Ember® EM358x Reference Manual

This reference manual accompanies several documents to provide the complete description of Silicon Labs EM358x 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.
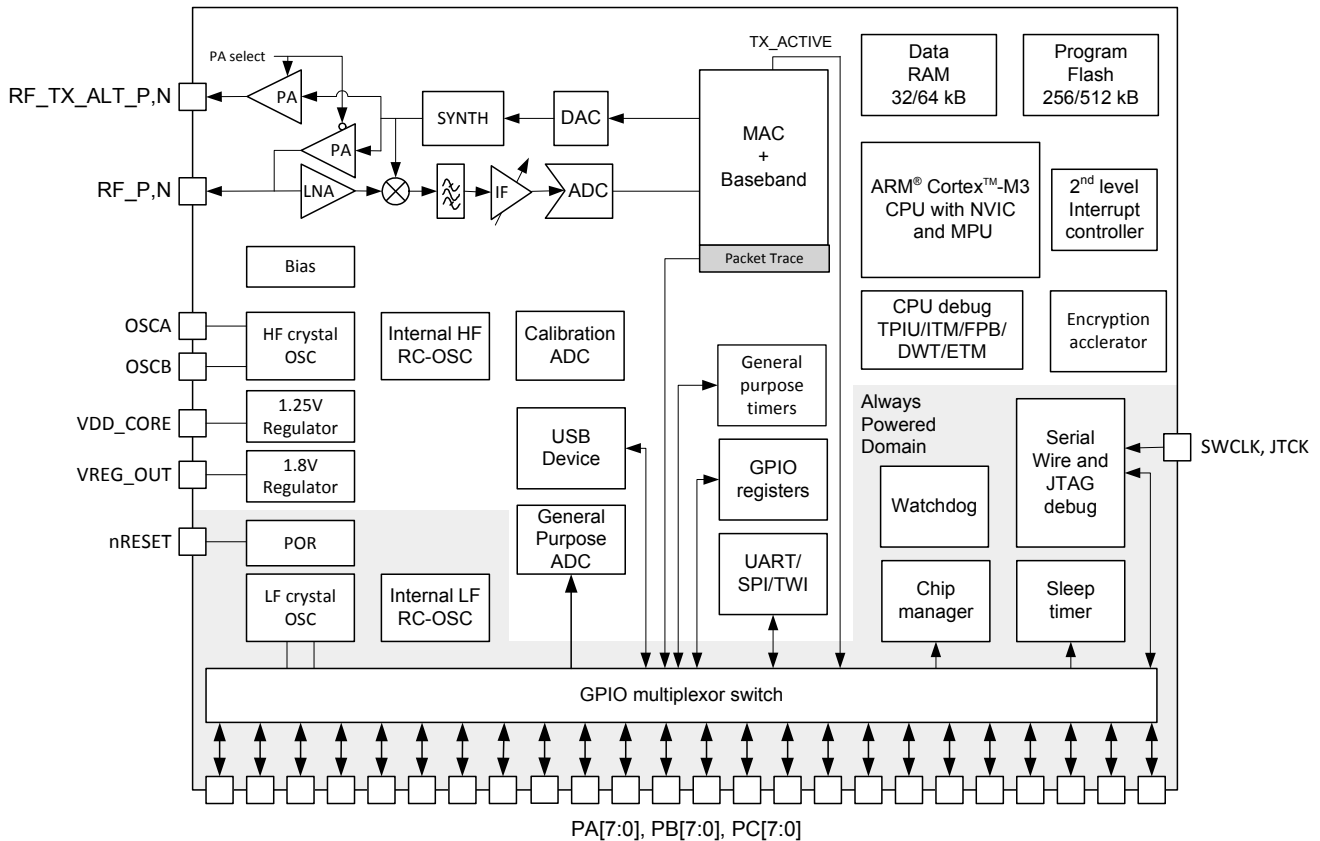
Table of Contents

SILICON LABS

SILICON LABS

1. Related Documents and Conventions

1.1. Related Documents

This reference manual accompanies several documents to provide the complete description of the EM358x devices.

1.1.1. Ember EM358x Data Sheet

The Silicon Laboratories Ember EM358x Data Sheet provides the configuration information for the EM358x 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

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

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

1.1.10. ARM® Cortex™-M3 Reference Manual

ARM-specific features like the Nested Vector Interrupt Controller are described in the ARM® Cortex™-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

SILICON LABS

1.2. Conventions

Abbreviations and acronyms used in this data sheet are explained in

Table 1.1. Acronyms and Abbreviations

Acronym/Abbreviation	Meaning
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^{TM}-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^{2}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

SILICON LABS

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

SILICON LABS

2. ARM® Cortex™-M3 and Memory Modules

This chapter discusses the ARM® Cortex™-M3 Microprocessor, and reviews the EM358x's flash and RAM memory modules as well as the Memory Protection Unit (MPU).

2.1. ARM® Cortex™-M3 Microprocessor

The EM358x integrates the ARM® Cortex™-M3 microprocessor, revision r1p1, developed by ARM Ltd., making the EM358x a true System-on-Chip solution. The ARM® Cortex™-M3 is an advanced 32-bit modified Harvard architecture processor that has separate internal program and data buses, but presents a unified program and data address space to software. The word width is 32 bits for both the program and data sides. The AARM® Cortex™-M3allows unaligned word and half-word data accesses to support efficiently-packed data structures.

The ARM® Cortex™-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® Cortex™-M3 in the EM358x 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 EM358x ARM® Cortex™-M3 memory map.

SILICON LABS

0xE00FFFFF
0xE00FF000
ROM table

Not used
0xE0042000
ETM
0xE0041000
TPIU
0xE0040000

0xE003FFFF
0xE000F000
Not used

NVIC
0xE000E000
Not used
0xE0003000
FPB
0xE0002000
DWT
0xE0001000
ITM
0xE0000000

0xFFFFFFFF
Not used

Private periph bus (external)
Private periph bus (internal)
0xE0000000
0xDFFFFFFF

Not used

0x43XXXXXX
Register bit band
alias region
mapped onto System
interface
(not used)
0x42000000

0xA0000000
0x9FFFFFFF

Not used

0x400XXXXX
Registers
mapped onto System
interface
0x40000000

0x22200000
RAM bit band
alias region
mapped onto System
interface
(not used)
0x22000000

0x60000000
0x5FFFFFFF

Peripheral

0x2000FFFF
RAM (64kB)
mapped onto System
interface
0x20000000

0x40000000
0x3FFFFFFF

0x08080FFF
Customer Info Block (2kB)
0x08080800
0x080807FF
Fixed Info Block (2kB)
0x08080000

RAM

0x0807FFFF

Main Flash Block (512kB)
Upper mapping
(Boot mode)

0x08000000

0x20000000
0x1FFFFFFF

Flash

Optional boot mode
maps Fixed Info Block
to the start of memory

0x0007FFFF

Main Flash Block (512kB)
Lower mapping
(Normal Mode)
0x000007FF

Fixed Info Block (2kB)
0x00000000

0x00000000

Figure 2.1. EM358x ARM® CortexTM-M3 Memory Map

SILICON LABS

2.2.1. Flash Memory

2.2.1.1. Flash Overview

The EM358x provides a total of 256 or 512 kB 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 EM358x 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[1]
0x08080806	Inverse Option Byte 3	Option Byte 3	Available for customer use[1]
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 EM358x 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 EM358x, 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 – 0x08013FFF
	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

SILICON LABS

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

SILICON LABS

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 EM358x has 32 or 64 kB of static RAM on-chip. The start of RAM is mapped to address 0x20000000. Although the ARM® CortexTM-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® CortexTM-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 EM358x 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. MAC
6. Serial Controller 1 Receive
7. Serial Controller 1 Transmit

2.2.2.3. RAM Memory Protection

The EM358x integrates a memory protection mechanism through the ARM® CortexTM-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 EM358x. 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^{TM}-M3, but the standard MPU configuration does not allow access to this alias address range.

2.3. Memory Protection Unit

The EM358x includes the ARM$^{®}$ Cortex^{TM}-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^{TM}-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 EM358x'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 EM358x.

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 EM358x 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.
—	33	Reserved.
—	34	Reserved.
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® CortexTM-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® CortexTM-M3 exceptions, refer to the ARM® CortexTM-M3 Technical Reference Manual and the ARM ARMv7-M Architecture Reference Manual.

3.2. Event Manager

While the standard ARM® CortexTM-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 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			16	INT_DEBUG		
		23	INT_USBWAKEUP		15	INT_IRQD		
		22	INT_USBRESUME		14	INT_IRQC		
		21	INT_USBSUSPEND		13	INT_IRQB		
		20	INT_USBRESET		12	INT_IRQA		
		19	INT_USBOF		11	INT_ADC	INT_ADCFLAG register	
		18	INT_USBNAK				4	INT_ADCOVF
		17	INT_USBPIPERXOVF				3	INT_ADCSAT
		16	INT_USBPIPETXUND				2	INT_ADCULDFULL
		15	INT_USBBUFRXOVF				1	INT_ADCULDHALF
		14	INT_USBBUFTXUND				0	INT_ADCDATA
		13	INT_USBRXVALDEP6		10	INT_MACRX		
		12	INT_USBRXVALDEP5		9	INT_MACTX		
		11	INT_USBRXVALDEP4		8	INT_MACTMR		
		10	INT_USBRXVALDEP3		7	INT_SEC		
		9	INT_USBRXVALDEP2		6	INT_SC2	INT_SC2FLAG register	
		8	INT_USBRXVALDEP1				12	INT_SCTXULDB
		7	INT_USBRXVALDEP0				11	INT_SCTXULDA
		6	INT_USBTXACTIVEEP6				10	INT_SCRXULDB
		5	INT_USBTXACTIVEEP5				9	INT_SCRXULDA
		4	INT_USBTXACTIVEEP4				8	INT_SCNAK
		3	INT_USBTXACTIVEEP3				7	INT_SCCDMFIN
		2	INT_USBTXACTIVEEP2				6	INT_SCTXFIN
		1	INT_USBTXACTIVEEP1				5	INT_SCRXFIN
		0	INT_USBTXACTIVEEP0				4	INT_SCTXUND
18	Reserved						3	INT_SCRXOVF
17	Reserved						2	INT_SCTXIDLE

SILICON LABS

Table 3.2. NVIC and EM Peripheral Interrupt Map

#	Name	Bit	Flag		#	Name	Bit	Flag
6	INT_SC2 (continue)	INT_SC2FLAG register (cont.)			1	INT_TMR2	INT_TMR2FLAG register	
		1	INT_SCTXFREE				6	INT_TMRTIF
		0	INT_SCRXVAL				4	INT_TMRCC4IF
5	INT_SC1	INT_SC1FLAG register					3	INT_TMRCC3IF
		14	INT_SC1PARERR				2	INT_TMRCC2IF
		13	INT_SC1FRMERR				1	INT_TMRCC1IF
		12	INT_SCTXULDB				0	INT_TMRUIF
		11	INT_SCTXULDA		0	INT_TMR1	INT_TMR1FLAG 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					
		3	INT_SCRXOVF					
		2	INT_SCTXIDLE					
		1	INT_SCTXFREE					
		0	INT_SCRXVAL					
4	INT_SLEEPTMR							
3	INT_BB							
2	INT_MGMT							

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 EM358x's main clock, SYSCLK, is operating from the 24 MHz crystal and the crystal fails, the EM358x detects the failure and automatically switches to the internal 12 MHz RC clock. When this failure detection and switch has occurred, the EM358x triggers the CLK24M_FAIL second-level interrupt, which then triggers the NMI.

2. **Watchdog low water mark:** If the EM358x'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 EM358x-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

SILICON LABS

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

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	0	0	0	0	INT_USB	INT_RSVD18	INT_RSVD17	INT_DEBUG
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	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_RSVD18	[18]	RW	Reserved: this bit should be ignored.
INT_RSVD17	[17]	RW	Reserved: this bit should be ignored.
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.)

Register 3.2. INT_CFGCLR: Top-Level Clear Interrupts Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	INT_USB	INT_RSVD18	INT_RSVD17	INT_DEBUG
Bit	15	14	13	12	11	10	9	8
Name	INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_SEC	INT_SC2	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	INT_TIM2	INT_TIM1

Address: 0xE000E180   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_USB	[19]	RW	Write 1 to disable USB interrupt. (Writing 0 has no effect.) (where applicable)
INT_RSVD18	[18]	RW	Reserved: this bit should be ignored.
INT_RSVD17	[17]	RW	Reserved: this bit should be ignored.
INT_DEBUG	[16]	RW	Write 1 to disable debug interrupt. (Writing 0 has no effect.)
INT_IRQD	[15]	RW	Write 1 to disable IRQD interrupt. (Writing 0 has no effect.)
INT_IRQC	[14]	RW	Write 1 to disable IRQC interrupt. (Writing 0 has no effect.)
INT_IRQB	[13]	RW	Write 1 to disable IRQB interrupt. (Writing 0 has no effect.)
INT_IRQA	[12]	RW	Write 1 to disable IRQA interrupt. (Writing 0 has no effect.)
INT_ADC	[11]	RW	Write 1 to disable ADC interrupt. (Writing 0 has no effect.)
INT_MACRX	[10]	RW	Write 1 to disable MAC receive interrupt. (Writing 0 has no effect.)
INT_MACTX	[9]	RW	Write 1 to disable MAC transmit interrupt. (Writing 0 has no effect.)
INT_MACTMR	[8]	RW	Write 1 to disable MAC timer interrupt. (Writing 0 has no effect.)
INT_SEC	[7]	RW	Write 1 to disable security interrupt. (Writing 0 has no effect.)
INT_SC2	[6]	RW	Write 1 to disable serial controller 2 interrupt. (Writing 0 has no effect.)
INT_SC1	[5]	RW	Write 1 to disable serial controller 1 interrupt. (Writing 0 has no effect.)
INT_SLEEPTMR	[4]	RW	Write 1 to disable sleep timer interrupt. (Writing 0 has no effect.)
INT_BB	[3]	RW	Write 1 to disable baseband interrupt. (Writing 0 has no effect.)
INT_MGMT	[2]	RW	Write 1 to disable management interrupt. (Writing 0 has no effect.)
INT_TIM2	[1]	RW	Write 1 to disable timer 2 interrupt. (Writing 0 has no effect.)
INT_TIM1	[0]	RW	Write 1 to disable timer 1 interrupt. (Writing 0 has no effect.)

Register 3.3. INT_PENDSET: Top-Level Set Interrupts Pending Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	INT_USB	INT_RSVD18	INT_RSVD17	INT_DEBUG
Bit	15	14	13	12	11	10	9	8
Name	INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_SEC	INT_SC2	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	INT_TIM2	INT_TIM1

Address: 0xE000E200   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_USB	[19]	RW	Write 1 to pend USB interrupt. (Writing 0 has no effect.) (where applicable)
INT_RSVD18	[18]	RW	Reserved: this bit should be ignored.
INT_RSVD17	[17]	RW	Reserved: this bit should be ignored.
INT_DEBUG	[16]	RW	Write 1 to pend debug interrupt. (Writing 0 has no effect.)
INT_IRQD	[15]	RW	Write 1 to pend IRQD interrupt. (Writing 0 has no effect.)
INT_IRQC	[14]	RW	Write 1 to pend IRQC interrupt. (Writing 0 has no effect.).
INT_IRQB	[13]	RW	Write 1 to pend IRQB interrupt. (Writing 0 has no effect.)
INT_IRQA	[12]	RW	Write 1 to pend IRQA interrupt. (Writing 0 has no effect.)
INT_ADC	[11]	RW	Write 1 to pend ADC interrupt. (Writing 0 has no effect.)
INT_MACRX	[10]	RW	Write 1 to pend MAC receive interrupt. (Writing 0 has no effect.)
INT_MACTX	[9]	RW	Write 1 to pend MAC transmit interrupt. (Writing 0 has no effect.)
INT_MACTMR	[8]	RW	Write 1 to pend MAC timer interrupt. (Writing 0 has no effect.)
INT_SEC	[7]	RW	Write 1 to pend security interrupt. (Writing 0 has no effect.)
INT_SC2	[6]	RW	Write 1 to pend serial controller 2 interrupt. (Writing 0 has no effect.)
INT_SC1	[5]	RW	Write 1 to pend serial controller 1 interrupt. (Writing 0 has no effect.)
INT_SLEEPTMR	[4]	RW	Write 1 to pend sleep timer interrupt. (Writing 0 has no effect.)
INT_BB	[3]	RW	Write 1 to pend baseband interrupt. (Writing 0 has no effect.)
INT_MGMT	[2]	RW	Write 1 to pend management interrupt. (Writing 0 has no effect.)
INT_TIM2	[1]	RW	Write 1 to pend timer 2 interrupt. (Writing 0 has no effect.)
INT_TIM1	[0]	RW	Write 1 to pend timer 1 interrupt. (Writing 0 has no effect.)

Register 3.4. INT_PENDCLR: Top-Level Clear Interrupts Pending Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	INT_USB	INT_RS-VD18	INT_RS-VD17	INT_DEBUG
Bit	15	14	13	12	11	10	9	8
Name	INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_SEC	INT_SC2	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	INT_TIM2	INT_TIM1

Address: 0xE000E280   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_USB	[19]	RW	Write 1 to unpend USB interrupt. (Writing 0 has no effect.) (where applicable)
INT_RSVD18	[18]	RW	Reserved: this bit should be ignored.
INT_RSVD17	[17]	RW	Reserved: this bit should be ignored.
INT_DEBUG	[16]	RW	Write 1 to unpend debug interrupt. (Writing 0 has no effect.)
INT_IRQD	[15]	RW	Write 1 to unpend IRQD interrupt. (Writing 0 has no effect.)
INT_IRQC	[14]	RW	Write 1 to unpend IRQC interrupt. (Writing 0 has no effect.)
INT_IRQB	[13]	RW	Write 1 to unpend IRQB interrupt. (Writing 0 has no effect.)
INT_IRQA	[12]	RW	Write 1 to unpend IRQA interrupt. (Writing 0 has no effect.)
INT_ADC	[11]	RW	Write 1 to unpend ADC interrupt. (Writing 0 has no effect.)
INT_MACRX	[10]	RW	Write 1 to unpend MAC receive interrupt. (Writing 0 has no effect.)
INT_MACTX	[9]	RW	Write 1 to unpend MAC transmit interrupt. (Writing 0 has no effect.)
INT_MACTMR	[8]	RW	Write 1 to unpend MAC timer interrupt. (Writing 0 has no effect.)
INT_SEC	[7]	RW	Write 1 to unpend security interrupt. (Writing 0 has no effect.)
INT_SC2	[6]	RW	Write 1 to unpend serial controller 2 interrupt. (Writing 0 has no effect.)
INT_SC1	[5]	RW	Write 1 to unpend serial controller 1 interrupt. (Writing 0 has no effect.)
INT_SLEEPTMR	[4]	RW	Write 1 to unpend sleep timer interrupt. (Writing 0 has no effect.)
INT_BB	[3]	RW	Write 1 to unpend baseband interrupt. (Writing 0 has no effect.)
INT_MGMT	[2]	RW	Write 1 to unpend management interrupt. (Writing 0 has no effect.)
INT_TIM2	[1]	RW	Write 1 to unpend timer 2 interrupt. (Writing 0 has no effect.)
INT_TIM1	[0]	RW	Write 1 to unpend timer 1 interrupt. (Writing 0 has no effect.)

Register 3.5. INT_ACTIVE: Top-Level Active Interrupts Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	INT_USB	INT_RS-VD18	INT_RS-VD17	INT_DEBUG
Bit	15	14	13	12	11	10	9	8
Name	INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_SEC	INT_SC2	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	INT_TIM2	INT_TIM1

Address: 0xE000E300   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_USB	[19]	RW	USB interrupt active (where applicable)
INT_RSVD18	[18]	RW	Reserved: this bit should be ignored.
INT_RSVD17	[17]	RW	Reserved: this bit should be ignored.
INT_DEBUG	[16]	R	Debug interrupt active.
INT_IRQD	[15]	R	IRQD interrupt active.
INT_IRQC	[14]	R	IRQC interrupt active.
INT_IRQB	[13]	R	IRQB interrupt active.
INT_IRQA	[12]	R	IRQA interrupt active.
INT_ADC	[11]	R	ADC interrupt active.
INT_MACRX	[10]	R	MAC receive interrupt active.
INT_MACTX	[9]	R	MAC transmit interrupt active.
INT_MACTMR	[8]	R	MAC timer interrupt active.
INT_SEC	[7]	R	Security interrupt active.
INT_SC2	[6]	R	Serial controller 2 interrupt active.
INT_SC1	[5]	R	Serial controller 1 interrupt active.
INT_SLEEPTMR	[4]	R	Sleep timer interrupt active.
INT_BB	[3]	R	Baseband interrupt active.
INT_MGMT	[2]	R	Management interrupt active.
INT_TIM2	[1]	R	Timer 2 interrupt active.
INT_TIM1	[0]	R	Timer 1 interrupt active.

Register 3.6. INT_MISS: Top-Level Missed Interrupts Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	INT_MIS-SUSB	INT_RSVD18	INT_RSVD17	0
Bit	15	14	13	12	11	10	9	8
Name	INT_MISSIRQD	INT_MISSIRQC	INT_MISSIRQB	INT_MISSIRQA	INT_MISSADC	INT_MISSMACRX	INT_MISSMACTX	INT_MISSMACTMR
Bit	7	6	5	4	3	2	1	0
Name	INT_MISSSEC	INT_MISSSC2	INT_MISSSC1	INT_MISSSLEEP	INT_MISSBB	INT_MISSMGMT	0	0

Address: 0x4000A820   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_MISSUSB	[19]	RW	USB interrupt missed (where applicable)
INT_RSVD18	[18]	RW	Reserved: this bit should be ignored.
INT_RSVD17	[17]	RW	Reserved: this bit should be ignored.
INT_MISSIRQD	[15]	RW	IRQD interrupt missed.
INT_MISSIRQC	[14]	RW	IRQC interrupt missed.
INT_MISSIRQB	[13]	RW	IRQB interrupt missed.
INT_MISSIRQA	[12]	RW	IRQA interrupt missed.
INT_MISSADC	[11]	RW	ADC interrupt missed.
INT_MISSMACRX	[10]	RW	MAC receive interrupt missed.
INT_MISSMACTX	[9]	RW	MAC transmit interrupt missed.
INT_MISSMACTMR	[8]	RW	MAC Timer interrupt missed.
INT_MISSSEC	[7]	RW	Security interrupt missed.
INT_MISSSC2	[6]	RW	Serial controller 2 interrupt missed.
INT_MISSSC1	[5]	RW	Serial controller 1 interrupt missed.
INT_MISSSLEEP	[4]	RW	Sleep timer interrupt missed.
INT_MISSBB	[3]	RW	Baseband interrupt missed.
INT_MISSMGMT	[2]	RW	Management interrupt missed.

SILICON LABS

Register 3.7. SCS_AFSR: Auxiliary Fault Status Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	WRONGSIZE	PROTECTED	RESERVED	MISSED

Address: 0xE000ED3C   Reset: 0x0

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

4. Radio Module

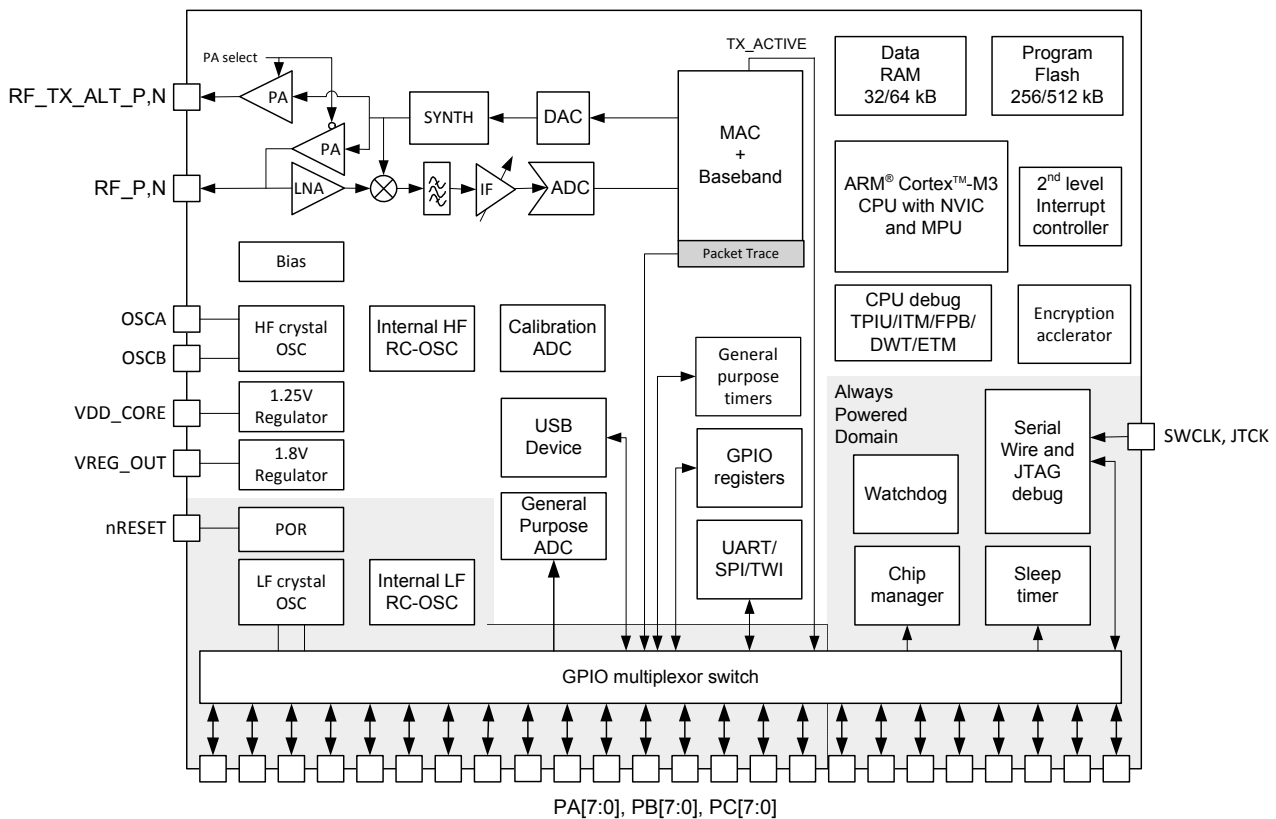The radio module consists of an analog front end and digital baseband as shown in Figure 4.1.

Figure 4.1. EM358x Block Diagram

4.1. Receive (RX) Path

The Rx path uses a low-IF, super-heterodyne receiver that rejects the image frequency using complex mixing and polyphase filtering. In the analog domain, the input RF signal from the antenna is first amplified and mixed down to a 4 MHz IF frequency. The mixers' output is filtered, combined, and amplified before being sampled by a 12 MSPS ADC. The digitized signal is then demodulated in the digital baseband. The filtering within the Rx path improves the EM358x's co-existence with other 2.4 GHz transceivers such as IEEE 802.15.4-2006, IEEE 802.11-2007, and Bluetooth radios. The digital baseband also provides gain control of the Rx path, both to enable the reception of small and large wanted signals and to tolerate large interferers.

