/O
	PTI_EN	O	Frame signal of Packet Trace Interface (PTI) Disable trace interface in ARM core Enable PTI in Ember software Select alternate output function with GPIO_PACFGH[3:0]
	TRACEDATA2	O	Synchronous CPU trace data bit 2 Select 4-wire synchronous trace interface in ARM core Enable trace interface in ARM core Select alternate output function with GPIO_PACFGH[3:0]

Table 17.1. EM342 Pin Descriptions (Continued)

Pin #	Signal	Direction	Description
27	PA5	I/O	Digital I/O
	PTI_DATA	O	Data signal of Packet Trace Interface (PTI) Disable trace interface in ARM core Enable PTI in Ember software Select alternate output function with GPIO_PACFGH[7:4]
	nBOOTMODE	I	Activate FIB monitor instead of main program or bootloader when coming out of reset. Signal is active during and immediately after a reset on nRESET. See "7.5. Boot Configuration" on page 51.
	TRACEDATA3	O	Synchronous CPU trace data bit 3 Select 4-wire synchronous trace interface in ARM core Enable trace interface in ARM core Select alternate output function with GPIO_PACFGH[7:4]
28	VDD_PADS	Power	Pads supply (2.1–3.6 V)
29	NC		Do not connect
30	PB1	I/O	Digital I/O
	SC1MISO	O	SPI slave data out of Serial Controller 1 Select SPI with SC1_MODE Select slave with SC1_SPICR Select alternate output function with GPIO_PBCFGL[7:4]
	SC1TXD	O	UART transmit data of Serial Controller 1 Select UART with SC1_MODE Select alternate output function with GPIO_PBCFGL[7:4]
31	PB2	I/O	Digital I/O
	SC1MOSI	I	SPI slave data in of Serial Controller 1 Select SPI with SC1_MODE Select slave with SC1_SPICR
	SC1RXD	I	UART receive data of Serial Controller 1 Select UART with SC1_MODE
32	SWCLK	I/O	Serial Wire clock input/output with debugger Selected when in Serial Wire mode (see JTMS description, Pin 35)
	JTCK	I	JTAG clock input from debugger Selected when in JTAG mode (default mode, see JTMS description, Pin 35) Internal pull-down is enabled

Table 17.1. EM342 Pin Descriptions (Continued)

Pin #	Signal	Direction	Description
33	PC2	I/O	Digital I/O Enable with GPIO_DBGCFG[5]
	JTDO	O	JTAG data out to debugger Selected when in JTAG mode (default mode, see JTMS description, Pin 35)
	SWO	O	Serial Wire Output asynchronous trace output to debugger Select asynchronous trace interface in ARM core Enable trace interface in ARM core Select alternate output function with GPIO_PCCFGL[11:8] Enable Serial Wire mode (see JTMS description, Pin 35) Internal pull-up is enabled
34	PC3	I/O	Digital I/O Either Enable with GPIO_DBGCFG[5], or enable Serial Wire mode (see JTMS description)
	JTDI	I	JTAG data in from debugger Selected when in JTAG mode (default mode, see JTMS description, Pin 35) Internal pull-up is enabled
35	PC4	I/O	Digital I/O Enable with GPIO_DBGCFG[5]
	JTMS	I	JTAG mode select from debugger Selected when in JTAG mode (default mode) JTAG mode is enabled after power-up or by forcing nRESET low Select Serial Wire mode using the ARM-defined protocol through a debugger Internal pull-up is enabled
	SWDIO	I/O	Serial Wire bidirectional data to/from debugger Enable Serial Wire mode (see JTMS description) Select Serial Wire mode using the ARM-defined protocol through a debugger Internal pull-up is enabled
36	PB0	I/O	Digital I/O
	IRQA	I	External interrupt source A
	TRACECLK	O	Synchronous CPU trace clock Enable trace interface in ARM core Select alternate output function with GPIO_PBCFGL[3:0]
37	VDD_PADS	Power	Pads supply (2.1–3.6 V)

Table 17.1. EM342 Pin Descriptions (Continued)

