EM260

ZigBee/802.15.4 Network Processor

This datasheet applies to EmberZNet PRO 3.1 or greater.

Integrated 2.4GHz, IEEE 802.15.4-compliant transceiver:

- Robust RX filtering allows co-existence with IEEE 802.11g and Bluetooth devices
- - 99dBm RX sensitivity (1% PER, 20byte packet)
- + 2.5dBm nominal output power
- Increased radio performance mode (boost mode) gives – 100dBm sensitivity and + 4.5dBm transmit power
- Integrated VCO and loop filter
- Secondary TX-only RF port for applications requiring external PA.

Integrated IEEE 802.15.4 PHY and MAC

ZigBee-compliant stack running on the dedicated network processor

Controlled by the Host using the EmberZNet Serial Protocol (EZSP)

- Standard SPI or UART Interfaces allow for connection to a variety of Host microcontrollers

Non-intrusive debug interface (SIF)

Integrated hardware and software support for the Ember development environment

Dedicated peripherals and integrated memory

Provides integrated RC oscillator for low power operation

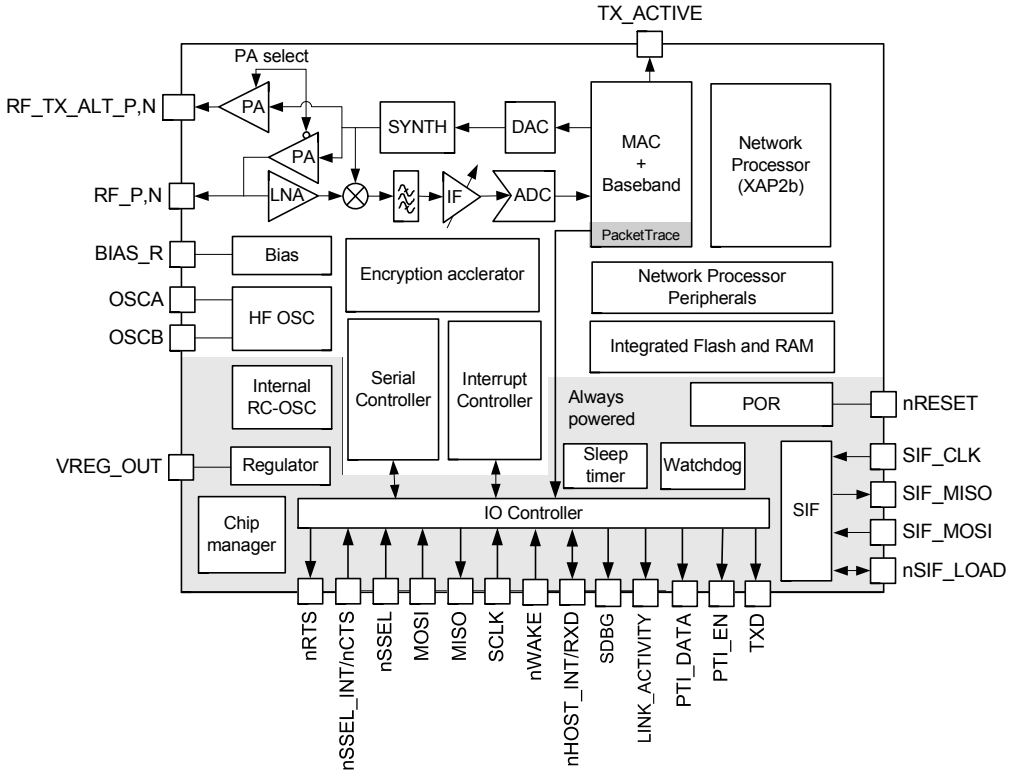
Three sleep modes:

- Processor idle (automatic)
- Deep sleep—1.0 µA
- Power down—1.0 µA

Watchdog timer and power-on-reset circuitry

Integrated AES encryption accelerator

Integrated 1.8V voltage regulator



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

May 29, 2013
120-0260-000M Rev 1.1

General Description

Note: Several important sections have been moved into standalone documents. Section 6 describing the ASH protocol has been moved to document UG101, *UART Gateway Protocol Reference*. Section 7 on EZSP has been moved to document UG100, *EZSP Reference Guide*.

The EM260 integrates a 2.4GHz, IEEE 802.15.4-compliant transceiver with a 16-bit network processor (XAP2b core) to run EmberZNet PRO, Silicon Labs' ZigBee-compliant network stack. The EM260 exposes access to the EmberZNet PRO API across a standard SPI module or a UART module, allowing application development on a Host platform. This means that the EM260 can be viewed as a ZigBee peripheral connected over a serial interface. The XAP2b microprocessor is a power-optimized core integrated in the EM260. It contains integrated Flash and RAM memory along with an optimized peripheral set to enhance the operation of the network stack.

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

The EM260 contains embedded Flash and integrated RAM for program and data storage. By employing an effective wear-leveling algorithm, the stack optimizes the lifetime of the embedded Flash, and affords the application the ability to configure stack and application tokens within the EM260.

To maintain the strict timing requirements imposed by ZigBee and the IEEE 802.15.4-2003 standard, the EM260 integrates a number of MAC functions into the hardware. The MAC hardware handles automatic ACK transmission and reception, automatic backoff delay, and clear channel assessment for transmission, as well as automatic filtering of received packets. In addition, the EM260 allows for true MAC level debugging by integrating the Packet Trace Interface.

An integrated voltage regulator, power-on-reset circuitry, sleep timer, and low-power sleep modes are available. The deep sleep and power down modes draw less than 1µA, allowing products to achieve long battery life.

Finally, the EM260 utilizes the non-intrusive SIF module for powerful software debugging and programming of the network processor.

Target applications for the EM260 include:

Building automation and control

Home automation and control

Home entertainment control

Asset tracking

The EM260 can only be purchased with the EmberZNet PRO stack. This technical datasheet details the EM260 features available to customers using it with the EmberZNet PRO stack.

Contents

1 Pin Assignments

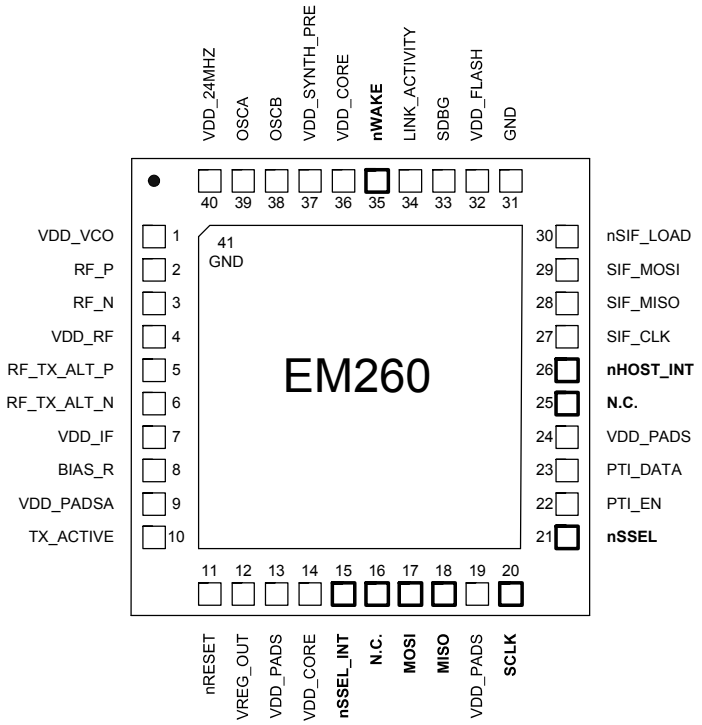


Figure 1. EM260 Pin Assignment for SPI Protocol

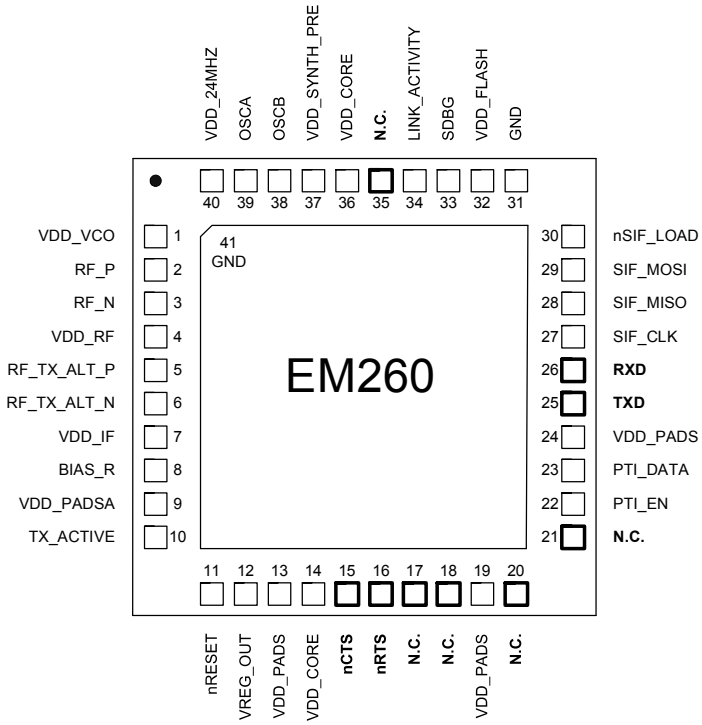


Figure 2. EM260 Pin Assignment for UART Protocol

Table 1. Pin Descriptions

Pin #	Signal	Direction	Description
1	VDD_VCO	Power	1.8V VCO supply; should be connected to VREG_OUT
2	RF_P	I/O	Differential (with RF_N) receiver input/transmitter output
3	RF_N	I/O	Differential (with RF_P) receiver input/transmitter output
4	VDD_RF	Power	1.8V RF supply (LNA and PA); should be connected to VREG_OUT
5	RF_TX_ALT_P	O	Differential (with RF_TX_ALT_N) transmitter output (optional)
6	RF_TX_ALT_N	O	Differential (with RF_TX_ALT_P) transmitter output (optional)
7	VDD_IF	Power	1.8V IF supply (mixers and filters); should be connected to VREG_OUT
8	BIAS_R	I	Bias setting resistor
9	VDD_PADSA	Power	Analog pad supply (1.8V); should be connected to VREG_OUT
10	TX_ACTIVE	O	Logic-level control for external RX/TX switch The EM260 baseband controls TX_ACTIVE and drives it high (1.8V) when in TX mode. (Refer to Table 6 and section 4.2.2.)
11	nRESET	I	Active low chip reset (internal pull-up)
12	VREG_OUT	Power	Regulator output (1.8V)
13	VDD_PADS	Power	Pads supply (2.1 – 3.6V)
14	VDD_CORE	Power	1.8V digital core supply; should be connected to VREG_OUT
15	nSSEL_INT	I	SPI Slave Select Interrupt (from Host to EM260) When using the SPI interface, this signal must be connected to nSSEL (Pin 21)
	nCTS	I	UART Clear To Send (enables EM260 transmission) When using the UART interface, this signal should be left unconnected if not used.
16	N.C.	I	When using the SPI interface, this signal is left not connected.
	nRTS	O	UART Request To Send (enables Host transmission) When using the UART interface, this signal should be left unconnected if not used.
17	MOSI	I	SPI Data, Master Out / Slave In (from Host to EM260)
	N.C.	I	When using the UART interface, this signal is left not connected.
18	MISO	O	SPI Data, Master In / Slave Out (from EM260 to Host)
	N.C.	I	When using the UART interface, this signal is left not connected.
19	VDD_PADS	Power	Pads supply (2.1 – 3.6V)
20	SCLK	I	SPI Clock (from Host to EM260)
	N.C.	I	When using the UART interface, this signal is left not connected.
21	nSSEL	I	SPI Slave Select (from Host to EM260)
	N.C.	I	When using the UART interface, this signal is left not connected.
22	PTI_EN	O	Frame signal of Packet Trace Interface (PTI)
23	PTI_DATA	O	Data signal of Packet Trace Interface (PTI)

Pin #	Signal	Direction	Description
24	VDD_PADS	Power	Pads supply (2.1 – 3.6V)
25	N.C.	I	When using the SPI interface, this signal is left not connected.
	TXD	O	UART Transmitted Data (from EM260 to Host)
26	nHOST_INT	O	SPI Host Interrupt signal (from EM260 to Host)
	RXD	I	UART Received Data (from Host to EM260)
27	SIF_CLK	I	Programming and Debug Interface, Clock (internal pull down)
28	SIF_MISO	O	Programming and Debug Interface, Master In / Slave Out
29	SIF_MOSI	I	Programming and Debug Interface, Master Out / Slave In (external pull-down required to guarantee state in Deep Sleep Mode)
30	nSIF_LOAD	I/O	Programming and Debug Interface, load strobe (open collector with internal pull up)
31	GND	Power	Ground Supply
32	VDD_FLASH	Power	1.8V Flash memory supply; should be connected to VREG_OUT
33	SDBG	O	Spare Debug signal
34	LINK_ACTIVITY	O	Link and Activity signal
35	nWAKE	I	SPI Wake Interrupt signal (from Host to EM260)
	N.C.	I	When using the UART interface, this signal is left not connected.
36	VDD_CORE	Power	1.8V digital core supply; should be connected to VREG_OUT
37	VDD_SYNTH_PRE	Power	1.8V synthesizer and prescalar supply; should be connected to VREG_OUT
38	OSCB	I/O	24MHz crystal oscillator or left open for when using an external clock input on OSCA
39	OSCA	I/O	24MHz crystal oscillator or external clock input
40	VDD_24MHZ	Power	1.8V high-frequency oscillator supply; should be connected to VREG_OUT
41	GND	Ground	Ground supply pad in the bottom center of the package forms Pin 41 (see the *EM260 Reference Design* for PCB considerations)

2 Top-Level Functional Description

Figure 3 shows a detailed block diagram of the EM260.