4.1.1. RX Baseband

The EM358x Rx digital baseband implements a coherent demodulator for optimal performance. The baseband demodulates the O-QPSK signal at the chip level and synchronizes with the IEEE 802.15.4-2006-defined preamble. An automatic gain control (AGC) module adjusts the analog gain continuously every ¼ symbol until the preamble is detected. Once detected, the gain is fixed for the remainder of the packet. The baseband despreads the demodulated data into 4-bit symbols. These symbols are buffered and passed to the hardware-based MAC module for packet assembly and filtering.

In addition, the Rx baseband provides the calibration and control interface to the analog Rx modules, including the LNA, Rx baseband filter, and modulation modules. The Silicon Labs software includes calibration algorithms that use this interface to reduce the effects of silicon process and temperature variation.

4.1.2. RSSI and CCA

The EM358x calculates the RSSI over every 8-symbol period as well as at the end of a received packet. The linear range of RSSI is specified to be at least 40 dB over temperature. At room temperature, the linear range is approximately 60 dB (-90 dBm to -30 dBm input signal).

The EM358x Rx baseband provides support for the IEEE 802.15.4-2006 RSSI CCA method. Clear channel reports busy medium if RSSI exceeds its threshold.

4.2. Transmit (TX) Path

The EM358x Tx path produces an O-QPSK-modulated signal using the analog front end and digital baseband. The area- and power-efficient Tx architecture uses a two-point modulation scheme to modulate the RF signal generated by the synthesizer. The modulated RF signal is fed to the integrated PA and then out of the EM358x.

4.2.1. TX Baseband

The EM358x Tx baseband in the digital domain spreads the 4-bit symbol into its IEEE 802.15.4-2006-defined 32-chip sequence. It also provides the interface for the Silicon Labs software to calibrate the Tx module to reduce silicon process, temperature, and voltage variations.

4.2.2. TX_ACTIVE and nTX_ACTIVE Signals

For applications requiring an external PA, two signals are provided called TX_ACTIVE and nTX_ACTIVE. These signals are the inverse of each other. They can be used for external PA power management and RF switching logic. In transmit mode the Tx baseband drives TX_ACTIVE high, as described in Table 7.5, "GPIO Signal Assignments," on page 61. In receive mode the TX_ACTIVE signal is low. TX_ACTIVE is the alternate function of PC5, and nTX_ACTIVE is the alternate function of PC6. See Chapter 7 GPIO for details of the alternate GPIO functions. The digital I/O that provide these signals have a 4 mA output sink and source capability.

4.3. Calibration

The Silicon Labs software calibrates the radio using dedicated hardware resources.

4.4. Integrated MAC Module

The EM358x integrates most of the IEEE 802.15.4-2006 MAC requirements in hardware. This allows the ARM$^{®}$ Cortex^{TM}-M3CPU to provide greater bandwidth to application and network operations. In addition, the hardware acts as a first-line filter for unwanted packets. The EM358x MAC uses a DMA interface to RAM to further reduce the overall ARM$^{®}$ Cortex^{TM}-M3 CPU interaction when transmitting or receiving packets.

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

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

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-pending on Tx packets
- Address recognition and packet filtering on Rx 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 received and transmitted messages
- Attaching packet information to received packets (LQI, RSSI, gain, time stamp, and packet status)
- IEEE 802.15.4-2006 timing and slotted/unslotted timing

4.5. Packet Trace Interface (PTI)

The EM358x integrates a true PHY-level PTI for effective network-level debugging. It monitors all the PHY Tx and Rx packets between the MAC and baseband modules without affecting their normal operation. It cannot be used to inject packets into the PHY/MAC interface. This 500 kbps asynchronous interface comprises the frame signal (PTI_EN, PA4) and the data signal (PTI_DATA, PA5). PTI is supported by the Silicon Labs development tools.

4.6. Random Number Generator

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

5. System Modules

System modules encompass power domains, resets, clocks, system timers, power management, and encryption. Figure 5.1 shows these modules and how they interact.
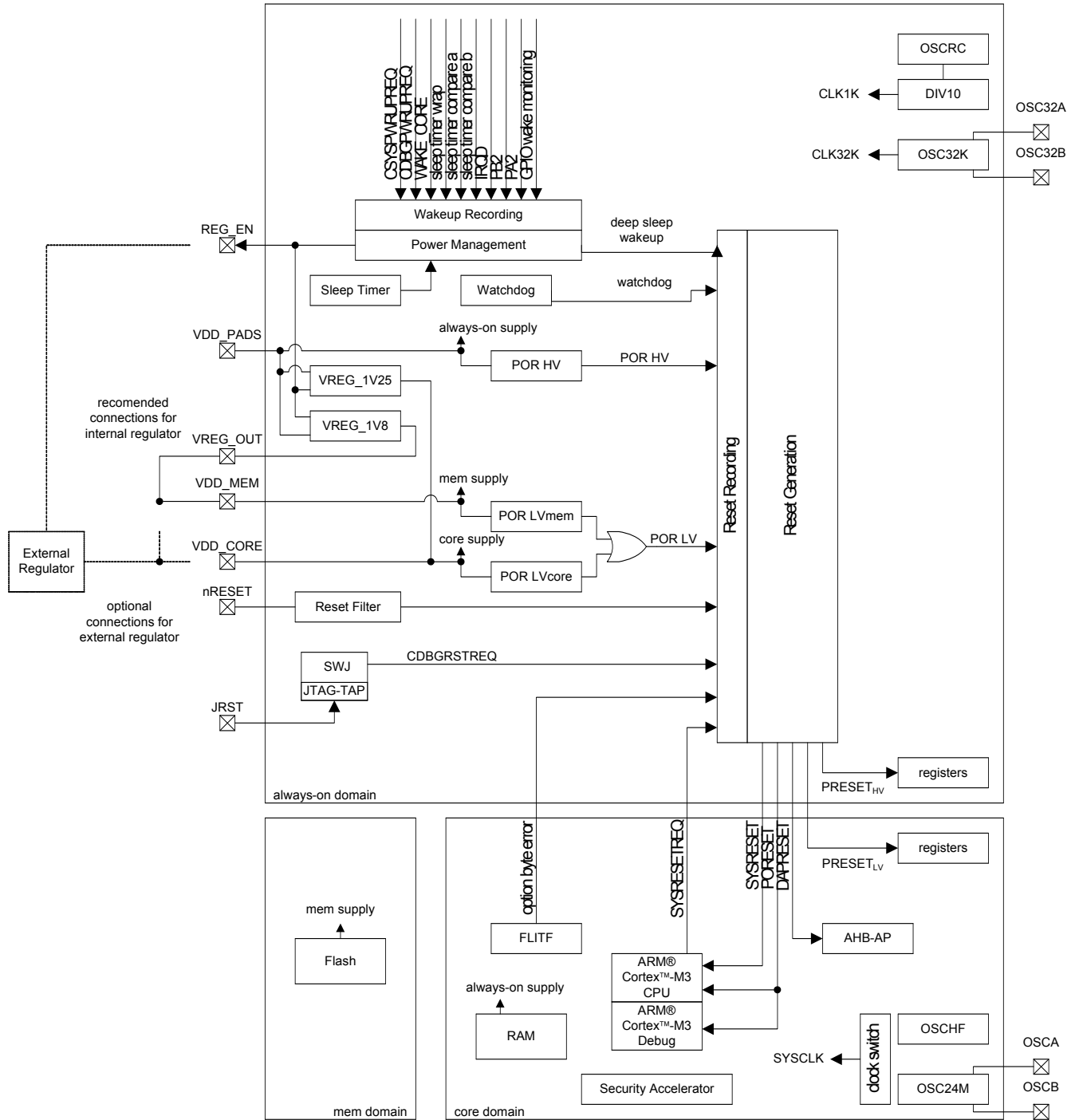

Figure 5.1. System Module Block Diagram

5.1. Power Domains

The EM358x contains three power domains:

- An "always-on domain" containing all logic and analog cells required to manage the EM358x's power modes, including the GPIO controller and sleep timer. This domain must remain powered.
- A "core domain" containing the CPU, Nested Vectored Interrupt Controller (NVIC), and peripherals. To save power, this domain can be powered down using a mode called deep sleep. In the EM358x the core domain also includes the RAM, which by default is powered down in deep sleep. An additional feature of the RAM is that blocks of RAM cells can optionally be retained in deep sleep. This is configured using a register, which must be written before entering deep sleep.
- A "flash domain" containing the flash memory. This domain is managed by the power management controller. During deep sleep the flash portion is completely powered down.

5.1.1. Internally Regulated Power

The preferred and recommended power configuration is to use the internal regulated power supplies to provide power to the core and memory domains. The internal regulators (VREG_1V25 and VREG_1V8) generate nominal 1.25 V and 1.8 V supplies. The 1.25 V supply is internally routed to the core domain, RAM, and to an external pin. The 1.8 V supply is routed to an external pin where it can be externally routed back into the chip to supply the memory domain. The internal regulators are described in Chapter 6, Integrated Voltage Regulator.

When using the internal regulators, the always-on domain must be powered between 1.8 V and 3.6 V at all four VDD_PADS pins.

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

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

5.1.2. Externally Regulated Power

Optionally, the on-chip regulators may be left unused, and the core and memory domains may instead be powered from external supplies. The nominal supply voltages of the internal power domains must be respected, that is core and RAM at nominally 1.25 V and flash at nominally 1.8 V. A regulator enable signal, REG_EN, is provided for control of external regulators. This is an open-drain signal that requires an external pull-up resistor. If REG_EN is not required to control external regulators it can be disabled (see section 7.3, Forced Functions in Chapter 7, GPIO).

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

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

When using an external regulator, the external nominal 1.25 V supply has to be connected to VDD_CORE pins. The external nominal 1.8 V supply must be connected to the VDD_MEM, VDD_PADSA, VDD_VCO, VDD_RF, VDD_IF, VDD_PRE and VDD_SYNTH pins.

5.2. Resets

The EM358x resets are generated from a number of sources. Each of these reset sources feeds into central reset detection logic that causes various parts of the system to be reset depending on the state of the system and the nature of the reset event.

5.2.1. Reset Sources

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

The EM358x measures the voltage levels supplied to the three power domains. If a supply voltage drops below a low threshold, then a reset is applied. The reset is released if the supply voltage rises above a high threshold. There are three detection circuits for power-on-reset as follows:

- POR HV monitors the always-on domain supply voltage. Thresholds are given in Table 5.1.
- POR LVcore monitors the core domain supply voltage. Thresholds are given in Table 5.2.
- POR LVmem monitors the memory supply voltage. Thresholds are given in Table 5.3.

Table 5.1. POR HV Thresholds

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

Table 5.2. POR LVcore Thresholds

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

Table 5.3. POR LVmem Thresholds

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

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

5.2.1.2. nRESET Pin

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

To afford good noise immunity and resistance to switch bounce, the pin is filtered with the Reset Filter module and generates the pin reset source, nRESET, to the Reset Generation module. Table 5.4 contains the specification for the filter.

Table 5.4. Reset Filter Specification for nRESET

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

5.2.1.3. Watchdog Reset

The EM358x contains a watchdog timer (see also the Watchdog Timer section) that is clocked by the internal 1 kHz timing reference. When the timer expires it generates the reset source WATCHDOG_RESET to the Reset Generation module.

5.2.1.4. Software Reset

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

5.2.1.5. Option Byte Error

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

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

5.2.1.6. Debug Reset

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

5.2.1.7. JRST

One of the EM358x's pins can function as the JTAG reset, conforming to the requirements of the JTAG standard. This input acts independently of all other reset sources and, when asserted, does not reset any on-chip hardware except for the JTAG TAP. If the EM358x is in the Serial Wire mode or if the SWJ is disabled, this input has no effect.

5.2.1.8. Deep Sleep Reset

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

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

5.2.2. Reset Recording

The EM358x records the last reset condition that generated a restart to the system. The reset conditions recorded are as follows:

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

Note: While CPU Lockup is shown as a reset condition in software, CPU Lockup is not specifically a reset event. CPU Lockup is set to indicate that the CPU entered an unrecoverable exception. Execution stops but a reset is not applied. This is so that a debugger can interpret the cause of the error. Silicon Labs recommends that in a live application (in other words, no debugger attached) the watchdog be enabled by default so that the EM358x can be restarted.

5.2.3. Reset Generation Module

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

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

Table 5.5 shows which reset sources generate certain resets.

Table 5.5. Generated Resets

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

5.3. Clocks

The EM358x integrates four oscillators:

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

Figure 5.2 shows a block diagram of the clocks in the EM358x. This simplified view shows all the clock sources and the general areas of the chip to which they are routed.
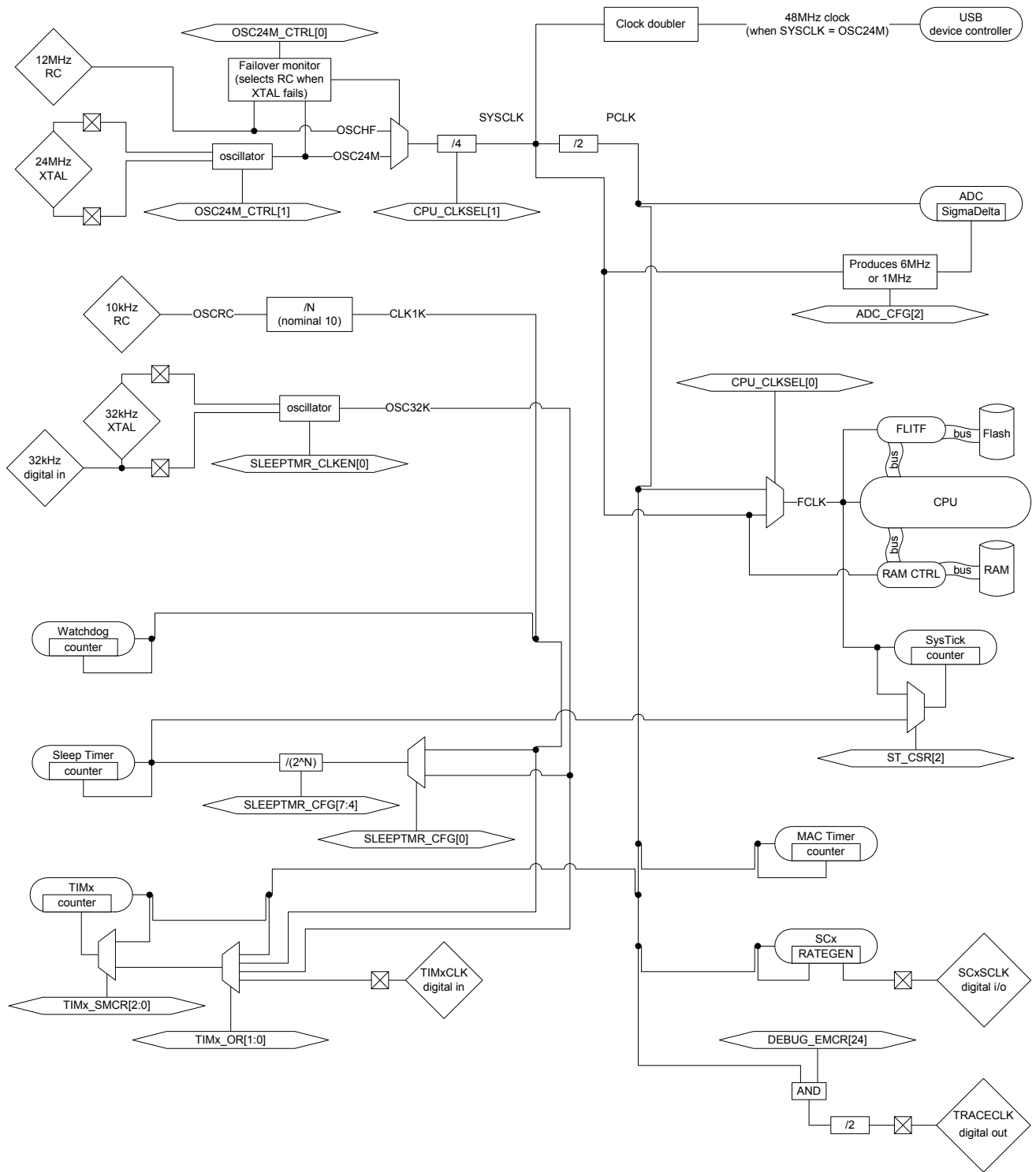
Figure 5.2. Clocks Block Diagram

5.3.1. High-Frequency Internal RC Oscillator (OSCHF)

The high-frequency RC oscillator (OSCHF) is used as the default system clock source when power is applied to the core domain. The nominal frequency coming out of reset is 12 MHz and Silicon Labs software calibrates this clock to 12 MHz. Table 5 7 contains the specification for the high frequency RC oscillator.

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

Table 5.6. High-Frequency RC Oscillator Specification

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

5.3.2. High-Frequency Crystal Oscillator (OSC24M)

The high-frequency crystal oscillator (OSC24M) requires an external 24 MHz crystal with an accuracy of ±40 ppm. Based upon the application's bill of materials and current consumption requirements, the external crystal may cover a range of ESR requirements. Table 5.6 contains the specification for the high frequency crystal oscillator.

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

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

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

SILICON LABS

Table 5.7. High-Frequency Crystal Oscillator Specification

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

5.3.3. Low-Frequency Internal RC Oscillator (OSCRC)

A low-frequency RC oscillator (OSCRC) is provided as an internal timing reference. The nominal frequency coming out of reset is 10 kHz, and Silicon Labs software calibrates this clock to 10 kHz. From the tuned 10 kHz oscillator (OSCRC) Silicon Labs software calibrates a fractional-N divider to produce a 1 kHz reference clock, CLK1K. Table 5.8 contains the specification for the low frequency RC oscillator.

Table 5.8. Low-Frequency RC Oscillator Specification

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

5.3.4. Low-Frequency Crystal Oscillator (OSC32K)

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

Table 5.9 contains the specification for the low frequency crystal oscillator.

Table 5.9. Low-Frequency Crystal Oscillator Specification

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

5.3.5. Clock Switching

The EM358x has two switching mechanisms for the main system clock, providing four clock modes. Table 5.10 shows these clock modes and how they affect the internal clocks.

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

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

The register bit USBSUSP_CLKSEL_FIELD in the CPU_CLKSEL register is used to divide the whole clock tree by 4 when the EM358x (variants that support USB) is operating as a bus-powered USB device and USB suspends the EM358x. Refer to Chapter 9, USB Device, for USB details.

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

Table 5.10. System Clock Modes

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

SILICON LABS

Table 5.10. System Clock Modes (Continued)

| 1 (OSC24M) | 0 (Normal CPU) | 24 MHz | 12 MHz | 12 MHz | 12 MHz |
| 1 (OSC24M) | 1 (Fast CPU) | 24 MHz | 12 MHz | 24 MHz | 12 MHz |

5.4. System Timers

5.4.1. Watchdog Timer

The EM358x integrates a watchdog timer which can be enabled to provide protection against software crashes and ARM® Cortex™-M3 CPU lockup. By default, it is disabled at power up of the always-on power domain. The watchdog timer uses the calibrated 1 kHz clock (CLK1K) as its reference and provides a nominal 2.048 s timeout. A low water mark interrupt occurs at 1.792 s and triggers an NMI to the ARM® Cortex™-M3 NVIC as an early warning. When the watchdog is enabled, the timer must be periodically reset before it expires. The watchdog timer is paused when the debugger halts the ARM® Cortex™-M3. Additionally, the Silicon Labs software that implements deep sleep functionality disables the watchdog when entering deep sleep and restores the watchdog, if it was enabled, when exiting deep sleep.

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

5.4.2. Sleep Timer

The EM358x integrates a 32-bit timer dedicated to system timing and waking from sleep at specific times. The sleep timer can use either the calibrated 1 kHz reference (CLK1K), or the 32 kHz crystal clock (CLK32K). The default clock source is the internal 1 kHz clock.

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

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

To save current during deep sleep, the low-frequency internal RC oscillator (OSCRC) can be turned off. If OSCRC is turned off during deep sleep and a low-frequency 32.768 kHz crystal oscillator is not being used, then the sleep timer will not operate during deep sleep and sleep timer wake events cannot be used to wake up the EM358x.

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

5.4.3. Event Timer

The SysTick timer is an ARM® standard system timer in the NVIC. The SysTick timer can be clocked from either the FCLK (the clock going into the CPU) or the Sleep Timer clock. FCLK is either the SYSCLK or PCLK as selected by CPU_CLKSEL register (see "5.3.5. Clock Switching").

5.5. Power Management

The EM358x's power management system is designed to achieve the lowest deep sleep current consumption possible while still providing flexible wakeup sources, timer activity, and debugger operation. The EM358x has four main sleep modes:

- Idle Sleep: Puts the CPU into an idle state where execution is suspended until any interrupt occurs. All power domains remain fully powered and nothing is reset.
- Deep Sleep 1: The primary deep sleep state. In this state, the core power domain is fully powered down and the sleep timer is active.
- Deep Sleep 2: The same as Deep Sleep 1 except that the sleep timer is inactive to save power. In this mode the sleep timer cannot wake up the EM358x.
- Deep Sleep 0 (also known as Emulated Deep Sleep): The chip emulates a true deep sleep without powering down the core domain. Instead, the core domain remains powered and all peripherals except the system debug components (ITM, DWT, FPB, NVIC) are held in reset. The purpose of this sleep state is to allow EM358x software to perform a deep sleep cycle while maintaining debug configuration such as breakpoints.

CSYSPWRUPREQ, CDBGPWRUPREQ, and the corresponding CSYSPWRUPACK and CDBGPWRUPACK are bits in the debug port's CTRL/STAT register in the SWJ. For further information on these bits and the operation of the SWJ-DP please refer to the ARM Debug Interface v5 Architecture Specification (ARM IHI 0031A).

For further power savings when not in deep sleep, the USB, ADC, Timer 1, Timer 2, Serial Controller 1, and Serial Controller 2 peripherals can be individually disabled through the PERIPHERAL_DISABLE register. Disabling a peripheral saves power by stopping the clock feeding that peripheral. A peripheral should only be disabled through the PERIPHERAL_DISABLE register when the peripheral is idle and disabled through the peripheral's own configuration registers, otherwise undefined behavior may occur. When a peripheral is disabled through the PERIPHERAL_DISABLE register, all registers associated with that peripheral ignore all subsequent writes, and subsequent reads return the value seen in the register at the moment the peripheral is disabled.

5.5.1. Wake Sources

When in deep sleep the EM358x can be returned to the running state in a number of ways, and the wake sources are split depending on deep sleep 1 or deep sleep 2.

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

- Wake on GPIO activity: Wake due to change of state on any GPIO.
- Wake on serial controller 1: Wake due to a change of state on GPIO Pin PB2.
- Wake on serial controller 2: Wake due to a change of state on GPIO Pin PA2.
- Wake on IRQD: Wake due to a change of state on IRQD. Since IRQD can be configured to point to any GPIO, this wake source is another means of waking on any GPIO activity.
- Wake on setting of CDBGPWRUPREQ: Wake due to setting the CDBGPWRUPREQ bit in the debug port in the SWJ.
- Wake on setting of CSYSPWRUPREQ: Wake due to setting the CSYSPWRUPREQ bit in the debug port in the SWJ.

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

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

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

- Wake on write to the WAKE_CORE register bit.

The Wakeup Recording module monitors all possible wakeup sources. More than one wakeup source may be recorded because events are continually being recorded (not just in deep-sleep) and another event may happen between the first wake event and when the EM358x wakes up.

5.5.2. Basic Sleep Modes

The power management state diagram in Figure 5.3 shows the basic operation of the power management controller.
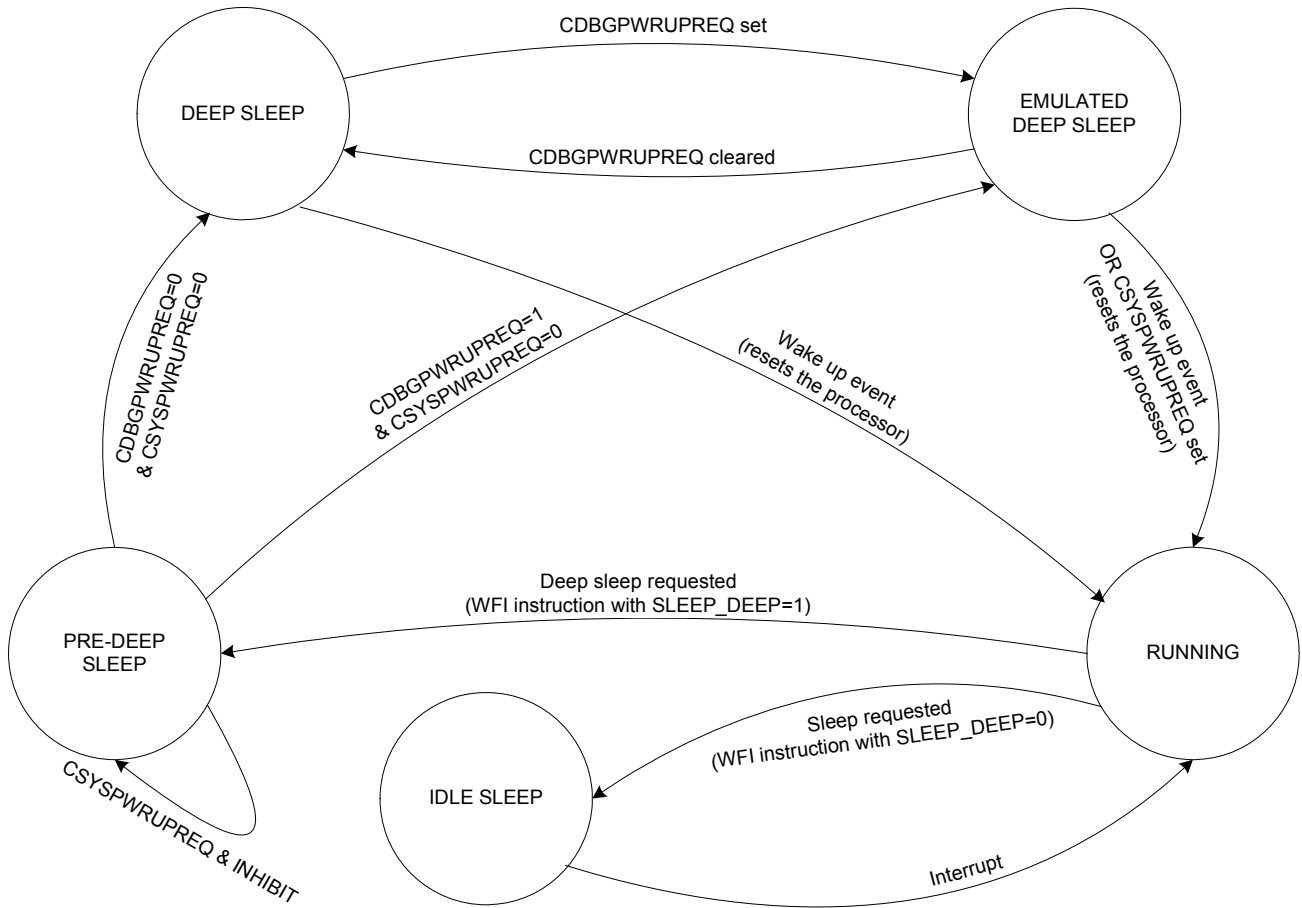
Figure 5.3. Power Management State Diagram

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

- Idle Sleep is achieved by executing a WFI instruction while the SLEEPDEEP bit in the Cortex System Control register (SCS_SCR) is clear. This puts the CPU into an idle state where execution is suspended until an interrupt occurs. This is indicated by the state at the bottom of the diagram. Power is maintained to the core logic of the EM358x during the Idle Sleeping state.