Pin #	Signal	Direction	Description
38	PC1	I/O	Digital I/O
	SWO (see also Pin 33)	O	Serial Wire Output asynchronous trace output to debugger Select asynchronous trace interface in ARM core Enable trace interface in ARM core Select alternate output function with GPIO_PCCFGL[7:4]
	TRACEDATA0	O	Synchronous CPU trace data bit 0 Select 1-, 2- or 4-wire synchronous trace interface in ARM core Enable trace interface in ARM core Select alternate output function with GPIO_PCCFGL[7:4]
39	VDD_MEM	Power	1.8 V supply (flash, RAM)
40	PC0	I/O High current	Digital I/O Either enable with GPIO_DBGCFG[5], or enable Serial Wire mode (see JTMS description, Pin 35) and disable TRACEDATA1
	JRST	I	JTAG reset input from debugger Selected when in JTAG mode (default mode, see JTMS description) and TRACEDATA1 is disabled Internal pull-up is enabled
	IRQD	I	Default external interrupt source D. IRQC and IRQD external interrupts can be mapped to any digital I/O pin using the GPIO_IRQSEL and GPIO_IRQDSEL registers.
	TRACEDATA1	O	Synchronous CPU trace data bit 1 Select 2- or 4-wire synchronous trace interface in ARM core Enable trace interface in ARM core Select alternate output function with GPIO_PCCFGL[3:0]
41	NC		Do not connect
42	NC		Do not connect
43	NC		Do not connect
44	VDD_CORE	Power	1.25 V digital core supply decoupling
45	VDD_PRE	Power	1.8 V prescaler supply
46	VDD_SYNTH	Power	1.8 V synthesizer supply
47	OSCB	I/O	24 MHz crystal oscillator or left open when using external clock input on OSCA

Table 17.1. EM342 Pin Descriptions (Continued)

Pin #	Signal	Direction	Description
48	OSCA	I/O	24 MHz crystal oscillator or external clock input. (An external clock input should only be used for test and debug purposes. If used in this manner, the external clock input should be a 1.8 V, 50% duty cycle, square wave.)
49	GND	Ground	Ground supply pad in the bottom center of the package forms Pin 49. See the various Ember *EM35x Reference Design* documentation for PCB considerations.

18. Package

The EM342 package is a plastic 48-pin QFN that is 7 mm x 7 mm. Figure 18.1 illustrates the package drawing.

