



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

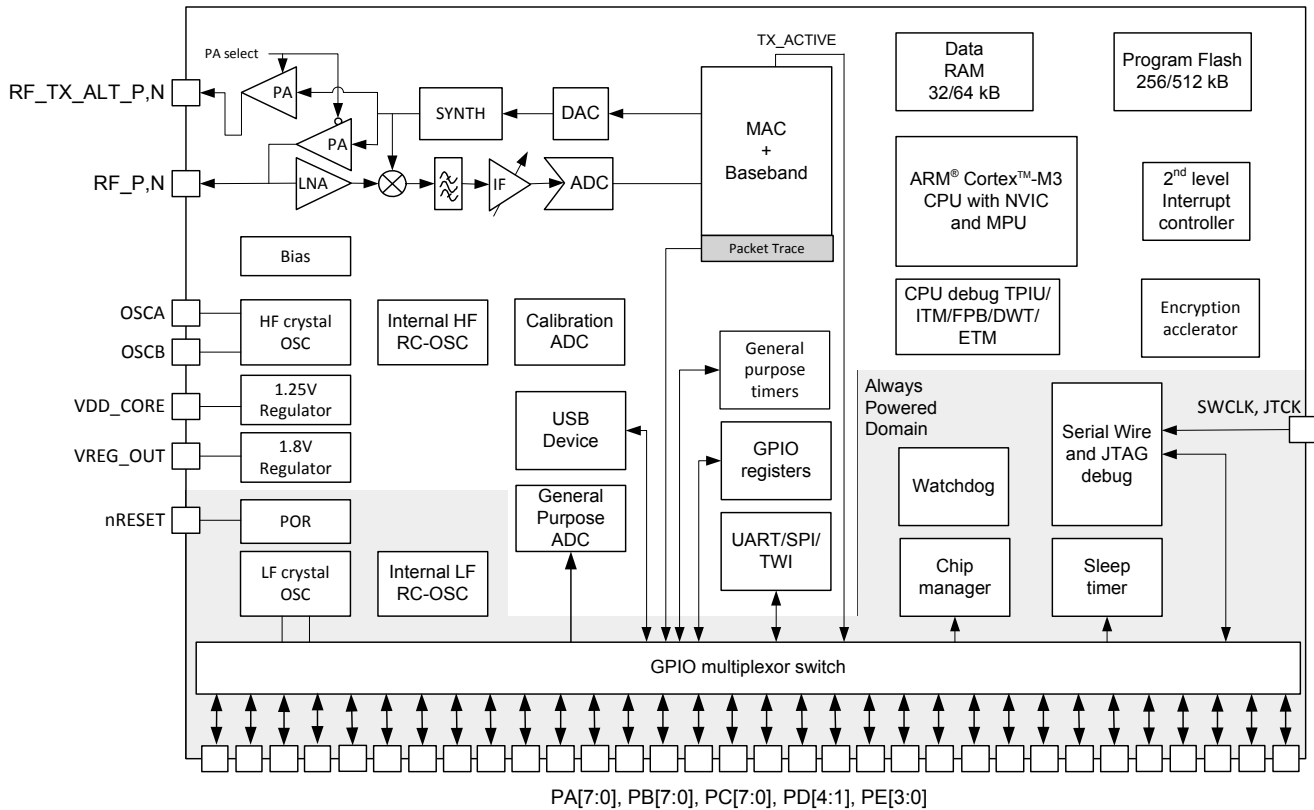
Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



# Ember® EM359x Reference Manual

This reference manual accompanies several documents to provide the complete description of Silicon Labs EM359x devices. In the event that the device data sheet and this document contain conflicting information, the device data sheet should be considered the authoritative source.



---

## Table of Contents

<b>1. Related Documents and Conventions</b> .....	<b>6</b>
1.1. Related Documents .....	6
1.1.1. Ember EM359x Data Sheet .....	6
1.1.2. ZigBee Specification .....	6
1.1.3. ZigBee PRO Stack Profile .....	6
1.1.4. ZigBee Stack Profile .....	6
1.1.5. Thread Specification .....	6
1.1.6. Bluetooth Core Specification .....	6
1.1.7. IEEE 802.15.4 .....	6
1.1.8. IEEE 802.11g .....	6
1.1.9. USB 2.0 Specification .....	6
1.1.10. ARM® Cortex™-M3 Reference Manual .....	6
1.2. Conventions .....	7
<b>2. ARM® Cortex™-M3 and Memory Modules</b> .....	<b>10</b>
2.1. ARM® Cortex™-M3 Microprocessor .....	10
2.2. Embedded Memory .....	10
2.2.1. Flash Memory .....	12
2.2.2. RAM .....	16
2.2.3. Registers .....	17
2.3. Memory Protection Unit .....	17
<b>3. Interrupt System</b> .....	<b>18</b>
3.1. Nested Vectored Interrupt Controller (NVIC) .....	18
3.2. Event Manager .....	20
3.3. Non-Maskable Interrupt (NMI) .....	24
3.4. Faults .....	24
3.5. Registers .....	25
<b>4. Radio Module</b> .....	<b>32</b>
4.1. Receive (RX) Path .....	32
4.1.1. RX Baseband .....	32
4.1.2. RSSI and CCA .....	33
4.2. Transmit (TX) Path .....	33
4.2.1. TX Baseband .....	33
4.2.2. TX_ACTIVE and nTX_ACTIVE Signals .....	33
4.3. Calibration .....	33
4.4. Integrated MAC Module .....	34
4.5. Packet Trace Interface (PTI) .....	34
4.6. Random Number Generator .....	34
<b>5. System Modules</b> .....	<b>35</b>
5.1. Power Domains .....	36
5.1.1. Internally Regulated Power .....	36
5.1.2. Externally Regulated Power .....	36
5.2. Resets .....	37
5.2.1. Reset Sources .....	37
5.2.2. Reset Recording .....	39