- Deep sleep is achieved by executing a WFI instruction with the SLEEPDEEP bit in SCS_SCR set. This triggers the state transitions around the main loop of the diagram, resulting in powering down the EM358x's core logic, and leaving only the always-on domain powered. Wake up is triggered when one of the pre-determined events occurs.

If a deep sleep is requested the EM358x first enters a pre-deep sleep state. This state prevents any section of the chip from being powered off or reset until the SWJ goes idle (by clearing CSYSPWRUPREQ). This pre-deep sleep state ensures debug operations are not interrupted.

In the deep sleep state the EM358x waits for a wake up event which will return it to the running state. In powering up the core logic the ARM® Cortex™-M3 is put through a reset cycle and Silicon Labs software restores the stack and application state to the point where deep sleep was invoked.

5.5.3. Further Options for Deep Sleep

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

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

5.5.4. RAM Retention in Deep Sleep

The RAM can optionally be configured using the RAM_RETAIN register to select banks of locations to be non-volatile. In deep sleep those banks selected are powered by a low leakage internal regulator that remains on during deep sleep, powered from the always-on supply.

The RAM_RETAIN[15:0] register acts as a bit map of 4k byte blocks whereby setting a bit to 1 indicates that a bank is to be retained. The default condition of 0xFFFF retains all banks in the RAM.

The bits in RAM_RETAIN are arranged so that bit [0] sets the retention option for bank 0, addresses from 0x20000000 to 0x20000FFF, bit [1] for addresses 0x20001000 to 0x20001FFF, and so on up to bit [15] for addresses 0x2000F000 to 0x2000FFFF. It is not necessary for retained banks to be contiguous. Some banks may need to be retained for correct operation of the stack and others may be defined according to the application.

5.5.5. Use of Debugger with Sleep Modes

The debugger communicates with the EM358x using the SWJ.

When the debugger is logically connected, the CDBGPWRUPREQ bit in the debug port in the SWJ is set, and the EM358x will only enter deep sleep 0 (the Emulated Deep Sleep state). The CDBGPWRUPREQ bit indicates that a debug tool is logically connected to the chip and therefore debug state may be in the system debug components. To maintain the debug state in the system debug components only deep sleep 0 may be used, since deep sleep 0 will not cause a power cycle or reset of the core domain. The CSYSPWRUPREQ bit in the debug port in the SWJ indicates that a debugger wants to access memory actively in the EM358x. Therefore, whenever the CSYSPWRUPREQ bit is set while the EM358x is awake, the EM358x cannot enter deep sleep until this bit is cleared. This ensures the EM358x does not disrupt debug communication into memory.

Clearing both CSYSPWRUPREQ and CDBGPWRUPREQ allows the EM358x to achieve a true deep sleep state (deep sleep 1 or 2). Both of these signals also operate as wake sources, so that when a debugger logically connects to the EM358x and begins accessing the chip, the EM358x automatically comes out of deep sleep. When the debugger initiates access while the EM358x is in deep sleep, the SWJ intelligently holds off the debugger for a brief period of time until the EM358x is properly powered and ready.

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

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

5.5.6. Registers

Register 5.1. RAM_RETAIN

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

Address: 0x4000403C Reset: 0xFFF

Bitname	Bitfield	Access	Description
RETAIN	[15]	RW	Sets the retention option for 0x2000F000 to 0x2000FFFF
RETAIN	[14]	RW	Sets the retention option for 0x2000E000 to 0x2000EFFF
RETAIN	[13]	RW	Sets the retention option for 0x2000D000 to 0x2000DFFF
RETAIN	[12]	RW	Sets the retention option for 0x2000C000 to 0x2000CFFF
RETAIN	[11]	RW	Sets the retention option for 0x2000B000 to 0x2000BFFF
RETAIN	[10]	RW	Sets the retention option for 0x2000A000 to 0x2000AFFF
RETAIN	[9]	RW	Sets the retention option for 0x20009000 to 0x20009FFF
RETAIN	[8]	RW	Sets the retention option for 0x20008000 to 0x20008FFF
RETAIN	[7]	RW	Sets the retention option for 0x20007000 to 0x20007FFF
RETAIN	[6]	RW	Sets the retention option for 0x20006000 to 0x20006FFF
RETAIN	[5]	RW	Sets the retention option for 0x20005000 to 0x20005FFF
RETAIN	[4]	RW	Sets the retention option for 0x20004000 to 0x20004FFF
RETAIN	[3]	RW	Sets the retention option for 0x20003000 to 0x20003FFF
RETAIN	[2]	RW	Sets the retention option for 0x20002000 to 0x20002FFF
RETAIN	[1]	RW	Sets the retention option for 0x20001000 to 0x20001FFF
RETAIN	[0]	RW	Sets the retention option for 0x20000000 to 0x20000FFF

Register 5.2. PERIPHERAL_DISABLE

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	PERIDIS_USB
Bit	7	6	5	4	3	2	1	0
Name	PER-IDIS_RSVD7	PERIDIS_RSVD6	PERIDIS_RSVD	PERIDIS_ADC	PERIDIS_TIM2	PERIDIS_TIM1	PERIDIS_SC1	PERIDIS_SC2

Address: 0x40004038 Reset: 0x0

Bitname	Bitfield	Access	Description
PERIDIS_USB	[8]	RW	Disable the clock to the USB periperal
PERIDIS_RSVD7	[7]	RW	Reserved: This bit must be set to 1.
PERIDIS_RSVD6	[6]	RW	Reserved: This bit must be set to 1.
PERIDIS_RSVD	[5]	RW	Reserved: This bit can change during normal operation. When writing to PERIPHERAL_DISABLE, the value of this bit must be preserved.
PERIDIS_ADC	[4]	RW	Disable the clock to the ADC peripheral.
PERIDIS_TIM2	[3]	RW	Disable the clock to the TIM2 peripheral.
PERIDIS_TIM1	[2]	RW	Disable the clock to the TIM1 peripheral.
PERIDIS_SC1	[1]	RW	Disable the clock to the SC1 peripheral.
PERIDIS_SC2	[0]	RW	Disable the clock to the SC2 peripheral.

5.6. Security Accelerator

The EM358x contains a hardware AES encryption engine accessible from the ARM® Cortex™-M3. NIST-based CCM, CCM*, CBC-MAC, and CTR modes are implemented in hardware. These modes are described in the IEEE 802.15.4-2006 specification, with the exception of CCM*, which is described in the ZigBee Security Services Specification 1.0.

6. Integrated Voltage Regulator

The EM358x integrates two low dropout regulators to provide 1.8 V and 1.25 V power supplies, as detailed in Table 6 1. The 1V8 regulator supplies the analog and memories, and the 1V25 regulator supplies the digital core. In deep sleep the voltage regulators are disabled.

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

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

The regulators are controlled by the digital portion of the chip as described in Chapter 5, System Modules.

An example of decoupling capacitors and PCB layout can be found in the application notes (see the various EM358x reference design documentation).

Table 6.1. Integrated Voltage Regulator Specifications

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

An external 1.8 V regulator may replace both internal regulators. The EM358x can control external regulators during deep sleep using open-drain GPIO PA7, as described in Chapter 7, GPIO. The EM358x drives PA7 low during deep sleep to disable the external regulator and an external pull-up is required to release this signal to indicate that supply voltage should be provided. Current consumption increases approximately 2 mA when using an external regulator. When using an external regulator the internal regulators should be disabled through Silicon Labs software. The always-on domain needs to be minimally powered at 2.1 V, and cannot be powered from the external 1.8 V regulator.

7. GPIO (General Purpose Input/Output)

The EM358x has 24 multipurpose GPIO pins, which may be individually configured as:

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

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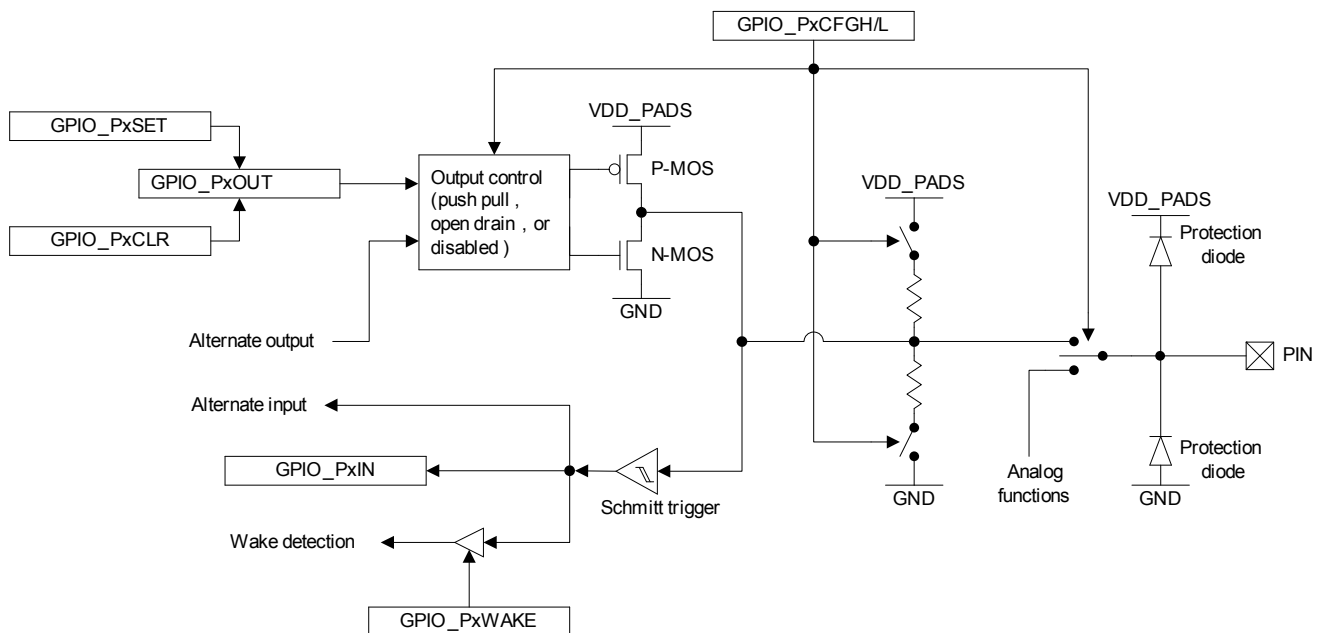
Figure 7.1. GPIO Block Diagram

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

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

7.1. GPIO Ports

The 24 GPIO pins are grouped into three ports: PA, PB, and PC. Individual GPIOs within a port are numbered according to their bit positions within the GPIO registers.

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

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

- GPIO_PxIN (input data register) returns the pin level (unless in analog mode).
- GPIO_PxOUT (output data register) controls the output level in normal output mode.
- GPIO_PxCLR (clear output data register) clears bits in GPIO_PxOUT.
- GPIO_PxSET (set output data register) sets bits in GPIO_PxOUT.
- GPIO_PxWAKE (wake monitor register) specifies the pins that can wake the EM358x.

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

Five GPIO pins (PA6, PA7, PB6, PB7 and PC0) can sink and source higher current than standard GPIO outputs. Refer to the *Ember EM358x Data Sheet*, Table 3-5, Digital I/O Specifications in Chapter 3, Electrical Characteristics, for more information.

7.2. Configuration

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

Table 7.1. GPIO Configuration Modes

GPIO Mode	GPIO_PxCFGH/L	Description
Analog	0x0	Analog input or output. When in analog mode, the digital input (GPIO_PxIN) always reads 1.
Input (floating)	0x4	Digital input without an internal pull up or pull down. Output is disabled.
SWDIO (bidirectional)	0x6	Bidirectional mode (push-pull output or floating input) only for retaining SWDIO functionality of PC4 when the GPIO_DEBUG-DIS bit in the GPIO_DBGCFG register is set.
Input (pull-up or pull-down)	0x8	Digital input with an internal pull up or pull down. A set bit in GPIO_PxOUT selects pull up and a cleared bit selects pull down. Output is disabled.
Output (push-pull)	0x1	Push-pull output. GPIO_PxOUT controls the output.
Output (open-drain)	0x5	Open-drain output. GPIO_PxOUT controls the output. If a pull up is required, it must be external.
Alternate Output (push-pull)	0x9	Push-pull output. An onboard peripheral controls the output.

SILICON LABS

Table 7.1. GPIO Configuration Modes

Alternate Output (open-drain)	0xD	Open-drain output. An onboard peripheral controls the output. If a pull up is required, it must be external.
Alternate Output (push-pull), SPI Slave MISO Mode	0xB	Push-pull output mode used only for SPI slave mode MISO pins.

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

Several GPIOs share an alternate output with Timer 2 and the Serial Controllers. Bits in Timer 2's TIM2_OR register control routing Timer 2 outputs to different GPIOs. Bits in Timer 2's TIM2_CCER register enable Timer 2 outputs. When Timer 2 outputs are enabled they override Serial Controller outputs. Table 7.2 indicates the GPIO mapping for Timer 2 outputs depending on the bits in the register TIM2_OR. Refer to Chapter 10, General Purpose Timers, for complete information on timer configuration.

Table 7.2. Timer 2 Output Configuration Controls

Timer 2 Output	Option Register Bit	GPIO Mapping Selected by TIM2_OR Bit	
		0	1
TIM2C1	TIM2_OR[4]	PA0	PB1
TIM2C2	TIM2_OR[5]	PA3	PB2
TIM2C3	TIM2_OR[6]	PA1	PB3
TIM2C4	TIM2_OR[7]	PA2	PB4

For outputs assigned to the serial controllers, the serial interface mode registers (SCx_MODE) determine how the GPIO pins are used.

The alternate outputs of PA4 and PA5 can either provide packet trace data (PTI_EN and PTI_DATA), or synchronous CPU trace data (TRACEDATA2 and TRACEDATA3). The selection of packet trace or CPU trace is made through the Silicon Labs software.

The alternate outputs of PB0 and PC1 can also provide TRACEDATA2 and TRACEDATA3 for situations where packet trace is also required.

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

7.3. Forced Functions

For some GPIOs the GPIO_PxCFGH/L configuration will be overridden. These functions are forced when the EM358x is reset and remain forced until software or an external debugger overrides the forced functions. Table 7.3 shows the GPIOs that have different functions forced on them regardless of the GPIO_PxCFGH/L registers.

Table 7.3. GPIO Forced Functions

GPIO	Forced Mode	Forced Signal
PA7	Open-drain output	REG_EN
PC0	Input with pull up	JRST
PC2	Push-pull output	JTDO

Table 7.3. GPIO Forced Functions

PC3	Input with pull up	JDTI
PC4*	Input with pull up	JTMS
PC4*	Bidirectional (push-pull output or floating input) controlled by debugger interface	SWDIO
*Note: The choice of PC4's forced signal is normally controlled by an external debug tool. JTMS is forced when the SWJ is in JTAG mode and SWDIO is forced when the SWJ is in Serial Wire mode. But, when GPIO_DEBUGDIS is set and PC4 is configured in SWDIO mode, then SWDIO is the only functionality available on PC4.		

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

PC0, PC2, PC3, and PC4 are forced to be the Serial Wire and JTAG (SWJ) Interface. When the EM358x resets, these four GPIOs are forced to operate in JTAG mode. Switching the debug interface between JTAG mode and Serial Wire mode can only be accomplished by the external debug tool and cannot be affected by software executing on the EM358x.

It is possible to either reclaim all of the four debugger pins (PC0, PC2, PC3, and P4), or reclaim the JTAG only debugger pins (PC0, PC2, and PC3) leaving Serial Wire operational.

Note: Disabling all debug functionality prevents external debug tools from operating, including flash programming and high-level debug tools.

Disabling the entire SWJ debugger interface is accomplished by setting the GPIO_DEBUGDIS bit in the GPIO_DBGCFG register and not having GPIO PC4 configured in SWDIO mode. In this configuration all debugger-related pins (PC0, PC2, PC3, PC4) behave as standard GPIOs.

Disabling only the JTAG debugger interface is accomplished by setting the GPIO_DEBUGDIS bit and configuring PC4 in SWDIO mode. When GPIO_DEBUGDIS is set and GPIO PC4 is in SWDIO mode, JTAG debugger-related pins (PC0, PC2, PC3) behave as standard GPIOs. Note that allowing the PC4 GPIO to operate as SWDIO does not affect the internal debug state of the chip.

If the SWJ debugger interface is already active (in either mode), the bit GPIO_DEBUGDIS cannot be set. When GPIO_DEBUGDIS is set, the SWJ debugger interface can be reclaimed by activating the SWJ while the EM358x is held in reset. If the SWJ debugger interface is forced active in this manner, the bit GPIO_FORCEDBG is set in the GPIO_DBGSTAT register. The SWJ debugger interface is defined as active when the CDBGPWRUPREQ signal, a bit in the debug port's CRTL/STAT register in the SWJ, is set high by an external debug tool.

If the SWJ debugger interface is active, and switched into Serial Wire mode (by the external debugger), then the JTAG only pins (PC0, PC2, PC3) behave as standard GPIOs. The use of SWDIO mode for GPIO PC4 allows reclaiming the JTAG only pins when an external debugger is not used.

7.4. Reset

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

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

7.5. Boot Configuration

nBOOTMODE is a special alternate function of PA5 that is active only during a pin reset (nRESET) or a power-on-reset of the always-powered domain (POR HV). If nBOOTMODE is asserted (pulled or driven low) when coming

SILICON LABS

out of reset, the processor starts executing an embedded serial-link-only monitor instead of its normal program.

While in reset and during the subsequent power-on-reset startup delay (512 OSCHF clocks), PA5 is automatically configured as an input with a pull-up resistor. At the end of this time, the EM358x samples nBOOTMODE: a high level selects normal boot mode, and a low level selects the embedded monitor. Figure 7 2 shows the timing parameters for invoking monitor mode from a pin (nRESET) reset. Because OSCHF is running uncalibrated during the reset sequence, the time for 512 OSCHF clocks may vary as indicated.
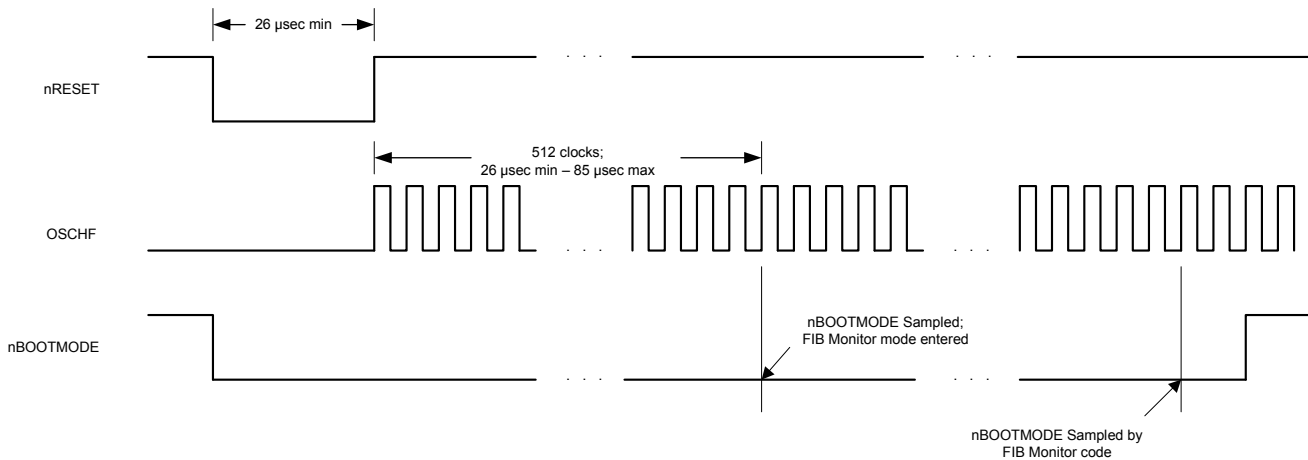
Figure 7.2. nBOOTMODE and nRESET Timing

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

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

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

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

7.6. GPIO Modes

7.6.1. Analog Mode

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

- PA0 and PA1 can be the differential IO pins for the USB device.
- PA4, PA5, PB5, PB6, PB7, and PC1 can be analog inputs to the ADC.
- PB0 can be an external analog voltage reference input to the ADC, or it can output the internal analog voltage reference from the ADC. The Silicon Labs software selects an internal or external voltage reference.
- PC6 and PC7 can connect to an optional 32.768 kHz crystal.

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

When configured in analog mode:

- The output drivers are disabled.

- The internal pull-up and pull-down resistors are disabled.
- The Schmitt trigger input is connected to a high logic level.
- Reading GPIO_PxIN returns a constant 1.

7.6.2. Input Mode

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

When configured in input mode:

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

7.6.3. SWDIO Mode

The SWDIO mode is only used with PC4 when the GPIO_DEBUGDIS bit in the GPIO_DBGCFG register is set.

Normally, the SWJ interface is a forced function of PC0, PC2, PC3, and PC4 so that the SWJ interface is always available. While the SWJ interface is being forced, the GPIO configurations of these four pins are ignored by the chip. The SWJ interface can be disabled in its entirety to reclaim these four pins as normal GPIO by setting the GPIO_DEBUGDIS bit. If the Serial Wire interface is desired but the JTAG interface is not, then PC4 can be configured in the SWDIO mode while GPIO_DEBUGDIS is set and therefore the Serial Wire interface will remain active.

7.6.4. Output Mode

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

When configured in output mode:

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

Note: Depending on configuration and usage, GPIO_PxOUT and GPIO_PxIN may not have the same value.

7.6.5. Alternate Output Mode

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

When configured in alternate output mode:

- The output drivers are enabled and are controlled by the output of an on-chip peripheral:
  - In open-drain mode: 0 activates the N-MOS current sink; 1 tri-states the pin.

- In push-pull mode: 0 activates the N-MOS current sink; 1 activates the P-MOS current source.
- ■ The internal pull-up and pull-down resistors are disabled.
- ■ The Schmitt trigger input is connected to the pin.
- ■ Reading GPIO_PxIN returns the input to the pin.

Note: Depending on configuration and usage, GPIO_PxOUT and GPIO_PxIN may not have the same value.

7.6.6. Alternate Output SPI Slave MISO Mode

This configuration mode is reserved for pins PB1 (SC1MISO) or PA1 (SC2MISO) when the associated serial controller is configured as an SPI slave. This configuration cannot be used with any other pins. This mode tri-states the pin when the respective SPI slave select signal (SCxnSSEL) is deasserted (goes high). When the SPI slave select signal is asserted (low), this pin functions as an alternate push-pull output.

7.7. Wake Monitoring

The GPIO_PxWAKE registers specify which GPIOs are monitored to wake the processor. If a GPIO's wake enable bit is set in GPIO_PxWAKE, then a change in the logic value of that GPIO causes the EM358x to wake from deep sleep. The logic values of all GPIOs are captured by hardware upon entering sleep. If any GPIO's logic value changes while in sleep and that GPIO's GPIO_PxWAKE bit is set, then the EM358x wakes from deep sleep. (There is no mechanism for selecting a specific rising-edge, falling-edge, or level on a GPIO: any change in logic value triggers a wake event.) Hardware records the fact that GPIO activity caused a wake event, but not which specific GPIO was responsible. Instead, the Silicon Labs software reads the state of the GPIOs on waking to determine this.

The register GPIO_WAKEFILT contains bits to enable digital filtering of the external wakeup event sources: the GPIO pins, SC1 activity, SC2 activity, and IRQD. The digital filter operates by taking samples based on the (nominal) 10 kHz RC oscillator. If three samples in a row all have the same logic value, and this sampled logic value is different from the logic value seen upon entering sleep, the filter outputs a wakeup event.

In order to use GPIO pins to wake the EM358x from deep sleep, the GPIO_WAKE bit in the WAKE_SEL register must be set. Waking up from GPIO activity does not work with pins configured for analog mode since the digital logic input is always set to 1 when in analog mode. Refer to Chapter 5, System Modules, for information on the EM358x's power management and sleep modes.

7.8. External Interrupts

The EM358x can use up to four external interrupt sources (IRQA, IRQB, IRQC, and IRQD), each with its own top-level NVIC interrupt vector. Since these external interrupt sources connect to the standard GPIO input path, an external interrupt pin may simultaneously be used by a peripheral device or even configured as an output. Analog mode is the only GPIO configuration that is not compatible with using a pin as an external interrupt.

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

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

Two of the four external interrupts, IRQA and IRQB, have fixed pin assignments. The other two external interrupts, IRQC and IRQD, can use any GPIO pin. The GPIO_IRQCSEL and GPIO_IRQDSEL registers specify the GPIO pins assigned to IRQC and IRQD, respectively. Table 7 4 shows how the GPIO_IRQCSEL and GPIO_IRQDSEL register values select the GPIO pin used for the external interrupt.

Table 7.4. IRQC/D GPIO Selection

GPIO_IRQxSEL	GPIO		GPIO_IRQxSEL	GPIO		GPIO_IRQxSEL	GPIO
0	PA0		11	PB3		22	PC6
1	PA1		12	PB4		23	PC7
2	PA2		13	PB5			
3	PA3		14	PB6			
4	PA4		15	PB7			
5	PA5		16	PC0			
6	PA6		17	PC1			
7	PA7		18	PC2			
8	PB0		19	PC3			
9	PB1		20	PC4			
10	PB2		21	PC5			

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

Refer to Chapter 3, Interrupt System, for further information regarding the EM358x interrupt system..

7.9. Debug Control and Status

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

SILICON LABS

7.10. GPIO Signal Assignment Summary

The GPIO signal assignments are shown in Table 7.5.