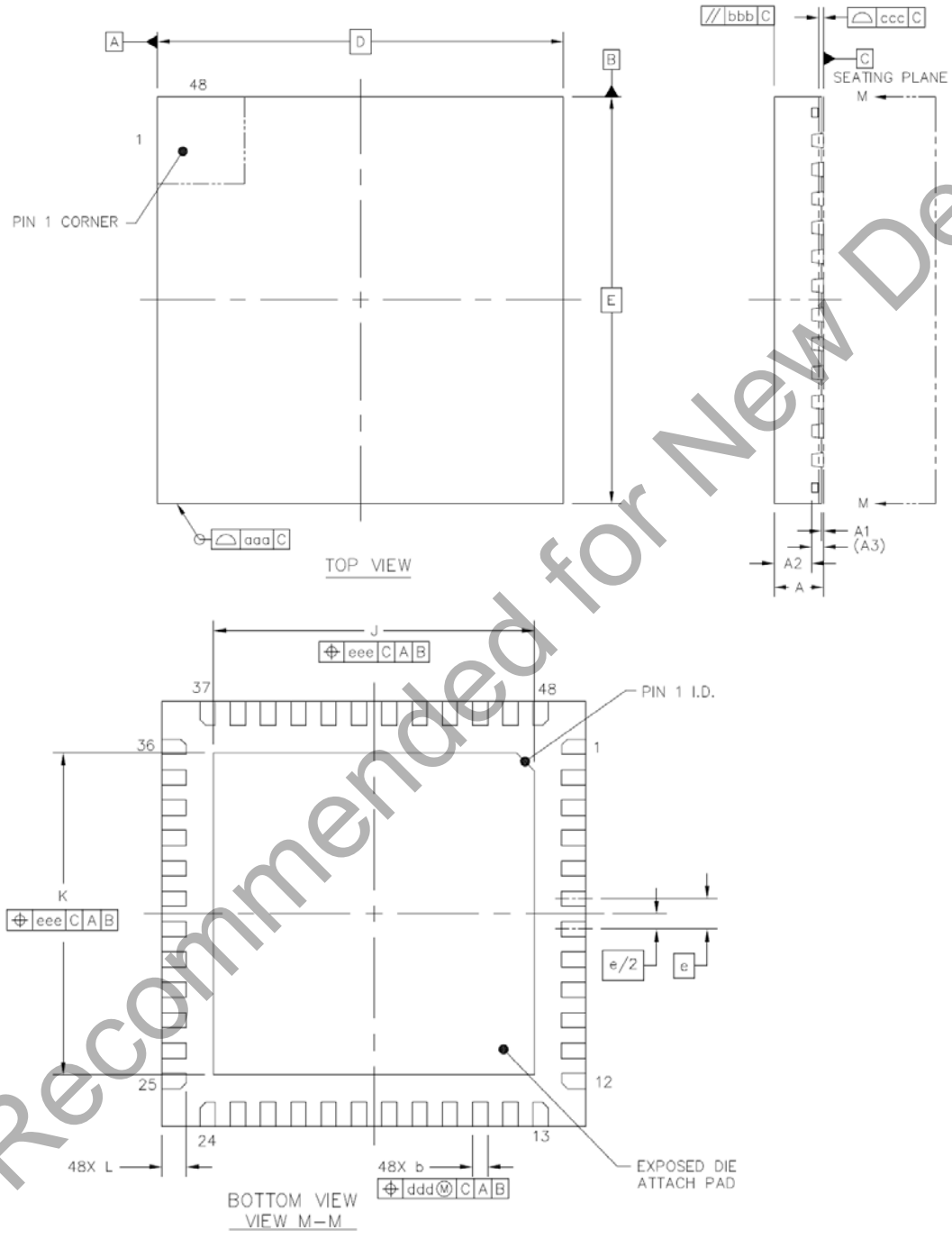


Figure 18.1. Package Drawing

Table 18.1. Package Dimensions

Dimension	MIN	NOM	MAX
A	0.80	0.85	0.90
A1	0	0.035	0.05
A2	—	0.65	0.67
A3	0.203 REF		
b	0.2	0.25	0.3
D	7 BSC		
E	7 BSC		
e	0.5 BSC		
J	5.2	5.3	5.4
K	5.2	5.3	5.4
L	0.35	0.40	0.45
aaa	0.10		
bbb	0.1		
ccc	0.08		
ddd	0.1		
eee	0.1		

Notes:

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. Dimensioning and Tolerancing per ANSI Y14.5M-1994.
3. This drawing conforms to the JEDEC Solid State Outline MO-220, Variation VKKD-4.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

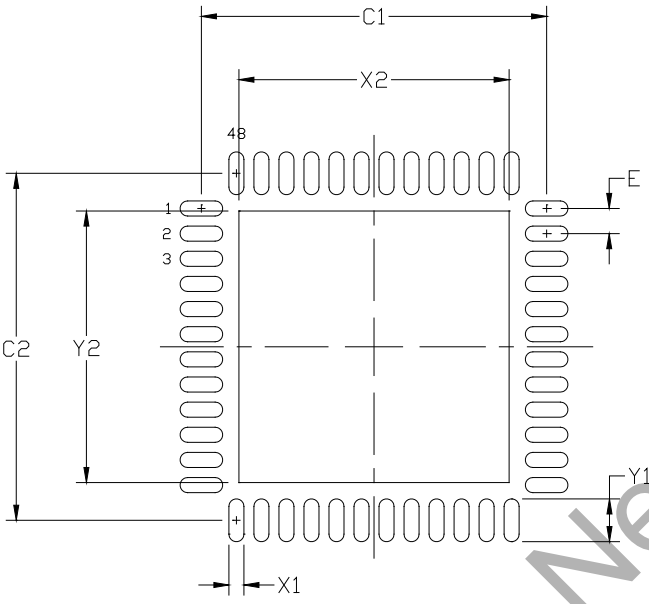


Figure 18.2. Solder Mask Dimensions

Table 18.2. PCB Land Pattern

Dimension	Min	Max
C1	6.80	6.90
C2	6.80	6.90
E	0.50 BSC	
X1	0.20	0.30
X2	5.20	5.40
Y1	0.75	0.85
Y2	5.20	5.40

Notes:

General

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. This Land Pattern Design is based on the IPC-7351 guidelines.