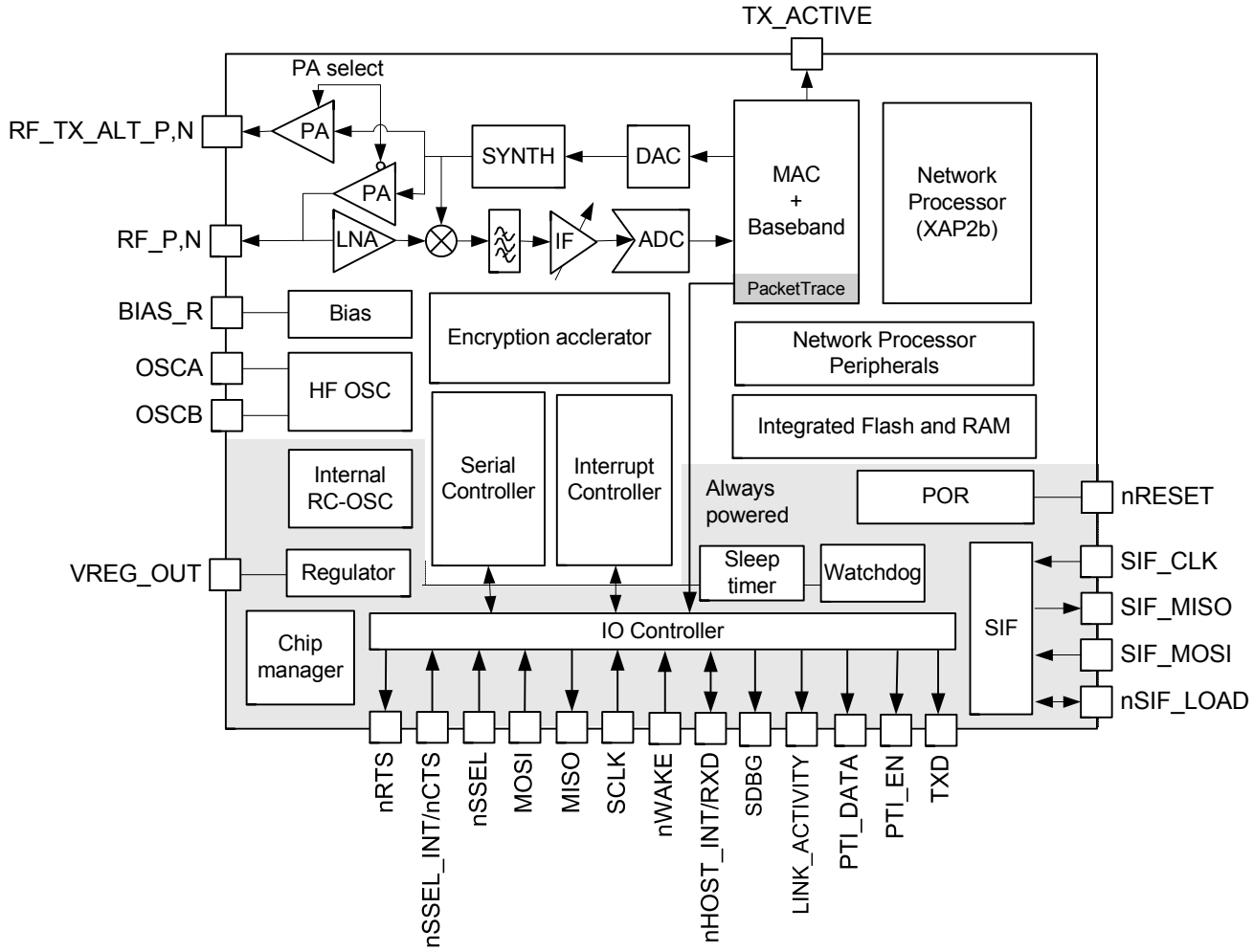


Figure 3. EM260 Block Diagram

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

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

The radio transmitter utilizes an efficient architecture in which the data stream directly modulates the VCO. An integrated PA boosts the output power. The calibration of the TX path as well as the output power is controlled by digital logic. If the EM260 is to be used with an external PA, the TX_ACTIVE signal should be used to control the timing of the external switching logic.

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

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

The EM260 integrates hardware support for a Packet Trace module, which allows robust packet-based debug. This element is a critical component of Ember Desktop, the Ember software IDE, providing advanced network debug capability when coupled with the Ember Debug Adapter (ISA).

The EM260 integrates a 16-bit XAP2b microprocessor developed by Cambridge Consultants Ltd. This power-efficient, industry-proven core provides the appropriate level of processing power to meet the needs of Silicon Lab's EmberZNet PRO ZigBee-compliant stack. In addition, the SIF module provides a non-intrusive programming and debug interface allowing for real-time application debugging.

The EM260 exposes the Ember Serial API over either a SPI or UART interface, which allows application development to occur on a Host platform of choice. The SPI interface uses the four standard SPI signals plus two additional signals, nHOST_INT and nWAKE, which provide an easy-to-use handshake mechanism between the Host and the EM260. The UART interface uses the two standard UART signals and also supports either standard RTS/CTS or XON/XOFF flow control.

The integrated voltage regulator generates a regulated 1.8V reference voltage from an unregulated supply voltage. This voltage is decoupled and routed externally to supply the 1.8V to the core logic. In addition, an integrated POR module allows for the proper cold start of the EM260.

The EM260 contains one high-frequency (24MHz) crystal oscillator and, for low-power operation, a second low-frequency internal 10 kHz oscillator.

The EM260 contains two power domains. The always-powered High Voltage Supply is used for powering the GPIO pads and critical chip functions. The rest of the chip is powered by a regulated Low Voltage Supply which can be disabled during deep sleep to reduce the power consumption.

3 Electrical Characteristics

3.1 Absolute Maximum Ratings

Table 2 lists the absolute maximum ratings for the EM260.

Table 2. Absolute Maximum Ratings

Parameter	Test Conditions	Min.	Max.	Unit
Regulator voltage (VDD_PADS)		- 0.3	3.6	V
Core voltage (VDD_24MHZ, VDD_VCO, VDD_RF, VDD_IF, VDD_PADSA, VDD_FLASH, VDD_SYNTH_PRE, VDD_CORE)		- 0.3	2.0	V
Voltage on RF_P,N; RF_TX_ALT_P,N		- 0.3	3.6	V
RF input power (for max level for correct packet reception, see Table 7)			+15	dBm
Voltage on nSSEL_INT, MOSI, MISO, SCLK, nSSEL, PTI_EN, PTI_DATA, nHOST_INT, SIF_CLK, SIF_MISO, SIF_MOSI, nSIF_LOAD, SDBG, LINK_ACTIVITY, nWAKE, nRESET, VREG_OUT		- 0.3	VDD_PADS+0.3	V
Voltage on TX_ACTIVE, BIAS_R, OSCA, OSCB		- 0.3	VDD_CORE+0.3	V
Storage temperature		- 40	+ 140	˚C

3.2 Recommended Operating Conditions

Table 3 lists the rated operating conditions of the EM260.

Table 3. Operating Conditions

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

3.3 Environmental Characteristics

Table 4 lists the environmental characteristics of the EM260.

Table 4. Environmental Characteristics

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

3.4 DC Electrical Characteristics

Table 5 lists the DC electrical characteristics of the EM260.

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

Table 5. DC Characteristics

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

3.5 Digital I/O Specifications

Table 6 contains the digital I/O specifications for the EM260. The digital I/O power (named VDD_PADS) comes from three dedicated pins (pins 13, 19, and 24). The voltage applied to these pins sets the I/O voltage.

Table 6. Digital I/O Specifications

Parameter	Name	Min.	Typ.	Max.	Unit
Voltage supply	VDD_PADS	2.1		3.6	V
Input voltage for logic 0	V_{IL}	0		0.2 x VDD_PADS	V
Input voltage for logic 1	V_{IH}	0.8 x VDD_PADS		VDD_PADS	V
Input current for logic 0	I_{IL}			-0.5	µA
Input current for logic 1	I_{IH}			0.5	µA
Input pull-up resistor value	R_{IPU}		30		kΩ
Input pull-down resistor value	R_{IPD}		30		kΩ
Output voltage for logic 0	V_{OL}	0		0.18 x VDD_PADS	V
Output voltage for logic 1	V_{OH}	0.82 x VDD_PADS		VDD_PADS	V
Output source current (standard current pad)	I_{OHS}			4	mA
Output sink current (standard current pad)	I_{OLS}			4	mA
Output source current (high current pad: pins 33, 34, and 35)	I_{OHH}			8	mA
Output sink current (high current pad: pins 33, 34, and 35)	I_{OLH}			8	mA
Total output current (for I/O pads)	$I_{OH} + I_{OL}$			40	mA
Input voltage threshold for OSCA		0.2 x VDD_CORE		0.8 x VDD_PADS	V
Output voltage level (TX_ACTIVE)		0.18 x VDD_CORE		0.82 x VDD_CORE	V
Output source current (TX_ACTIVE)				1	mA

3.6 RF Electrical Characteristics

3.6.1 Receive

Table 7 lists the key parameters of the integrated IEEE 802.15.4 receiver on the EM260.

Note: Receive Measurements were collected with the Silicon Labs EM260 Ceramic Balun Reference Design at 2440MHz and using the EmberZNet software stack Version 3.0.1. The Typical number indicates one standard deviation above the mean, measured at room temperature (25C). The Min and Max numbers are measured over process corners at room temperature (25C).

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

Table 7. Receive Characteristics

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

3.6.2 Transmit

Table 8 lists the key parameters of the integrated IEEE 802.15.4 transmitter on the EM260.

Note: Transmit Measurements were collected with the Silicon Labs EM260 Ceramic Balun Reference Design at 2440MHz and using the EmberZNet software stack Version 3.0.1. The Typical number indicates one standard deviation below the mean, measured at room temperature (25C). The Min and Max numbers are measured over process corners at room temperature (25C).

Table 8. Transmit Characteristics

Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Maximum output power (boost mode)	At highest power setting		4.5		dBm
Maximum output power	At highest power setting	- 0.5	2.5		dBm
Minimum output power	At lowest power setting		- 32		dBm
Error vector magnitude	As defined by IEEE 802.15.4, which sets a 35% maximum		15	25	%
Carrier frequency error		- 40		+ 40	ppm
Load impedance			200		Ω
PSD mask relative	3.5MHz away	- 20			dB
PSD mask absolute	3.5MHz away	- 30			dBm

3.6.3 Synthesizer

Table 9 lists the key parameters of the integrated synthesizer on the EM260.

Table 9. Synthesizer Characteristics

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

4 Functional Description

The EM260 connects to the Host platform through either a standard SPI interface or a standard UART interface. The EmberZNet Serial Protocol (EZSP) has been defined to allow an application to be written on a host platform of choice. Therefore, the EM260 comes with a license to EmberZNet PRO, Silicon Labs' ZigBee-compliant software stack. The following brief description of the hardware modules provides the necessary background on the operation of the EM260. For more information, contact .

4.1 Receive (RX) Path

The EM260 RX path spans the analog and digital domains. The RX architecture is based on a low-IF, super-heterodyne receiver. It utilizes differential signal paths to minimize noise interference. The input RF signal is mixed down to the IF frequency of 4MHz by I and Q mixers. The output of the mixers is filtered and combined prior to being sampled by a 12Msps ADC. The RX filtering within the RX path has been designed to optimize the co-existence of the EM260 with other 2.4GHz transceivers, such as the IEEE 802.11g and Bluetooth.

4.1.1 RX Baseband

The EM260 RX baseband (within the digital domain) implements a coherent demodulator for optimal performance. The baseband demodulates the O-QPSK signal at the chip level and synchronizes with the IEEE 802.15.4-2003 preamble. An automatic gain control (AGC) module adjusts the analog IF gain continuously (every ¼ symbol) until the preamble is detected. Once the packet preamble is detected, the IF gain is fixed during the packet reception. The baseband de-spreads the demodulated data into 4-bit symbols. These symbols are buffered and passed to the hardware-based MAC module for filtering.