Table 7.5. GPIO Signal Assignments

GPIO	Analog	Alternate Output	Input	Output Current Drive
PA0	USBDM	TIM2C1[1], SC2MOSI	TIM2C1[1], SC2MOSI	Standard
PA1	USBDP	TIM2C3[1], SC2MISO, SC2SDA	TIM2C3[1], SC2MISO, SC2SDA	Standard
PA2		TIM2C4[1], SC2SCLK, SC2SCL	TIM2C4[1], SC2SCLK	Standard
PA3		TIM2C2[1]	TIM2C2[1], SC2nSSEL	Standard
PA4	ADC4	PTI_EN, TRACEDATA2		Standard
PA5	ADC5	PTI_DATA, TRACEDATA3	nBOOTMODE[2]	Standard
PA6		TIM1C3	TIM1C3	High
PA7		TIM1C4, REG_EN[3]	TIM1C4	High
PB0	VREF	TRACEDATA2	TIM1CLK, TIM2MSK, IRQA	Standard
PB1		TIM2C1[4], SC1TXD, SC1MOSI, SC1MISO, SC1SDA	TIM2C1[4], SC1SDA	Standard
PB2		TIM2C2[4]	TIM2C2[4], SC1MISO, SC1MOSI, SC1SCL, SC1RXD	Standard
PB3		TIM2C3[4], SC1SCLK	TIM2C3[4], SC1SCLK, SC1nCTS	Standard
PB4		TIM2C4[4], SC1nRTS	TIM2C4[4], SC1nSSEL	Standard
PB5	ADC0		TIM2CLK, TIM1MSK	Standard
PB6	ADC1	TIM1C1	TIM1C1, IRQB	High
PB7	ADC2	TIM1C2	TIM1C2	High
PC0		TRACEDATA1	JRST[5]	High
PC1	ADC3	TRACEDATA3		Standard
PC2		JTDO[6], SWO, TRACEDATA0		Standard
PC3		TRACECLK	JTDI[5]	Standard
PC4		SWDIO[7]	SWDIO[7], JTMS[7]	Standard
PC5		TX_ACTIVE		Standard
PC6	OSC32B	nTX_ACTIVE		Standard
PC7	OSC32A		OSC32_EXT	Standard

Note:
1. Default signal assignment (not remapped).
2. Overrides during reset as an input with pull up.
3. Overrides after reset as an open-drain output.
4. Alternate signal assignment (remapped).
5. Overrides in JTAG mode as a input with pull up.
6. Overrides in JTAG mode as a push-pull output.
7. Overrides in Serial Wire mode as either a push-pull output, or a floating input, controlled by the debugger

SILICON LABS

7.11. Registers

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

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

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

GPIO_PACFGL: Address: 0x4000B000   Reset: 0x4444
GPIO_PBCFGL: Address: 0x4000B200   Reset: 0x4444
GPIO_PCCFGL: Address: 0x4000B400   Reset: 0x4444

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

SILICON LABS

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

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

GPIO_PACFGH: Address: 0x4000B004   Reset: 0x4444
GPIO_PBCFGH: Address: 0x4000B204   Reset: 0x4444
GPIO_PCCFGH: Address: 0x4000B404   Reset: 0x4444

Bitname	Bitfield	Access	Description
Px7_CFG	[15:12]	RW	GPIO configuration control. 0x0: Analog, input or output (GPIO_PxIN always reads 1). 0x1: Output, push-pull (GPIO_PxOUT controls the output). 0x4: Input, floating. 0x5: Output, open-drain (GPIO_PxOUT controls the output). 0x6: SWDIO, bidirectional (only for retaining SWDIO functionality of PC4 when the GPIO_DEBUGDIS bit of the GPIO_DBGCFG register is set). 0x8: Input, pulled up or down (selected by GPIO_PxOUT: 0 = pull-down, 1 = pull-up). 0x9: Alternate output, push-pull (peripheral controls the output). 0xB: Alternate output SPI slave MISO, push-pull (only for SPI slave mode MISO) 0xD: Alternate output, open-drain (peripheral controls the output).
Px6_CFG	[11:8]	RW	GPIO configuration control: see Px7_CFG above.
Px5_CFG	[7:4]	RW	GPIO configuration control: see Px7_CFG above.
Px4_CFG	[3:0]	RW	GPIO configuration control: see Px7_CFG above.

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

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

GPIO_PAIN: Address: 0x4000B008   Reset: 0x0
GPIO_PBIN: Address: 0x4000B208   Reset: 0x0
GPIO_PCIN: Address: 0x4000B408   Reset: 0x0

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

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

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

GPIO_PAOUT: Address: 0x4000B00C   Reset: 0x0
GPIO_PBOUT: Address: 0x4000B20C   Reset: 0x0
GPIO_PCOUT: Address: 0x4000B40C   Reset: 0x0

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

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

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

GPIO_PACLR: Address: 0x4000B014   Reset: 0x0
GPIO_PBCLR: Address: 0x4000B214   Reset: 0x0
GPIO_PCCLR: Address: 0x4000B414   Reset: 0x0

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

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

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

GPIO_PASET: Address: 0x4000B010   Reset: 0x0
GPIO_PBSET: Address: 0x4000B210   Reset: 0x0
GPIO_PCSET: Address: 0x4000B410   Reset: 0x0

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

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

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

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

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

Register 7.8. GPIO_WAKEFILT: GPIO Wakeup Filtering Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	IRQD_WAKE_ FILTER	SC2_WAKE_ FILTER	SC1_WAKE_ FILTER	GPIO_WAKE_ FILTER

Address: 0x4000BC28 Reset: 0x0

Bitname	Bitfield	Access	Description
IRQD_WAKE_FILTER	[3]	RW	Enable filter on GPIO wakeup source IRQD.
SC2_WAKE_FILTER	[2]	RW	Enable filter on GPIO wakeup source SC2 (PA2).
SC1_WAKE_FILTER	[1]	RW	Enable filter on GPIO wakeup source SC1 (PB2).
GPIO_WAKE_FILTER	[0]	RW	Enable filter on GPIO wakeup sources enabled by the GPIO_PnWAKE registers.

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

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

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

GPIO_IRQCSEL: Address: 0x4000BC20 Reset: 0xF
GPIO_IRQDSEL: Address: 0x4000BC24 Reset: 0x10

Bitname	Bitfield	Access	Description
SEL_GPIO	[4:0]	RW	Pin assigned to IRQx. 0x00: PA0. 0x01: PA1. 0x02: PA2. 0x03: PA3. 0x04: PA4. 0x05: PA5. 0x06: PA6. 0x07: PA7. 0x08: PB0. 0x09: PB1. 0x0A: PB2. 0x0B: PB3. 0x0C: PB4. 0x0D: PB5. 0x0E: PB6. 0x0F: PB7. 0x10: PC0. 0x11: PC1. 0x12: PC2. 0x13: PC3. 0x14: PC4. 0x15: PC5. 0x16: PC6. 0x17: PC7. 0x18 – 0x1F: Reserved.

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

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

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	GPIO_INTFILT

Bit	7	6	5	4	3	2	1	0
Name	GPIO_INTMOD			0	0	0	0	0

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

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

SILICON LABS

Register 7.11. INT_GPIOFLAG: GPIO Interrupt Flag Register

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

Address: 0x4000A814   Reset: 0x0

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

Register 7.12. GPIO_DBGCFG: GPIO Debug Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	0	0	GPIO_DEBUGDIS	GPIO_EXTREGEN	GPIO_DBGCFGRSVD	0	0	0

Address: 0x4000BC00   Reset: 0x10

Bitname	Bitfield	Access	Description
GPIO_DEBUGDIS	[5]	RW	Disable debug interface override of normal GPIO configuration. Configuring PC4 in SWDIO mode will retain the Serial Wire SWDIO functionality. 0: Permit debug interface to be active. 1: Disable debug interface (if it is not already active).
GPIO_EXTREGEN	[4]	RW	Enable REG_EN override of PA7's normal GPIO configuration. 0: Disable override. 1: Enable override.
GPIO_DBGCFGRSVD	[3]	RW	Reserved: this bit can change during normal operation. When writing to GPIO_DBGCFG, the value of this bit must be preserved.

Register 7.13. GPIO_DBGSTAT: GPIO Debug Status Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	GPIO_BOOTMODE	0	GPIO_FORCEDBG	GPIO_SWEN

Address: 0x4000BC04   Reset: 0x0

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

8. Serial Controllers

8.1. Overview

The EM358x has two serial controllers, SC1 and SC2, which provide several options for full-duplex synchronous and asynchronous serial communications.

- SPI (Serial Peripheral Interface), master or slave
- TWI (Two Wire serial Interface), master only
- UART (Universal Asynchronous Receiver/Transmitter), SC1 only
- Receive and transmit FIFOs and DMA channels, SPI and UART modes

Receive and transmit FIFOs allow faster data speeds using byte-at-a-time interrupts. For the highest SPI and UART speeds, dedicated receive and transmit DMA channels reduce CPU loading and extend the allowable time to service a serial controller interrupt. Polled operation is also possible using direct access to the serial data registers. Figure 8 1 shows the components of the serial controllers..

Note: The notation SCx means that either SC1 or SC2 may be substituted to form the name of a specific register or field within a register.
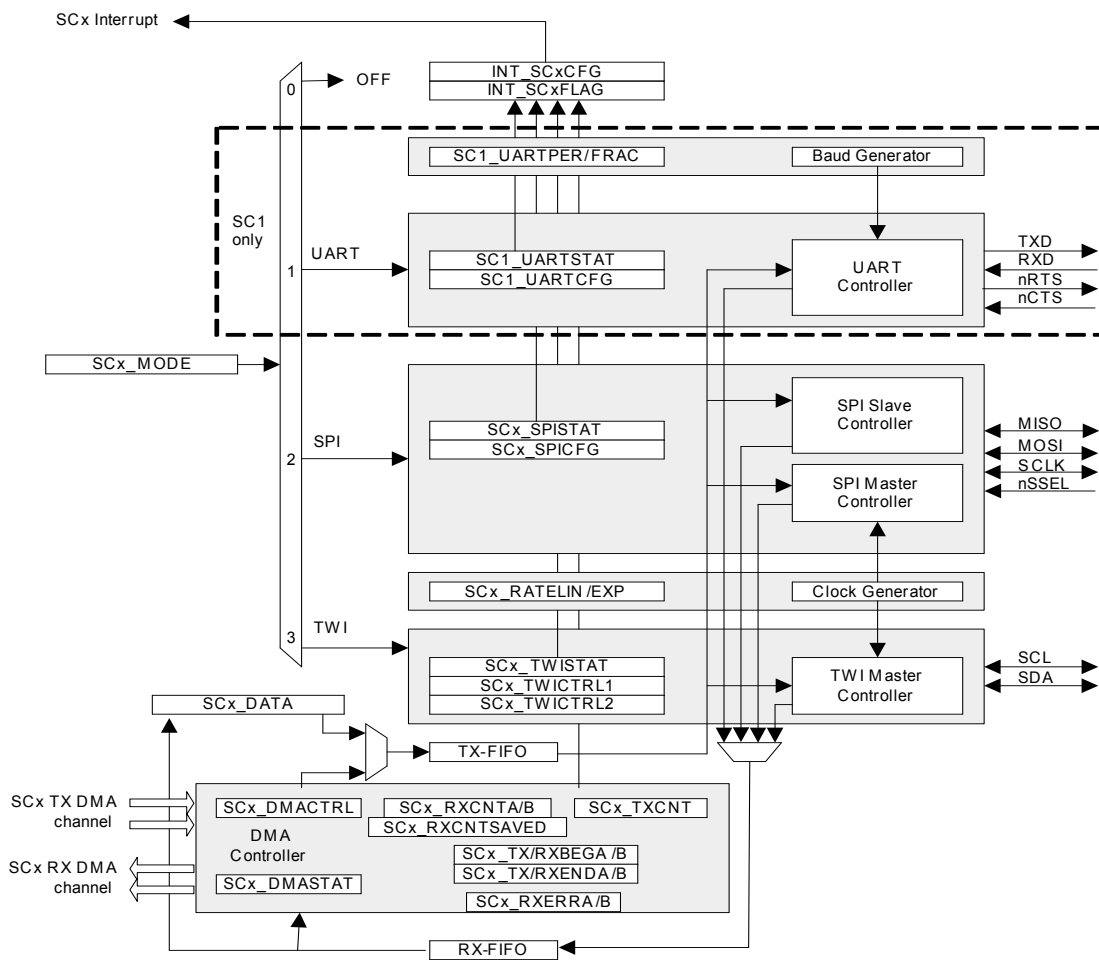

Figure 8.1. Serial Controller Block Diagram

8.2. Configuration

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

1. Set up the parameters specific to the operating mode (master/slave for SPI, baud rate for UART, etc.).

2. Configure the GPIO pins used by the serial controller as shown in Table 8.1 and Table 8.2. Section 2 in Chapter 7, GPIO shows how to configure GPIO pins.

3. If using DMA, set up the DMA and buffers. This is described fully in Section 8.7.

4. If using interrupts, select edge- or level-triggered interrupts with the SCx_INTMODE register, enable the desired second-level interrupt sources in the INT_SCxCFG register, and finally enable the top-level SCx interrupt in the NVIC.

5. Write the serial interface operating mode — SPI, TWI, or UART — to the SCx_MODE register.

Table 8.1. SC1 GPIO Usage and Configuration

	PB1	PB2	PB3	PB4
SPI - Master	SC1MOSI Alternate Output (push-pull)	SC1MISO Input	SC1SCLK Alternate Output (push-pull)	(not used)
SPI - Slave	SC1MISO Alternate Output (push-pull), SPI Slave MISO Mode	SC1MOSI Input	SC1SCLK Input	SC1nSSEL Input
TWI - Master	SC1SDA Alternate Output (open-drain)	SC1SCL Alternate Output (open-drain)	(not used)	(not used)
UART	TXD Alternate Output (push-pull)	RXD Input	nCTS Input[1]	nRTS Alternate Output (push-pull)[1]
Note:				
1. Used if RTS/CTS hardware flow control is enabled.				

Table 8.2. SC2 GPIO Usage and Configuration

	PA0	PA1	PA2	PA3
SPI - Master	SC2MOSI Alternate Output (push-pull)	SC2MISO Input	SC2SCLK Alternate Output (push-pull)	(not used)
SPI - Slave	SC2MOSI Input	SC2MISO Alternate Output (push-pull), SPI Slave MISO Mode	SC2SCLK Input	SC2nSSEL Input
TWI - Master	(not used)	SC2SDA Alternate Output (open-drain)	SC2SCL Alternate Output (open-drain)	(not used)

8.2.1. Registers

Note: Substitute "1" or "2" for "x" in the following detailed descriptions.

Register 8.1. SCx_MODE
SC1_MODE: Serial Mode Register
SC2_MODE: Serial Mode Register

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

SC1_MODE: Address: 0x4000C854   Reset: 0x0
SC2_MODE: Address: 0x4000C054   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_MODE	[1:0]	RW	Serial controller mode. 0: Disabled. 1: UART mode (valid only for SC1). 2: SPI mode. 3: TWI mode.

Register 8.2. INT_SCxFLAG
INT_SC1FLAG: Serial Controller 1 Interrupt Flag Register
INT_SC2FLAG: Serial Controller 2 Interrupt Flag Register

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

INT_SC1FLAG: Address: 0x4000A808   Reset: 0x0
INT_SC2FLAG: Address: 0x4000A80C   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_SC1PARERR	[14]	RW	Parity error received (UART) interrupt pending.
INT_SC1FRMERR	[13]	RW	Frame error received (UART) interrupt pending.
INT_SCTXULDB	[12]	RW	DMA transmit buffer B unloaded interrupt pending.
INT_SCTXULDA	[11]	RW	DMA transmit buffer A unloaded interrupt pending.
INT_SCRXULDB	[10]	RW	DMA receive buffer B unloaded interrupt pending.
INT_SCRXULDA	[9]	RW	DMA receive buffer A unloaded interrupt pending.
INT_SCNAK	[8]	RW	NACK received (TWI) interrupt pending.
INT_SCCMDFIN	[7]	RW	START/STOP command complete (TWI) interrupt pending.
INT_SCTXFIN	[6]	RW	Transmit operation complete (TWI) interrupt pending.
INT_SCRXFIN	[5]	RW	Receive operation complete (TWI) interrupt pending.
INT_SCTXUND	[4]	RW	Transmit buffer underrun interrupt pending.
INT_SCRXOVF	[3]	RW	Receive buffer overrun interrupt pending.
INT_SCTXIDLE	[2]	RW	Transmitter idle interrupt pending.
INT_SCTXFREE	[1]	RW	Transmit buffer free interrupt pending.
INT_SCRXVAL	[0]	RW	Receive buffer has data interrupt pending.

Register 8.3. INT_SCxCFG
INT_SC1CFG: Serial Controller 1 Interrupt Configuration Register
INT_SC2CFG: Serial Controller 2 Interrupt Configuration Register

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

INT_SC1CFG: Address: 0x4000A848   Reset: 0x0
INT_SC2CFG: Address: 0x4000A84C   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_SC1PARERR	[14]	RW	Parity error received (UART) interrupt enable.
INT_SC1FRMERR	[13]	RW	Frame error received (UART) interrupt enable.
INT_SCTXULDB	[12]	RW	DMA transmit buffer B unloaded interrupt enable.
INT_SCTXULDA	[11]	RW	DMA transmit buffer A unloaded interrupt enable.
INT_SCRXULDB	[10]	RW	DMA receive buffer B unloaded interrupt enable.
INT_SCRXULDA	[9]	RW	DMA receive buffer A unloaded interrupt enable.
INT_SCNAK	[8]	RW	NACK received (TWI) interrupt enable.
INT_SCCMDFIN	[7]	RW	START/STOP command complete (TWI) interrupt enable.
INT_SCTXFIN	[6]	RW	Transmit operation complete (TWI) interrupt enable.
INT_SCRXFIN	[5]	RW	Receive operation complete (TWI) interrupt enable.
INT_SCTXUND	[4]	RW	Transmit buffer underrun interrupt enable.
INT_SCRXOVF	[3]	RW	Receive buffer overrun interrupt enable.
INT_SCTXIDLE	[2]	RW	Transmitter idle interrupt enable.
INT_SCTXFREE	[1]	RW	Transmit buffer free interrupt enable.
INT_SCRXVAL	[0]	RW	Receive buffer has data interrupt enable.

Register 8.4. SCx_INTMODE
SC1_INTMODE: Serial Controller 1 Interrupt Mode Register
SC2_INTMODE: Serial Controller 2 Interrupt Mode Register

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

SC1_INTMODE: Address: 0x4000A854   Reset: 0x0
SC2_INTMODE: Address: 0x4000A858   Reset: 0x0

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

8.3. SPI—Master Mode

The SPI master controller has the following features:

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

8.3.1. GPIO Usage

The SPI master controller uses the three signals:

- MOSI (Master Out, Slave In) - outputs serial data from the master
- MISO (Master In, Slave Out) - inputs serial data from a slave
- SCLK (Serial Clock) - outputs the serial clock used by MOSI and MISO

The GPIO pins used for these signals are shown in Table 8.3. Additional outputs may be needed to drive the nSSEL signals on slave devices.

Table 8.3. SPI Master GPIO Usage

	MOSI	MISO	SCLK
Direction	Output	Input	Output
GPIO Configuration	Alternate Output (push-pull)	Input	Alternate Output (push-pull)
SC1 pin	PB1	PB2	PB3
SC2 pin	PA0	PA1	PA2

8.3.2. Set Up and Configuration

The serial controllers support SPI master mode. SPI master mode is enabled by the following register settings:

- The serial controller mode register (SCx_MODE) is 2.
- The SC_SPIMST bit in the SPI configuration register (SCx_SPICFG) is 1.

The SPI serial clock (SCLK) is produced by a programmable clock generator. The serial clock is produced by dividing down 12 MHz according to this equation:

$$rate = \frac{12\ MHz}{(LIN + 1) \times 2^{EXP}}$$

EXP is the value written to the SCx_RATEEXP register, and LIN is the value written to the SCx_RATELIN register. EXP and LIN can both be zero, so the SPI master mode clock may be 12 Mbps.

The SPI master controller supports various frame formats depending upon the clock polarity (SC_SPIPOL), clock phase (SC_SPIPHA), and direction of data (SC_SPIORD) (see Table 8.4). The bits SC_SPIPOL, SC_SPIPHA, and SC_SPIORD are defined within the SCx_SPICFG register.

Table 8.4. SPI Master Mode Formats

<table>
<tr><th colspan="4">SCx_SPICFG</th><th rowspan="3">Frame Formats</th></tr>
<tr><th colspan="4">SC_SPIxxx*</th></tr>
<tr><th>MST</th><th>ORD</th><th>PHA</th><th>POL</th></tr>
<tr>
<td>1</td><td>0</td><td>0</td><td>0</td>
<td>

SCLK_{out}
MOSI_{out}: TX[7] TX[6] TX[5] TX[4] TX[3] TX[2] TX[1] TX[0]
MISO_{in}: RX[7] RX[6] RX[5] RX[4] RX[3] RX[2] RX[1] RX[0]

</td>
</tr>
<tr>
<td>1</td><td>0</td><td>0</td><td>1</td>
<td>

SCLK_{out}
MOSI_{out}: TX[7] TX[6] TX[5] TX[4] TX[3] TX[2] TX[1] TX[0]
MISO_{in}: RX[7] RX[6] RX[5] RX[4] RX[3] RX[2] RX[1] RX[0]

</td>
</tr>
<tr>
<td>1</td><td>0</td><td>1</td><td>0</td>
<td>

SCLK_{out}
MOSI_{out}: TX[7] TX[6] TX[5] TX[4] TX[3] TX[2] TX[1] TX[0]
MISO_{in}: RX[7] RX[6] RX[5] RX[4] RX[3] RX[2] RX[1] RX[0]

</td>
</tr>
<tr>
<td>1</td><td>0</td><td>1</td><td>1</td>
<td>

SCLK_{out}
MOSI_{out}: TX[7] TX[6] TX[5] TX[4] TX[3] TX[2] TX[1] TX[0]
MISO_{in}: RX[7] RX[6] RX[5] RX[4] RX[3] RX[2] RX[1] RX[0]

</td>
</tr>
<tr>
<td>1</td><td>1</td><td>—</td><td>—</td>
<td>Same as above except data is sent LSB first instead of MSB first</td>
</tr>
<tr>
<td colspan="5">*Note: The notation xxx means that the corresponding column header below is inserted to form the field name.</td>
</tr>
</table>

8.3.3. Operation

Characters transmitted and received by the SPI master controller are buffered in transmit and receive FIFOs that are both 4 entries deep. When software writes a character to the SCx_DATA register, the character is pushed onto the transmit FIFO. Similarly, when software reads from the SCx_DATA register, the character returned is pulled from the receive FIFO. If the transmit and receive DMA channels are used, they also write to and read from the transmit and receive FIFOs.

When the transmit FIFO and the serializer are both empty, writing a character to the transmit FIFO clears the SC_SPITXIDLE bit in the SCx_SPISTAT register. This indicates that some characters have not yet been transmitted. If characters are written to the transmit FIFO until it is full, the SC_SPITXFREE bit in the SCx_SPISTAT register is cleared. Shifting out a character to the MOSI pin sets the SC_SPITXFREE bit in the SCx_SPISTAT register. When the transmit FIFO empties and the last character has been shifted out, the SC_SPITXIDLE bit in the SCx_SPISTAT register is set.

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

To receive a character, you must transmit a character. If a long stream of receive characters is expected, a long sequence of dummy transmit characters must be generated. To avoid software or transmit DMA initiating these transfers and consuming unnecessary bandwidth, the SPI serializer can be instructed to retransmit the last

SILICON LABS

transmitted character or to transmit a busy token (0xFF), which is determined by the SC_SPIRPT bit in the SCx_SPICFG register. This functionality can only be enabled or disabled when the transmit FIFO is empty and the transmit serializer is idle, indicated by a cleared SC_SPITXIDLE bit in the SCx_SPISTAT register. Refer to the register description of SCx_SPICFG for more detailed information about SC_SPIRPT.

Every time an automatic character transmission starts, a transmit underrun is detected as there is no data in the transmit FIFO, and the INT_SCTXUND bit in the INT_SCxFLAG register is set. After automatic character transmission is disabled, no more new characters are received. The receive FIFO holds characters just received.

Note: The Receive DMA complete event does not always mean the receive FIFO is empty.

"8.7. DMA Channels" on page 109 describes how to configure and use the serial receive and transmit DMA channels.

8.3.4. Interrupts

SPI master controller second-level interrupts are generated by the following events:

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

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

8.3.5. Registers

Note: Substitute "1" or "2" for "x" in the following detailed descriptions.

Register 8.5. SCx_DATA
SC1_DATA: Serial Data Register
SC2_DATA: Serial Data Register

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

SC1_DATA: Address: 0x4000C83C   Reset: 0x0
SC2_DATA: Address: 0x4000C03C   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_DATA	[7:0]	RW	Transmit and receive data register. Writing to this register adds a byte to the transmit FIFO. Reading from this register takes the next byte from the receive FIFO and clears the overrun error bit if it was set. In UART mode (SC1 only), reading from this register loads the UART status register with the parity and frame error status of the next byte in the FIFO, and clears these bits if the FIFO is now empty.

SILICON LABS

Register 8.6. SCx_SPICFG
SC1_SPICFG: SPI Configuration Register
SC2_SPICFG: SPI Configuration Register

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

SC1_SPICFG: Address: 0x4000C858   Reset: 0x0
SC2_SPICFG: Address: 0x4000C058   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_SPIRXDRV	[5]	RW	Receiver-driven mode selection bit (SPI master mode only). Clear this bit to initiate transactions when transmit data is available. Set this bit to initiate transactions when the receive buffer (FIFO or DMA) has space.
SC_SPIMST	[4]	RW	Set this bit to put the SPI in master mode, clear this bit to put the SPI in slave mode.
SC_SPIRPT	[3]	RW	This bit controls behavior when the transmit serializer must send a byte and there is no data already available in/to the serializer. The conditions for sending this "busy" token are transmit buffer underrun condition when using DMA in master or slave mode, empty FIFO in slave mode, and the busy token will always be sent as the first byte every time nSSEL is asserted while operating in slave mode. Clear this bit to send the BUSY token (0xFF) and set this bit to repeat the last byte. Changes to this bit take effect when the transmit FIFO is empty and the transmit serializer is idle. Note that when the chip comes out of reset, if SC_SPIRPT is set before any data has been transmitted and no data is available (in the FIFO), the "last byte" that will be transmitted after the padding byte is 0x00 due to the FIFO having been reset to 0x00.
SC_SPIORD	[2]	RW	This bit specifies the bit order in which SPI data is transmitted and received. 0: Most significant bit first. 1: Least significant bit first.
SC_SPIPHA	[1]	RW	Clock phase configuration: clear this bit to sample on the leading (first edge) and set this bit to sample on the second edge.
SC_SPIPOL	[0]	RW	Clock polarity configuration: clear this bit for a rising leading edge and set this bit for a falling leading edge.

Register 8.7. SCx_SPISTAT
SC1_SPISTAT: SPI Status Register
SC2_SPISTAT: SPI Status Register

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

SC1_SPISTAT: Address: 0x4000C840   Reset: 0x0
SC2_SPISTAT: Address: 0x4000C040   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_SPITXIDLE	[3]	R	This bit is set when both the transmit FIFO and the transmit serializer are empty.
SC_SPITXFREE	[2]	R	This bit is set when the transmit FIFO has space to accept at least one byte.
SC_SPIRXVAL	[1]	R	This bit is set when the receive FIFO contains at least one byte.
SC_SPIRXOVF	[0]	R	This bit is set if a byte is received when the receive FIFO is full. This bit is cleared by reading the data register.

Register 8.8. SCx_RATELIN
SC1_RATELIN: Serial Clock Linear Prescaler Register
SC2_RATELIN: Serial Clock Linear Prescaler Register

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

SC1_RATELIN: Address: 0x4000C860   Reset: 0x0
SC2_RATELIN: Address: 0x4000C060   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_RATELIN	[3:0]	RW	The linear component (LIN) of the clock rate in the equation: $$\text{rate} = \frac{12\text{ MHz}}{(\text{LIN} + 1) \times 2^{\text{EXP}}}$$

Register 8.9. SCx_RATEEXP
SC1_RATEEXP: Serial Clock Exponential Prescaler Register
SC2_RATEEXP: Serial Clock Exponential Prescaler Register

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

SC1_RATEEXP: Address: 0x4000C864   Reset: 0x0
SC2_RATEEXP: Address: 0x4000C064   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_RATEEXP	[3:0]	RW	The exponential component (EXP) of the clock rate in the equation: $$\text{rate} = \frac{12\,\text{MHz}}{(\text{LIN} + 1) \times 2^{\text{EXP}}}$$

8.4. SPI—Slave Mode

The SPI controller has the following features:

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

8.4.1. GPIO Usage

The SPI slave controller uses four signals:

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

The GPIO pins that can be assigned to these signals are shown in Table 8.5.