Solder Mask Design

1. All metal pads are to be non-solder mask defined (NSMD). Clearance between the solder mask and the metal pad is to be 60mm minimum, all the way around the pad.

Stencil Design

1. A stainless steel, laser-cut and electro-polished stencil with trapezoidal walls should be used to assure good solder paste release.
2. The stencil thickness should be 0.125 mm (5 mils).
3. The ratio of stencil aperture to land pad size should be 1:1 for all perimeter pads.
4. A 4x4 array of 1.1 mm square openings on 1.3 mm pitch can be used for the center ground pad.

Card Assembly

1. A No-Clean, Type-3 solder paste is recommended.
2. The recommended card reflow profile is per the JEDEC/IPC J-STD-020C specification for Small Body Components.

19. Top Marking

Figure 19.1 shows the part marking for the EM342 Series. The circle in the top corner indicates Pin 1. Pins are numbered counter-clockwise from Pin 1 with 12 pins per package edge.

Figure 19.1. Part Marking for EM342

Table 19.1. 48-Pin QFN Top Marking Explanation

Mark Method:	Laser	
Pin 1 Marking:	Circle = 0.40 mm Diameter (Top-Left Justified)	
Line 1 Marking:	Logo and Device Part Number Right Justified	Silicon Labs logo .
Line 2 Marking:	TTTTTT = Mfg Code YY=Year WW-Work Week	Manufacturing Code from the Assembly Purchase form. Assigned by the Assembly House. Corresponds to the year and work week. Right Justified
Line 3 Marking:	Circle = 1.3 mm Diameter Center Justified Country of Origin ISO abbreviation Right Justified	“e3” indicates Sn solder finish. TW

APPENDIX A—REGISTER ADDRESS TABLE

BLOCK	CM_LV	40004000–40004038 CM_LV		
Address	**Name**	**Type**	**Reset**	**Description**
40004038	PERIPHERAL_DISABLE	RW	0	Peripheral Disable Register

BLOCK	INTERRUPTS	4000A000–4000AFFF Interrupts		
Address	**Name**	**Type**	**Reset**	**Description**
4000A808	INT_SC1FLAG	RW	0	Serial Controller 1 Interrupt Flag Register
4000A814	INT_GPIOFLAG	RW	0	GPIO Interrupt Flag Register
4000A820	INT_MISS	RW	0	Top-Level Missed Interrupts Register
4000A848	INT_SC1CFG	RW	0	Serial Controller 1 Interrupt Configuration Register
4000A854	SC1_INTMODE	RW	0	Serial Controller 1 Interrupt Mode Register
4000A860	GPIO_INTCFGA	RW	0	GPIO Interrupt A Configuration Register
4000A864	GPIO_INTCFGB	RW	0	GPIO Interrupt B Configuration Register
4000A868	GPIO_INTCFGC	RW	0	GPIO Interrupt C Configuration Register
4000A86C	GPIO_INTCFGD	RW	0	GPIO Interrupt D Configuration Register

BLOCK	GPIO	4000B000–4000BFFF General Purpose IO		
Address	**Name**	**Type**	**Reset**	**Description**
4000B000	GPIO_PACFGL	RW	4444	Port A Configuration Register (Low)
4000B004	GPIO_PACFGH	RW	4444	Port A Configuration Register (High)
4000B008	GPIO_PAIN	RW	0	Port A Input Data Register
4000B00C	GPIO_PAOUT	RW	0	Port A Output Data Register
4000B010	GPIO_PASET	RW	0	Port A Output Set Register
4000B014	GPIO_PACLR	RW	0	Port A Output Clear Register
4000B400	GPIO_PBCFGL	RW	4444	Port B Configuration Register (Low)
4000B404	GPIO_PBCFGH	RW	4444	Port B Configuration Register (High)
4000B408	GPIO_PBIN	RW	0	Port B Input Data Register
4000B40C	GPIO_PBOUT	RW	0	Port B Output Data Register
4000B410	GPIO_PBSET	RW	0	Port B Output Set Register
4000B414	GPIO_PBCLR	RW	0	Port B Output Clear Register
4000B800	GPIO_PCCFGL	RW	4444	Port C Configuration Register (Low)
4000B804	GPIO_PCCFGH	RW	4444	Port C Configuration Register (High)
4000B808	GPIO_PCIN	RW	0	Port C Input Data Register
4000B80C	GPIO_PCOUT	RW	0	Port C Output Data Register
4000B810	GPIO_PCSET	RW	0	Port C Output Set Register
4000B814	GPIO_PCCLR	RW	0	Port C Output Clear Register
4000BC00	GPIO_DBGCFG	RW	10	GPIO Debug Configuration Register
4000BC04	GPIO_DBGSTAT	R	0	GPIO Debug Status Register
4000BC08	GPIO_PAWAKE	RW	0	Port A Wakeup Monitor Register
4000BC0C	GPIO_PBWAKE	RW	0	Port B Wakeup Monitor Register
4000BC10	GPIO_PCWAKE	RW	0	Port C Wakeup Monitor Register
4000BC14	GPIO_IRQCSEL	RW	F	Interrupt C Select Register
4000BC18	GPIO_IRQDSEL	RW	10	Interrupt D Select Register
4000BC1C	GPIO_WAKEFILT	RW	0	GPIO Wakeup Filtering Register