---

5.2.3. Reset Generation Module .....	39
5.3. Clocks.....	40
5.3.1. High-Frequency Internal RC Oscillator (OSCHF) .....	42
5.3.2. High-Frequency Crystal Oscillator (OSC24M).....	42
5.3.3. Low-Frequency Internal RC Oscillator (OSCRC) .....	43
5.3.4. Low-Frequency Crystal Oscillator (OSC32K) .....	44
5.3.5. Clock Switching .....	44
5.4. System Timers .....	45
5.4.1. Watchdog Timer .....	45
5.4.2. Sleep Timer .....	45
5.4.3. Event Timer .....	45
5.5. Power Management .....	46
5.5.1. Wake Sources .....	46
5.5.2. Basic Sleep Modes .....	47
5.5.3. Further Options for Deep Sleep.....	48
5.5.4. RAM Retention in Deep Sleep.....	48
5.5.5. Use of Debugger with Sleep Modes .....	48
5.5.6. Registers.....	49
5.6. Security Accelerator .....	50
<b>6. Integrated Voltage Regulator.....</b>	<b>51</b>
<b>7. GPIO (General Purpose Input/Output) .....</b>	<b>53</b>
7.1. GPIO Ports .....	54
7.2. Configuration .....	54
7.3. Forced Functions.....	55
7.4. Reset.....	56
7.5. Boot Configuration.....	56
7.6. GPIO Modes.....	57
7.6.1. Analog Mode.....	57
7.6.2. Input Mode.....	58
7.6.3. SWDIO Mode.....	58
7.6.4. Output Mode .....	58
7.6.5. Alternate Output Mode.....	58
7.6.6. Alternate Output SPI Slave MISO Mode.....	59
7.7. Wake Monitoring .....	59
7.8. External Interrupts .....	59
7.9. Debug Control and Status .....	60
7.10. GPIO Signal Assignment Summary .....	61
7.11. Registers.....	63
<b>8. Serial Controllers .....</b>	<b>77</b>
8.1. Overview .....	77
8.2. Configuration .....	78
8.2.1. Registers.....	80
8.3. SPI—Master Mode .....	84
8.3.1. GPIO Usage .....	84
8.3.2. Set Up and Configuration .....	84
8.3.3. Operation.....	85

---

8.3.4. Interrupts.....	86
8.3.5. Registers.....	87
8.4. SPI—Slave Mode .....	92
8.4.1. GPIO Usage .....	92
8.4.2. Set Up and Configuration .....	93
8.4.3. Operation.....	94
8.4.4. DMA .....	94
8.4.5. Interrupts.....	95
8.4.6. Registers.....	95
8.5. TWI—Two Wire Serial Interfaces .....	95
8.5.1. GPIO Usage .....	95
8.5.2. Set Up and Configuration .....	96
8.5.3. Constructing Frames .....	97
8.5.4. Interrupts.....	99
8.5.5. Registers.....	99
8.6. UART—Universal Asynchronous Receiver/Transmitter .....	102
8.6.1. GPIO Usage .....	102
8.6.2. Set Up and Configuration .....	103
8.6.3. FIFOs .....	105
8.6.4. RTS/CTS Flow control.....	105
8.6.5. DMA .....	106
8.6.6. Interrupts.....	106
8.6.7. Registers.....	107
8.7. DMA Channels .....	111
8.7.1. Registers.....	112
<b>9. USB Device.....</b>	<b>130</b>
9.1. Overview .....	130
9.2. Host Drivers.....	130
9.3. Normal Serial COM Port Operation.....	130
9.4. References .....	130
9.5. GPIO Usage and USB Pin Assignments.....	131
9.6. Application Schematics .....	131
9.7. Endpoints .....	132
9.8. Buffers and DMA .....	133
9.9. Standard Commands .....	133
9.10. Set Up and Configuration.....	134
9.11. DMA Usage and Transfers .....	136
9.12. Suspend and Resume .....	136
9.13. Interrupts.....	137
9.14. Registers.....	138
<b>10. General Purpose Timers (TIM1 and TIM2) .....</b>	<b>171</b>
10.1. Introduction .....	171
10.2. GPIO Usage.....	173
10.3. Timer Functional Description .....	173
10.3.1. Time-Base Unit.....	173
10.3.2. Counter Modes .....	174