Table 8.5. SPI Slave GPIO Usage

	MOSI	MISO	SCLK	nSSEL
Direction	Input	Output	Input	Input
GPIO Configuration	Input	Alternate Output (push-pull), SPI Slave MISO Mode	Input	Input
SC1 pin	PB2	PB1	PB3	PB4
SC2 pin	PA0	PA1	PA2	PA3

SILICON LABS

8.4.2. Set Up and Configuration

The serial controllers support SPI slave mode. SPI slave mode is enabled by the following register settings:

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

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

The SPI slave controller supports various frame formats depending upon the clock polarity (SC_SPIPOL), clock phase (SC_SPIPHA), and direction of data (SC_SPIORD) (see Table 8.6). The SC_SPIPOL, SC_SPIPHA, and SC_SPIORD bits are defined within the SCx_SPICFG registers.

Table 8.6. SPI Slave Formats

SCx_SPICFG				Frame Format
SC_SPIxxx*				
MST	ORD	PHA	POL	
0	0	0	0	nSSEL / SCLK$_{in}$ / MOSI$_{in}$: RX[7] RX[6] RX[5] RX[4] RX[3] RX[2] RX[1] RX[0] / MISO$_{out}$: TX[7] TX[6] TX[5] TX[4] TX[3] TX[2] TX[1] TX[0]
0	0	0	1	SCLK$_{in}$ / MOSI$_{in}$: RX[7] RX[6] RX[5] RX[4] RX[3] RX[2] RX[1] RX[0] / MISO$_{out}$: TX[7] TX[6] TX[5] TX[4] TX[3] TX[2] TX[1] TX[0]
0	0	1	0	nSSEL / SCLK$_{in}$ / MOSI$_{in}$: RX[7] RX[6] RX[5] RX[4] RX[3] RX[2] RX[1] RX[0] / MISO$_{out}$: TX[7] TX[6] TX[5] TX[4] TX[3] TX[2] TX[1] TX[0]
0	0	1	1	nSSEL / SCLK$_{in}$ / MOSI$_{in}$: RX[7] RX[6] RX[5] RX[4] RX[3] RX[2] RX[1] RX[0] / MISO$_{out}$: TX[7] TX[6] TX[5] TX[4] TX[3] TX[2] TX[1] TX[0]
0	1	—	—	Same as above except LSB first instead of MSB first
*Note: The notation "xxx" means that the corresponding column header below is inserted to form the field name.				

8.4.3. Operation

When the slave select (nSSEL) signal is asserted by the master, SPI transmit data is driven to the output pin MISO, and SPI data is received from the input pin MOSI. The nSSEL pin has to be asserted to enable the transmit serializer to drive data to the output signal MISO. When the nSSEL pin is deasserted, no data is transferred on the MISO or MOSI pins and the output pin MISO is tri-stated (when the MISO pin is configured as Alternate Output (push-pull), SPI Slave MISO Mode). A falling edge on nSSEL resets the SPI slave shift registers.

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

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

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

When the transmit FIFO and the serializer are both empty, writing a character to the transmit FIFO clears the SC_SPITXIDLE bit in the SCx_SPISTAT register. This indicates that not all characters have been transmitted. If characters are written to the transmit FIFO until it is full, the SC_SPITXFREE bit in the SCx_SPISTAT register is cleared. Shifting out a transmit character to the MISO pin causes the SC_SPITXFREE bit in the SCx_SPISTAT register to get set. When the transmit FIFO empties and the last character has been shifted out, the SC_SPITXIDLE bit in the SCx_SPISTAT register is set.

The SPI slave controller must guarantee that there is time to move new transmit data from the transmit FIFO into the hardware serializer. To provide sufficient time, the SPI slave controller inserts a byte of padding at the start of every new string of transmit data defined by **every time nSSEL is asserted**. This byte is inserted as if this byte was placed there by software. The value of the byte of padding is always 0xFF.

8.4.4. DMA

The DMA Channels section describes how to configure and use the serial receive and transmit DMA channels.

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

8.4.5. Interrupts

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

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

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

8.4.6. Registers

Refer to Registers (in the SPI Master Mode "8.3. SPI—Master Mode" on page 82) for a description of the SCx_DATA, SCx_SPICFG, and SCx_SPISTAT registers.

8.5. TWI—Two Wire Serial Interfaces

The Two Wire serial Interface (TWI) master controller has the following features:

- Uses only two bidirectional GPIO pins
- Programmable clock frequency (up to 400 kHz)
- Supports both 7-bit and 10-bit addressing
- Compatible with Philips' I^2C-bus slave devices

8.5.1. GPIO Usage

The TWI master controller uses just two signals:

- SDA (Serial Data) - bidirectional serial data
- SCL (Serial Clock) - bidirectional serial clock

Table 8.7 lists the GPIO pins used by the SC1 and SC2 TWI master controllers. Because the pins are configured as open-drain outputs, they require external pull-up resistors.

Table 8.7. TWI Master GPIO Usage

	SDA	SCL
Direction	Input / Output	Input / Output
GPIO Configuration	Alternate Output (Open Drain)	Alternate Output (Open Drain)
SC1 Pin	PB1	PB2
SC2 Pin	PA1	PA2

8.5.2. Set Up and Configuration

The TWI controller is enabled by writing 3 to the SCx_MODE register. The TWI controller operates only in master mode and supports both Standard (100 kbps) and Fast (400 kbps) TWI modes. Address arbitration is not implemented, so multiple master applications are not supported.

The TWI master controller's serial clock (SCL) is produced by a programmable clock generator. SCL is produced by dividing down 12 MHz according to this equation:

$$rate = \frac{12\ \text{MHz}}{(LIN + 1) \times 2^{EXP}}$$

EXP is the value written to the SCx_RATEEXP register and LIN is the value written to the SCx_RATELIN register. Table 8.8 shows the rate settings for Standard-Mode TWI (100 kbps) and Fast-Mode TWI (400 kbps) operation.

Table 8.8. TWI Clock Rate Programming

Clock Rate	SCx_RATELIN	SCx_RATEEXP
100 kbps	14	3
375 kbps*	15	1
400 kbps*	14	1
*Note: At 400 kbps, the Philips I^{2}C Bus specification requires the minimum low period of SCL to be 1.3 µs, but on the EM358x it is 1.25 µs. If a slave device requires strict compliance with SCL timing, the clock rate must be lowered to 375 kbps.		

The EM358x supports clock stretching. The slave device can hold SCL low on any received or transmitted data bit. This inhibits further data transfers until SCL is allowed to go high again.

SILICON LABS

8.5.3. Constructing Frames

The TWI master controller supports generating various frame segments by means of the SC_TWISTART, SC_TWISTOP, SC_TWISEND, and SC_TWIRECV bits in the SCx_TWICTRL1 registers. Table 8.9 summarizes these frames.

Table 8.9. TWI Master Frame Segments

SCx_TWICTRL1				Frame Segments
SC_TWIxxxx*				
START	SEND	RECV	STOP	
1	0	0	0	TWI start segment / TWI re-start segment - after transmit or frame with NACK
0	1	0	0	TWI transmit segment - after (re-)start frame / TWI transmit segment – after transmit with ACK
0	0	1	0	TWI receive segment – transmit with ACK / TWI receive segment - after receive with ACK
0	0	0	1	TWI stop segment - after frame with NACK or stop
0	0	0	0	No pending frame segment
1	1	—	—	Illegal
—	1	1	—	
—	—	1	1	
1	—	—	1	
Note: The notation "xxx" means that the corresponding column header below is inserted to form the field name.				

SILICON LABS

Full TWI frames have to be constructed by software from individual TWI segments. All necessary segment transitions are shown in Figure 8.2. ACK or NACK generation of a TWI receive frame segment is determined with the SC_TWIACK bit in the SCx_TWICTRL2 register.
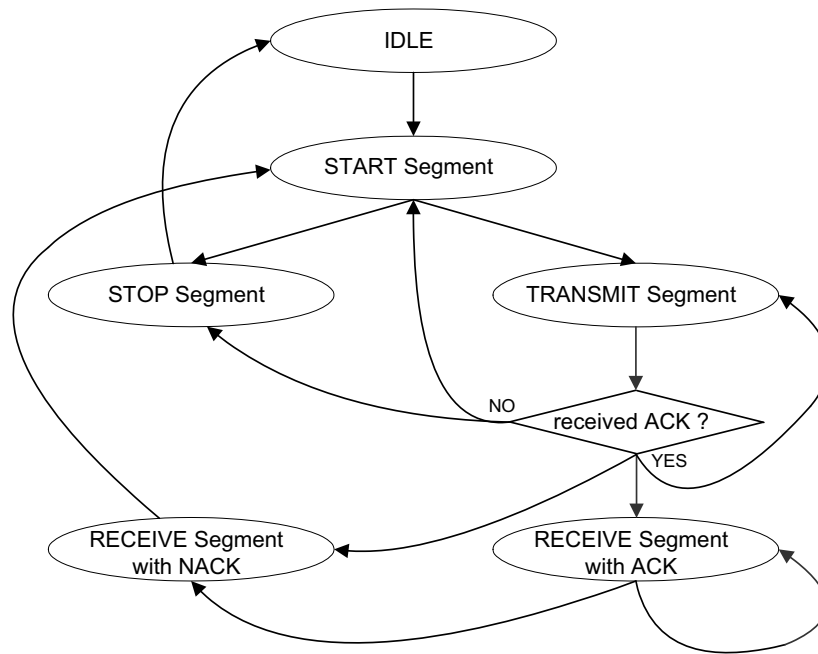

Figure 8.2. TWI Segment Transitions

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

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

Transmitted and received characters are accessed through the SCx_DATA register.

To initiate (re)start and stop segments, set the SC_TWISTART or SC_TWISTOP bit in the SCx_TWICTRL1 register, then wait until the bit is clear. Alternatively, the SC_TWICMDFIN bit in the SCx_TWISTAT can be used for waiting.

To initiate a transmit segment, write the data to the SCx_DATA data register, then set the SC_TWISEND bit in the SCx_TWICTRL1 register, and finally wait until the bit is clear. Alternatively the SC_TWITXFIN bit in the SCx_TWISTAT register can be used for waiting.

To initiate a receive segment, set the SC_TWIRECV bit in the SCx_TWICTRL1 register, wait until it is clear, and then read from the SCx_DATA register. Alternatively, the SC_TWIRXFIN bit in the SCx_TWISTAT register can be used for waiting. Now the SC_TWIRXNAK bit in the SCx_TWISTAT register indicates if a NACK or ACK was received from a TWI slave device..

8.5.4. Interrupts

TWI master controller interrupts are generated on the following events:

- Bus command (SC_TWISTART/SC_TWISTOP) completed (0 to 1 transition of SC_TWICMDFIN)
- Character transmitted and slave device responded with NACK
- Character transmitted (0 to 1 transition of SC_TWITXFIN)
- Character received (0 to 1 transition of SC_TWIRXFIN)
- Received and lost character while receive FIFO was full (receive overrun error)
- Transmitted character while transmit FIFO was empty (transmit underrun error)

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

8.5.5. Registers

Refer to "8.3.5. Registers" on page 85 (in "8.3. SPI—Master Mode") for a description of the SCx_DATA, SCx_RATELIN, and SCx_RATEEXP registers.

Note: Substitute "1" or "2" for "x" in the following detailed descriptions.

Register 8.10. SCx_TWISTAT
SC1_TWISTAT: TWI Status Register
SC2_TWISTAT: TWI Status Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	SC_TWICMDFIN	SC_TWIRXFIN	SC_TWITXFIN	SC_TWIRXNAK

SC1_TWISTAT: Address: 0x4000C844   Reset: 0x0
SC2_TWISTAT: Address: 0x4000C044   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_TWICMDFIN	[3]	R	This bit is set when a START or STOP command completes. It clears on the next TWI bus activity.
SC_TWIRXFIN	[2]	R	This bit is set when a byte is received. It clears on the next TWI bus activity.
SC_TWITXFIN	[1]	R	This bit is set when a byte is transmitted. It clears on the next TWI bus activity.
SC_TWIRXNAK	[0]	R	This bit is set when a NACK is received from the slave. It clears on the next TWI bus activity.

Register 8.11. SCx_TWICTRL1
SC1_TWICTRL1: TWI Control Register 1
SC2_TWICTRL1: TWI Control Register 1

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

SC1_TWICTRL1: Address: 0x4000C84C   Reset: 0x0
SC2_TWICTRL1: Address: 0x4000C04C   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_TWISTOP	[3]	RW	Setting this bit sends the STOP command. It clears when the command completes.
SC_TWISTART	[2]	RW	Setting this bit sends the START or repeated START command. It clears when the command completes.
SC_TWISEND	[1]	RW	Setting this bit transmits a byte. It clears when the command completes.
SC_TWIRECV	[0]	RW	Setting this bit receives a byte. It clears when the command completes.

Rev 1.1

Register 8.12. SCx_TWICTRL2
SC1_TWICTRL2: TWI Control Register 2
SC2_TWICTRL2: TWI Control Register 2

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

SC1_TWICTRL2: Address: 0x4000C850   Reset: 0x0
SC2_TWICTRL2: Address: 0x4000C050   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_TWIACK	[0]	RW	Setting this bit signals ACK after a received byte. Clearing this bit signals NACK after a received byte.

8.6. UART—Universal Asynchronous Receiver/Transmitter

The SC1 UART is enabled by writing 1 to SC1_MODE. The SC2 serial controller does not include UART functions.

The UART supports the following features:

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

8.6.1. GPIO Usage

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

- TXD (Transmitted Data) - serial data sent by the EM358x
- RXD (Received Data) - serial data received by the EM358x

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

- nRTS (Request To Send) - indicates the EM358x is able to receive data
- nCTS (Clear To Send) - inhibits sending data from the EM358x if not asserted

The GPIO pins assigned to these signals are shown in Table 8.10.

Table 8.10. UART GPIO Usage

	TXD	RXD	nCTS[1]	nRTS[1]
Direction	Output	Input	Input	Output
GPIO Configuration	Alternate Output (push-pull)	Input	Input	Alternate Output (push-pull)
SC1 pin	PB1	PB2	PB3	PB4

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

SILICON LABS

8.6.2. Set Up and Configuration

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

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

The integer portion of the divisor, N, is written to the SC1_UARTPER register and the fractional part, F, to the SC1_UARTFRAC register. Table 8 11 shows the values used to generate some common baud rates and their associated clock frequency error. The UART requires an internal clock that is at least eight times the baud rate clock, so the minimum allowable setting for SC1_UARTPER is 8.

Table 8.11. UART Baud Rate Divisors for Common Baud Rates

Baud Rate (bits/sec)	SC1_UARTPER	SC1_UARTFRAC	Baud Rate Error (%)
300	40000	0	0
2400	5000	0	0
4800	2500	0	0
9600	1250	0	0
19200	625	0	0
38400	312	1	0
57600	208	1	– 0.08
115200	104	0	+ 0.16
230400	52	0	+ 0.16
460800	26	0	+ 0.16
921600	13	0	+ 0.16

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

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

Errors may occur when:

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

Where:

T_{gap}= inter-byte gap in seconds

baud = baud rate in bps

Ferror = relative frequency error in ppm

For example, if the baud rate tolerance between receive and transmit is 200 ppm (reasonable if both sides are

SILICON LABS

derived from a crystal), and the baud rate is 115200 bps, then errors will not occur until the inter-byte gap exceeds 43 ms. If the gap is exceeded then the chance of an error is essentially random, with a probability of approximately P = baud / 24e6. At 115200 bps, the probability of corruption is 0.5%.

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

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

A UART character frame contains, in sequence:

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

Figure 8.3 shows the UART character frame format, with optional bits indicated. Depending on the options chosen for the character frame, the length of a character frame ranges from 9 to 12 bit times.

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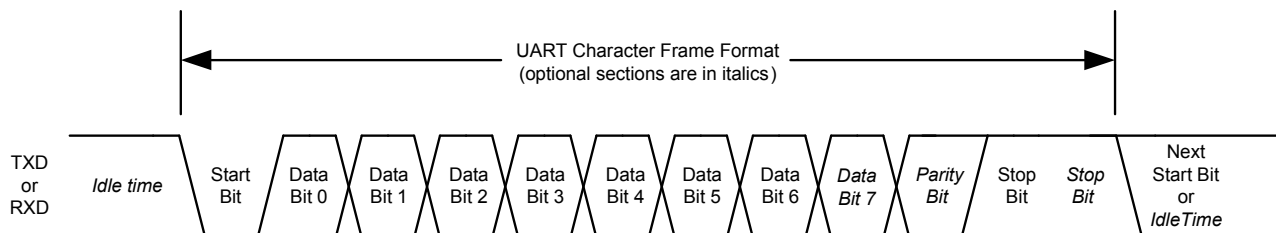

Figure 8.3. UART Character Frame Format

8.6.3. FIFOs

Characters transmitted and received by the UART are buffered in the transmit and receive FIFOs that are both 4 entries deep (see Figure 8.4). When software writes a character to the SC1SCy_DATA register, it is pushed onto the transmit FIFO. Similarly, when software reads from the SC1_DATA register, the character returned is pulled from the receive FIFO. If the transmit and receive DMA channels are used, the DMA channels also write to and read from the transmit and receive FIFOs.
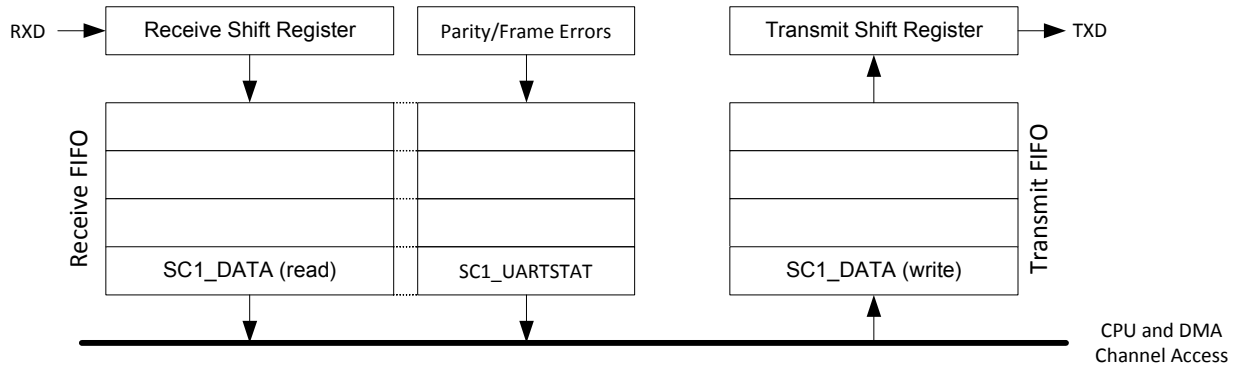

Figure 8.4. UART FIFOs

8.6.4. RTS/CTS Flow control

RTS/CTS flow control, also called hardware flow control, uses two signals (nRTS and nCTS) in addition to received and transmitted data (see Figure 8.5). Flow control is used by a data receiver to prevent buffer overflow, by signaling an external device when it is and is not allowed to transmit.
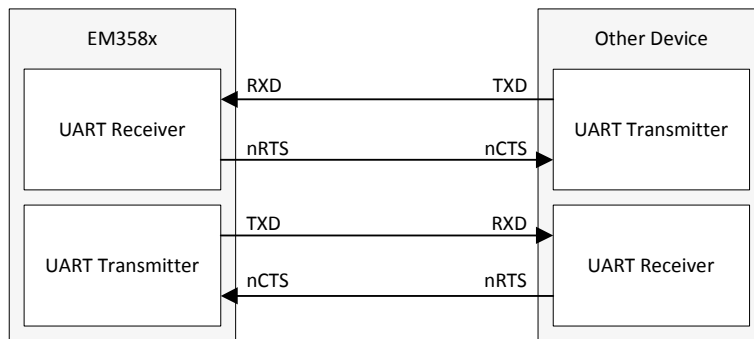

Figure 8.5. RTS/CTS Flow Control Connections

The UART RTS/CTS flow control options are selected by the SC_UARTFLOW and SC_UARTAUTO bits in the SC1_UARTCFG register (see Table 8.12). Whenever the SC_UARTFLOW bit is set, the UART will not start transmitting a character unless nCTS is low (asserted). If nCTS transitions to the high state (deasserts) while a character is being transmitted, transmission of that character continues until it is complete.

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

Table 8.12. UART RTS/CTS Flow Control Configurations

SC1_UARTCFG			Pins Used	Operating Mode
SC_UARTxxx*				
FLOW	AUTO	RTS		
0	—	—	TXD, RXD	No RTS/CTS flow control
1	0	0/1	TXD, RXD, nCTS, nRTS	Flow control using RTS/CTS with software control of nRTS: nRTS controlled by SC_UARTRTS bit in SC1_UARTCFG register
1	1	—	TXD, RXD, nCTS, nRTS	Flow control using RTS/CTS with hardware control of nRTS: nRTS is asserted if room for at least 2 characters in receive FIFO
*Note: The notation "xxx" means that the corresponding column header below is inserted to form the field name.				

8.6.5. DMA

The DMA Channels section describes how to configure and use the serial receive and transmit DMA channels.

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

8.6.6. Interrupts

UART interrupts are generated on the following events:

- Transmit FIFO empty and last character shifted out (depending on SC1_INTMODE, either the 0 to 1 transition or the high level of SC_UARTTXIDLE)
- Transmit FIFO changed from full to not full (depending on SC1_INTMODE, either the 0 to 1 transition or the high level of SC_UARTTXFREE)
- Receive FIFO changed from empty to not empty (depending on SC1_INTMODE, either the 0 to 1 transition or the high level of SC_UARTRXVAL)
- Transmit DMA buffer A/B complete (1 to 0 transition of SC_TXACTA/B)
- Receive DMA buffer A/B complete (1 to 0 transition of SC_RXACTA/B)
- Character received with parity error
- Character received with frame error
- Character received and lost when receive FIFO was full (receive overrun error)

To enable CPU interrupts, set the desired interrupt bits in the second-level INT_SC1CFG register, and enable the top-level SC1 interrupt in the NVIC by writing the INT_SC1 bit in the INT_CFGSET register.

8.6.7. Registers

Refer to "8.3.5. Registers" on page 85 (in "8.3. SPI—Master Mode") for a description of the SCx_DATA register.

Register 8.13. SC1_UARTSTAT: UART Status Register

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

SC1_UARTSTAT: Address: 0x4000C848   Reset: 0x40

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

Register 8.14. SC1_UARTCFG: UART Configuration Register

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

SC1_UARTCFG: Address: 0x4000C85C   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_UARTAUTO	[6]	RW	Set this bit to enable automatic nRTS control by hardware (SC_UARTFLOW must also be set). When automatic control is enabled, nRTS will be deasserted when the receive FIFO has space for only one more byte (inhibits transmission from the other device) and will be asserted if it has space for more than one byte (enables transmission from the other device). The SC_UARTRTS bit in this register has no effect if this bit is set.
SC_UARTFLOW	[5]	RW	Set this bit to enable using nRTS/nCTS flow control signals. Clear this bit to disable the signals. When this bit is clear, the UART transmitter will not be inhibited by nCTS.
SC_UARTODD	[4]	RW	If parity is enabled, specifies the kind of parity. 0: Even parity. 1: Odd parity.
SC_UARTPAR	[3]	RW	Specifies whether to use parity bits. 0: Don't use parity. 1: Use parity.
SC_UART2STP	[2]	RW	Number of stop bits transmitted. 0: 1 stop bit. 1: 2 stop bits.
SC_UART8BIT	[1]	RW	Number of data bits. 0: 7 data bits. 1: 8 data bits.
SC_UARTRTS	[0]	RW	nRTS is an output to control the flow of serial data sent to the EM358x from another device. This bit directly controls the output at the nRTS pin (SC_UARTFLOW must be set and SC_UARTAUTO must be cleared). When this bit is set, nRTS is asserted (pin is low, 'XON', RS232 positive voltage); the other device's transmission is enabled. When this bit is cleared, nRTS is deasserted (pin is high, 'XOFF', RS232 negative voltage), the other device's transmission is inhibited.

Register 8.15. SC1_UARTPER: UART Baud Rate Period Register

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

SC1_UARTPER: Address: 0x4000C868   Reset: 0x0

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

Register 8.16. SC1_UARTFRAC: UART Baud Rate Fractional Period Register

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

SC1_UARTFRAC: Address: 0x4000C86C   Reset: 0x0

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

8.7. DMA Channels

The EM358x serial DMA channels enable efficient, high-speed operation of the SPI and UART controllers by reducing the load on the CPU as well as decreasing the frequency of interrupts that it must service. The transmit and receive DMA channels can transfer data between the transmit and receive FIFOs and the DMA buffers in main memory as quickly as it can be transmitted or received. Once software defines, configures, and activates the DMA, it only needs to handle an interrupt when a transmit buffer has been emptied or a receive buffer has been filled. The DMA channels each support two memory buffers, labeled A and B, and can alternate ("ping-pong") between them automatically to allow continuous communication without critical interrupt timing.

Note: DMA memory buffer terminology

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

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

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

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

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

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

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

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

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

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

8.7.1. Registers

Note: Substitute "1" or "2" for "x" in the following detailed descriptions.

Register 8.17. SCx_DMACTRL
SC1_DMACTRL: Serial DMA Control Register
SC2_DMACTRL: Serial DMA Control Register

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

SC1_DMACTRL: Address: 0x4000C830   Reset: 0x0
SC2_DMACTRL: Address: 0x4000C030   Reset: 0x0

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

Register 8.18. SCx_DMASTAT
SC1_DMASTAT: Serial DMA Status Register
SC2_DMASTAT: Serial DMA Status Register

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

SC1_DMASTAT: Address: 0x4000C82C   Reset: 0x0
SC2_DMASTAT: Address: 0x4000C02C   Reset: 0x0

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

SILICON LABS

Bitname	Bitfield	Access	Description
SC_RXOVFA	[4]	R	This bit is set when DMA receive buffer A was passed an overrun error from the receive FIFO. Neither receive buffer was capable of accepting any more bytes (unloaded), and the FIFO filled up. Buffer A was the next buffer to load, and when it drained the FIFO the overrun error was passed up to the DMA and flagged with this bit. Cleared the next time buffer A is loaded and when the receive DMA is reset.
SC_TXACTB	[3]	R	This bit is set when DMA transmit buffer B is active.
SC_TXACTA	[2]	R	This bit is set when DMA transmit buffer A is active.
SC_RXACTB	[1]	R	This bit is set when DMA receive buffer B is active.
SC_RXACTA	[0]	R	This bit is set when DMA receive buffer A is active.

Register 8.19. SCx_TXBEGA
SC1_TXBEGA: Transmit DMA Begin Address Register A
SC2_TXBEGA: Transmit DMA Begin Address Register A

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

SC1_TXBEGA: Address: 0x4000C810   Reset: 0x20000000
SC2_TXBEGA: Address: 0x4000C010   Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_TXBEGA	[15:0]	RW	DMA transmit buffer A start address.