BLOCK	SERIAL	4000C000–4000CFFF Serial Controllers		
Address	**Name**	**Type**	**Reset**	**Description**
4000C800	SC1_RXBEGA	RW	20000000	Receive DMA Begin Address Register A
4000C804	SC1_RXENDA	RW	20000000	Receive DMA End Address Register A
4000C808	SC1_RXBEGB	RW	20000000	Receive DMA Begin Address Register B
4000C80C	SC1_RXENDB	RW	20000000	Receive DMA End Address Register B
4000C810	SC1_TXBEGA	RW	20000000	Transmit DMA Begin Address Register A
4000C814	SC1_TXENDA	RW	20000000	Transmit DMA End Address Register A
4000C818	SC1_TXBEGB	RW	20000000	Transmit DMA Begin Address Register B
4000C81C	SC1_TXENDB	RW	20000000	Transmit DMA End Address Register B
4000C820	SC1_RXCNTA	R	0	Receive DMA Count Register A
4000C824	SC1_RXCNTB	R	0	Receive DMA Count Register B
4000C828	SC1_TXCNT	R	0	Transmit DMA Count Register
4000C82C	SC1_DMASTAT	R	0	Serial DMA Status Register
4000C830	SC1_DMACTRL	RW	0	Serial DMA Control Register
4000C834	SC1_RXERRA	R	0	DMA First Receive Error Register A
4000C838	SC1_RXERRB	R	0	DMA First Receive Error Register B
4000C83C	SC1_DATA	RW	0	Serial Data Register
4000C840	SC1_SPISTAT	R	0	SPI Status Register
4000C848	SC1_UARTSTAT	R	40	UART Status Register
4000C854	SC1_MODE	RW	0	Serial Mode Register
4000C858	SC1_SPICFG	RW	0	SPI Configuration Register
4000C85C	SC1_UARTCFG	RW	0	UART Configuration Register
4000C860	SC1_RATELIN	RW	0	Serial Clock Linear Prescaler Register
4000C864	SC1_RATEEXP	RW	0	Serial Clock Exponential Prescaler Register
4000C868	SC1_UARTPER	RW	0	UART Baud Rate Period Register
4000C86C	SC1_UARTFRAC	RW	0	UART Baud Rate Fractional Period Register
4000C870	SC1_RXCNTSAVED	R	0	Saved Receive DMA Count Register

BLOCK	NVIC	E000E000 - E000EFFF Nested Vectored Interrupt Controller		
Address	**Name**	**Type**	**Reset**	**Description**
E000E100	INT_CFGSET	RW	0	Top-Level Set Interrupts Configuration Register
E000E180	INT_CFGCLR	RW	0	Top-Level Clear Interrupts Configuration Register
E000E200	INT_PENDSET	RW	0	Top-Level Set Interrupts Pending Register
E000E280	INT_PENDCLR	RW	0	Top-Level Clear Interrupts Pending Register
E000E300	INT_ACTIVE	R	0	Top-Level Active Interrupts Register
E000ED3C	SCS_AFSR	RW	0	Auxiliary Fault Status Register