---

10.3.3.Clock Selection.....	179
10.3.4.Capture/Compare Channels.....	182
10.3.5.Input Capture Mode.....	183
10.3.6.PWM Input Mode.....	184
10.3.7.Forced Output Mode.....	185
10.3.8.Output Compare Mode.....	185
10.3.9.PWM Mode.....	186
10.3.10.One-Pulse Mode.....	189
10.3.11.Encoder Interface Mode.....	190
10.3.12.Timer Input XOR Function.....	192
10.3.13.Timers and External Trigger Synchronization.....	192
10.3.14.Timer Synchronization.....	195
10.3.15.Timer Signal Descriptions.....	199
10.4.Interrupts.....	201
10.5.Registers.....	202
<b>11.ADC (Analog to Digital Converter).....</b>	<b>229</b>
11.1.Setup and Configuration.....	230
11.1.1.GPIO Usage.....	230
11.1.2.Voltage Reference.....	230
11.1.3.Offset/Gain Correction.....	231
11.1.4.DMA.....	231
11.1.5.ADC Configuration Register.....	231
11.2.Interrupts.....	233
11.3.Operation.....	234
11.4.Calibration.....	235
11.5.ADC Key Parameters.....	236
11.6.Registers.....	242
<b>12.Trace Port Interface Unit (TPIU).....</b>	<b>249</b>
<b>13.Instrumentation Trace Macrocell (ITM).....</b>	<b>250</b>
<b>14.Embedded Trace Macrocell (ETM).....</b>	<b>251</b>
<b>15.Data Watchpoint and Trace (DWT).....</b>	<b>252</b>
<b>16.Flash Patch and Breakpoint (FPB).....</b>	<b>253</b>
<b>17.Serial Wire and JTAG (SWJ) Interface.....</b>	<b>254</b>
<b>Appendix A—Register Address Table.....</b>	<b>255</b>
<b>Document Change List.....</b>	<b>266</b>
<b>Contact Information.....</b>	<b>267</b>

---

# 1. Related Documents and Conventions

## 1.1. Related Documents

This reference manual accompanies several documents to provide the complete description of the EM359x devices.

### 1.1.1. Ember EM359x Data Sheet

The Silicon Laboratories Ember EM359x Data Sheet provides the configuration information for the EM359x devices.

### 1.1.2. ZigBee Specification

The core ZigBee specification (Document 053474) defines ZigBee's smart, cost-effective, and energy-efficient mesh network. It can be downloaded from the ZigBee website (111.zigbee.org). ZigBee Alliance membership is required.

### 1.1.3. ZigBee PRO Stack Profile

The ZigBee PRO Stack Profile specification (Document 074855) is optimized for low power consumption and to support large networks with thousands of devices. It can be downloaded from the ZigBee website (111.zigbee.org). ZigBee Alliance membership is required.

### 1.1.4. ZigBee Stack Profile

The ZigBee Stack Profile specification (Document 064321) is designed to support smaller networks with hundreds of devices in a single network. It can be downloaded from the ZigBee website (111.zigbee.org). ZigBee Alliance membership is required.

### 1.1.5. Thread Specification

The Thread specification can be found through the Thread Group website:

[www.threadgroup.org](http://www.threadgroup.org)

### 1.1.6. Bluetooth Core Specification

The Bluetooth specification is the global short-range wireless standard enabling connectivity for a broad range of electronic devices. Version 2.1 + EDR (Enhanced Data Rate) can be found here:

[http://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc\\_id=241363](http://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=241363)

### 1.1.7. IEEE 802.15.4

This standard defines the protocol and compatible interconnection for data communication devices using low data rate, low power, and low complexity, short-range radio frequency (RF) transmissions in a wireless personal area network (WPAN). It can be found here:

IEEE 802.15.4-2003 (<http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf>)

IEEE 802-15.4-2006 (<http://standards.ieee.org/findstds/standard/802.15.4-2006.html>)

### 1.1.8. IEEE 802.11g

This version provides changes and additions to support the further higher data rate extension for operation in the 2.4 GHz band. It can be found here:

<http://standards.ieee.org/getieee802/download/802.11g-2003.pdf>

### 1.1.9. USB 2.0 Specification

The Universal Serial Bus Revision 2.0 specification provides the technical details to understand USB requirements and design USB compatible products. The main specification (usb\_20.pdf) is part of the zipfile found here:

[http://www.usb.org/developers/docs/usb\\_20\\_101111.zip](http://www.usb.org/developers/docs/usb_20_101111.zip)

### 1.1.10. ARM<sup>®</sup> Cortex<sup>™</sup>-M3 Reference Manual

ARM-specific features like the Nested Vector Interrupt Controller are described in the ARM<sup>®</sup> Cortex<sup>™</sup>-M3 reference documentation. The online reference manual can be found here:

<http://infocenter.arm.com/help/topic/com.arm.doc.subset.cortexm.m3/index.html#cortexm3>

---

## 1.2. Conventions

Abbreviations and acronyms used in this data sheet are explained in

**Table 1.1. Acronyms and Abbreviations**

<b>Acronym/Abbreviation</b>	<b>Meaning</b>
ACK	Acknowledgement
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
FIB	Fixed Information Block
FIFO	First-in, First-out



**Table 1.1. Acronyms and Abbreviations**

FPB	Flash Patch and Breakpoint
GPIO	General Purpose I/O (pins)
HF	High Frequency
I <sup>2</sup> C	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
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

---

**Table 1.1. Acronyms and Abbreviations**

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
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
USB	Universal Serial Bus
VCO	Voltage Controlled Oscillator

---

## 2. ARM<sup>®</sup> Cortex<sup>™</sup>-M3 and Memory Modules

This chapter discusses the ARM<sup>®</sup> Cortex<sup>™</sup>-M3 Microprocessor, and reviews the EM359x's flash and RAM memory modules as well as the Memory Protection Unit (MPU).

### 2.1. ARM<sup>®</sup> Cortex<sup>™</sup>-M3 Microprocessor

The EM359x integrates the ARM<sup>®</sup> Cortex<sup>™</sup>-M3 microprocessor, revision r1p1, developed by ARM Ltd., making the EM359x a true System-on-Chip solution. The ARM<sup>®</sup> Cortex<sup>™</sup>-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<sup>®</sup> Cortex<sup>™</sup>-M3 allows unaligned word and half-word data accesses to support efficiently-packed data structures.

The ARM<sup>®</sup> Cortex<sup>™</sup>-M3 clock speed is configurable to 6 MHz, 12 MHz, or 24 MHz. For normal operation 24 MHz is preferred over 12 MHz due to improved performance for all applications and improved duty cycling for applications using sleep modes. The 6 MHz operation can only be used when radio operations are not required since the radio requires an accurate 12 MHz clock.

The ARM<sup>®</sup> Cortex<sup>™</sup>-M3 in the EM359x 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..

### 2.2. Embedded Memory

Figure 2.1 shows the EM359x ARM<sup>®</sup> Cortex<sup>™</sup>-M3 memory map.