Register 8.20. SCx_TXBEGB
SC1_TXBEGB: Transmit DMA Begin Address Register B
SC2_TXBEGB: Transmit DMA Begin Address Register B

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

SC1_TXBEGB: Address: 0x4000C818 Reset: 0x20000000
SC2_TXBEGB: Address: 0x4000C018 Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_TXBEGB	[15:0]	RW	DMA transmit buffer B start address.

Register 8.21. SCx_TXENDA
SC1_TXENDA: Transmit DMA End Address Register A
SC2_TXENDA: Transmit DMA End Address Register A

Bit	31	30	29	28	27	26	25	24
Name	0	0	1	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Name	SC_TXENDA								

Bit	7	6	5	4	3	2	1	0
Name	SC_TXENDA								

SC1_TXENDA: Address: 0x4000C814   Reset: 0x20000000
SC2_TXENDA: Address: 0x4000C014   Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_TXENDA	[15:0]	RW	Address of the last byte that will be read from the DMA transmit buffer A.

Register 8.22. SCx_TXENDB
SC1_TXENDB: Transmit DMA End Address Register B
SC2_TXENDB: Transmit DMA End Address Register B

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

SC1_TXENDB: Address: 0x4000C81C   Reset: 0x20000000
SC2_TXENDB: Address: 0x4000C01C   Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_TXENDB	[15:0]	RW	Address of the last byte that will be read from the DMA transmit buffer B.

Register 8.23. SCx_TXCNT
SC1_TXCNT: Transmit DMA Count Register
SC2_TXCNT: Transmit DMA Count Register

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

SC1_TXCNT: Address: 0x4000C828   Reset: 0x0
SC2_TXCNT: Address: 0x4000C028   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_TXCNT	[15:0]	R	The offset from the start of the active DMA transmit buffer from which the next byte will be read. This register is set to zero when the buffer is loaded and when the DMA is reset.

Register 8.24. SCx_RXBEGA
SC1_RXBEGA: Receive DMA Begin Address Register A
SC2_RXBEGA: Receive DMA Begin Address Register A

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

SC1_RXBEGA: Address: 0x4000C800   Reset: 0x20000000
SC2_RXBEGA: Address: 0x4000C000   Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_RXBEGA	[15:0]	RW	DMA receive buffer A start address.

Register 8.25. SCx_RXBEGB
SC1_RXBEGB: Receive DMA Begin Address Register B
SC2_RXBEGB: Receive DMA Begin Address Register B

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

SC1_RXBEGB: Address: 0x4000C808   Reset: 0x20000000
SC2_RXBEGB: Address: 0x4000C008   Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_RXBEGB	[15:0]	RW	DMA receive buffer B start address.

Register 8.26. SCx_RXENDA
SC1_RXENDA: Receive DMA End Address Register A
SC2_RXENDA: Receive DMA End Address Register A

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

SC1_RXENDA: Address: 0x4000C804   Reset: 0x20000000
SC2_RXENDA: Address: 0x4000C004   Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_RXENDA	[15:0]	RW	Address of the last byte that will be written in the DMA receive buffer A.

Register 8.27. SCx_RXENDB
SC1_RXENDB: Receive DMA End Address Register B
SC2_RXENDB: Receive DMA End Address Register B

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

SC1_RXENDB: Address: 0x4000C80C   Reset: 0x20000000
SC2_RXENDB: Address: 0x4000C00C   Reset: 0x20000000

Bitname	Bitfield	Access	Description
SC_RXENDB	[15:0]	RW	Address of the last byte that will be written in the DMA receive buffer B.

Register 8.28. SCx_RXCNTA
SC1_RXCNTA: Receive DMA Count Register A
SC2_RXCNTA: Receive DMA Count Register A

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

SC1_RXCNTA: Address: 0x4000C820   Reset: 0x0
SC2_RXCNTA: Address: 0x4000C020   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_RXCNTA	[15:0]	RW	The offset from the start of DMA receive buffer A at which the next byte will be written. This register is set to zero when the buffer is loaded and when the DMA is reset. If this register is written when the buffer is not loaded, the buffer is loaded.

Register 8.29. SCx_RXCNTB
SC1_RXCNTB: Receive DMA Count Register B
SC2_RXCNTB: Receive DMA Count Register B

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

SC1_RXCNTB: Address: 0x4000C824   Reset: 0x0
SC2_RXCNTB: Address: 0x4000C024   Reset: 0x0

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

Register 8.30. SCx_RXCNTSAVED
SC1_RXCNTSAVED: Saved Receive DMA Count Register
SC2_RXCNTSAVED: Saved Receive DMA Count Register

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

SC1_RXCNTSAVED: Address: 0x4000C870   Reset: 0x0
SC2_RXCNTSAVED: Address: 0x4000C070   Reset: 0x0

Bitname	Bitfield	Access	Description
SC_RXCNTSAVED	[15:0]	R	Receive DMA count saved in SPI slave mode when nSSEL deasserts. The count is only saved the first time nSSEL deasserts.

Register 8.31. SCx_RXERRA
SC1_RXERRA: DMA First Receive Error Register A
SC2_RXERRA: DMA First Receive Error Register A

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

SC1_RXERRA: Address: 0x4000C834 Reset: 0x0
SC2_RXERRA: Address: 0x4000C034 Reset: 0x0

Bitname	Bitfield	Access	Description
SC_RXERRA	[15:0]	R	The offset from the start of DMA receive buffer A of the first byte received with a parity, frame, or overflow error. Note that an overflow error occurs at the input to the receive FIFO, so this offset is 4 bytes before the overflow position. If there is no error, it reads zero. This register will not be updated by subsequent errors until the buffer unloads and is reloaded, or the receive DMA is reset.

Register 8.32. SCx_RXERRB
SC1_RXERRB: DMA First Receive Error Register B
SC2_RXERRB: DMA First Receive Error Register B

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

SC1_RXERRB: Address: 0x4000C838   Reset: 0x0
SC2_RXERRB: Address: 0x4000C038   Reset: 0x0

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

EM358x

9. USB Device

9.1. Overview

The EM3582, EM3586, and EM3588 have a USB 2.0-compliant full-speed (12 Mbps) device peripheral, with on-chip transceiver. Other EM358x variants (EM3581, EM3585, and EM3587) do not support USB.

The EM358x only supports one configuration (configuration 0) and two interfaces. By default all logical endpoints are on the first interface (interface 0), while the USB_INTF1SEL register enables associating logical endpoints with interface 1.

The EM358x supports up to six endpoints (in addition to the control endpoint 0). There are five endpoints that can be used as either interrupt or bulk and one isochronous endpoint.

The USB peripheral is interfaced to the CPU through memory mapped registers for control, and DMA for data. The USB device generates its own 48 MHz internal clock from the main 24 MHz crystal clock.

The EM358x fully supports USB suspend and resume modes, and can meet the USB specification suspend current of <2.5 mA. It achieves this by switching the chip to run from a divided down version of the system clock.

Note: Fully supporting all of electrical compliance for the purpose of an EM358x device passing USB Certification is tightly coupled with the application being run on the EM358x and the larger system using the EM358x. From the perspective of the EM358x device, suspend is an asynchronous event. Application designs will have to change to effectively handle suspend, resume, remote wakeup and their clocking requirements.

Note: The device is USB 2.0 compliant but only supports full-speed (12 Mbps).

9.2. Host Drivers

The goal of the USB COM port functionality Silicon Labs provides is to make it simple for an EM358x to interact with a PC without needing a physical UART / RS-232 but still use the basic application serial functionality that has been available on a UART.

There are two options for host drivers:

- The Silicon Labs supplied driver: There is an "EM358VPInstaller" available for both x64 and x86 Windows. The device driver is signed and certified by Microsoft Windows Hardware Compatibility.
- The Windows PC USB built-in driver: Silicon Labs provides the only Windows file that is necessary to use the communications device class (CDC). This is a .inf file needed to allow Windows to recognize the device correctly and load the built-in driver.

Once the driver is installed and the EM358x is connected to the computer via USB, nothing more is needed on the host to configure this new COM port device.

9.3. Normal Serial COM Port Operation

From the perspective of the software developer, in addition to the existing serial port 1 - the UART – the use of USB allows for the serial port 3 - the USB COM port. Port 1 and Port 3 will not affect each other. The majority of the basic functionality available for the UART exists for the USB COM port. This includes the emberSerialPrintf() and companion printing functions as well as emberSerialReadLine() and companion reading functions. Some of the UART-style functionality, such as hardware flow control, is not available on the USB COM port.

9.4. References

The Original USB 2.0 specification released on April 27, 2000 is available from the USB Implementers Forum (USB-IF) at:

http://www.usb.org/developers/docs/

The zip file http://www.usb.org/developers/docs/usb_20_070113.zip has the original specification along with supporting information.

The information for testing a full-speed peripheral device can be found on the USB-IF Compliance Program page at: http://www.usb.org/developers/compliance/

Compliance test requirements are found at: http://www.usb.org/developers/compliance/peripheral_low/

9.5. GPIO Usage and USB Pin Assignments

Three GPIO are required for all USB design. A fourth GPIO is required when the system is a self powered device.

The USB interface is available as an alternate function on 2 GPIO pins: PA0 and PA1. USBDM is on PA0, and USBDP is on PA1. PA0 and PA1 need to be put in analog mode for USBDM on PA0 and USBDP on PA1 (through register PACFGL[7:0]).

PA0 and PA1 also share some signals for Serial Controller 2 functionality, and hence SC2 and the USB are mutually exclusive. This means the EM358x gains UART style functionality over USB (Windows COM port) at the expense of SC2's SPI and TWI, but the existing UART and SPI functionality over SC1 is still available.

Software on the device must be able to control enumeration. Any GPIO can be chosen for push-pull output to control a 1.5kOhm pull-up resistor. A separate circuit can be used instead of connecting the pull-up directly to the EM358x chip, as long as software on the EM358x chip is capable of controlling the enumeration pull-up. PA2 is a logical choice since SC2 won't be available due to the USB signals being on PA0 and PA1. It is possible for any GPIO other than PA2 to be used for this functionality.

For a self-powered device a GPIO must be chosen to sense the USB voltage (VBUS) so the device knows when the USB is physically connected or not connected. This VBUS monitoring ability is needed for software to control the pull-up resistor appropriately since the USB specification requires that the pull-up resistor is disconnected if VBUS is not connected. PA3 is a logical choice for VBUS monitoring since SC2 won't be available due to USB being on PA0 and PA1. If PA3 is used, one of the configurable external interrupts (IRQC or IRQD) must be used to trigger execution based on a change of state of VBUS. PB0 could be used instead since PB0 is tied to IRQA as an external interrupt. PB6 could be used instead since PB6 is tied to IRQB as an external interrupt.

Note: While any GPIO can be used for VBUS monitoring, no GPIO is 5 V tolerant so external circuitry such as a voltage divider is required to sense the nominally 5 V VBUS.

9.6. Application Schematics

A USB application requires a small number of external passive components: series resistors, a pull-up to indicate a full speed device and control enumeration, and capacitors to ground.

The two power configurations shown are:

- Bus powered. The product is only powered when plugged into a host, for example a USB dongle. A 3.3 V regulator is required as the EM358x maximum voltage is 3.6 V.
- Self powered. The product is always powered from its own power source.

Note: Refer to Figure 2.1. Typical Application Circuit in the EM358x Datasheet for circuit specifics.
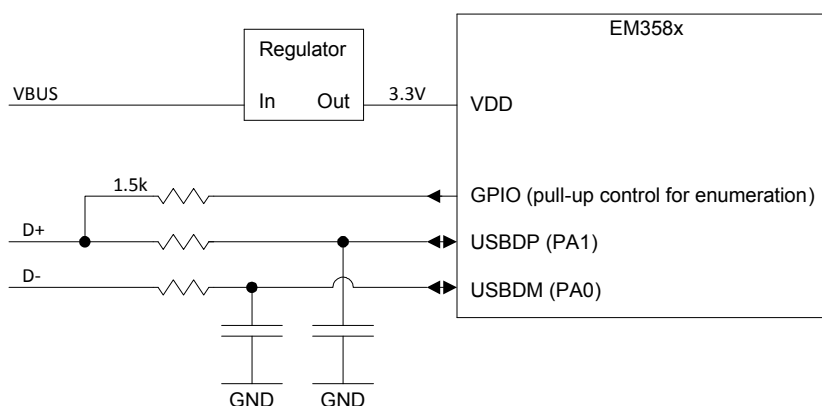
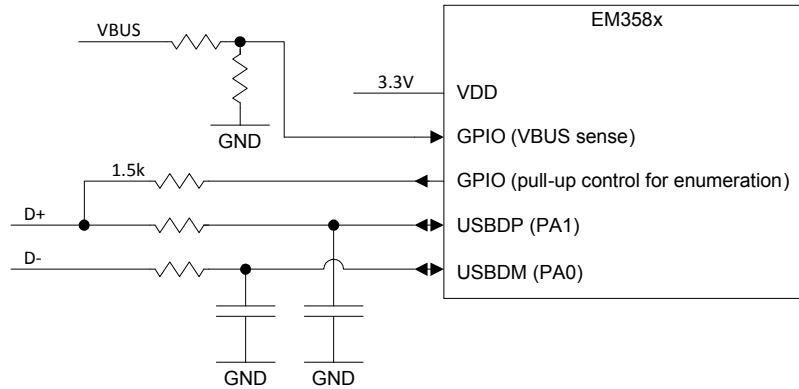**Figure 9.1. EM358x USB Application Circuit – Bus Powered Device**

EM358x

Figure 9.2. EM358x USB Application Circuit – Self Powered Device

9.7. Endpoints

EM358x supports up to twelve endpoints (in addition to the control endpoint 0). The USB peripheral is interfaced to the CPU through memory mapped registers for control, and DMA for data.

Before the device will respond to the USB bus, it must be configured to define its endpoint status and enable the device. Various decisions must be made as to the nature of the device — i.e. what endpoint types and their number. All devices must have endpoint 0, which is a control endpoint. There are 10 endpoints that can be used as either bulk or interrupt and two isochronous endpoint. The choice of bulk versus interrupt is done in the endpoint descriptor.

Each endpoint can be enabled independently, except endpoint 0 which is always enabled. Endpoints can be enabled independently. Endpoints are individually enabled through the USB_ENABLEIN and USB_ENABLEOUT registers. It is possible to explicitly stall individual endpoint directions with the registers USB_STALLIN and USB_STALLOUT. Explicitly stalled endpoints can only be un-stalled explicitly.

Both directions on a given logical endpoint have the same max packet size.

Table 9.1. Endpoints

Endpoint Number	Max Packet Size in Bytes	Type	Direction	Buffer Offset
Endpoint 0	8	Control	In	0x000
			Out	0x008
Endpoint 1	8	Bulk or Interrupt	In	0x010
			Out	0x018
Endpoint 2	8	Bulk or Interrupt	In	0x020
			Out	0x028
Endpoint 3	64	Bulk or Interrupt	In	0x030
			Out	0x070

SILICON LABS

Table 9.1. Endpoints (Continued)

Endpoint Number	Max Packet Size in Bytes	Type	Direction	Buffer Offset
Endpoint 4	32	Bulk or Interrupt	In	0x0b0
			Out	0x0d0
Endpoint 5	64	Bulk or Interrupt	In	0x0f0
			Out	0x130
Endpoint 6	512	Isochronous	In	0x170
			Out	0x370

9.8. Buffers and DMA

All physical endpoints are packed together and accessed through a contiguous block of RAM. The buffer's RAM can be written or read at any time, therefore a buffer's section of RAM should not be modified while the USB DMA is transmitting or receiving. To assist with throughput, the DMA interface supports double buffering with RAM buffers A and B. All functionality can be performed with just buffer A. Buffer B is only useful in systems needing enhanced throughput while the EM358x device is busy with other functionality. Due to the intricacies of handling two buffers, it is recommended that designs start with buffer A and only add buffer B if necessary.

There are two registers to allow positioning of the A and B buffers independently in RAM. Register USB_BUFBASEA is for positioning buffer A and register USB_BUFBASEB is for positioning buffer B when double buffering is enabled.

Note: 8 byte alignment is the only requirement as to where the buffer can be located in RAM.

9.9. Standard Commands

A set of USB standard commands are defined in the USB specification which, for the purposes of the EM358x USB device, are split into two types; those handled by the hardware and those passed on for the software to handle. These standard commands as denoted in the setup packet by bmRequestType where Type is Standard.

From the interface point of view, data appears like a packet being sent to the endpoint 0 OUT buffer and hardware writes this data to the buffer in RAM. However, as the hardware handled core commands are fully decoded and acted on by the hardware, even though the data actually appears in RAM, this data stage is hidden from software with no interrupts generated. The exception to this rule and the only indication that might be seen by software is an INT_USBNACK if the host should retry the command.

Since there is no means for software to explicitly know if the hardware handled commands have occurred, any protocol running on the device must depend on activity in the software handled commands to infer the device state.

Table 9.2. Software Handled Standard Commands

Command	Target	Data?	Comments
Get Descriptor	Device	IN	Get the device descriptor.
Set Descriptor	Device	OUT	Set the device descriptor.
Get Descriptor	Configuration	IN	Get the configuration descriptor.
Set Descriptor	Configuration	OUT	Set the configuration descriptor.
Synch Frame	Endpoint	IN	Synchronize frames in an endpoint.

EM358x

Table 9.3. Hardware Handled Standard Commands

Command	Target	Data?	Comments
Get Configuration	Device	1 byte, IN	Returns currently selected configuration.
Set Configuration	Device	None	Sets the configuration to use.
Get Interface	Interface	1 byte, IN	Returns the alternate setting selected for the addressed interface
Set Interface	Interface	None	Set interface and alternate setting to use.
Set Address	Device	None	Set the device address for the interface.
Clear Feature	Device	None	Clear a device feature. The only valid features are DEVICE_RE-MOTE_WAKEUP and TEST_MOD as defined in the USB specification.
Set Feature	Device	None	Set a device feature. The only valid features are DEVICE_RE-MOTE_WAKEUP and TEST_MOD as defined in the USB specification.
Clear Feature	Interface	None	Does nothing. (There are no interface features.)
Set Feature	Interface	None	Does nothing. (There are no interface features.)
Clear Feature	Endpoint	None	Can only clear the HALT feature of an endpoint.
Set Feature	Endpoint	None	Can only set the HALT feature of an endpoint.
Get Status	Device	2 bytes, IN	Returns remote wakeup and self powered status.
Get Status	Interface	2 bytes, IN	Always returns 2 bytes of 0x00.
Get Status	Endpoint	2 bytes, IN	Returns endpoint HAL status.

9.10. Set Up and Configuration

It is best to have the enumeration pull-up resistor configured for the disconnected state whenever modifying the USB configuration so that the host does not attempt enumeration before the USB core and the USB software on the device is fully configured and ready to interact with the host. The disconnected state means the GPIO used for control of the enumeration pull-up is configured as a floating input.

The GPIOs PA0 and PA1, which are dedicated to USB data D- and D+, are configured into the analog mode.

The USB_BUFBASEA register is written with a RAM address of where buffer A exists. If double buffering is used, the USB_BUFBASEB register is also written with a RAM address of where buffer B exists. These addresses are the root of the DMA interface and must not change once USB becomes active.

The IN and OUT directions of physical endpoints that will be used are enabled with the bits USB_ENABLEINEPx in the USB_ENABLEIN register and the bits USB_ENABLEOUTEPx in the USB_ENABLEOUT register. These physical endpoints can be enabled independently of each other.

Double buffering can be independently enabled for each physical IN endpoint and OUT endpoint separately. If double buffering is not enabled, then only buffer A is used. Double buffering is enabled in the USB_CTRL register with the bits USB_ENBUFOUTEPxB and USB_ENBUFINEPxB.

Additional configuration decisions must be made before the USB is fully configured.

- Is this a self-powered device?
- Does this device support remote-wakeup?

The bit USB_SELFPWRD in the register USB_CTRL is used to configure the USB peripheral core if the device is self powered. This self-powered choice must also be indicated in the device descriptor reported to the host when the device enumerates.

The bit USB_REMOTEWKUPEN in the register USB_CTRL is used to configure the USB peripheral core indicating if the remote-wakeup feature is enabled. With this bit clear, the host will get a STALL response to the ClearFeature or SetFeature transaction for remote-wakeup of the device. Otherwise the host can inspect and control this feature. This remote-wakeup choice must also be indicated in the device descriptor reported to the host when the device enumerates.

When beginning to enable interrupts, it's always best to start with clearing any possible stale interrupt flags. This is done by writing a 1 to every bit in the INT_USBFLAG register.

The register INT_USBCFG is used for configuring the USB interrupts. From the perspective of the EM358x device, the concept of RX behavior relates to an OUT transaction from the host and TX behavior relates to an IN transaction to the host. Since EP0 is always enabled in the USB core and is always necessary, the two EP0 interrupt enable bits in INT_USBCFG must be set: INT_USBTXACTIVEEP0 and INT_USBRXVALIDEP0.

In addition to EP0, the interrupts for other endpoints being used must also be enabled. Every physical endpoint has separate interrupt enables for the TX and RX direction since not every physical endpoint direction is necessary in every possible configuration. A common use is to fully enable interrupts for bulk data transactions, such as EP3 with the bits INT_USBTXACTIVEEP3 and INT_USBRXVALIDEP3.

The interrupt bit INT_USBRESET in INT_USBCFG must be set to service a USB reset event that occurs across the bus. While the reset across the bus will automatically reset the USB core in the EM358x, the DMA interacting with the USB core is not reset. Therefore software must service the INT_USBRESET event by clearing out the DMA buffers to put the buffers back to a reset state to match the core. Clearing out the DMA is done by writing a 1 to all bits in the register USB_BUFCLR.

The interrupt bits INT_USBSUSPEND and INT_USBRESUME must also be enabled to service the suspend and resume events across the bus.

Once the second-level interrupts in the INT_USBCFG register have been appropriately set, USB must be enabled at the top level in the NVIC. This is done by setting the bit INT_USB in the register INT_CFGSET.

If the device is self-powered, the USB specification requires that the pull-up resistor for enumeration is disconnected when VBUS is not connected. This requirement means that the GPIO used to sense the VBUS presence and the interrupt used to track the VBUS changing state must be configured before starting the enumeration process. The GPIO and interrupt configuration used follows the same procedure described for External Interrupts.

- Configure the GPIO as a simple input.
- Disable interrupt triggering by clearing to 0 the IRQ register being used for the GPIO. This will be one of the registers GPIO_INTCFGA, GPIO_INTCFGB, GPIO_INTCFGC, or GPIO_INTCFGD.
- Ensure the interrupts start from a clean state by writing the necessary bit in INT_CFGCLR register and the INT_GPIOFLAG register.
- If using the selectable interrupt IRQC or IRQD, write the register GPIO_IRQCSEL or GPIO_IRQDSEL with the value corresponding to the GPIO being used.
- Set the IRQ configuration, as defined in the register GPIO_INTCFGA, GPIO_INTCFGB, GPIO_INTCFGC, or GPIO_INTCFGD, to the value 3. The value 3 enables triggering on a falling or rising edge.
- Enable the top level interrupt in the NVIC by setting the appropriate bit the INT_CFGSET register.

Because the USB specification requires the enumeration pull-up to track the state of VBUS, software on the EM358x must be sure to specifically read the VBUS monitoring GPIO to check the state of VBUS before the monitoring IRQ edge detection interrupt has taken over.

Once the USB registers and software configuration has been set up, enumeration can be begin by setting the

EM358x

GPIO used for control of the enumeration pull-up to an push-pull output set high.

9.11. DMA Usage and Transfers

To transfer IN data, data must first be put in an endpoint buffer's RAM. Once the buffer's RAM has data, the register USB_TXBUFSIZEEPxy for that endpoint must be set with the number of bytes to be transmitted. Transmission is then started with loading the buffer by setting the appropriate USB_TXLOADEPxy bit in the USB_TXLOAD register.

A buffer should not be modified while the USB_TXLOADxy bit is still set. Modifying a buffer's RAM before the buffer has unloaded could result in incorrect data being transferred.

Setting the register USB_TXBUFSIZEEPxy to 0 to transmit a zero length packet is valid and in this situation the data in buffer's RAM does not matter.

Because the interrupt INT_USBTXACTIVEEPx will fire on the falling edge of the status bit USB_TXACTIVEEPxy, the INT_USBTXACTIVEEPx interrupt can be used to keep loading data transmission without any foreground processing involvement.

Buffer B will only be used if double buffering is enabled. The USB peripheral will always use buffer A if buffer A has been loaded when the host begins a transfer, regardless of buffer B having data. The USB peripheral will only transfer from buffer B if buffer B is loaded and buffer A is not loaded.

Receiving OUT data is directly put into the appropriate endpoint buffer's RAM. Handling this OUT data is best driven by the interrupt INT_USBRXVALIDEPx. This interrupt will fire on the rising edge of the appropriate USB_RXVALIDEPxy status bit. Once the received data is in RAM and marked valid, the number of bytes received is shown in the register USB_RXBUFSIZEEPxy. Reception of a zero length OUT transaction will set the RX registers and interrupt bits but will not affect any data already in the buffer's RAM.

When reception processing is finished, the appropriate USB_RXVALIDEPxy bit in the USB_RXVALID register must be set. If USB_RXVALID is not cleared by writing a 1 to the USB_RXVALIDEPxy bit, further reception will STALL.

Like the IN direction, buffer B will only be used if double buffering is enabled. The hardware will always transfer into buffer A if it is free, and will only transfer into buffer B if the USB_RXVALIDEPxA bit for buffer A has not been cleared.

9.12. Suspend and Resume

The USB specification details the maximum allowable current draw while suspended and how quickly a device must achieve this current draw. Fully supporting all of electrical compliance for the purpose of an EM358x device passing USB Certification is tightly coupled with the application being run on the EM358x and the larger EM358x system. From the perspective of the EM358x device, suspend is an asynchronous event. Application designs will have to be written to effectively handle suspend, resume, remote wakeup and their clocking requirements

To achieve the lower current required, some peripherals need to be disabled. Especially the radio must be shutdown. To keep the chip functional enough to stay on the USB bus and resume from the suspended state, the system clock needs to be divided by 4. Refer to figure 5-2 Clocks Block Diagram to see exactly where in the clocking of the chip the division occurs. Peripherals that are kept active might need to be adjusted for the system clock being divided by 4.

When the bus suspends this device, the interrupt INT_USBSUSPEND will pend. To divide the system clock by 4, set the register bit USBSUSP_CLKSEL in the register CPU_CLKSEL and enter idle sleep. Clock division will not occur until the system enters idle sleep. An exit from idle sleep will automatically disable the divide by 4, restoring the system clock, and clearing the USBSUSP_CLKSEL bit. Refer to section 5.5 Power Management for general information about idle sleep.

When the bus resumes the device, the interrupt INT_USBRESUME will pend. The act of servicing the interrupt will take the system out of idle sleep and disable the divide by 4, restoring the system clock.

The interrupt INT_USBWAKEUP will pend after a successful remote wakeup. Because code on this device is the action that instigated a remote wakeup, this device is already running and will not have been in idle sleep with the slower clock mode. With respect to current consumption in the larger system, INT_USBWAKEUP can be treated like INT_USBRESUME with respect to using pieces of the system that consume higher current, such as the radio.

SILICON LABS

9.13. Interrupts

USB interrupts are generated on the following events:

- INT_USBRXVALIDEPx: Reception becoming valid is indicated with the INT_USBRXVALIDEPx bits. The interrupt pends on the rising edge of either of the corresponding USB_RXVALIDEPxy bits for endpoint x in the USB_RXVALID register.