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

4.1.2 RSSI and CCA

The EM260 calculates the RSSI over an 8-symbol period as well as at the end of a received packet. It utilizes the RX gain settings and the output level of the ADC within its algorithm. The linear range of RSSI is specified to be 40dB over all temperatures. At room temperature, the linear range is approximately 60dB (-90 dBm to -30dBm).

The EM260 RX baseband provides support for the IEEE 802.15.4-2003 required CCA methods summarized in Table 10. Modes 1, 2, and 3 are defined by the 802.15.4-2003 standard; Mode 0 is a proprietary mode.

Table 10. CCA Mode Behavior

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

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

At RX input powers higher than –25dBm, there is some compression in the receive chain where the gain is not properly adjusted. In the worst case, this has resulted in packet loss of up to 0.1%. This packet loss can be

seen in range testing measurements when nodes are closely positioned and transmitting at high power or when receiving from test equipment. There is no damage to the EM260 from this problem. This issue will rarely occur in the field as ZigBee Nodes will be spaced far enough apart. If nodes are close enough for it to occur in the field, the MAC and networking software treat the packet as not having been received and therefore the MAC level and network level retries resolve the problem without needing to notify the upper level application.

4.2 Transmit (TX) Path

The EM260 transmitter utilizes both analog circuitry and digital logic to produce the O-QPSK modulated signal. The area-efficient TX architecture directly modulates the spread symbols prior to transmission. The differential signal paths increase noise immunity and provide a common interface for the external balun.

4.2.1 TX Baseband

The EM260 TX baseband (within the digital domain) performs the spreading of the 4-bit symbol into its IEEE 802.15.4-2003-defined 32-chip I and Q sequence. In addition, it provides the interface for software to perform the calibration of the TX module in order to reduce process, temperature, and voltage variations.

4.2.2 TX_ACTIVE Signal

Even though the EM260 provides an output power suitable for most ZigBee applications, some applications will require an external power amplifier (PA). Due to the timing requirements of IEEE 802.15.4-2003, the EM260 provides a signal, TX_ACTIVE, to be used for external PA power management and RF Switching logic. When in TX, the TX Baseband drives TX_ACTIVE high (as described in Table 6). When in RX, the TX_ACTIVE signal is low. If an external PA is not required, then the TX_ACTIVE signal should be connected to GND through a 100k Ohm resistor, as shown in the application circuit in Figure 14.

The TX_ACTIVE signal can only source 1mA of current, and it is based upon the 1.8V signal swing. If the PA Control logic requires greater current or voltage potential, then TX_ACTIVE should be buffered externally to the EM260.

4.3 Integrated MAC Module

The EM260 integrates critical portions of the IEEE 802.15.4-2003 MAC requirements in hardware. This allows the EM260 to provide greater bandwidth to application and network operations. In addition, the hardware acts as a first-line filter for non-intended packets. The EM260 MAC utilizes a DMA interface to RAM memory to further reduce the overall microcontroller interaction when transmitting or receiving packets.

When a packet is ready for transmission, the software configures the TX MAC DMA by indicating the packet buffer RAM location. The MAC waits for the backoff period, then transitions the baseband to TX mode and performs channel assessment. When the channel is clear, the MAC reads data from the RAM buffer, calculates the CRC, and provides 4-bit symbols to the baseband. When the final byte has been read and sent to the baseband, the CRC remainder is read and transmitted.

The MAC resides in RX mode most of the time, and different format and address filters keep non-intended packets from using excessive RAM buffers, as well as preventing the EM260 CPU from being interrupted. When the reception of a packet begins, the MAC reads 4-bit symbols from the baseband and calculates the CRC. It assembles the received data for storage in a RAM buffer. A RX MAC DMA provides direct access to the RAM memory. Once the packet has been received, additional data is appended to the end of the packet in the RAM buffer space. The appended data provides statistical information on the packet for the software stack.

The primary features of the MAC are:

- CRC generation, appending, and checking
- Hardware timers and interrupts to achieve the MAC symbol timing
- Automatic preamble, and SFD pre-pended to a TX packet
- Address recognition and packet filtering on received packets

- Automatic acknowledgement transmission
- Automatic transmission of packets from memory
- Automatic transmission after backoff time if channel is clear (CCA)
- Automatic acknowledgement checking
- Time stamping of received and transmitted messages
- Attaching packet information to received packets (LQI, RSSI, gain, time stamp, and packet status)
- IEEE 802.15.4-2003 timing and slotted/unslotted timing

4.4 Packet Trace Interface (PTI)

The EM260 integrates a true PHY-level PTI for effective network-level debugging. This two-signal interface monitors all the PHY TX and RX packets (in a non-intrusive manner) between the MAC and baseband modules. It is an asynchronous 500kbps interface and cannot be used to inject packets into the PHY/MAC interface. The two signals from the EM260 are the frame signal (PTI_EN) and the data signal (PTI_DATA). The PTI is supported by Ember Desktop.

4.5 XAP2b Microprocessor

The EM260 integrates the XAP2b microprocessor developed by Cambridge Consultants Ltd., making it a true network processor solution. The XAP2b is a 16-bit Harvard architecture processor with separate program and data address spaces. The word width is 16 bits for both the program and data sides.

The standard XAP2 microprocessor and accompanying software tools have been enhanced to create the XAP2b microprocessor used in the EM260. The XAP2b adds data-side byte addressing support to the XAP2 allowing for more productive usage of RAM and optimized code.

The XAP2b clock speed is 12MHz. When used with the EmberZNet PRO stack, firmware may be loaded into Flash memory using the SIF mechanism (described in section 7) or over the air or by a serial link using a built-in bootloader[1] in a reserved area of the Flash. Alternatively, firmware may be loaded via the SIF interface with the assistance of RAM-based utility routines also loaded via SIF.

4.6 Embedded Memory

The EM260 contains embedded Flash and RAM memory for firmware storage and execution. In addition it partitions a portion of the Flash for Simulated EEPROM and token storage.

4.6.1 Simulated EEPROM

The EmberZNet PRO stack reserves a section of Flash memory to provide Simulated EEPROM storage area for stack and customer tokens. The Flash cell has been qualified for a data retention time of >100 years at room temperature and is rated to have a guaranteed 1,000 write/erase cycles. Because the Flash cells are qualified for up to 1,000 write cycles, the Simulated EEPROM implements an effective wear-leveling algorithm which effectively extends the number of write cycles for individual tokens.

The number of set-token operations is finite due to the write cycle limitation of the Flash. It is not possible to guarantee an exact number of set-token operations because the life of the Simulated EEPROM depends on which tokens are written and how often.

The EM260 stores non-volatile information necessary for network operation as well as 8 tokens available to the Host (see document UG100, *EZSP Reference Guide*). The majority of internal tokens is only written when the EM260 performs a network join or leave operation. As a simple ballpark estimate of possible set-token

[1] See document UG103.6, *Ember Application Development Fundamentals: Bootloading*, for more information on the bootloader.

operations, consider an EM260 in a stable network (no joins or leaves) not sending any messages and the Host is using only one of the 8-byte tokens available to it. Under this scenario, a very rough estimate results in approximately 330,000 possible set-token operations. The number of possible set-token calls, though, depends on which tokens are being set, so the ratios of set-token calls for each token play a large factor. A very rough estimate for the total number of times an App token can bet set is approximately 320,000.

These estimates would typically increase if the EM260 is kept closer to room temperature, since the 1,000 guaranteed write cycles of the Flash is for across temperature.

4.6.2 Flash Information Area (FIA)

The EM260 also includes a separate 1024-byte FIA that can be used for storage of data during manufacturing, including serial numbers and calibration values. Programming of this special Flash page can only be enabled using the SIF interface to prevent accidental corruption or erasure. The EmberZNet PRO stack reserves a small portion of this space for its own use and in addition makes eight manufacturing tokens available to the application. See document UG100, *EZSP Reference Guide*, for more information.

4.7 Encryption Accelerator

The EM260 contains a hardware AES encryption engine that is attached to the CPU using a memory-mapped interface. The CBC-MAC and CTR modes are implemented in hardware, and CCM* is implemented in software. The first two modes are described in the IEEE 802.15.4-2003 specification. CCM* is described in the ZigBee Specification (ZigBee document 053474). The EmberZNet PRO stack implements a security API for applications that require security at the application level.

4.8 nRESET Signal

When the asynchronous external reset signal, nRESET (Pin 13), is driven low for a time greater than 200ns, the EM260 resets to its default state. An integrated glitch filter prevents noise from causing an inadvertent reset to occur. If the EM260 is to be placed in a noisy environment, an external LC Filter or supervisory reset circuit is recommended to guarantee the integrity of the reset signal.

When nRESET asserts, all EM260 registers return to their reset state. In addition, the EM260 consumes 1.5mA (typical) of current when held in RESET.

4.9 Reset Detection

The EM260 contains multiple reset sources. The reset event is logged into the reset source register, which lets the CPU determine the cause of the last reset. The following reset causes are detected:

Power-on-Reset

Watchdog

PC rollover

Software reset

Core Power Dip

4.10 Power-on-Reset (POR)

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

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

The high-voltage Pads Supply VDD_PADS voltage rises above a threshold.

The internal RC clock starts and generates three clock pulses.

The 1.8V POR cell holds the main digital core in reset until the regulator output voltage rises above a threshold.

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

Table 11 lists the features of the EM260 POR circuitry.

Table 11. POR Specifications

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

4.11 Clock Sources

The EM260 integrates two oscillators: a high-frequency 24MHz crystal oscillator and a low-frequency internal 10kHz RC oscillator.

4.11.1 High-Frequency Crystal Oscillator

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

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

Table 12. High-Frequency Crystal Specifications

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

4.11.2 Internal RC Oscillator

The EM260 has a low-power, low-frequency RC oscillator that runs all the time. Its nominal frequency is 10kHz.

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

Timekeeping accuracy depends on temperature fluctuations the chip is exposed to, power supply impedance, and the calibration interval, but in general it will be better than 150ppm (including crystal error of 40ppm).

Table 13 lists the specifications of the RC oscillator.

Table 13. RC Oscillator Specifications

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

4.12 Random Number Generator

The EM260 allows for the generation of random numbers by exposing a randomly generated bit from the RX ADC. Analog noise current is passed through the RX path, sampled by the receive ADC, and stored in a

register. The value contained in this register could be used to seed a software-generated random number. The EmberZNet PRO stack utilizes these random numbers to seed the Random MAC Backoff and Encryption Key Generators.

4.13 Watchdog Timer

The EM260 contains an internal watchdog timer clocked from the internal oscillator. If the timer reaches its time-out value of approximately 2 seconds, it will reset the EM260. This reset signal cannot be routed externally to the Host.

The EM260 firmware will periodically restart the watchdog timer while the firmware is running normally. The Host cannot effect or configure the watchdog timer.

4.14 Sleep Timer

The 16-bit sleep timer is contained in the always-powered digital block. The clock source for the sleep timer is a calibrated 1kHz clock. The frequency is slowed down with a 2^N prescaler to generate a final timer resolution of 1ms. With a 1ms tick and a 16-bit timer, the timer wraps about every 65.5 seconds. The EmberZNet PRO stack appropriately handles timer wraps allowing the Host to order a theoretical maximum sleep delay of 4 million seconds.

4.15 Power Management

The EM260 supports four different power modes: active, idle, deep sleep, and power down.

Active mode is the normal, operating state of the EM260.

While in idle mode, code execution halts until any interrupt occurs. All modules of the EM260 including the radio continue to operate normally. The EmberZNet PRO stack automatically invokes idle as appropriate.

Deep sleep mode and power down mode both power off most of the EM260, including the radio, and leave only the critical chip functions powered. The internal regulator is disabled and VREG_OUT is turned off. All output signals are maintained in a frozen state. Upon waking from deep sleep or power down mode, the internal regulator is re-enabled. Deep sleep and power down result in the same sleep current consumption. The two sleep modes differ as follows: the EM260 can wake on both an internal timer and an external signal from deep sleep mode; power down mode can only wake on an external signal.

5 SPI Protocol

The EM260 Low Level Protocol centers on the SPI interface for communication with a pair of GPIO for handshake signaling.

The EM260 looks like a hardware peripheral.

The EM260 is the slave device and all transactions are initiated by the Host (the master).

The EM260 supports a reasonably high data rate.

5.1 Physical Interface Configuration

The EM260 supports both SPI Slave Mode 0 (clock is idle low, sample on rising edge) and SPI Slave Mode 3 (clock is idle high, sample on rising edge) at a maximum SPI clock rate of 5MHz, as illustrated in Figure 4. The convention for the waveforms in this document is to show Mode 0.