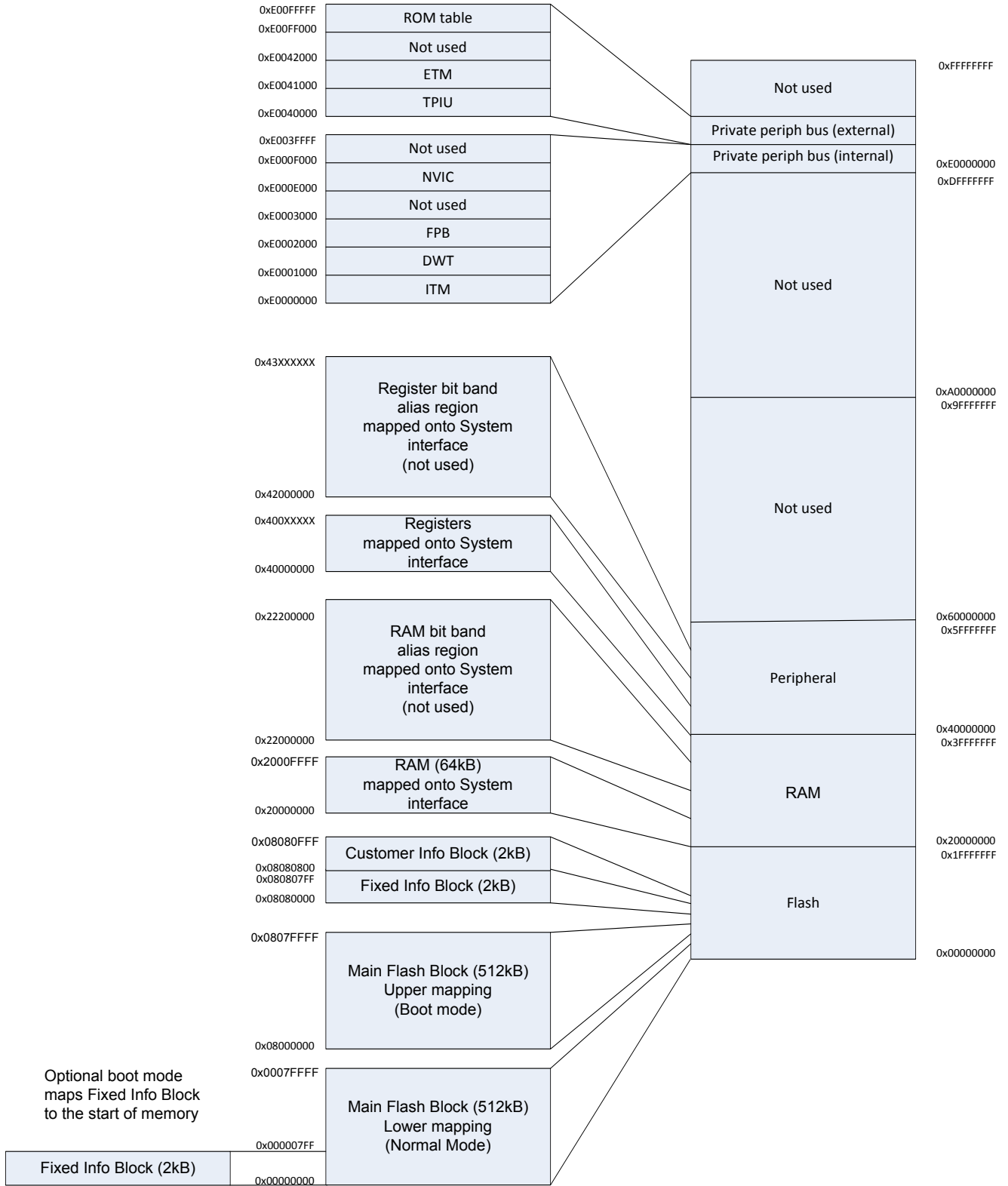


Figure 2.1. EM359x ARM® Cortex™-M3 Memory Map

---

## 2.2.1. Flash Memory

### 2.2.1.1. Flash Overview

The EM359x provides a total of 256 or 512 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 EM359x has either 128 or 256 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 Silicon Labs software for over-the-air loading or serial link loading.

#### 2.2.1.2. 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 section “7.5. Boot Configuration”, in Chapter 7, GPIO). 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 Silicon Labs proprietary Simulated EEPROM system..

#### 2.2.1.3. 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 0x08080000. 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 access. In FIB monitor mode (see section “7.5. Boot Configuration” in Chapter 7, GPIO), the start of the FIB is mapped to both address 0x00000000 and address 0x08080000 so the monitor may be executed out of reset

#### 2.2.1.4. Customer Information Block

The 2048 byte CIB can be used to store customer data. The start of the CIB is mapped to address 0x08080800. 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 2.1.

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 2.1. Option Byte Storage**

Address	bits [15:8]	bits [7:0]	Notes
0x08080800	Inverse Option Byte 0	Option Byte 0	Configures flash read protection
0x08080802	Inverse Option Byte 1	Option Byte 1	Reserved
0x08080804	Inverse Option Byte 2	Option Byte 2	Available for customer use <sup>1</sup>
0x08080806	Inverse Option Byte 3	Option Byte 3	Available for customer use <sup>1</sup>
0x08080808	Inverse Option Byte 4	Option Byte 4	Configures flash write protection
0x0808080A	Inverse Option Byte 5	Option byte 5	Configures flash write protection
0x0808080C	Inverse Option Byte 6	Option Byte 6	Configures flash write protection
0x0808080E	Inverse Option Byte 7	Option Byte 7	Configures flash write protection

**Note:**

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 2 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 EM359x has up to 32 regions and therefore option bytes 4, 5, 6, and 7 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 EM359x, 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.

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. To obtain fully protected flash, both read protection and write protection should be active.

**Table 2.2. 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 – 0x08003FFF
	bit [1]	Write protection of address range 0x08004000 – 0x08007FFF
	bit [2]	Write protection of address range 0x08008000 – 0x0800BFFF
	bit [3]	Write protection of address range 0x0800C000 – 0x0800FFFF
	bit [4]	Write protection of address range 0x08010000 – 0x08013FFF
	bit [5]	Write protection of address range 0x08014000 – 0x08017FFF
	bit [6]	Write protection of address range 0x08018000 – 0x0801BFFF
	bit [7]	Write protection of address range 0x0801C000 – 0x0801FFFF
Option Byte 5	bit [0]	Write protection of address range 0x08020000 – 0x08023FFF
	bit [1]	Write protection of address range 0x08024000 – 0x08027FFF
	bit [2]	Write protection of address range 0x08028000 – 0x0802BFFF
	bit [3]	Write protection of address range 0x0802C000 – 0x0802FFFF
	bit [4]	Write protection of address range 0x08030000 – 0x08033FFF
	bit [5]	Write protection of address range 0x08034000 – 0x08037FFF
	bit [6]	Write protection of address range 0x08038000 – 0x0803BFFF
	bit [7]	Write protection of address range 0x0803C000 – 0x0803FFFF
Option Byte 6	bit [0]	Write protection of address range 0x08040000 – 0x08043FFF
	bit [1]	Write protection of address range 0x08044000 – 0x08047FFF
	bit [2]	Write protection of address range 0x08048000 – 0x0804BFFF
	bit [3]	Write protection of address range 0x0804C000 – 0x0804FFFF
	bit [4]	Write protection of address range 0x08050000 – 0x08053FFF
	bit [5]	Write protection of address range 0x08054000 – 0x08057FFF
	bit [6]	Write protection of address range 0x08058000 – 0x0805BFFF
	bit [7]	Write protection of address range 0x0805C000 – 0x0805FFFF