APPENDIX B—ABBREVIATIONS AND ACRONYMS

Acronym/Abbreviation	Meaning
ACK	Acknowledgment
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
AGC	Automatic Gain Control
AHB	Advanced High Speed Bus
APB	Advanced Peripheral Bus
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
CIB	Customer Information Block
CLK1K	1 kHz Clock
CLK32K	32.768 kHz Crystal Clock
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA-CA	Carrier Sense Multiple Access-Collision Avoidance
CTR	Counter Mode
CTS	Clear to Send
DNL	Differential Non-Linearity
DMA	Direct Memory Access
DWT	Data Watchpoint and Trace
EEPROM	Electrically Erasable Programmable Read Only Memory
EM	Event Manager
ENOB	effective number of bits
ESD	Electro Static Discharge
ESR	Equivalent Series Resistance
ETR	External Trigger Input
FCLK	ARM® Cortex™-M3 CPU Clock

Acronym/Abbreviation	Meaning
FIB	Fixed Information Block
FIFO	First-in, First-out
FPB	Flash Patch and Breakpoint
GPIO	General Purpose I/O (pins)
HF	High Frequency
I^2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IF	Intermediate Frequency
IEEE	Institute of Electrical and Electronics Engineers
INL	Integral Non-linearity
ITM	Instrumentation Trace Macrocell
JTAG	Joint Test Action Group
LF	Low Frequency
LNA	Low Noise Amplifier
LQI	Link Quality Indicator
LSB	Least significant bit
MAC	Medium Access Control
MFB	Main Flash Block
MISO	Master in, slave out
MOS	Metal Oxide Semiconductor (P-channel or N-channel)
MOSI	Master out, slave in
MPU	Memory Protection Unit
MSB	Most significant bit
MSL	Moisture Sensitivity Level
NACK	Negative Acknowledge
NIST	National Institute of Standards and Technology
NMI	Non-Maskable Interrupt
NVIC	Nested Vectored Interrupt Controller
OPM	One-Pulse Mode

Acronym/Abbreviation	Meaning
O-QPSK	Offset-Quadrature Phase Shift Keying
OSC24M	High Frequency Crystal Oscillator
OSC32K	Low-Frequency 32.768 kHz Oscillator
OSCHF	High-Frequency Internal RC Oscillator
OSCRC	Low-Frequency RC Oscillator
PA	Power Amplifier
PCLK	Peripheral clock
PER	Packet Error Rate
PHY	Physical Layer
PLL	Phase-Locked Loop
POR	Power-On-Reset
PRNG	Pseudo Random Number Generator
PSD	Power Spectral Density
PTI	Packet Trace Interface
PWM	Pulse Width Modulation
QFN	Quad Flat Pack
RAM	Random Access Memory
RC	Resistive/Capacitive
RF	Radio Frequency
RMS	Root Mean Square
RoHS	Restriction of Hazardous Substances
RSSI	Receive Signal Strength Indicator
RTS	Request to Send
Rx	Receive
SYSCLK	System clock
SDFR	Spurious Free Dynamic Range
SFD	Start Frame Delimiter
SINAD	Signal-to-noise and distortion ratio
SPI	Serial Peripheral Interface

Acronym/Abbreviation	Meaning
SWJ	Serial Wire and JTAG Interface
THD	Total Harmonic Distortion
TRNG	True random number generator
TWI	Two Wire serial interface
Tx	Transmit
UART	Universal Asynchronous Receiver/Transmitter
UEV	Update event
VCO	Voltage Controlled Oscillator

Abbreviation	Meaning
dB	decibel
dBc	decibels relative to the carrier
dBm	decibels relative to 1 mW
GHz	GigaHerz
kB	Kilobyte
kbps	kilobits/second
kHz	kiloherz
kΩ	kiloOhm
kV	kiloVolt
mA	milliAmpere
Mbps	Megabits per second
MHz	megaherz
MΩ	megaOhm
MSPS	Megasamples per second
µA	microAmpere
µsec	microsecond
nH	nanohenry
ns	nanoseconds
Ω	Ohm
pF	picofarad
ppm	part per million
V	Volt