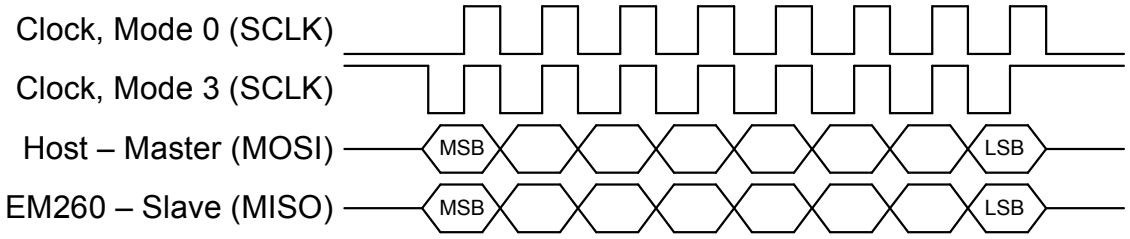


Figure 4. SPI Transfer Format, Mode 0 and Mode 3

The nHOST_INT signal and the nWAKE signal are both active low. The Host must supply a pull-up resistor on the nHOST_INT signal to prevent errant interruptions during undefined events such as the EM260 resetting. The EM260 supplies an internal pull-up on the nWAKE signal to prevent errant interruptions during undefined events such as the Host resetting.

5.2 SPI Transaction

The basic EM260 SPI transaction is half-duplex to ensure proper framing and to give the EM260 adequate response time. The basic transaction, as shown in Figure 5, is composed of three sections: Command, Wait, and Response. The transaction can be considered analogous to a function call. The Command section is the function call, and the Response section is the return value.

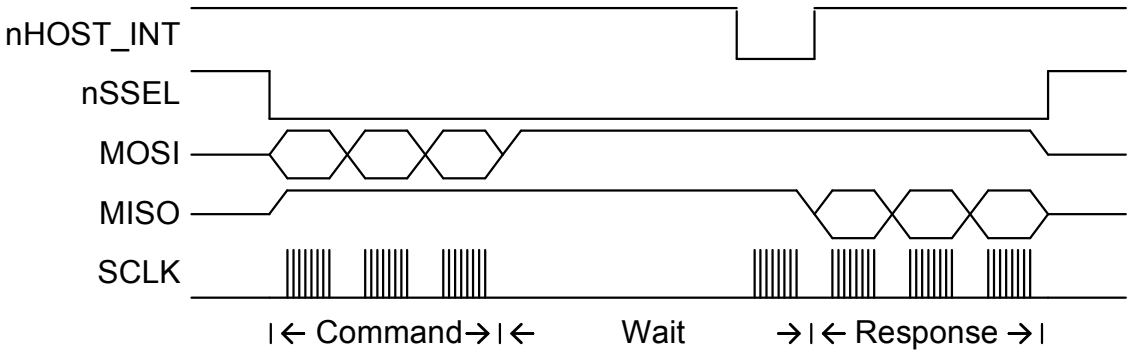


Figure 5. General Timing Diagram for a SPI Transaction

5.2.1 Command Section

The Host begins the transaction by asserting the Slave Select and then sending a command to the EM260. This command can be of any length from 2 to 136 bytes and must not begin with 0xFF. During the Command section, the EM260 will respond with only 0xFF. The Host should ignore data on MISO during the Command section. Once the Host has completed transmission of the entire message, the transaction moves to the Wait section.

5.2.2 Wait Section

The Wait section is a period of time during which the EM260 may be processing the command or performing other operations. Note that this section can be any length of time up to 300 milliseconds. Because of the variable size of the Wait section, an interrupt-driven or polling-driven method is suggested for clocking the SPI as opposed to a DMA method. Since the EM260 can require up to 300 milliseconds to respond, as long as the Host keeps Slave Select active, the Host can perform other tasks while waiting for a Response.

To determine when a Response is ready, use one of two methods:

Clock the SPI until the EM260 transmits a byte other than 0xFF.

Interrupt on the falling edge of nHOST_INT.

The first method, clocking the SPI, is recommended due to simplicity in implementing. During the Wait section, the EM260 will transmit only 0xFF and will ignore all incoming data until the Response is ready. When the EM260 transmits a byte other than 0xFF, the transaction has officially moved into the Response section. Therefore, the Host can poll for a Response by continuing to clock the SPI by transmitting 0xFF and waiting for the EM260 to transmit a byte other than 0xFF. The EM260 will also indicate that a Response is ready by asserting the nHOST_INT signal. The falling edge of nHOST_INT is the indication that a Response is ready. Once the nHOST_INT signal asserts, nHOST_INT will return to idle after the Host begins to clock data.

5.2.3 Response Section

When the EM260 transmits a byte other than 0xFF, the transaction has officially moved into the Response section. The data format is the same format used in the Command section. The response can be of any length from 2 to 136 bytes and will not begin with 0xFF. Depending on the actual response, the length of the response is known from the first or second byte and this length should be used by the Host to clock out exactly the correct number of bytes. Once all bytes have been clocked, it is allowable for the Host to deassert chip select. Since the Host is in control of clocking the SPI, there are no ACKs or similar signals needed back from the Host because the EM260 will assume the Host could accept the bytes being clocked on the SPI. After every transaction, the Host must hold the Slave Select high for a minimum of 1ms. This timing requirement is called the inter-command spacing and is necessary to allow the EM260 to process a command and become ready to accept a new command.

5.2.4 Asynchronous Signaling

When the EM260 has data to send to the Host, it will assert the nHOST_INT signal. The nHOST_INT signal is designed to be an edge-triggered signal as opposed to a level-triggered signal; therefore, the falling edge of nHOST_INT is the true indicator of data availability. The Host then has the responsibility to initiate a transaction to ask the EM260 for its output. The Host should initiate this transaction as soon as possible to prevent possible backup of data in the EM260. The EM260 will deassert the nHOST_INT signal after receiving a byte on the SPI. Due to inherent latency in the EM260, the timing of when the nHOST_INT signal returns to idle can vary between transactions. nHOST_INT will always return to idle for a minimum of 10us before asserting again. If the EM260 has more output available after the transaction has completed, the nHOST_INT signal will assert again after Slave Select is deasserted and the Host must make another request.

5.2.5 Spacing

To ensure that the EM260 is always able to deal with incoming commands, a minimum inter-command spacing is defined at 1ms. After every transaction, the Host must hold the Slave Select high for a minimum of 1ms. The Host must respect the inter-command spacing requirement, or the EM260 will not have time to operate on the command; additional commands could result in error conditions or undesired behavior. If the nHOST_INT signal is not already asserted, the Host is allowed to use the Wake handshake instead of the inter-command spacing to determine if the EM260 is ready to accept a command.

5.2.6 Waking the EM260 from Sleep

Waking up the EM260 involves a simple handshaking routine as illustrated in Figure 6. This handshaking insures that the Host will wait until the EM260 is fully awake and ready to accept commands from the Host. If the EM260 is already awake when the handshake is performed (such as when the Host resets and the EM260 is already operating), the handshake will proceed as described below with no ill effects.

Note: A wake handshake cannot be performed if nHOST_INT is already asserted.

Note: nWAKE should not be asserted after the EM260 has been reset until the EM260 has fully booted, as indicated by the EM260 asserting nHOST_INT. If nWAKE is asserted during this boot time, the EM260 may enter bootloader mode. See section 5.6.1, Bootloading the EM260.

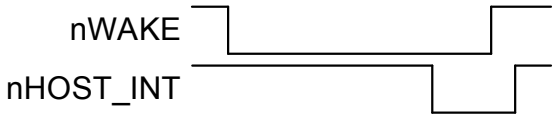


Figure 6. EM260 Wake Sequence

Waking the EM260 involves the following steps:

1. Host asserts nWAKE.
2. EM260 interrupts on nWAKE and exits sleep.
3. EM260 performs all operations it needs to and will not respond until it is ready to accept commands.
4. EM260 asserts nHOST_INT within 300ms of nWAKE asserting. If the EM260 does not assert nHOST_INT within 300ms of nWAKE, it is valid for the Host to consider the EM260 unresponsive and to reset the EM260.
5. Host detects nHOST_INT assertion. Since the assertion of nHOST_INT indicates the EM260 can accept SPI transactions, the Host does not need to hold Slave Select high for the normally required minimum 1ms of inter-command spacing.
6. Host deasserts nWAKE after detecting nHOST_INT assertion.
7. EM260 will deassert nHOST_INT within 25µs of nWAKE deasserting.
8. After 25µs, any change on nHOST_INT will be an indication of a normal asynchronous (callback) event.

5.2.7 Error Conditions

If two or more different error conditions occur back to back, only the first error condition will be reported to the Host (if it is possible to report the error). The following are error conditions that might occur with the EM260.

Unsupported SPI Command: If the SPI Byte of the command is unsupported, the EM260 will drop the incoming command and respond with the Unsupported SPI Command Error Response. This error means the SPI Byte is

unsupported by the current Mode the EM260 is in. Bootloader Frames can only be used with the bootloader and EZSP Frames can only be used with the EZSP.

Oversized Payload Frame: If the transaction includes a Payload Frame, the Length Byte cannot be a value greater than 133. If the EM260 detects a length byte greater than 133, it will drop the incoming Command and abort the entire transaction. The EM260 will then assert nHOST_INT after Slave Select returns to Idle to inform the Host through an error code in the Response section what has happened. Not only is the Command in the problematic transaction dropped by the EM260, but the next Command is also dropped, because it is responded to with the Oversized Payload Frame Error Response.

Aborted Transaction: An aborted transaction is any transaction where Slave Select returns to Idle prematurely and the SPI Protocol dropped the transaction. The most common reason for Slave Select returning to Idle prematurely is the Host unexpectedly resetting. If a transaction is aborted, the EM260 will assert nHOST_INT to inform the Host through an error code in the Response section what has happened. When a transaction is aborted, not only does the Command in the problematic transaction get dropped by the EM260, but the next Command also gets dropped since it is responded to with the Aborted Transaction Error Response.

Missing Frame Terminator: Every Command and Response must be terminated with the Frame Terminator byte. The EM260 will drop any Command that is missing the Frame Terminator. The EM260 will then immediately provide the Missing Frame Terminator Error Response.

Long Transaction: A Long Transaction error occurs when the Host clocks too many bytes. As long as the inter-command spacing requirement is met, this error condition should not cause a problem, since the EM260 will send only `0xFF` outside of the Response section as well as ignore incoming bytes outside of the Command section.

Unresponsive: Unresponsive can mean the EM260 is not powered, not fully booted yet, incorrectly connected to the Host, or busy performing other tasks. The Host must wait the maximum length of the Wait section before it can consider the EM260 unresponsive to the Command section. This maximum length is 300 milliseconds, measured from the end of the last byte sent in the Command Section. If the EM260 ever fails to respond during the Wait section, it is valid for the Host to consider the EM260 unresponsive and to reset the EM260. Additionally, if nHOST_INT does not assert within 300ms of nWAKE asserting during the wake handshake, the Host can consider the EM260 unresponsive and reset the EM260.

5.3 SPI Protocol Timing

Figure 7 illustrates all critical timing parameters in the SPI Protocol. These timing parameters are a result of the EM260's internal operation and both constrain Host behavior and characterize EM260 operation. The parameters shown are discussed elsewhere in this document. Note that Figure 7 is not drawn to scale, but is instead drawn only to illustrate where the parameters are measured.

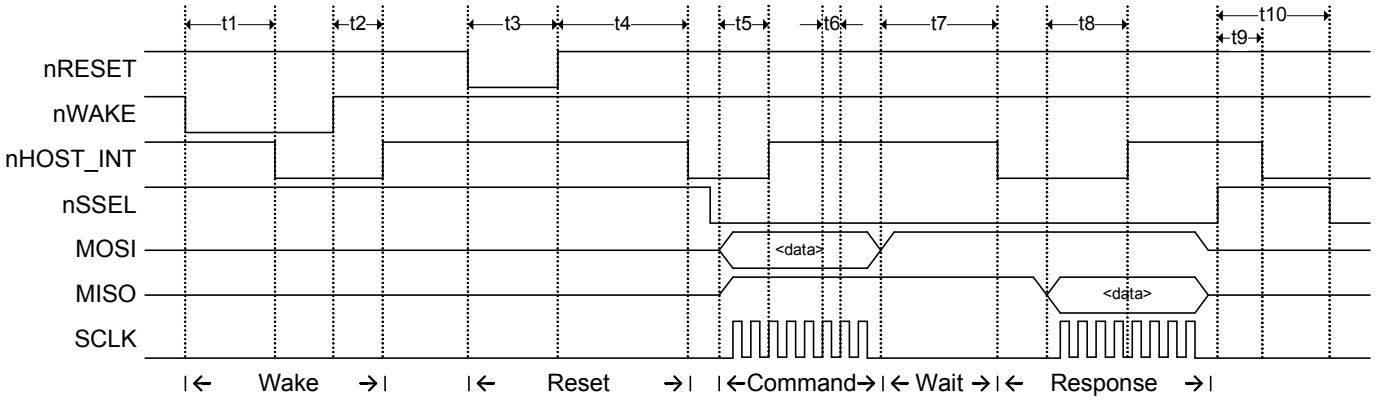


Figure 7. SPI Protocol Timing Waveform

Table 14 lists the timing parameters of the SPI Protocol. These parameters are illustrated in Figure 7.