---

**Table 2.2. Option Byte Write Protection Bit Map**

Option Byte 7	bit [0]	Write protection of address range 0x08060000 – 0x08063FFF
	bit [1]	Write protection of address range 0x08064000 – 0x08067FFF
	bit [2]	Write protection of address range 0x08068000 – 0x0806BFFF
	bit [3]	Write protection of address range 0x0806C000 – 0x0806FFFF
	bit [4]	Write protection of address range 0x08070000 – 0x08073FFF
	bit [5]	Write protection of address range 0x08074000 – 0x08077FFF
	bit [6]	Write protection of address range 0x08078000 – 0x0807BFFF
	bit [7]	Write protection of address range 0x0807C000 – 0x0807FFFF



---

### 2.2.1.5. Simulated EEPROM

Silicon Labs 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.

## 2.2.2. RAM

### 2.2.2.1. RAM Overview

The EM359x has 32 or 64 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 one wait state. This is handled by hardware transparent to the user application with no configuration required.

### 2.2.2.2. 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 (802.15.4-2006 MAC), general purpose ADC, USB device controller and the two serial controllers. In the case of the serial controllers, 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 Chapter 8, Section 8.7 and Chapter 11, Section 11.1.4 for a description of how to configure the serial controllers and ADC for DMA operation. The DMA channels do not use AHB system bus bandwidth as they access the RAM directly.

The EM359x 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. USB Device Controller (where applicable)
2. General Purpose ADC
3. Serial Controller 2 Receive
4. Serial Controller 2 Transmit
5. Serial Controller 4 Receive
6. Serial Controller 4 Transmit
7. MAC
8. Serial Controller 1 Receive
9. Serial Controller 1 Transmit
10. Serial Controller 3 Receive
11. Serial Controller 3 Transmit

### 2.2.2.3. RAM Memory Protection

The EM359x integrates a memory protection mechanism through the ARM® Cortex™-M3 Memory Protection Unit (MPU) described in the Memory Protection Unit section. The MPU may be used to protect any area of memory. MPU configuration is normally handled by Silicon Labs software.

---

### 2.2.3. Registers

“ Appendix A—Register Address Table” provides a short description of all application-accessible registers within the EM359x. 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.

### 2.3. Memory Protection Unit

The EM359x 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.

Silicon Labs 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.

---

### 3. Interrupt System

The EM359x'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.

The Nested Vectored Interrupt Controller (NVIC) section provides a description of the NVIC and an overview of the exception table (ARM nomenclature refers to interrupts as exceptions). The Event Manager section 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.

#### 3.1. 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 18 individually vectored peripheral interrupts specific to the EM359x.

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. Equation 3.1 lists the entire exception table. Exceptions 0 (stack pointer) through 15 (SysTick) are part of the standard ARM® Cortex™-M3 NVIC, while exceptions 16 (Timer 1) through 35 (USB, where applicable) are the peripheral interrupts specific to the EM359x peripherals. The peripheral interrupts are listed in greater detail in Table 3.2.

**Table 3.1. 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.

**Table 3.1. NVIC Exception Table**

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.
Timer 1	16	Timer 1 peripheral interrupt.
Timer 2	17	Timer 2 peripheral interrupt.
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.
Serial Controller 2	22	Serial Controller 2 peripheral interrupt.
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.
ADC	27	ADC peripheral interrupt.
IRQA	28	IRQA peripheral interrupt.
IRQB	29	IRQB peripheral interrupt.
IRQC	30	IRQC peripheral interrupt.
IRQD	31	IRQD peripheral interrupt.
Debug	32	Debug peripheral interrupt.
Serial Controller 3	33	Serial Controller 3 peripheral interrupt.
Serial Controller 4	34	Serial Controller 4 peripheral interrupt.
USB	35	USB peripheral interrupt (where applicable).

---

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 IRQB 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 IRQB with priority number 29.

The top-level interrupts are controlled through five ARM<sup>®</sup> Cortex<sup>™</sup>-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.