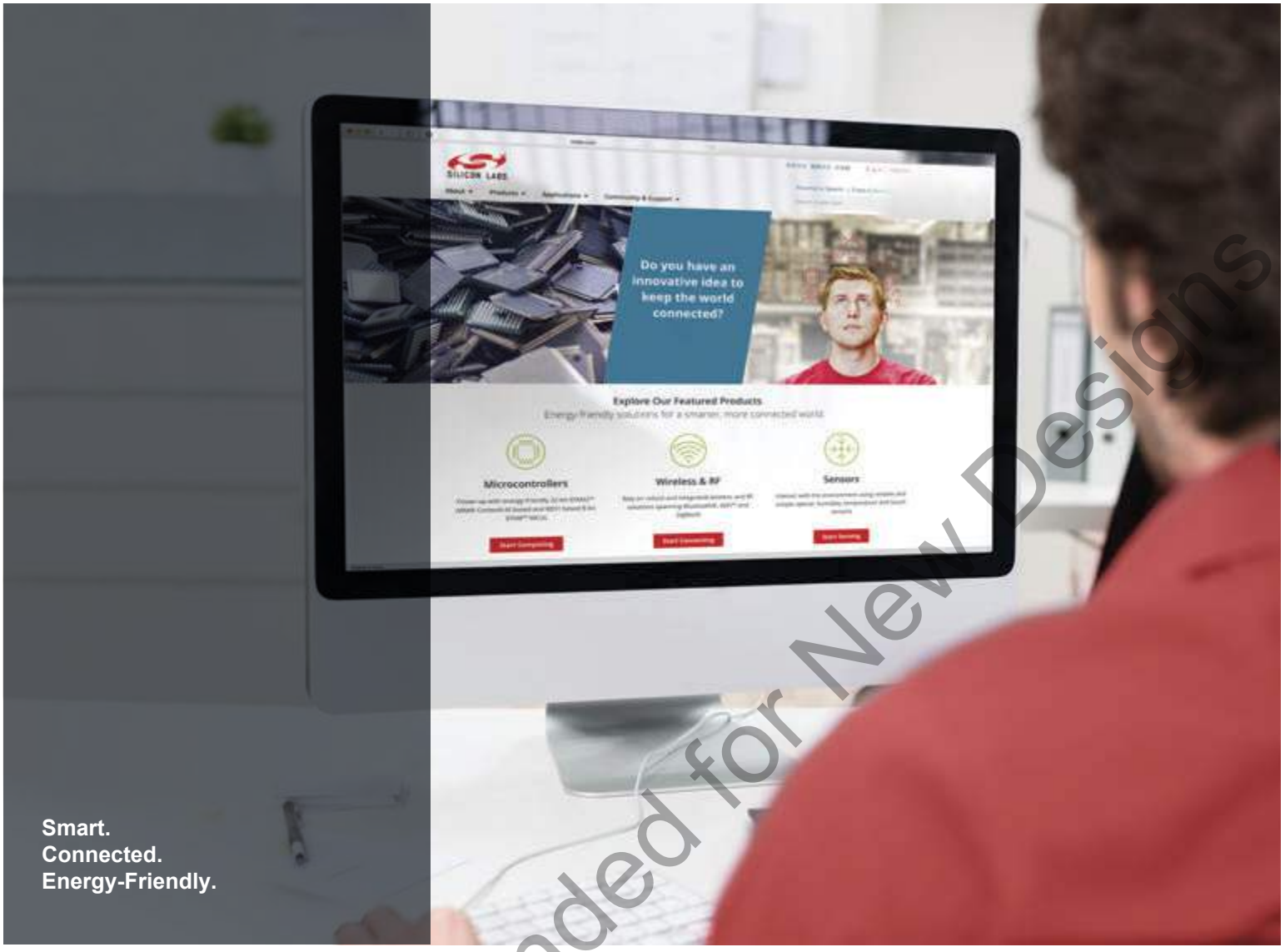
APPENDIX C—REFERENCES

- ZigBee Specification (www.zigbee.org; ZigBee Document 053474)
- ZigBee-PRO Stack Profile (www.zigbee.org; ZigBee Document 074855)
- ZigBee Stack Profile (www.zigbee.org; ZigBee Document 064321)
- 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)
- ARM® Cortex™-M3 reference manual (http://infocenter.arm.com/help/topic/com.arm.doc.subset.cortexm.m3/index.html#cortexm3)

DOCUMENT CHANGE LIST

Revision 1.0

- Initial Release



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Labs shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information

Silicon Laboratories Inc.® , Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOmodem®, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress® and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com