- INT_USBTXACTVEEPx: Transmit activity completing is indicated with the INT_USBTXACTVEEPx bit. The interrupt pends on the falling edge of either of the corresponding USB_TXACTIVEEPxy bits for endpoint x in the USB_ACTIVE register.

- INT_USBRESET: A reset on the bus will reset the USB core and pend the INT_USBRESET bit. A USB reset of the core will not affect USB DMA. Therefore USB DMA needs to be manually reset.

- INT_USBSUSPEND: A suspend on the bus will pend the INT_USBSUSPEND bit.

- INT_USBRESUME: When in the suspended state, a resume on the bus will pend the INT_USBRESUME bit.

- INT_USBWAKEUP: A successful remote wakeup from the suspended state (performed with the USB_RESUME register) pends the INT_USBWAKEUP bit. This activity is very similar to INT_USBRESUME, except it is initiated by this device instead of the bus.

- INT_USBSOF: Reception of a Start of Frame packet will pend the INT_USBSOF bit. Because a SOF packet can be used by the host to prevent the idle state (suspending), this interrupt could burden the device and therefore might be best left disabled.

- INT_USBBUFRXOVF: Buffer reception overflow will pend the INT_USBBUFRXOVF bit.

- INT_USBBUFTXUND: Buffer transmit underflow will pend the INT_USBBUFTXUND bit.

- INT_USBPIPERXOVF: Pipeline reception overflow will pend the INT_USBPIPERXOVF bit.

- INT_USBPIPETXUND: Pipeline transmit underflow will pend the INT_USBPIPETXUND bit.

- INT_USBNAK: Reception of a NAK packet will pend the INT_USBNAK bit. This interrupt is best left disabled. Some NAKs are not an error, other NAKs will occur on errors that will be retried automatically by the core.

EM358x

9.14. Registers

Note: x = EP number. y = buffer A or B

Register 9.1. USB_CTRL: USB Control Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	USB_RE-SETCTRL	USB_ENBU-FOUTEP6B	USB_ENBU-FOUTEP5B	USB_ENBU-FOUTEP4B	USB_ENBU-FOUTEP3B	USB_ENBU-FOUTEP2B	USB_ENBU-FOUTEP1B	USB_ENBU-FOUTEP0B
Bit	15	14	13	12	11	10	9	8
Name	0	USB_ENBU FINEP6B	USB_ENBU FINEP5B	USB_ENBU FINEP4B	USB_ENBU-FINEP3B	USB_ENBU FINEP2B	USB_ENBU FINEP1B	USB_ENDU FINEP0B
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	0	USB_REMO TEWKUPEN	USB_CLRFE P0STALL	USB_SELFP WRD

USB_CTRL: Address: 0x4001105C Reset: 0x4

Bitname	Bitfield	Access	Description
USB_RESETCTRL	[23]	RW	Force a reset state internally to the USB core. Setting this bit will cause the USB core to go back to its default sate, as if the core has just been connected to the bus. Because this state will require re-enumerating, it is recommended that setting this bit is only done while disconnected. This bit is not self clearing since this bit may have to be asserted for multiple cycles.
USB_ENBU-FOUTEP6B	[22]	RW	Set this bit to enable endpoint 6 buffer B OUT.
USB_ENBU-FOUTEP5B	[21]	RW	Set this bit to enable endpoint 5 buffer B OUT.
USB_ENBU-FOUTEP4B	[20]	RW	Set this bit to enable endpoint 4 buffer B OUT.
USB_ENBU-FOUTEP3B	[19]	RW	Set this bit to enable endpoint 3 buffer B OUT.
USB_ENBU-FOUTEP2B	[18]	RW	Set this bit to enable endpoint 2 buffer B OUT.
USB_ENBU-FOUTEP1B	[17]	RW	Set this bit to enable endpoint 1 buffer B OUT.
USB_ENBU-FOUTEP0B	[16]	RW	Set this bit to enable endpoint 0 buffer B OUT.
USB_ENBUFINEP6B	[14]	RW	Set this bit to enable endpoint 6 buffer B IN.
USB_ENBUFINEP5B	[13]	RW	Set this bit to enable endpoint 5 buffer B IN.

SILICON LABS

USB_ENBUFINEP4B	[12]	RW	Set this bit to enable endpoint 4 buffer B IN.
USB_ENBUFINEP3B	[11]	RW	Set this bit to enable endpoint 3 buffer B IN.
USB_ENBUFINEP2B	[10]	RW	Set this bit to enable endpoint 2 buffer B IN.
USB_ENBUFINEP1B	[9]	RW	Set this bit to enable endpoint 1 buffer B IN.
USB_ENBUFINEP0B	[8]	RW	Set this bit to enable endpoint 0 buffer B IN.
USB_REMOTEWKU-PEN	[2]	RW	Set this this bit to enable the remote-wakeup feature. With this bit set, the host can inspect and control this feature. With this bit cleared, the host will get a STALL response to a ClearFeature or SetFeature control command.
USB_CLR-FEP0STALL	[1]	RW	Set this bit for the USB core to send STALL on ClearFeature (EPO), else send ACK. EP0 must always process the control commands so the USB core will not actually respond with STALL to subsequent control commands. Instead this bit is a means of indicating to the host that ClearFeature(EP0) is acknowledged or unexpected.
USB_SELFPWRD	[0]	RW	Set this bit to indicate whether USB is self-powered. The state of this register is reflected in the data returned from a GetStatus(Device) standard command, handled by the USB core. The device descriptor returned to the host during enumeration must match the self-powered configuration set in this register.

EM358x

Register 9.2. USB_STATUS: USB Status Register

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

USB_STATUS: Address: 0x4001106C Reset: 0x0

Bitname	Bitfield	Access	Description
USB_RESETSTAT	[12]	R	This bit is set while USB reset is active. The rising edge of the reset status generates the INT_USBRESET interrupt.
USB_SUSPENDED	[11]	R	This bit is set while the device is suspended. The rising of edge of the suspended status generates the INT_USBSUSPEND interrupt.
USB_TIMESTAMP	[10:0]	R	The timestamp of the reception of the last Start of Frame (SOF) packet.

SILICON LABS

Register 9.3. USB_ENABLEIN: USB Endpoint IN Enables Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	USB_EN-ABLEINEP6	USB_EN-ABLEINEP5	USB_EN-ABLEINEP4	USB_EN-ABLEINEP3	USB_EN-ABLEINEP2	USB_EN-ABLEINEP1	USB_EN-ABLEINEP0

USB_ENABLEIN: Address: 0x4001104C Reset: 0x1

Bitname	Bitfield	Access	Description
USB_ENABLEINEP6	[6]	RW	Set this bit to enable endpoint 6 buffer A IN.
USB_ENABLEINEP5	[5]	RW	Set this bit to enable endpoint 5 buffer A IN.
USB_ENABLEINEP4	[4]	RW	Set this bit to enable endpoint 4 buffer A IN.
USB_ENABLEINEP3	[3]	RW	Set this bit to enable endpoint 3 buffer A IN.
USB_ENABLEINEP2	[2]	RW	Set this bit to enable endpoint 2 buffer A IN.
USB_ENABLEINEP1	[1]	RW	Set this bit to enable endpoint 1 buffer A IN.
USB_ENABLEINEP0	[0]	R	Enable endpoint 0 buffer A IN. EP0 must always be enabled.

EM358x

Register 9.4. USB_ENABLEOUT: USB Endpoint OUT Enables Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	USB_EN-ABLE-OUTEP6	USB_EN-ABLE-OUTEP5	USB_EN-ABLE-OUTEP4	USB_EN-ABLE-OUTEP3	USB_EN-ABLE-OUTEP2	USB_EN-ABLE-OUTEP1	USB_EN-ABLE-OUTEP0

USB_ENABLEOUT: Address: 0x40011050 Reset: 0x01

Bitname	Bitfield	Access	Description
USB_ENABLE-OUTEP6	[6]	RW	Set this bit to enable endpoint 6 buffer A OUT.
USB_ENABLE-OUTEP5	[5]	RW	Set this bit to enable endpoint 5 buffer A OUT.
USB_ENABLE-OUTEP4	[4]	RW	Set this bit to enable endpoint 4 buffer A OUT.
USB_ENABLE-OUTEP3	[3]	RW	Set this bit to enable endpoint 3 buffer A OUT.
USB_ENABLE-OUTEP2	[2]	RW	Set this bit to enable endpoint 2 buffer A OUT.
USB_ENABLE-OUTEP1	[1]	RW	Set this bit to enable endpoint 1 buffer A OUT.
USB_ENABLE-OUTEP0	[0]	R	Enable endpoint 0 buffer A OUT. EP0 must always be enabled.

Register 9.5. USB_INTF1SEL: USB Endpoint Interface 1 Select Register

Bit	31	30	29	28	27	26	25	24
Name	USB_INTF1 SELEN	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	USB_INTF1 SELEP6	USB_INTF1 SELEP5	USB_INTF1 SELEP4	USB_INTF1 SELEP3	USB_INTF1 SELEP2	USB_INTF1 SELEP1	0

USB_INTF1SEL: Address: 0x40011074   Reset: 0x0

Bitname	Bitfield	Access	Description
USB_INTF1SELEN	[31]	RW	Set this bit to enable endpoints to be associated with Interface 1. Clear this bit to keep endpoints only associated with Interface 0.
USB_INTF1SELEP6	[6]	RW	Set this bit to associate endpoint 6 IN with Interface 1.
USB_INTF1SELEP5	[5]	RW	Set this bit to associate endpoint 5 IN with Interface 1.
USB_INTF1SELEP4	[4]	RW	Set this bit to associate endpoint 4 IN with Interface 1.
USB_INTF1SELEP3	[3]	RW	Set this bit to associate endpoint 3 IN with Interface 1.
USB_INTF1SELEP2	[2]	RW	Set this bit to associate endpoint 2 IN with Interface 1.
USB_INTF1SELEP1	[1]	RW	Set this bit to associate endpoint 1 IN with Interface 1.

Register 9.6. USB_BUFBASEy
USB_BUFBASEA: Base Address of Endpoint A Buffers in Main Memory Register
USB_BUFBASEB: Base Address of Endpoint B Buffers in Main Memory Register

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

USB_BUFBASEA: Address: 0x40011000   Reset: 0x20000000
USB_BUFBASEB: Address: 0x40011004 Reset: 0x20000000

Bitname	Bitfield	Access	Description
USB_BUFBASEy_-FIELD	[31:0]	RW	Write this register with a RAM address to position buffer y in RAM. 8 byte alignment is the only requirement as to where the buffer can be located in RAM.

Register 9.7. USB_TXLOAD: USB TX Buffer A and B Load Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	USB_TX-LOADEP6B	USB_TX-LOADEP5B	USB_TX-LOADEP4B	USB_TX-LOADEP3B	USB_TX-LOADEP2B	USB_TX-LOADEP1B	USB_TX-LOADEP0B
Bit	7	6	5	4	3	2	1	0
Name	0	USB_TX-LOADEP6A	USB_TX-LOADEP5A	USB_TX-LOADEP4A	USB_TX-LOADEP3A	USB_TX-LOADEP2A	USB_TX-LOADEP1A	USB_TX-LOADEP0A

USB_TXLOAD: Address: 0x40011008   Reset: 0x0

Bitname	Bitfield	Access	Description
USB_TXLOADEP6B	[14]	RW	Write this bit to load endpoint 6, buffer B for transmit. Auto-clears to 0.
USB_TXLOADEP5B	[13]	RW	Write this bit to load endpoint 5, buffer B for transmit. Auto-clears to 0.
USB_TXLOADEP4B	[12]	RW	Write this bit to load endpoint 4, buffer B for transmit. Auto-clears to 0.
USB_TXLOADEP3B	[11]	RW	Write this bit to load endpoint 3, buffer B for transmit. Auto-clears to 0.
USB_TXLOADEP2B	[10]	RW	Write this bit to load endpoint 2, buffer B for transmit. Auto-clears to 0.
USB_TXLOADEP1B	[9]	RW	Write this bit to load endpoint 1, buffer B for transmit. Auto-clears to 0.
USB_TXLOADEP0B	[8]	RW	Write this bit to load endpoint 0, buffer B for transmit. Auto-clears to 0.
USB_TXLOADEP6A	[6]	RW	Write this bit to load endpoint 6, buffer A for transmit. Auto-clears to 0.
USB_TXLOADEP5A	[5]	RW	Write this bit to load endpoint 5, buffer A for transmit. Auto-clears to 0.
USB_TXLOADEP4A	[4]	RW	Write this bit to load endpoint 4, buffer A for transmit. Auto-clears to 0.
USB_TXLOADEP3A	[3]	RW	Write this bit to load endpoint 3, buffer A for transmit. Auto-clears to 0.
USB_TXLOADEP2A	[2]	RW	Write this bit to load endpoint 2, buffer A for transmit. Auto-clears to 0.
USB_TXLOADEP1A	[1]	RW	Write this bit to load endpoint 1, buffer A for transmit. Auto-clears to 0.
USB_TXLOADEP0A	[0]	RW	Write this bit to load endpoint 0, buffer A for transmit. Auto-clears to 0.

Register 9.8. USB_TXACTIVE: USB Buffer A and Buffer B Transmit Active Status Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	USB_ TXAC- TIVEEP6B	USB_ TXAC- TIVEEP5B	USB_ TXAC- TIVEEP4B	USB_ TXAC- TIVEEP3B	USB_ TXAC- TIVEEP2B	USB_ TXAC- TIVEEP1B	USB_ TXAC- TIVEEP0B
Bit	7	6	5	4	3	2	1	0
Name	0	USB_TXAC- TIVEEP6A	USB_TXAC- TIVEEP5A	USB_TXAC- TIVEEP4A	USB_TXAC- TIVEEP3A	USB_TXAC- TIVEEP2A	USB_TXAC- TIVEEP1A	USB_TXAC- TIVEEP0A

USB_TXACTIVE: Address: 0x4001100C Reset: 0x0

Bitname	Bitfield	Access	Description
USB_TXACTIVEEP6B	[14]	R	This bit is set while endpoint 6, buffer B is active.
USB_TXACTIVEEP5B	[13]	R	This bit is set while endpoint 5, buffer B is active.
USB_TXACTIVEEP4B	[12]	R	This bit is set while endpoint 4, buffer B is active.
USB_TXACTIVEEP3B	[11]	R	This bit is set while endpoint 3, buffer B is active.
USB_TXACTIVEEP2B	[10]	R	This bit is set while endpoint 2, buffer B is active.
USB_TXACTIVEEP1B	[9]	R	This bit is set while endpoint 1, buffer B is active.
USB_TXACTIVEEP0B	[8]	R	This bit is set while endpoint 0, buffer B is active.
USB_TXACTIVEEP6A	[6]	R	This bit is set while endpoint 6, buffer A is active.
USB_TXACTIVEEP5A	[5]	R	This bit is set while endpoint 5, buffer A is active.
USB_TXACTIVEEP4A	[4]	R	This bit is set while endpoint 4, buffer A is active.
USB_TXACTIVEEP3A	[3]	R	This bit is set while endpoint 3, buffer A is active.
USB_TXACTIVEEP2A	[2]	R	This bit is set while endpoint 2, buffer A is active.
USB_TXACTIVEEP1A	[1]	R	This bit is set while endpoint 1, buffer A is active.
USB_TXACTIVEEP0A	[0]	R	This bit is set while endpoint 0, buffer A is active.

SILICON LABS

Register 9.9. USB_TXBUFSIZEEP0y
USB_TXBUFSIZEEP0A: Number of Bytes to Transmit for Endpoint 0, Buffer A Register
USB_TXBUFSIZEEP0B: Number of Bytes to Transmit for Endpoint 0, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0			USB_TXBUFSIZEEP0y		

USB_TXBUFSIZEEP0A: Address: 0x40011010 Reset: 0x0
USB_TXBUFSIZEEP0B: Address: 0x4001102C Reset: 0x0

Bitname	Bitfield	Access	Description
USB_ TXBUFSI- ZEEP0y	[3:0]	RW	Size, in bytes, of data to transmit for endpoint 0, buffer y. Must be written before TX_LOAD is set and remain unmodified while TX_LOAD is set.

Register 9.10. USB_TXBUFSIZEEP1y
USB_TXBUFSIZEEP1A: Number of Bytes to Transmit for Endpoint 1, Buffer A Register
USB_TXBUFSIZEEP1B: Number of Bytes to Transmit for Endpoint 1, Buffer B Register

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

USB_TXBUFSIZEEP1A: Address: 0x40011014 Reset: 0x0
USB_TXBUFSIZEEP1B: Address: 0x40011030 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_ TXBUFSIZEEP1y	[3:0]	RW	Size, in bytes, of data to transmit for endpoint 1, buffer y. Must be written before TX_LOAD is set and remain unmodified while TX_-LOAD is set.

SILICON LABS

Register 9.11. USB_TXBUFSIZEEP2y
USB_TXBUFSIZEEP2A: Number of Bytes to Transmit for Endpoint 2, Buffer A Register
USB_TXBUFSIZEEP2B: Number of Bytes to Transmit for Endpoint 2, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	USB_TXBUFSIZEEP2y			

USB_TXBUFSIZEEP2A: Address: 0x40011018 Reset: 0x0
USB_TXBUFSIZEEP2A: Address: 0x40011034 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_ TXBUFSIZEEP2y	[3:0]	RW	Size, in bytes, of data to transmit for endpoint 2, buffer y. Must be written before TX_LOAD is set and remain unmodified while TX_LOAD is set.

EM358x

Register 9.12. USB_TXBUFSIZEEP3y
USB_TXBUFSIZEEP3A: Number of Bytes to Transmit for Endpoint 3, Buffer A Register
USB_TXBUFSIZEEP3B: Number of Bytes to Transmit for Endpoint 3, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	USB_TXBUFSIZEEP3y						

USB_TXBUFSIZEEP3A: Address: 0x4001101C Reset: 0x0
USB_TXBUFSIZEEP3B: Address: 0x40011038 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_TXBUFSI-ZEEP3y	[6:0]	RW	Size, in bytes, of data to transmit for endpoint 3, buffer y. Must be written before TX_LOAD is set and remain unmodified while TX_LOAD is set.

Register 9.13. USB_TXBUFSIZEEP4y
USB_TXBUFSIZEEP4A: Number of Bytes for Endpoint 4, Buffer A Register
USB_TXBUFSIZEEP4B: Number of Bytes for Endpoint 4, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	USB_TXBUFSIZEEP4y					

USB_TXBUFSIZEEP4A: Address: 0x40011020 Reset: 0x0
USB_TXBUFSIZEEP4B: Address: 0x4001103C Reset: 0x0

Bitname	Bitfield	Access	Description
USB_ TXBUFSIZEEP4y	[5:0]	RW	Size, in bytes, of data to transmit for endpoint 4, buffer y. Must be written before TX_LOAD is set and remain unmodified while TX_LOAD is set.

Register 9.14. USB_TXBUFSIZEEP5y
USB_TXBUFSIZEEP5A: Number of Bytes for Endpoint 5, Buffer A Register
USB_TXBUFSIZEEP5B: Number of Bytes for Endpoint 5, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	USB_TXBUFSIZEEP5y						

USB_TXBUFSIZEEP5A: Address: 0x40011024 Reset: 0x0
USB_TXBUFSIZEEP5B: Address: 0x40011040 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_ TXBUFSIZEEP5y	[6:0]	RW	Size, in bytes, of data to transmit for endpoint 5, buffer y. Must be written before TX_LOAD is set and remain unmodified while TX_LOAD is set.

SILICON LABS

Register 9.15. USB_TXBUFSIZEEP6y
USB_TXBUFSIZEEP6A: Number of Bytes for Endpoint 6, Buffer A Register
USB_TXBUFSIZEEP6B: Number of Bytes for Endpoint 6, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	USB_TXBUFSIZEEP6y	
Bit	7	6	5	4	3	2	1	0
Name	USB_TXBUFSIZEEP6y							

USB_TXBUFSIZEEP6A: Address: 0x40011028 Reset: 0x0
USB_TXBUFSIZEEP6B: Address: 0x40011044 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_TXBUFSIZEEP6y	[9:0]	RW	Size, in bytes, of data to transmit for endpoint 5, buffer y. Must be written before TX_LOAD is set and remain unmodified while TX_LOAD is set.

Register 9.16. USB_RXVALID: USB Buffer A and Buffer B Reception Valid Status Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	USB_RXVA-LIDEP6B	USB_RXVA-LIDEP5B	USB_RXVA-LIDEP4B	USB_RXVALIDEP3B	USB_RXVA-LIDEP2B	USB_RXVA-LIDEP1B	USB_RXVA-LIDEP0B
Bit	7	6	5	4	3	2	1	0
Name	0	USB_RXVA-LIDEP6A	USB_RXVA-LIDEP5A	USB_RXVA-LIDEP4A	USB_RXVALIDEP3A	USB_RXVA-LIDEP2A	USB_RXVA-LIDEP1A	USB_RXVA-LIDEP0A

USB_RXVALID: Address: 0x40011048 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_RXVALIDEP6B	[14]	RW	This bit is set when endpoint 6, buffer B reception is valid.
USB_RXVALIDEP5B	[13]	RW	This bit is set when endpoint 5, buffer B reception is valid.
USB_RXVALIDEP4B	[12]	RW	This bit is set when endpoint 4, buffer B reception is valid.
USB_RXVALIDEP3B	[11]	RW	This bit is set when endpoint 3, buffer B reception is valid.
USB_RXVALIDEP2B	[10]	RW	This bit is set when endpoint 2, buffer B reception is valid.
USB_RXVALIDEP1B	[9]	RW	This bit is set when endpoint 1, buffer B reception is valid.
USB_RXVALIDEP0B	[8]	RW	This bit is set when endpoint 0, buffer B reception is valid.
USB_RXVALIDEP6A	[6]	RW	This bit is set when endpoint 6, buffer A reception is valid.
USB_RXVALIDEP5A	[5]	RW	This bit is set when endpoint 5, buffer A reception is valid.
USB_RXVALIDEP4A	[4]	RW	This bit is set when endpoint 4, buffer A reception is valid.
USB_RXVALIDEP3A	[3]	RW	This bit is set when endpoint 3, buffer A reception is valid.
USB_RXVALIDEP2A	[2]	RW	This bit is set when endpoint 2, buffer A reception is valid.
USB_RXVALIDEP1A	[1]	RW	This bit is set when endpoint 1, buffer A reception is valid.
USB_RXVALIDEP0A	[0]	RW	This bit is set when endpoint 0, buffer A reception is valid.

*Note: USB_RXVALIDxy bits are set when endpoint x, buffer y reception is valid. USB_RXVALIDxy bits must be written 1 to confirm software acceptances of the data. If the USB_RXVALIDxy bit is not written 1, reception on endpoint x, buffer y will stall.

SILICON LABS

Register 9.17. USB_RXBUFSIZEEP0y
USB_RXBUFSIZEEP0A: Number of Bytes Received in Endpoint 0, Buffer A Register
USB_RXBUFSIZEEP0B: Number of Bytes Received in Endpoint 0, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	USB_RXBUFSIZEEP0y			

USB_RXBUFSIZEEP0A: Address: 0x40011078 Reset: 0x0
USB_RXBUFSIZEEP0B: Address: 0x40011094 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_ RXBUFSIZEEP0y	[3:0]	R	Size, in bytes, of data received on endpoint 0, buffer y. This register is valid only after the corresponding USB_RXVALIDEP0y bit is set.

Register 9.18. USB_RXBUFSIZEEP1y
USB_RXBUFSIZEEP1A: Number of Bytes Received in Endpoint 1, Buffer A Register
USB_RXBUFSIZEEP1B: Number of Bytes Received in Endpoint 1, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	USB_RXBUFSIZEEP1y			

USB_RXBUFSIZEEP1A: Address: 0x4001107C Reset: 0x0
USB_RXBUFSIZEEP1B: Address: 0x40011098 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_RXBUFSIZEEP1y	[3:0]	RW	Size, in bytes, of data received on endpoint 1, buffer y. This register is valid only after the corresponding USB_RXVALIDEP1y bit is set.

SILICON LABS

Register 9.19. USB_RXBUFSIZEEP2y
USB_RXBUFSIZEEP2A: Number of Bytes Received in Endpoint 2, Buffer A Register
USB_RXBUFSIZEEP2B: Number of Bytes Received in Endpoint 2, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	USB_RXBUFSIZEEP2y			

USB_RXBUFSIZEEP2A: Address: 0x40011080 Reset: 0x0
USB_RXBUFSIZEEP2B: Address: 0x4001109C Reset: 0x0

Bitname	Bitfield	Access	Description
USB_RXBUFSIZEEP2y	[3:0]	RW	Size, in bytes, of data received on endpoint 2, buffer y. This register is valid only after the corresponding USB_RXVALIDEP2y bit is set.

EM358x

Register 9.20. USB_RXBUFSIZEEP3y
USB_RXBUFSIZEEP3A: Number of Bytes Received in Endpoint 3, Buffer A Register
USB_RXBUFSIZEEP2B: Number of Bytes Received in Endpoint 3, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	USB_RXBUFSIZEEP3y						

USB_RXBUFSIZEEP3A: Address: 0x40011084 Reset: 0x0
USB_RXBUFSIZEEP3B: Address: 0x400110A0 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_RXBUFSIZEEP3y	[6:0]	RW	Size, in bytes, of data received on endpoint 3, buffer y. This register is valid only after the corresponding USB_RXVALIDEP3y bit is set.

Register 9.21. USB_RXBUFSIZEEP4y
USB_RXBUFSIZEEP4A: Number of Bytes Received in Endpoint 4, Buffer A Register
USB_RXBUFSIZEEP4B: Number of Bytes Received in Endpoint 4, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	0	USB_RXBUFSIZEEP3y					

USB_RXBUFSIZEEP4A: Address: 0x40011084 Reset: 0x0
USB_RXBUFSIZEEP4B: Address: 0x400110A0 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_ RXBUFSIZEEP4y	[5:0]	RW	Size, in bytes, of data received on endpoint 4, buffer y. This register is valid only after the corresponding USB_RXVALIDEP4y bit is set.

EM358x

Register 9.22. USB_RXBUFSIZEEP5y
USB_RXBUFSIZEEP5A: Number of Bytes Received in Endpoint 5, Buffer A Register
USB_RXBUFSIZEEP5B: Number of Bytes Received in Endpoint 5, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	USB_RXBUFSIZEEP5y						

USB_RXBUFSIZEEP5A: Address: 0x4001108C Reset: 0x0
USB_RXBUFSIZEEP5B: Address: 0x400110A8 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_RXBUFSIZEEP5y	[6:0]	RW	Size, in bytes, of data received on endpoint 5, buffer y. This register is valid only after the corresponding USB_RXVALIDEP5y bit is set.

SILICON LABS

Register 9.23. USB_RXBUFSIZEEP6y
USB_RXBUFSIZEEP6A: Number of Bytes Received in Endpoint 6, Buffer A Register
USB_RXBUFSIZEEP6B: Number of Bytes Received in Endpoint 6, Buffer B Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	USB_RXBUFSIZEEP6y	
Bit	7	6	5	4	3	2	1	0
Name	USB_RXBUFSIZEEP6y							

USB_RXBUFSIZEEP6A: Address: 0x40011090 Reset: 0x0
USB_RXBUFSIZEEP6B: Address: 0x400110AC Reset: 0x0

Bitname	Bitfield	Access	Description
USB_RXBUFSIZEEP6y	[9:0]	RW	Size, in bytes, of data received on endpoint 6, buffer y. This register is valid only after the corresponding USB_RXVALIDEP6y bit is set.