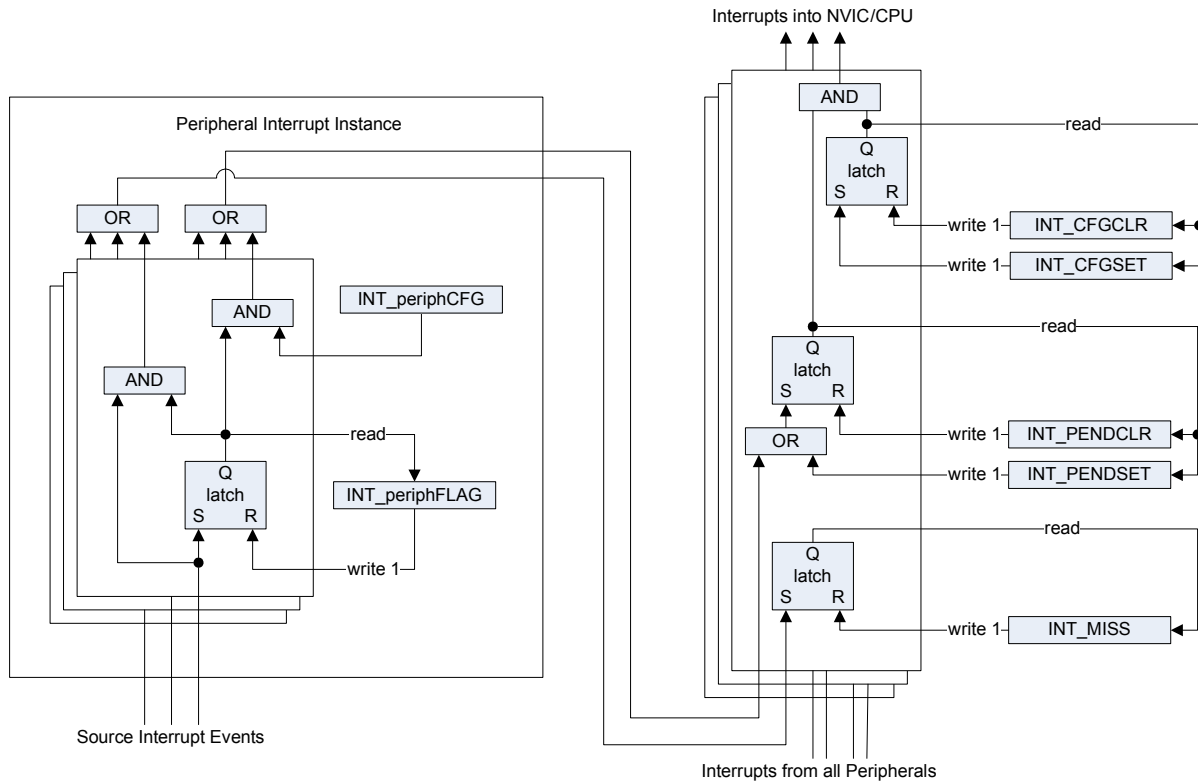
The second-level interrupt registers, which provide control of the second-level Event Manager peripheral interrupts, are described in the Event Manager section.

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

### 3.2. Event Manager

While the standard ARM<sup>®</sup> Cortex<sup>™</sup>-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 “OR’d” 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 reference manual describing each peripheral. Figure 3.1 shows the Peripheral Interrupts Block Diagram.



**Figure 3.1. 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 3.2 on page 22 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 3.2. NVIC and EM Peripheral Interrupt Map**

NVIC Interrupt (Top-Level)		EM Interrupt (Second-Level)			NVIC Interrupt (Top-Level)		EM Interrupt (Second-Level)	
19	INT_USB	INT_USBFLAG Register					9	INT_SCRXULDA
		23	INT_USBWAKEUP				8	INT_SCNAK
		22	INT_USBRESUME				7	INT_SCCDMFIN
		21	INT_USBSUSPEND				6	INT_SCTXFIN
		20	INT_USBRESET				5	INT_SCRXFIN
		19	INT_USBOF				4	INT_SCTXUND
		18	INT_USBNAK				3	INT_SCRXOVF
		17	INT_USBPIPERXOVF				2	INT_SCTXIDLE
		16	INT_USBPIPETXUND				1	INT_SCTXFREE
		15	INT_USBBUFRXOVF				0	INT_SCRXVAL
		14	INT_USBBUFTXUND	17	INT_SC3	INT_SC3FLAG Register		
		13	INT_USBRXVALDEP6			14	INT_SC3PARERR	
		12	INT_USBRXVALDEP5			13	INT_SC3FRMERR	
		11	INT_USBRXVALDEP4			12	INT_SCTXULDB	
		10	INT_USBRXVALDEP3			11	INT_SCTXULDA	
		9	INT_USBRXVALDEP2			10	INT_SCRXULDB	
		8	INT_USBRXVALDEP1			9	INT_SCRXULDA	
		7	INT_USBRXVALDEP0			8	INT_SCNAK	
		6	INT_USBTXACTIVEEP6			7	INT_SCCDMFIN	
		5	INT_USBTXACTIVEEP5			6	INT_SCTXFIN	
		4	INT_USBTXACTIVEEP4			5	INT_SCRXFIN	
		3	INT_USBTXACTIVEEP3			4	INT_SCTXUND	
		2	INT_USBTXACTIVEEP2			3	INT_SCRXOVF	
		1	INT_USBTXACTIVEEP1			2	INT_SCTXIDLE	
		0	INT_USBTXACTIVEEP0			1	INT_SCTXFREE	
						0	INT_SCRXVAL	
18	INT_SC4	INT_SC4FLAG Register						
		12	INT_SCTXULDB	16	INT_DEBUG			
		11	INT_SCTXULDA	15	INT_IRQD			
		10	INT_SCRXULDB	14	INT_IRQC			