Table 14. SPI Protocol Timing Parameters

Parameter	Description	Min.	Typ.	Max.	Unit
t1 (a)	Wake handshake, while 260 is awake		133	150	µs
t1 (b)	Wake handshake, while 260 is asleep		7.3	300	ms
t2	Wake handshake finish	1.1	1.2	25	µs
t3	Reset pulse width	8			µs
t4 (a)	Startup time, entering application		250	1500	ms
t4 (b)	Startup time, entering bootloader		2.5	7.5	s
t5	nHOST_INT deasserting after Command	13	35	75	µs
t6	Clock rate	200			ns
t7	Wait section	25	755	300000	µs
t8	nHOST_INT deasserting after Response	20	130	800	µs
t9	nHOST_INT asserting after transaction	25	70	800	µs
t10	Inter-command spacing	1			ms

5.4 Data Format

The data format, also referred to as a *command*, is the same for both the Command section and the Response section. The data format of the SPI Protocol is straightforward, as illustrated in Figure 8.

Figure 8. SPI Protocol Data Format

The total length of a command must not exceed 136 bytes.

All commands must begin with the *SPI Byte*. Some commands are only two bytes—that is, they contain the SPI Byte and Frame Terminator only.

The *Length Byte* is only included if there is information in the Payload Frame and the Length Byte defines the length of just the Payload Frame. Therefore, if a command includes a Payload Frame, the Length Byte can have a value from 2 through 133 and the overall command size will be 5 through 136 bytes. The SPI Byte can be a specific value indicating if there is a Payload Frame or not, and if there is a Payload Frame, then the Length Byte can be expected.

The *Error Byte* is used by the error responses to provide additional information about the error and appears in place of the length byte. This additional information is described in the following sections.

The *Payload Frame* contains the data needed for operating EmberZNet PRO. The EZSP Frame and its format are explained in document UG100, *EZSP Reference Guide*. The Payload Frame may also contain the data needed for operating the bootloader, which is called a Bootloader Frame. Refer to document UG103.6, *Ember Application Development Fundamentals: Bootloading*, for more information on the bootloader.

The *Frame Terminator* is a special control byte used to mark the end of a command. The Frame Terminator byte is defined as `0xA7` and is appended to all Commands and Responses immediately after the final data byte. The purpose of the Frame Terminator is to provide a known byte the SPI Protocol can use to detect a

corrupt command. For example, if the EM260 resets during the Response Section, the Host will still clock out the correct number of bytes. But when the host attempts to verify the value 0xA7 at the end of the Response, it will see either the value 0x00 or 0xFF and know that the EM260 just reset and the corrupt Response should be discarded.

Note: The Length Byte only specifies the length of the Payload Frame. It does not include the Frame Terminator.

5.5 SPI Byte

Table 15 lists the possible commands and their responses in the SPI Byte.

Table 15. SPI Commands & Reponses

Command Value	Command	Response Value	Response
Any	Any	0x00	EM260 reset occurred—This is never used in another Response; it always indicates an EM260 Reset.
Any	Any	0x01	Oversized Payload Frame received—This is never used in another Response; it always indicates an overflow occurred.
Any	Any	0x02	Aborted Transaction occurred—This is never used in another Response; it always indicates an aborted transaction occurred.
Any	Any	0x03	Missing Frame Terminator—This is never used in another Response; it always indicates a Missing Frame Terminator in the Command.
Any	Any	0x04	Unsupported SPI Command—This is never used in another Response; it always indicates an unsupported SPI Byte in the Command.
0x00 – 0x09	Reserved	[none]	[none]
0x0A	SPI Protocol Version	0x81 – 0xBF	bit[7] is always set. bit[6] is always cleared. bit[5:0] is a number from 1–63.
0x0B	SPI Status	0xC0 – 0xC1	bit[7] is always set. bit[6] is always set. bit[0]—Set if Alive.
0x0C – 0xFC	Reserved	[none]	[none]
0xFD	Bootloader Frame	0xFD	Bootloader Frame
0xFE	EZSP Frame	0xFE	EZSP Frame
0xFF	Invalid	0xFF	Invalid

5.5.1 Primary SPI Bytes

There are four primary SPI Bytes: SPI Protocol Version, SPI Status, Bootloader Frame, and EZSP Frame.

SPI Protocol Version [0x0A]: Sending this command requests the SPI Protocol Version number from the SPI Interface. The response will always have bit 7 set and bit 6 cleared. In this current version, the response

will be 0x82, since the version number corresponding to this set of Command-Response values is version number 2. The version number can be a value from 1 to 63 (0x81–0xBF).

SPI Status [0x0B]: Sending this command asks for the EM260 status. The response status byte will always have the upper 2 bits set. In this current version, the status byte only has one status bit [0], which is set if the EM260 is alive and ready for commands.

Bootloader Frame [0xFD]: This byte indicates that the current transaction is a Bootloader transaction and there is more data to follow. This SPI Byte will cause the transaction to look like the full data format illustrated in Figure 8. The byte immediately after this SPI Byte will be a Length Byte, and it is used to identify the length of the Bootloader Frame. Refer to document UG103.6, *Ember Application Development Fundamentals: Bootloading*, for more information on the bootloader. If the SPI Byte is 0xFD, it means the minimum transaction size is four bytes.

EZSP Frame [0xFE]: This byte indicates that the current transaction is an EZSP transaction and there is more data to follow. This SPI Byte will cause the transaction to look like the full data format illustrated in Figure 8. The byte immediately after this SPI Byte will be a Length Byte, and it is used to identify the length of the EZSP Frame. (The EZSP Frame is defined in document UG100, *EZSP Reference Guide*.) If the SPI Byte is 0xFE, it means the minimum transaction size is five bytes.

5.5.2 Special Response Bytes

There are only five SPI Byte values, 0x00–0x04, ever used as error codes (see Table 16). When the error condition occurs, any command sent to the EM260 will be ignored and responded to with one of these codes. These special SPI Bytes must be trapped and dealt with. In addition, for each error condition the Error Byte (instead of the Length Byte) is also sent with the SPI Byte.

Table 16. Byte Values Used as Error Codes

SPI Byte Value	Error Message	Error Description	Error Byte Description
0x00	EM260 Reset	See section 5.6, Powering On, Power Cycling, and Rebooting.	The reset type. Refer to API documentation discussing EmberResetType.
0x01	Oversized EZSP Frame	The command contained an EZSP frame with a Length Byte greater than 133. The EM260 was forced to drop the entire command.	Reserved
0x02	Aborted Transaction	The transaction was not completed properly and the EM260 was forced to abort the transaction.	Reserved
0x03	Missing Frame Terminator	The command was missing the Frame Terminator. The EM260 was forced to drop the entire command.	Reserved
0x04	Unsupported SPI Command	The command contained an unsupported SPI Byte. The EM260 was forced to drop the entire command.	Reserved

5.6 Powering On, Power Cycling, and Rebooting

When the Host powers on (or reboots), it cannot guarantee that the EM260 is awake and ready to receive commands. Therefore, the Host should always perform the Wake EM260 handshake to guarantee that the EM260 is awake. If the EM260 resets, it needs to inform the Host so that the Host can reconfigure the stack if needed.

When the EM260 resets, it will assert the nHOST_INT signal, telling the Host that it has data. The Host should request data from the EM260 as usual. The EM260 will ignore whatever command is sent to it and respond only with two bytes. The first byte will always be `0x00` and the second byte will be the reset type as defined by `EmberResetType`. This specialty SPI Byte is never used in another Response SPI Byte. If the Host sees `0x00` from the EM260, it knows that the EM260 has been reset. The EM260 will deassert the nHOST_INT signal shortly after receiving a byte on the SPI and process all further commands in the usual manner. In addition to the Host having control of the reset line of the EM260, the EmberZNet Serial Protocol also provides a mechanism for a software reboot.

5.6.1 Bootloading the EM260

The SPI Protocol supports a Payload Frame called the Bootloader Frame for communicating with the EM260 when the EM260 is in bootloader mode. The EM260 can enter bootload mode through either an EZSP command or holding one of two pins low while the EM260 exits reset. Both the nWAKE pin and the PTI_DATA pin are capable of activating the bootloader while performing a standard EM260 reset procedure. Assert nRESET to hold the EM260 in reset. While nRESET is asserted, assert (active low) either nWAKE or PTI_DATA and then deassert nRESET to boot the EM260. Do not deassert nWAKE or PTI_DATA until the EM260 asserts nHOST_INT, indicating that the EM260 has fully booted and is ready to accept data over the SPI Protocol. Once nHOST_INT is asserted, nWAKE or PTI_DATA way be deasserted. Refer to document UG103.6, *Ember Application Development Fundamentals: Bootloading*, for more information on the bootloader and the format of the Bootloader Frame.

5.6.2 Unexpected Resets

The EM260 is designed to protect itself against undefined behavior due to unexpected resets. The protection is based on the state of Slave Select since the inter-command spacing mandates that Slave Select must return to idle. The EM260's internal SPI Protocol uses Slave Select returning to idle as a trigger to reinitialize its SPI Protocol. By always reinitializing, the EM260 is protected against the Host unexpectedly resetting or terminating a transaction. Additionally, if Slave Select is active when the EM260 powers on, the EM260 will ignore SPI data until Slave Select returns to idle. By ignoring SPI traffic until idle, the EM260 will not begin receiving in the middle of a transaction.

If the Host resets, in most cases it should reset the EM260 as well so that both devices are once again in the same state: freshly booted. Alternately, the Host can attempt to recover from the reset by recovering its previous state and resynchronizing with the state of the EM260.

If the EM260 resets during a transaction, the Host can expect either a Wait Section timeout or a missing Frame Terminator indicating an invalid Response.

If the EM260 resets outside of a transaction, the Host should proceed normally.

5.7 Transaction Examples

This section contains the following transaction examples:

SPI Protocol Version

EmberZNet Serial Protocol Frame-Version Command

EM260 Reset

Three-Part Transaction: Wake, Get Version, Stack Status Callback

5.7.1 SPI Protocol Version

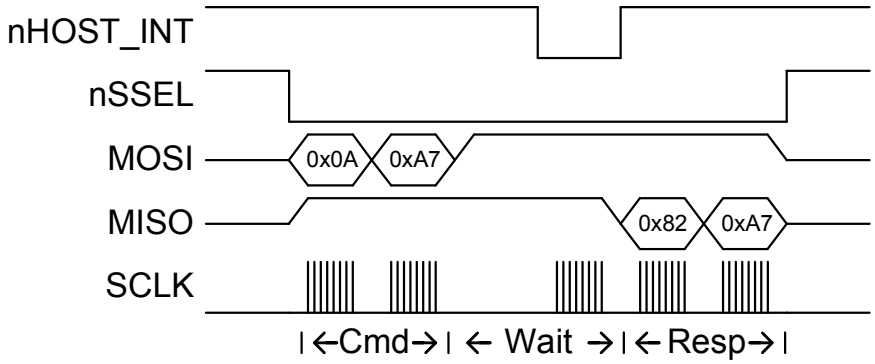


Figure 9. SPI Protocol Version Example

1. Activate Slave Select (nSSEL).
2. Transmit the command `0x0A` - SPI Protocol Version Request.
3. Transmit the Frame Terminator, `0xA7`.
4. Wait for nHOST_INT to assert.
5. Transmit and receive `0xFF` until a byte other than `0xFF` is received.
6. Receive response `0x82` (a byte other than `0xFF`), then receive the Frame Terminator, `0xA7`.
7. Bit 7 is always set and bit 6 is always cleared in the Version Response, so this is Version 2.
8. Deactivate Slave Select.

5.7.2 EmberZNet Serial Protocol Frame—Version Command

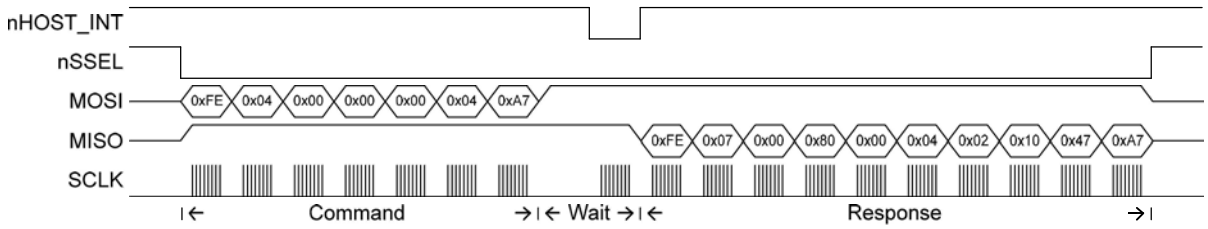


Figure 10. EmberZNet Serial Protocol Frame - Version Command Example

1. Activate Slave Select (nSSEL).
2. Transmit the appropriate command:
   - `0xFE`: SPI Byte indicating an EZSP Frame
   - `0x04`: Length Byte showing the EZSP Frame is 4 bytes long
   - `0x00`: EZSP Sequence Byte (Note that this value should vary based upon previous sequence bytes)
   - `0x00`: EZSP Frame Control Byte indicating a command with no sleeping
   - `0x00`: EZSP Frame ID Byte indicating the `Version` command
   - `0x04`: EZSP Parameter for this command (`desiredProtocolVersion`)
   - `0xA7`: Frame Terminator
3. Wait for nHOST_INT to assert.
4. Transmit and receive `0xFF` until a byte other than `0xFF` is received.
5. Receive response `0xFE` (a byte other than `0xFF`) and read the next byte for a length.
6. Stop transmitting after the number of bytes (length) is received plus the Frame Terminator.
7. Decode the response:
   - `0xFE`: SPI Byte indicating an EZSP Frame
   - `0x07`: Length Byte showing the EZSP Frame is 7 bytes long
   - `0x00`: EZSP Sequence Byte (Note that this value should vary based upon previous sequence bytes)
   - `0x80`: EZSP Frame Control Byte indicating a response with no overflow
   - `0x00`: EZSP Frame ID Byte indicating the `Version` response
   - `0x04`: EZSP Parameter for this response (`protocolVersion`)
   - `0x02`: EZSP Parameter for this response (`stackType`)
   - `0x10`: EZSP Parameter for this response (`stackVersion`, Note that this value may vary)
   - `0x47`: EZSP Parameter for this response (`stackVersion`, Note that this value may vary)
   - `0xA7`: Frame Terminator
8. Deactivate Slave Select.

5.7.3 EM260 Reset

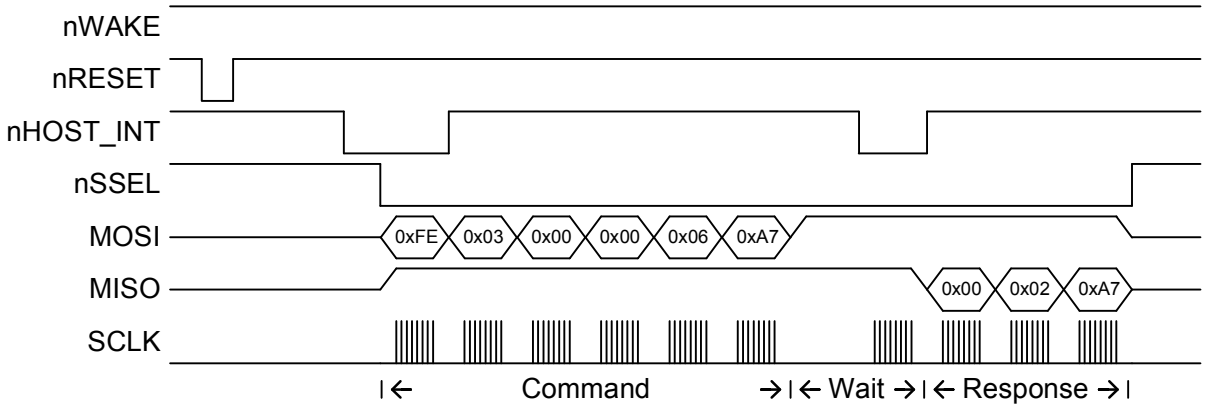


Figure 11. EM260 Reset Example

1. nRESET toggles active low to reset the EM260.
2. nWAKE stays idle high between nRESET and nHOST_INT indicating the EM260 should continue with normal booting (do not enter the bootloader).
3. nHOST_INT asserts.
4. Activate Slave Select (nSSEL).
5. Transmit the command:
   - `0xFE`: SPI Byte indicating an EZSP Frame
   - `0x03`: Length Byte showing the EZSP Frame is 3 bytes long
   - `0x00`: EZSP Sequence Byte (Note that this value should vary based upon previous sequence bytes)
   - `0x00`: EZSP Frame Control Byte indicating a command with no sleeping
   - `0x06`: EZSP Frame ID Byte indicating the `callback` command
   - `0xA7`: Frame Terminator
6. Wait for nHOST_INT to assert.
7. Transmit and receive `0xFF` until a byte other than `0xFF` is received.
8. Receive response `0x00` (a byte other than `0xFF`).
9. Receive the Error Byte and decode (`0x02` is enumerated as RESET_POWERON).
10. Receive the Frame Terminator (`0xA7`).
11. Response `0x00` indicates the EM260 has reset and the Host should respond appropriately.
12. Deactivate Slave Select.
13. Since nHOST_INT does not assert again, there is no more data for the Host.

5.7.4 Three-Part Transaction: Wake, Get Version, Stack Status Callback

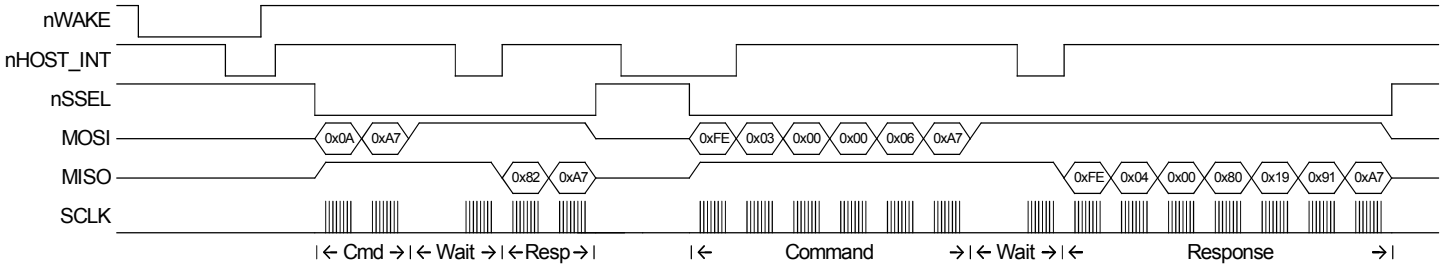


Figure 12. Timing Diagram of the Three-Part Transaction

1. Activate nWAKE and activate timeout timer.
2. EM260 wakes up (if not already) and enables communication.
3. nHOST_INT asserts, indicating the EM260 can accept commands.
4. Host sees nHOST_INT activation within 300ms and deactivates nWAKE and timeout timer.
5. nHOST_INT deasserts immediately after nWAKE.
6. Activate Slave Select.
7. Transmit the Command `0x0A` - SPI Protocol Version Request.
8. Transmit the Frame Terminator, `0xA7`.
9. Wait for nHOST_INT to assert.
10. Transmit and receive `0xFF` until a byte other than `0xFF` is received.
11. Receive response `0x82` (a byte other than `0xFF`), then receive the Frame Terminator, `0xA7`.
12. Bit 7 is always set and bit 6 is always cleared in the Version Response, so this is Version 2.
13. Deactivate Slave Select.
14. Host begins timing the inter-command spacing of 1ms in preparation for sending the next command.
15. nHOST_INT asserts shortly after deactivating Slave Select, indicating a callback.
16. Host sees nHOST_INT, but waits for the 1ms before responding.
17. Activate Slave Select.
18. Transmit the command:
    - `0xFE`: SPI Byte indicating an EZSP Frame
    - `0x03`: Length Byte showing the EZSP Frame is 3 bytes long
    - `0x00`: EZSP Sequence Byte (Note that this value should vary based upon previous sequence bytes)
    - `0x00`: EZSP Frame Control Byte indicating a command with no sleeping
    - `0x06`: EZSP Frame ID Byte indicating the `callback` command
    - `0xA7`: Frame Terminator
19. Wait for nHOST_INT to assert.
20. Transmit and receive `0xFF` until a byte other than `0xFF` is received.
21. Receive response `0xFE` (a byte other than `0xFF`), read the next byte for a length.

22. Stop transmitting after the number of bytes (length) is received plus the Frame Terminator.
23. Decode the response:
    - `0xFE`: SPI Byte indicating an EZSP Frame
    - `0x04`: Length Byte showing the EZSP Frame is 3 bytes long
    - `0x00`: EZSP Sequence Byte (Note that this value should vary based upon previous sequence bytes)
    - `0x80`: EZSP Frame Control Byte indicating a response with no overflow
    - `0x19`: EZSP Frame ID Byte indicating the `stackStatusHandler` command
    - `0x91`: EZSP Parameter for this response `(EmberStatus EMBER_NETWORK_DOWN)`
    - `0xA7` - Frame Terminator
24. Deactivate Slave Select.
25. Since nHOST_INT does not assert again, there is no more data for the Host.

6 UART Gateway Protocol

The UART Gateway protocol is designed for network gateway systems in which the host processor is running a full-scale operating system such as embedded Linux or Windows. The host sends EmberZNet Serial Protocol (EZSP) commands to the UART interface using the Ember Asynchronous Serial Host (ASH) protocol. The EZSP commands are the same as those used in the SPI protocol, but the SPI protocol is better suited for resource-constrained microcontroller hosts since ASH uses considerably more host RAM and program storage.

ASH implements error detection/recovery and tolerates latencies on multi-tasking hosts due to scheduling and I/O buffering. The ASH protocol is described in detail in document UG101, *UART Gateway Protocol Reference*.

Silicon Labs supplies ASH host software in source form compatible with Linux and Windows. In most cases it will need only a few simple edits to adapt it to a particular host system.

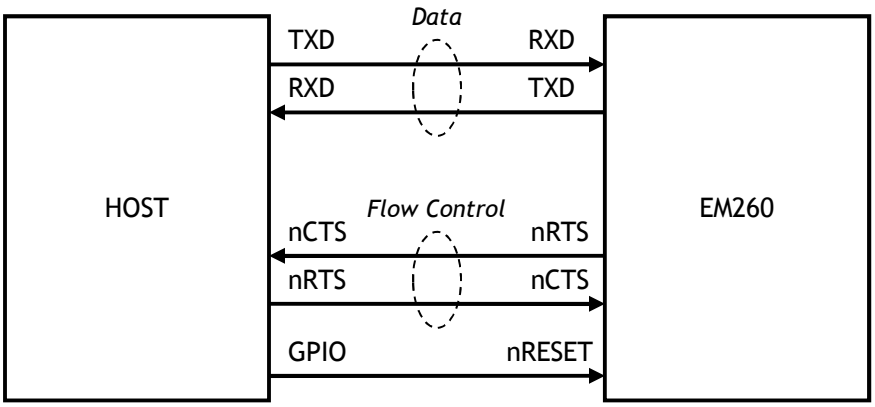


Figure 13. UART Interface Signals

The UART hardware interface uses the following EM260 signals:

- Serial data: TXD and RXD

  The ASH protocol sends data in both directions, so both TXD and RXD signals are required. An external pull-up resistor should be connected to TXD to avoid data glitches while the EM260 is resetting.

- Flow control: nRTS and nCTS (optional)

  ASH uses hardware handshaking for flow control: nRTS enables transmission from the host to the EM260, and nCTS enables EM260 transmissions to the host. If the host serial port cannot support RTS/CTS, XON/XOFF flow control may be used instead. But, XON/XOFF will deliver slightly lower performance.

  When using hardware flow control, the EM260's nRTS must be able to control host serial output. However, in many gateway systems, the host will not need to throttle transmission by the EM260. In those systems nCTS may be left unconnected since it has an internal pull-down and will be continuously asserted.

- Reset control: nRESET

  The host must be able to reset the EM260 to run the ASH protocol. The best way to do this is to use a host output connected to nRESET. If this is not feasible, the host can send a special ASH frame that requests the EM260 to reboot, but this method is less reliable than asserting nRESET and is not recommended for normal use.

The UART signals follow the usual conventions:

- When idle, serial data is high (marking)
- The start bit is low (spacing), and the stop bit is high (marking)
- Data bits are sent least-significant bit first, with positive (non-inverting) logic
- The flow control signals are asserted low

Note that commonly used EIA transceivers invert these logic levels.

Silicon Labs supplies the UART Gateway protocol software in two versions: one uses RTCS/CTS flow control and the other uses XON/XOFF. The UART is set up as follows for these versions:

- 115,200 bps for the RTS/CTS version
- 57,600 bps for the XON/XOFF version
- No parity bit
- 8 data bits
- 1 stop bit

The ASH protocol has been tuned for optimal operation with the two configurations listed here. These configurations can be changed through manufacturing tokens, but doing so may result in a degradation of performance. To learn how to change the configuration, contact customer support at www.silabs.com/zigbee-support.

7 SIF Module Programming and Debug Interface

SIF is a synchronous serial interface developed by Cambridge Consultants Ltd. It is the primary programming and debug interface of the EM260. The SIF module allows external devices to read and write memory-mapped registers in real-time without changing the functionality or timing of the XAP2b core. See document AN696, *PCB Design with an EM260,* for the PCB-level design details regarding the implementation of the SIF interface.