**Table 3.2. NVIC and EM Peripheral Interrupt Map**

13	INT_IRQB					10	INT_SCRXULDB
12	INT_IRQA					9	INT_SCRXULDA
11	INT_ADC	INT_ADCFLAG register				8	INT_SCNAK
		4	INT_ADCOVF			7	INT_SCCDMFIN
		3	INT_ADCSAT			6	INT_SCTXFIN
		2	INT_ADCULDFULL			5	INT_SCRXFIN
		1	INT_ADCULDHAF			4	INT_SCTXUND
		0	INT_ADCDATA			3	INT_SCRXOVF
10	INT_MACRX					2	INT_SCTXIDLE
9	INT_MACTX					1	INT_SCTXFREE
8	INT_MACTMR					0	INT_SCRXVAL
7	INT_SEC			4	INT_SLEEPTMR		
6	INT_SC2	INT_SC2FLAG register			3	INT_BB	
		12	INT_SCTXULDB		2	INT_MGMT	
		11	INT_SCTXULDA		1	INT_TMR2	INT_TMR2FLAG register
		10	INT_SCRXULDB			6	INT_TMRTIF
		9	INT_SCRXULDA			4	INT_TMRCC4IF
		8	INT_SCNAK			3	INT_TMRCC3IF
		7	INT_SCCDMFIN			2	INT_TMRCC2IF
		6	INT_SCTXFIN			1	INT_TMRCC1IF
		5	INT_SCRXFIN			0	INT_TMRUIF
		4	INT_SCTXUND		0	INT_TMR1	INT_TMR1FLAG register
		3	INT_SCRXOVF			6	INT_TMRTIF
		2	INT_SCTXIDLE			4	INT_TMRCC4IF
		1	INT_SCTXFREE			3	INT_TMRCC3IF
		0	INT_SCRXVAL			2	INT_TMRCC2IF
5	INT_SC1	INT_SC1FLAG register				1	INT_TMRCC1IF
		14	INT_SC1PARERR			0	INT_TMRUIF
		13	INT_SC1FRMERR				
		12	INT_SCTXULDB				
		11	INT_SCTXULDA				



---

### 3.3. Non-Maskable Interrupt (NMI)

The non-maskable interrupt (NMI) is a special case. Despite being one of the 10 standard ARM® Cortex™-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 EM359x's main clock, SYSCLK, is operating from the 24 MHz crystal and the crystal fails, the EM359x detects the failure and automatically switches to the internal 12 MHz RC clock. When this failure detection and switch has occurred, the EM359x triggers the CLK24M\_FAIL second-level interrupt, which then triggers the NMI.
2. **Watchdog low water mark:** If the EM359x's watchdog is active and the watchdog counter has not been reset for nominally 1.792 seconds, the watchdog triggers the WATCHDOG\_INT second-level interrupt, which then triggers the NMI.

### 3.4. 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® Cortex™-M3 exceptions.

The Bus Fault, though, is derived from EM359x-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 Chapter 2, Section 2.2.2.3).
- **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

### 3.5. Registers

**Register 3.1. INT\_CFGSET: Top-Level Set Interrupts Configuration Register**

<b>Bit</b>	<b>31</b>	<b>30</b>	<b>29</b>	<b>28</b>	<b>27</b>	<b>26</b>	<b>25</b>	<b>24</b>
<b>Name</b>	0	0	0	0	0	0	0	0
<b>Bit</b>	<b>23</b>	<b>22</b>	<b>21</b>	<b>20</b>	<b>19</b>	<b>18</b>	<b>17</b>	<b>16</b>
<b>Name</b>	0	0	0	0	INT_USB	INT_SC4	INT_SC3	INT_DEBUG
<b>Bit</b>	<b>15</b>	<b>14</b>	<b>13</b>	<b>12</b>	<b>11</b>	<b>10</b>	<b>9</b>	<b>8</b>
<b>Name</b>	INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR
<b>Bit</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
<b>Name</b>	INT_SEC	INT_SC2	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	INT_TIM2	INT_TIM1

Address: 0xE000E100; Reset: 0x0

Bit Name	Bit Field	Access	Description
INT_USB	[19]	RW	Write 1 to enable USB interrupt. (Writing 0 has no effect.) (where applicable)
INT_SC4	[18]	RW	Write 1 to enable serial controller 4 interrupt. (Writing 0 has no effect.)
INT_SC3	[17]	RW	Write 1 to enable serial controller 4 interrupt. (Writing 0 has no effect.)
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_IRQB	[13]	RW	Write 1 to enable IRQB interrupt. (Writing 0 has no effect.)
INT_IRQA	[12]	RW	Write 1 to enable IRQA interrupt. (Writing 0 has no effect.)
INT_ADC	[11]	RW	Write 1 to enable ADC 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_SC2	[6]	RW	Write 1 to enable serial controller 2 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.)
INT_TIM2	[1]	RW	Write 1 to enable timer 2 interrupt. (Writing 0 has no effect.)
INT_TIM1	[0]	RW	Write 1 to enable timer 1 interrupt. (Writing 0 has no effect.)