The EM260 pins involved in the SIF Interface:

nSIF_LOAD

SIF_CLK

SIF_MOSI

SIF_MISO

- nRESET

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

The SIF interface provides the following:

PCB production test interface via Virtual UART and a Debug Adapter (ISA)

Programming and debug interface during EmberZNet application development

In order to achieve the deep sleep currents specified in Table 5, a pull-down resistor must be connected to the SIF_MOSI pin. In addition, Silicon Labs recommends a pull-up resistor to be placed on the nSIF_LOAD pin in order to prevent noise from coupling onto the signal. Both of these recommendations are documented within the EM260 Reference designs.

When developing application-specific manufacturing test procedures, Silicon Labs recommends the designer refer to document AN700, *Manufacturing Test Guidelines*. This document provides more detail regarding importance of designing the proper SIF interface as well as timing of the SIF.

8 Typical Application

Figure 14 illustrates the typical application circuit for the EM260 using the SPI Protocol. This figure does not contain all decoupling capacitance required by the EM260. The Balun provides the impedance transformation from the antenna to the EM260 for both TX and RX modes. The harmonic filter provides additional suppression of the second harmonic, which increases the margin over the FCC limit. The 24MHz crystal with loading capacitors is required and provides the high frequency source for the EM260. The RC debounce filter (R4 and C7) is suggested to improve the noise immunity of the nRESET logic (Pin 11).

The SIF (nSIF_LOAD, SIF_MOSI, SIF_MISO, and SIF_CLK), Packet Trace (PTI_EN and PTI_DATA), and SDBG signals should be brought out test points or, if space permits to a 10-pin, dual row, 0.05-inch pitch header footprint. With a header populated, a direct connection to the Debug Adapter is possible which enhances the debug capability of the EM260. For more information, refer to the *EM260 Reference Design*.

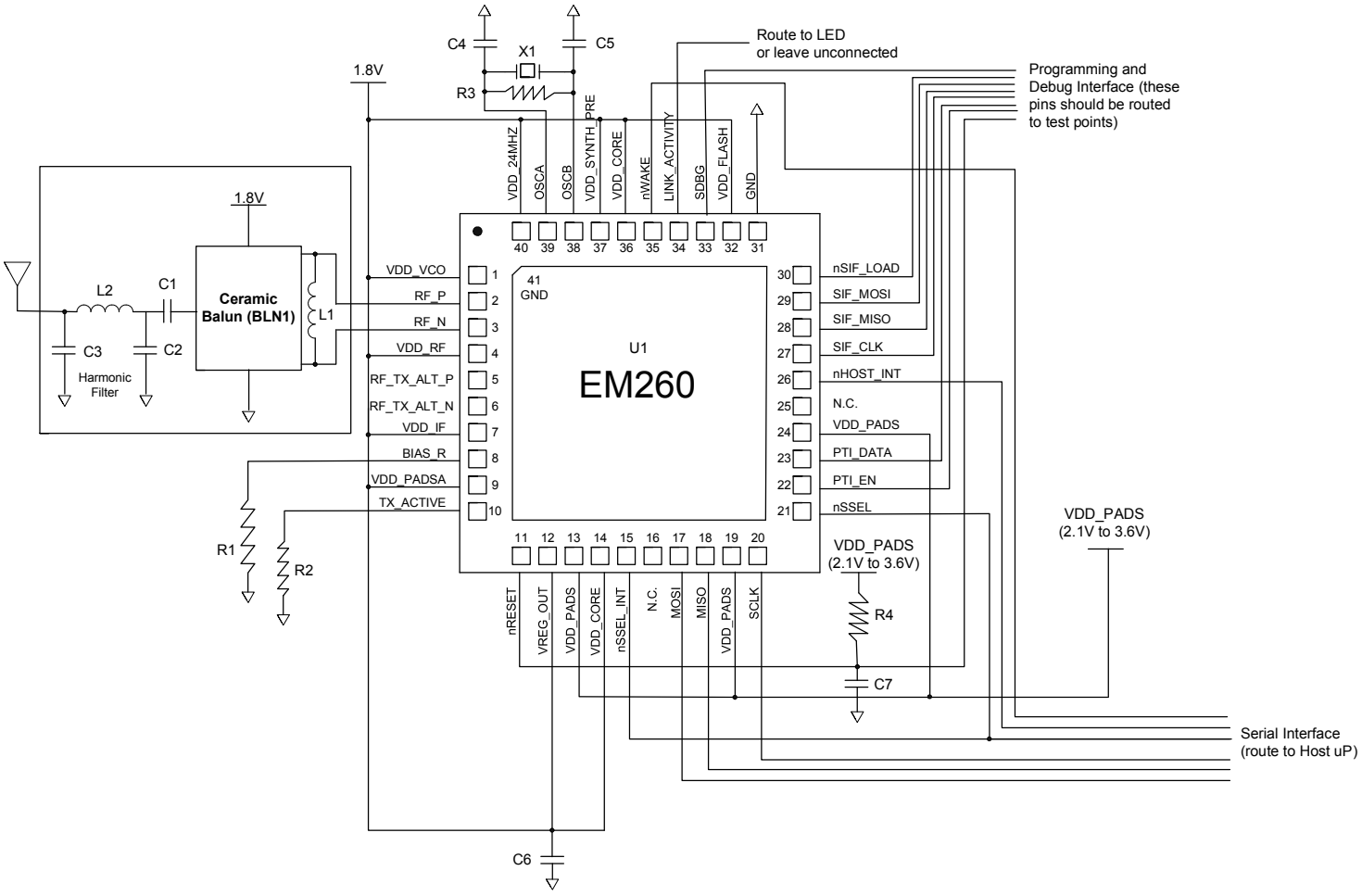


Figure 14. Typical Application Circuit for SPI Protocol

Table 17 contains the Bill of Materials for the application circuit shown in Figure 14.

Table 17. Bill of Materials

Item	Quantity	Reference	Description	Manufacturer/Part No.
1	1	C2	CAPACITOR, 5PF, 50V, NPO, 0402	<not specified>
2	2	C1,C3	CAPACITOR, 0.5PF, 50V, NPO, 0402	<not specified>
3	4	C4,C5	CAPACITOR, 27PF, 50V, NPO, 0402	<not specified>
4	1	C6	CAPACITOR, 10UF, 10V, TANTALUM, 3216 (SIZE A)	<not specified>
5	1	C7	CAPACITOR, 10PF, 5V, NPO, 0402	<not specified>
6	1	L1	INDUCTOR, 2.7NH, +/- 5%, 0603, MULTILAYER	MURATA LQG18HN2N7
7	2	L2	INDUCTOR, 3.3NH, +/- 5%, 0603, MULTILAYER	MURATA LQG18HN3N3
8	1	R1	RESISTOR, 169 KOHM, 1%, 0402	<not specified>
9	1	R2	RESISTOR, 100 KOHM, 5% O402	<not specified>
10	1	R3	RESISTOR, 3.3 KOHM, 5% 0402	<not specified>
11	1	R4	RESISTOR, 10 KOHM, 5%, 0402	<not specified>
12	1	U1	EM260 SINGLE-CHIP ZIGBEE/802.15.4 SOLUTION	SILICON LABS EM260
13	1	X1	CRYSTAL, 24.000MHZ, +/- 10PPM TOLERANCE, +/- 25PPM STABILITY, 18PF, - 40 TO + 85C	ILSI ILCX08-JG5F18-24.000MHZ
14	1	BLN1	BALUN, CERAMIC	TDK HHM1521

9 Mechanical Details

The EM260 package is a plastic 40-pin QFN that is 6 mm x 6 mm x 0.9 mm. Figure 15 and Figure 16 illustrate the Malaysia (SCM) and Taiwan (AESCL) package mechanical drawings, respectively.

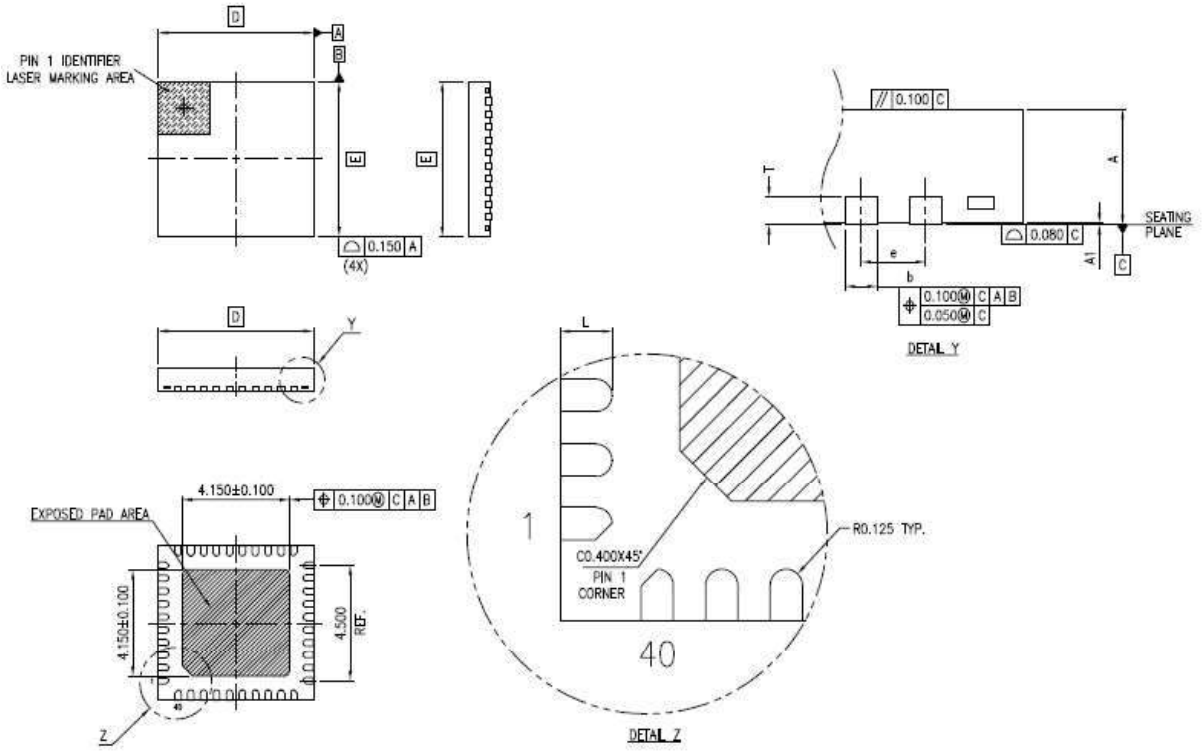
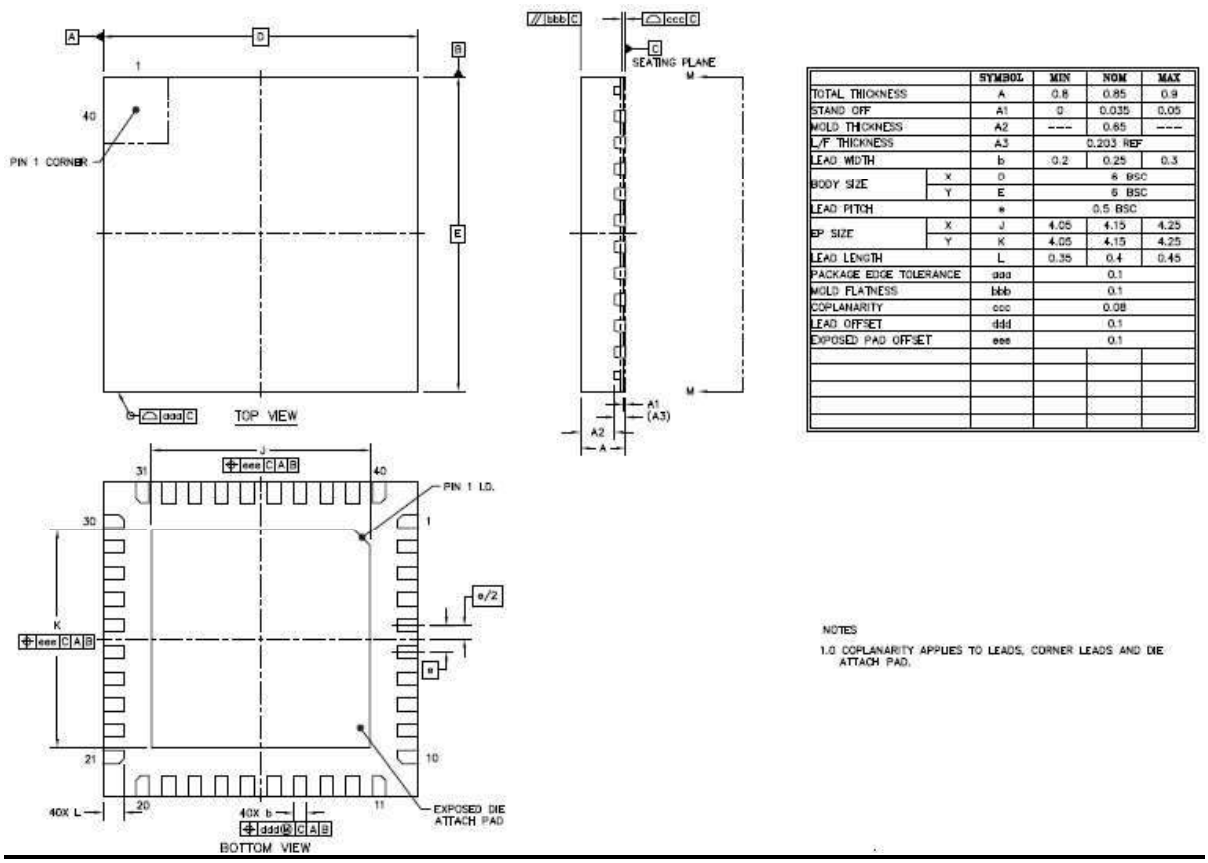


Figure 15: EM260 Malaysia (SCM) Package Drawing



		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	0.8	0.85	0.9
STAND OFF		A1	0	0.035	0.05
MOLD THICKNESS		A2	---	0.65	---
L/F THICKNESS		A3		0.203 REF	
LEAD WIDTH		b	0.2	0.25	0.3
BODY SIZE	X	D		6 BSC	
	Y	E		6 BSC	
LEAD PITCH		e		0.5 BSC	
EP SIZE	X	J	4.05	4.15	4.25
	Y	K	4.05	4.15	4.25
LEAD LENGTH		L	0.35	0.4	0.45
PACKAGE EDGE TOLERANCE		aaa		0.1	
MOLD FLATNESS		bbb		0.1	
COPLANARITY		ccc		0.08	
LEAD OFFSET		ddd		0.1	
EXPOSED PAD OFFSET		eee		0.1	

Figure 16. EM260 Taiwan (AESCL) Package Drawing

Table 18. Package Dimensions

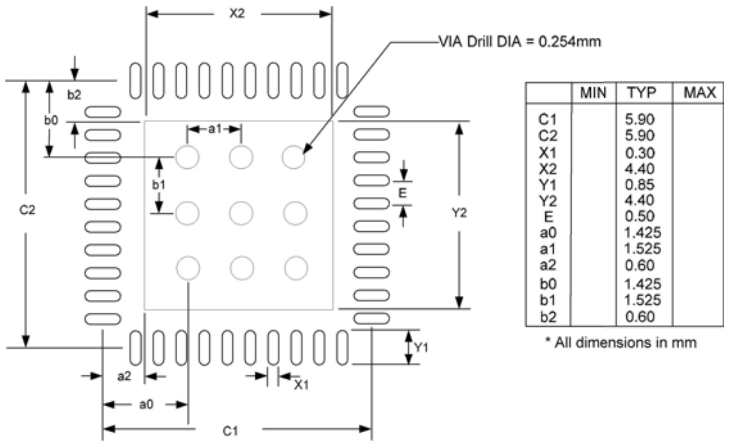
Package type	SCM 40QFN 6x6			ASECL 40QFN 6x6		
Dimensions	Minimum (mm)	Maximum (mm)	Nominal (mm)	Minimum (mm)	Maximum (mm)	Nominal (mm)
Package thickness	0.80	1.00	0.90	0.80	0.90	0.85
Lead Length	0.30	0.50	0.40	0.35	0.45	0.40

10 QFN40 Footprint

Figure 17 demonstrates the IPC-7351 recommended PCB footprint for the EM260 (QFN50P600x600x90-41). A ground pad in the bottom center of the package forms a 41st pin.

A 3 x 3 array of non-thermal vias should connect the EM260 decal center shown in Figure 17 to a PCB ground plane. To properly solder the EM260 to the footprint, the Paste Mask layer (shown in Figure 18) should have a 3 x 3 array of circular openings at 0.076mm diameter spaced approximately 1.525mm apart (center to center). The solder mask (shown in Figure 19) has the same dimensions as the copper pour. This will force an evenly distributed solder flow and coplanar attachment to the PCB.

For more information on the package footprint, please refer to the *EM260 Reference Design*.



	MIN	TYP	MAX
C1		5.90	
C2		5.90	
X1		0.30	
X2		4.40	
Y1		0.85	
Y2		4.40	
E		0.50	
a0		1.425	
a1		1.525	
a2		0.60	
b0		1.425	
b1		1.525	
b2		0.60	

* All dimensions in mm

Figure 17. PCB Footprint for the EM260

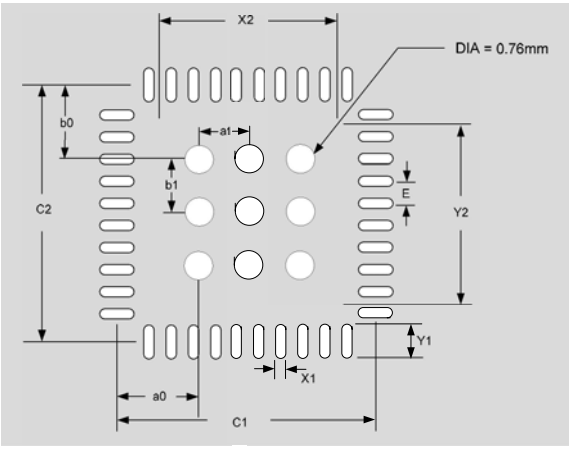


Figure 18. Paste Mask Dimensions

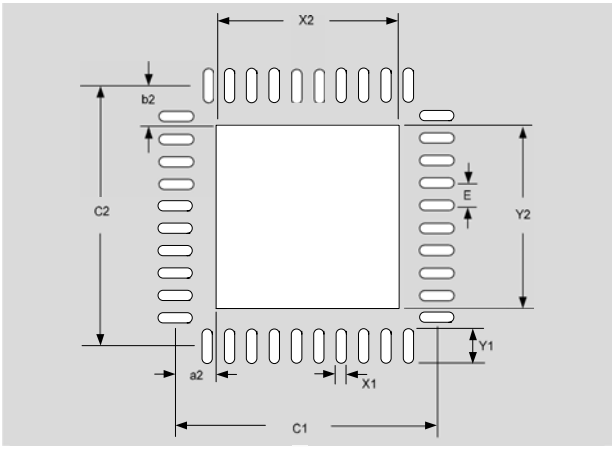


Figure 19. Solder Mask Dimensions

11 Part Marking

The EM260 part marking is shown in Figure 20 and Figure 21. The circle in the top left corner indicates Pin 1. Pins are numbered counter-clockwise from Pin 1, 10 pins per package edge.

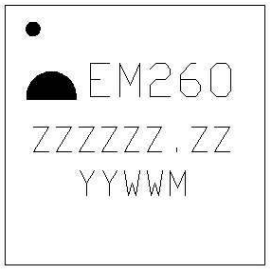


Figure 20. Stats Chippac Malaysia Part marking for EM260

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

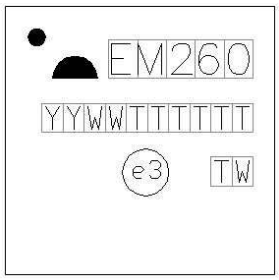


Figure 21. ASE Chung Li Part marking for EM260

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

12 Ordering Information

Use the following part numbers to order the EM260:

- EM260-RTR Reel, RoHS – contains 3500 units / reel

EM260-RTR Reel conforms to EIA specification 481.

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

13 Shipping Box Label

Silicon Labs includes the following information on each tape and reel box label (EM260-RTR):

Package

Device type

Quantity (Bar coded)

Box ID (Bar coded)

Lot Number (Bar coded)

Date Code (Bar coded)

Figure 22 depicts the label position on the box. As shown in Figure 22, there can be up to two date codes in a single tape and reel.

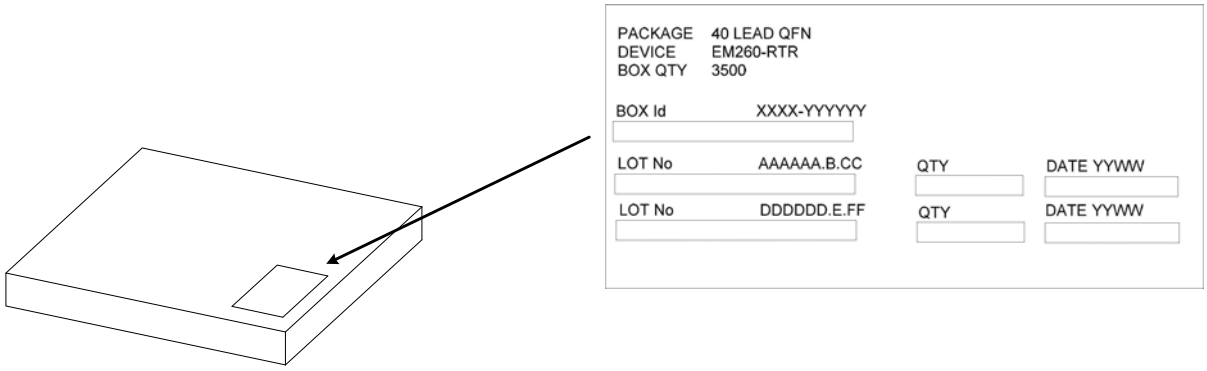


Figure 22. Shipping Label

14 Abbreviations and Acronyms

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

Acronym/Abbreviation	Meaning
PA	Power Amplifier
PER	Packet Error Rate
PHY	Physical Layer
PLL	Phase-Locked Loop
POR	Power-On-Reset
PSD	Power Spectral Density
PSRR	Power Supply Rejection Ratio
PTI	Packet Trace Interface
PWM	Pulse Width Modulation
RoHS	Restriction of Hazardous Substances
RSSI	Receive Signal Strength Indicator
RTS	Request To Send
RXD	Received Data
SFD	Start Frame Delimiter
SIF	Serial Interface
SPI	Serial Peripheral Interface
TXD	Transmitted Data
UART	Universal Asynchronous Receiver/Transmitter
VCO	Voltage Controlled Oscillator
VDD	Voltage Supply

15 References

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

16 Revision History

Revision	Location	Description of Change
1.1	Chapter 9 and Chapter 12 updated.	SCM package updated and AESCL package added.
M	Datasheet	Update cross references to Silicon Labs documentation.
L	Chapter 5	Updates to Wait and Wake Handshake times.
	Chapter 13	Updated Figure 23.
K	Datasheet	Rebranding to Silicon Labs.
J	Chapter 9 (new Figure 17) and Chapter 12	Content additions for Malaysian manufacture.
	Table 7. Receive Characteristics and Table 8. Transmit Characteristics	Changed notes above tables to reflect collection using a Ceramic Balun.
	Figure 18. PCB Footprint for the EM260	Corrected X2 and Y2 dimensions.
I	Section 5.2.6, Waking the EM260 from Sleep	Added new text about when to assert nWAKE.
	Chapter 15, References	Update references.
H	Figure 23. EM260 Tray Information #1	Updated figure per PCN 121-1009-000.
	Figure 24. EM260 Tray Information #2	Updated figure per PCN 121-1009-000.
G	Figure 13. UART Interface Signals	Corrected the direction of the nRTS to nCTS signals.
	Table 15. SPI Commands & Reponses	Updated table response column.
	Figure 16. “M” = C (Chengdu) Package Drawing	Updated figure.
	Figure 22. Detailed Tape and Reel Dimensions	Updated figure.
F	Table 1. Pin Descriptions	Revised descriptions for the following pins: 1, 4, 7, 9, 14, 32, 36, 37, 40
	Figure 16. “M” = C (Chengdu) Package Drawing	Added new figure with mechanical drawing details for the China assembled product as described in PCN 121-1005-000.
E	Table 16. Byte Values Used as Error Codes	Corrected error code 0x01 error description.
	Chapter 6, UART Gateway Protocol	Deleted note at end of chapter.
	Figure 16. PCB Footprint for the EM260	Corrected via dimensions.
	Figure 17. Paste Mask Dimensions	Corrected via dimensions.
	Figure 18. Solder Mask Dimensions	Corrected via dimensions.
D	Section 4.1.1, RX Baseband	Updated AGC information.
	Table 5. DC Characteristics	Added nRESET active current parameter.

Revision	Location	Description of Change
	Figure 17. Paste Mask Dimensions	Added new figure.
	Figure 18. Solder Mask Dimensions	Added new figure.
	Chapter 11, IR Temperature Profile	Added new chapter and figures.
C	Chapter, 6 UART Gateway Protocol	Added new text about XON/XOFF flow control.
	Chapter 16, References	Added new chapter.
B	Section 4.5, XAP2b Microprocessor	Added new text about Application Note 120-3021-000.
	Figure 24, Shipping Label	Added new figure.
A	N/A	First release.