Register 9.24. USB_BUFCLR: USB IN Buffer Clear Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	USB_BUF-CLRINEP6	USB_BUF-CLRINEP5	USB_BUF-CLRINEP4	USB_BUF-CLRINEP3	USB_BUF-CLRINEP2	USB_BUF-CLRINEP1	USB_BUF-CLRINEP0

USB_BUFCLR: Address: 0x40011064 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_BUFCLRINEP6	[6]	W	Write this bit to force clearing of endpoint 6 IN buffer. Auto-clears to 0.
USB_BUFCLRINEP5	[5]	W	Write this bit to force clearing of endpoint 5 IN buffer. Auto-clears to 0.
USB_BUFCLRINEP4	[4]	W	Write this bit to force clearing of endpoint 4 IN buffer. Auto-clears to 0.
USB_BUFCLRINEP3	[3]	W	Write this bit to force clearing of endpoint 3 IN buffer. Auto-clears to 0.
USB_BUFCLRINEP2	[2]	W	Write this bit to force clearing of endpoint 2 IN buffer. Auto-clears to 0.
USB_BUFCLRINEP1	[1]	W	Write this bit to force clearing of endpoint 1 IN buffer. Auto-clears to 0.
USB_BUFCLRINEP0	[0]	W	Write this bit to force clearing of endpoint 6 IN buffer. Auto-clears to 0.

*Note: USB_BUFCLR forces clearing of the internal state of the IN buffer with respect to DMA operation. A USB reset will reset the core so USB_BUFCLR must be used to keep the DMA in sync with the USB core state. Some protocols might require aborting further IN data based upon received OUT data and therefore specific endpoints might have to be cleared.

SILICON LABS

Register 9.25. USB_STALLIN: USB Endpoint IN Stall Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	USB_STAL-LINEP6	USB_STAL-LINEP5	USB_STAL-LINEP4	USB_STAL-LINEP3	USB_STAL-LINEP2	USB_STAL-LINEP1	USB_STAL-LINEP0

USB_STALLIN: Address: 0x40011054 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_STALLINEP6	[6]	RW	Write this bit to stall endpoint 6 IN. This bit will not auto-clear.
USB_STALLINEP5	[5]	RW	Write this bit to stall endpoint 5 IN. This bit will not auto-clear.
USB_STALLINEP4	[4]	RW	Write this bit to stall endpoint 4 IN. This bit will not auto-clear.
USB_STALLINEP3	[3]	RW	Write this bit to stall endpoint 3 IN. This bit will not auto-clear.
USB_STALLINEP2	[2]	RW	Write this bit to stall endpoint 2 IN. This bit will not auto-clear.
USB_STALLINEP1	[1]	RW	Write this bit to stall endpoint 1 IN. This bit will not auto-clear.
USB_STALLINEP0	[0]	RW	Write this bit to stall endpoint 0 IN. This bit will not auto-clear.

Register 9.26. USB_STALLOUT: USB Endpoint OUT Stall Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	USB_STALL OUTEP6	USB_STALL OUTEP5	USB_STALL OUTEP4	USB_STALL OUTEP3	USB_STALL OUTEP2	USB_STALL OUTEP1	USB_STALL OUTEP0

USB_STALLOUT: Address: 0x40011054 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_STALLOUTEP6	[6]	RW	Write this bit to stall endpoint 6 OUT. This bit will not auto-clear.
USB_STALLOUTEP5	[5]	RW	Write this bit to stall endpoint 5 OUT. This bit will not auto-clear.
USB_STALLOUTEP4	[4]	RW	Write this bit to stall endpoint 4 OUT. This bit will not auto-clear.
USB_STALLOUTEP3	[3]	RW	Write this bit to stall endpoint 3 OUT. This bit will not auto-clear.
USB_STALLOUTEP2	[2]	RW	Write this bit to stall endpoint 2 OUT. This bit will not auto-clear.
USB_STALLOUTEP1	[1]	RW	Write this bit to stall endpoint 1 OUT. This bit will not auto-clear.
USB_STALLOUTEP0	[0]	RW	Write this bit to stall endpoint 0 OUT. This bit will not auto-clear.

Register 9.27. USB_RESUME: USB Resume Register

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

USB_RESUME: Address: 0x40011068 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_RESUME	[0]	W	Write this bit to resume from the suspended state. This activity is also known as remote-wakeup. Auto-clears to 0.

Register 9.28. USB_PIPECLR: USB Force DMA Pipeline Clearing Register

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

USB_PIPECLR: Address: 0x40011060 Reset: 0x0

Bitname	Bitfield	Access	Description
USB_RXPIPECLR	[1]	R	Write this bit to force clearing of the receive DMA pipeline. Auto-clears to 0.
USB_TXPIPECLR	[0]	R	Write this bit to force clearing of the transmit DMA pipeline. Auto-clears to 0.

SILICON LABS

Register 9.29. INT_USBFLAG: USB Interrupt Flag Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	INT_USB-WAKEUP	INT_US-BRESUME	INT_USB-SUSPEND	INT_US-BRESET	INT_USB-SOF	INT_USB-NAK	INT_USB-PIPERXOVF	INT_USBPI-PETXUND
Bit	15	14	13	12	11	10	9	8
Name	INT_USB-BUFRXOVF	INT_USB-BUFTXUND	INT_US-BRXVA-LIDEP6	INT_US-BRXVA-LIDEP5	INT_US-BRXVA-LIDEP4	INT_US-BRXVA-LIDEP3	INT_US-BRXVA-LIDEP3	INT_US-BRXVA-LIDEP1
Bit	7	6	5	4	3	2	1	0
Name	INT_US-BRXVA-LIDEP0	INT_USBTX ACTIVEEP6	INT_USBTX ACTIVEEP5	INT_USBTX ACTIVEEP4	INT_USBTX ACTIVEEP3	INT_USBTX ACTIVEEP2	INT_USBTX ACTIVEEP1	INT_USBTX ACTIVEEP0

INT_USBFLAG: Address: 0x4000A888 Reset: 0x0

Bitname	Bitfield	Access	Description
INT_USBWAKEUP	[23]	RW	A successful remote wakeup by this device pends this interrupt.
INT_USBRESUME	[22]	RW	When suspended, a resume on the bus pends this interrupt.
INT_USBSUSPEND	[21]	RW	Suspending this device pends this interrupt.
INT_USBRESET	[20]	RW	When a USB reset occurs it resets the core and pends this interrupt.
INT_USBSOF	[19]	RW	A start of frame packet pends this interrupt.
INT_USBNAK	[18]	RW	A NAK handshake packet pends this interrupt.
INT_USBPIPERXOVF	[17]	RW	Pipeline reception overflow pends this interrupt.
INT_USBPIPETXUND	[16]	RW	Pipeline transmit underflow pends this interrupt.
INT_USBBUFRXOVF	[15]	RW	Buffer reception overflow pends this interrupt.
INT_USBBUFTXUND	[14]	RW	Buffer transmit underflow pends this interrupt.
INT_USBRXVALIDEP6	[13]	RW	The rising edge of either USB_RXVALIDEP6y bit pends this interrupt.
INT_USBRXVALIDEP5	[12]	RW	The rising edge of either USB_RXVALIDEP5y bit pends this interrupt.
INT_USBRXVALIDEP4	[11]	RW	The rising edge of either USB_RXVALIDEP4y bit pends this interrupt.
INT_USBRXVALIDEP3	[10]	RW	The rising edge of either USB_RXVALIDEP3y bit pends this interrupt.
INT_USBRXVALIDEP2	[9]	RW	The rising edge of either USB_RXVALIDEP2y bit pends this interrupt.

Bitname	Bitfield	Access	Description
INT_USBRXVALIDEP1	[8]	RW	The rising edge of either USB_RXVALIDEP1y bit pends this interrupt.
INT_USBRXVALIDEP0	[7]	RW	The rising edge of either USB_RXVALIDEP0y bit pends this interrupt.
INT_USBTXACTIVEEP6	[6]	RW	The falling edge of either USB_TXACTIVEEP6y bit pends this interrupt.
INT_USBTXACTIVEEP5	[5]	RW	The falling edge of either USB_TXACTIVEEP5y bit pends this interrupt.
INT_USBTXACTIVEEP4	[4]	RW	The falling edge of either USB_TXACTIVEEP4y bit pends this interrupt.
INT_USBTXACTIVEEP3	[3]	RW	The falling edge of either USB_TXACTIVEEP3y bit pends this interrupt.
INT_USBTXACTIVEEP2	[2]	RW	The falling edge of either USB_TXACTIVEEP2y bit pends this interrupt.
INT_USBTXACTIVEEP1	[1]	RW	The falling edge of either USB_TXACTIVEEP1y bit pends this interrupt.
INT_USBTXACTIVEEP0	[0]	RW	The falling edge of either USB_TXACTIVEEP0y bit pends this interrupt.

SILICON LABS

Register 9.30. INT_USBCFG: USB Interrupt Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	INT_USB-WAKEUP	INT_US-BRESUME	INT_USB-SUSPEND	INT_USBRESET	INT_USB-SOF	INT_USB-NAK	INT_USB-PIPERXOVF	INT_USBPI-PETXUND
Bit	15	14	13	12	11	10	9	8
Name	INT_USB-BUFRXOVF	INT_USB-BUFTXUND	INT_USBRXVA-LIDEP6	INT_USBRXVA-LIDEP5	INT_USBRXVA-LIDEP4	INT_USBRXVA-LIDEP3	INT_USBRXVA-LIDEP3	INT_USBRXVA-LIDEP1
Bit	7	6	5	4	3	2	1	0
Name	INT_US-BRXVA-LIDEP0	INT_USBTXACTIVEEP6	INT_USBTXACTIVEEP5	INT_USBTXACTIVEEP4	INT_USBTXACTIVEEP3	INT_USBTXACTIVEEP2	INT_USBTXACTIVEEP1	INT_USBTXACTIVEEP0

INT_USBCFG: Address: 0x4000A88C Reset: 0x0

Bitname	Bitfield	Access	Description
INT_USBWAKEUP	[23]	RW	A successful remote wakeup by this device interrupt enable.
INT_USBRESUME	[22]	RW	When suspended, a resume on the bus interrupt enable.
INT_USBSUSPEND	[21]	RW	Suspending this device interrupt enable.
INT_USBRESET	[20]	RW	When a USB reset occurs it resets the core interrupt enable.
INT_USBSOF	[19]	RW	A start of frame packet interrupt enable.
INT_USBNAK	[18]	RW	A NAK handshake packet interrupt enable.
INT_USBPIPERXOVF	[17]	RW	Pipeline reception overflow interrupt enable.
INT_USBPIPETXUND	[16]	RW	Pipeline transmit underflow interrupt enable.
INT_USBBUFRXOVF	[15]	RW	Buffer reception overflow interrupt enable.
INT_USBBUFTXUND	[14]	RW	Buffer transmit underflow interrupt enable.
INT_USBRXVALIDEP6	[13]	RW	The rising edge of either USB_RXVALIDEP6y bit interrupt enable.
INT_USBRXVALIDEP5	[12]	RW	The rising edge of either USB_RXVALIDEP5y bit interrupt enable.
INT_USBRXVALIDEP4	[11]	RW	The rising edge of either USB_RXVALIDEP4y bit interrupt enable.
INT_USBRXVALIDEP3	[10]	RW	The rising edge of either USB_RXVALIDEP3y bit interrupt enable.
INT_USBRXVALIDEP2	[9]	RW	The rising edge of either USB_RXVALIDEP2y bit interrupt enable.

EM358x

Bitname	Bitfield	Access	Description
INT_USBRXVALIDEP1	[8]	RW	The rising edge of either USB_RXVALIDEP1y bit interrupt enable.
INT_USBRXVALIDEP0	[7]	RW	The rising edge of either USB_RXVALIDEP0y bit interrupt enable.
INT_USBTXACTIVEEP6	[6]	RW	The falling edge of either USB_TXACTIVEEP6y bit interrupt enable.
INT_USBTXACTIVEEP5	[5]	RW	The falling edge of either USB_TXACTIVEEP5y bit interrupt enable.
INT_USBTXACTIVEEP4	[4]	RW	The falling edge of either USB_TXACTIVEEP4y bit interrupt enable.
INT_USBTXACTIVEEP3	[3]	RW	The falling edge of either USB_TXACTIVEEP3y bit interrupt enable.
INT_USBTXACTIVEEP2	[2]	RW	The falling edge of either USB_TXACTIVEEP2y bit interrupt enable.
INT_USBTXACTIVEEP1	[1]	RW	The falling edge of either USB_TXACTIVEEP1y bit interrupt enable.
INT_USBTXACTIVEEP0	[0]	RW	The falling edge of either USB_TXACTIVEEP0y bit interrupt enable.

SILICON LABS

10. General Purpose Timers (TIM1 and TIM2)

10.1. Introduction

Each of the EM358xs two general-purpose timers consists of a 16-bit auto-reload counter driven by a programmable prescaler. They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM). Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler. The timers are completely independent, and do not share any resources. They can be synchronized together as described in the Timer Synchronizationsection.

The two general-purpose timers, TIM1 and TIM2, have the following features:

- 16-bit up, down, or up/down auto-reload counter.
- Programmable prescaler to divide the counter clock by any power of two from 1 through 32768.
- 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge- and center-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect the timers.
- Flexible clock source selection:
  - Peripheral clock (PCLK at 6 or 12 MHz)
  - 32.768 kHz external clock (if available)
  - 1 kHz clock
  - GPIO input
- Interrupt generation on the following events:
  - Update: counter overflow/underflow, counter initialization (software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoders and Hall sensors for positioning applications.
- Trigger input for external clock or cycle-by-cycle current management.

Figure 10.1 shows an overview of a timer's internal structure.

Note: Because the two timers are identical, the notation TIMx refers to either TIM1 or TIM2. For example, TIMx_PSC refers to both TIM1_PSC and TIM2_PSC. Similarly, "y" refers to any of the four channels of a given timer, so for example, OCy refers to OC1, OC2, OC3, and OC4.
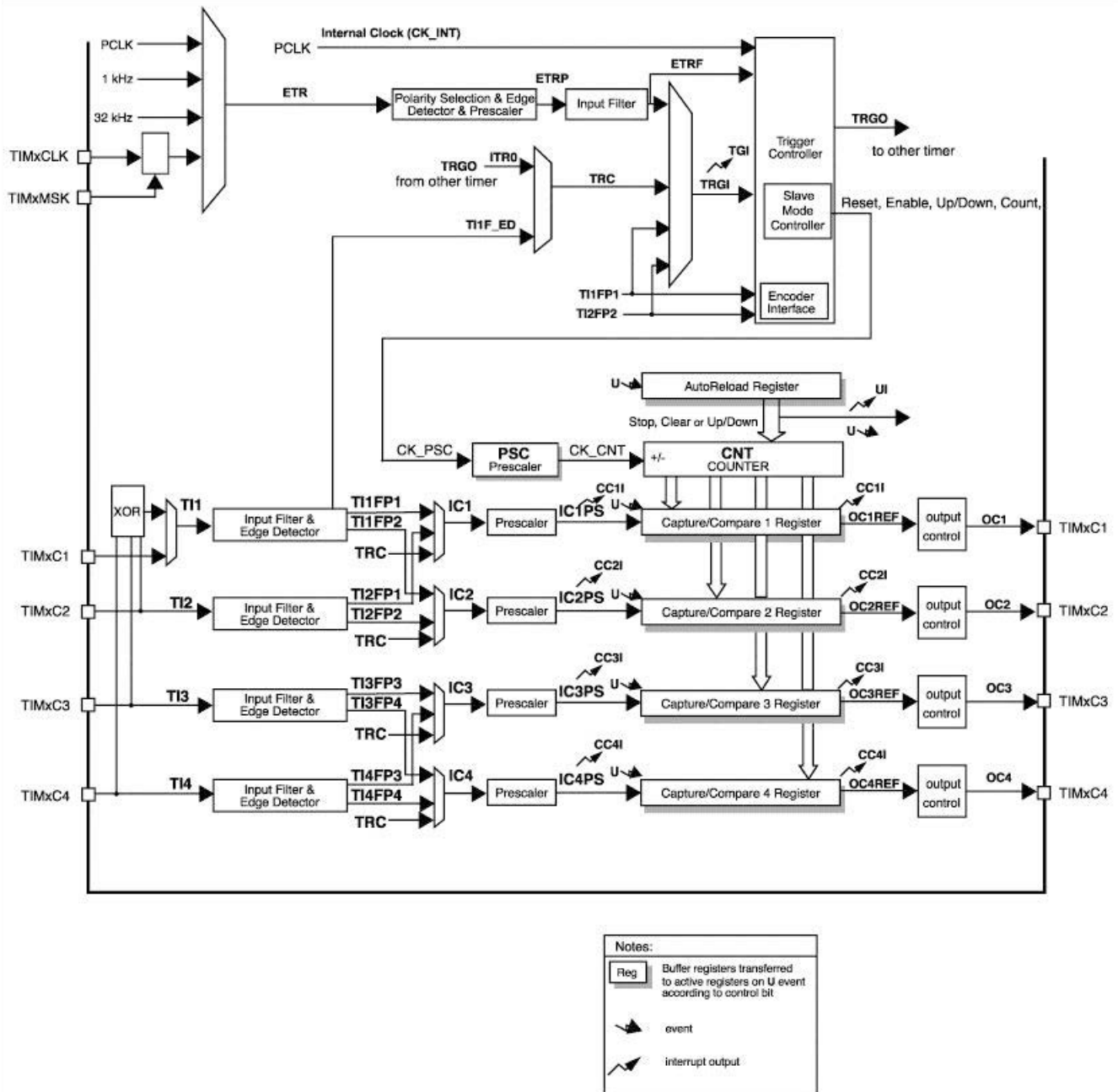
Figure 10.1. General-Purpose Timer Block Diagram

Note: The internal signals shown in Figure 10.1 are described in the Timer Signal Descriptions "10.3.15. Timer Signal Descriptions" on page 195, and are used throughout the text to describe how the timer components are interconnected.

10.2. GPIO Usage

The timers can optionally use GPIOs in the PA and PB ports for external inputs or outputs. As with all EM358x digital inputs, a GPIO used as a timer input can be shared with other uses of the same pin. Available timer inputs include an external timer clock, a clock mask, and four input channels. Any GPIO used as a timer output must be configured as an alternate output and is controlled only by the timer.

Many of the GPIOs that can be assigned as timer outputs can also be used by another on-chip peripheral such as a serial controller. Using a GPIO as a timer output takes precedence over another peripheral function, as long as the channel is configured as an output in the TIMx_CCMR1 register and is enabled in the TIMx_CCER register.

The GPIOs that can be used by Timer 1 are fixed, but the GPIOs that can be used as Timer 2 channels can be mapped to either of two pins, as shown in Table 10.1. The Timer 2 Option Register (TIM2_OR) has four single bit fields (TIM_REMAPCy) that control whether a Timer 2 channel is mapped to its default GPIO in port PA, or remapped to a GPIO in PB.

Table 10.1 specifies the pins that may be assigned to Timer 1 and Timer 2 functions.

Table 10.1. Timer GPIO Usage

Signal (Direction)	TIMxC1 (In or Out)	TIMxC2 (In or Out)	TIMxC3 (In or Out)	TIMxC4 (In or Out)	TIMxCLK (In)	TIMxMSK (In)
Timer 1	PB6	PB7	PA6	PA7	PB0	PB5
Timer 2 (TIM_REMAPCy = 0)	PA0	PA3	PA1	PA2	PB5	PB0
Timer 2 (TIM_REMAPCy = 1)	PB1	PB2	PB3	PB4	PB5	PB0

The TIMxCLK and TIMxMSK inputs can be used only in the external clock modes; refer to "10.3.3.2. External Clock Source Mode 1" on page 176 and "10.3.3.3. External Clock Source Mode 2" on page 177 for details concerning their use.

10.3. Timer Functional Description

10.3.1. Time-Base Unit

The main block of the general purpose timer is a 16-bit counter with its related auto-reload register. The counter can count up, down, or alternate up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register, and the prescaler register can be written to or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

Some timer registers cannot be directly accessed by software, which instead reads and writes a "buffer register". The internal registers actually used for timer operations are called "shadow registers".

The auto-reload register is buffered. Writing to or reading from the auto-reload register accesses the buffer register. The contents of the buffer register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload buffer enable bit (TIM_ARBE) in the TIMx_CR1 register. The UEV is generated when both the counter reaches the overflow (or underflow when down-counting) and when the TIM_UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. UEV generation is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (TIM_CEN) in the TIMx_CR1 register is set. Refer also to the slave mode controller description in the Timers and

External Trigger Synchronization section to get more details on counter enabling.

Note that the actual counter enable signal CNT_EN is set one clock cycle after TIM_CEN.

Note: When the EM358x enters debug mode and the ARM® Cortex™-M3 core is halted, the counters continue to run normally.

10.3.1.1. Prescaler

The prescaler can divide the counter clock frequency by power of two from 1 through 32768. It is based on a 16-bit counter controlled through the 4-bit TIM_PSCEXP bit field in the TIMx_PSC register. The factor by which the counter is divided is two raised to the power TIM_PSCEXP (2^{TIM_PSCEXP}).

It can be changed on the fly as this control register is buffered. The new prescaler ratio is used starting at the next UEV.

Figure 10.2 gives an example of the counter behavior when the prescaler ratio is changed on the fly.
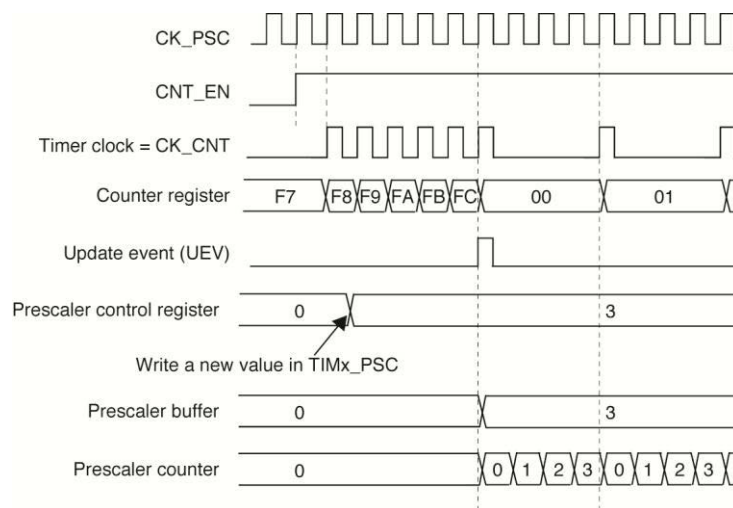
Figure 10.2. Counter Modes

10.3.2. Counter Modes

10.3.2.1. Up-Counting Mode

In up-counting mode, the counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

A UEV can be generated at each counter overflow, by setting the TIM_UG bit in the TIMx_EGR register, or by using the slave mode controller.

Software can disable the UEV by setting the TIM_UDIS bit in the TIMx_CR1 register, to avoid updating the shadow registers while writing new values in the buffer registers. No UEV will occur until the TIM_UDIS bit is written to 0. Both the counter and the prescaler counter restart from 0, but the prescale rate does not change. In addition, if the TIM_URS bit in the TIMx_CR1 register is set, setting the TIM_UG bit generates a UEV but without setting the INT_TIMUIF flag. Thus no interrupt request is sent. This avoids generating both update and capture interrupts when clearing the counter on the capture event.

When a UEV occurs, the update flag (the INT_TIMUIF bit in the INT_TIMxFLAG register) is set (unless TIM_URS is 1) and the following registers are updated:

- The buffer of the prescaler is reloaded with the buffer value (contents of the TIMx_PSC register).
- The auto-reload shadow register is updated with the buffer value (TIMx_ARR).

Figures 10.3, 10.4, 10.5, and 10.6 show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.
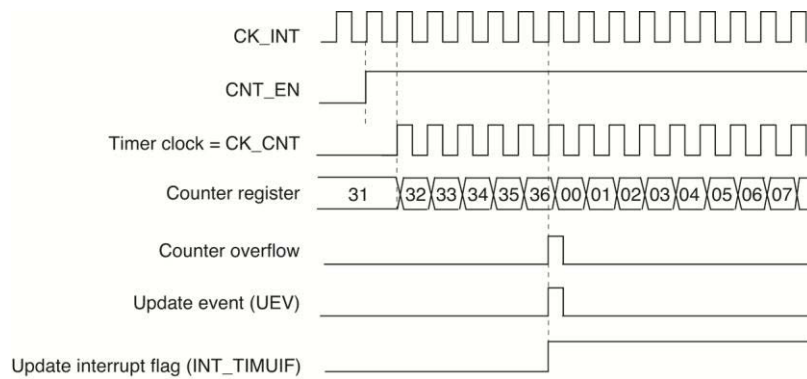
Figure 10.3. Counter Timing Diagram, Internal Clock Divided by 1
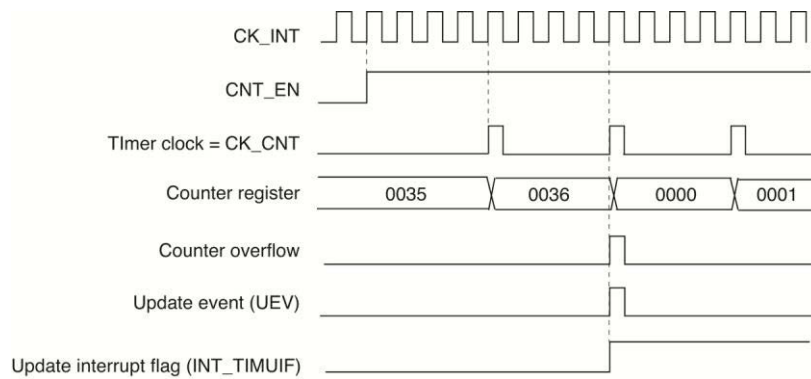
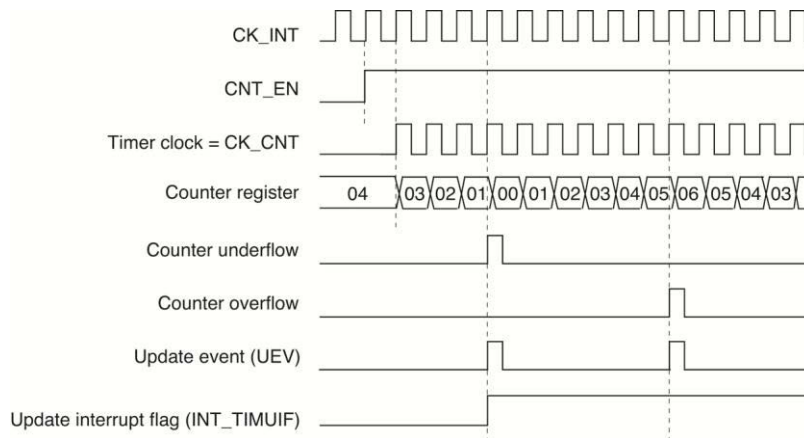**Figure 10.4. Counter Timing Diagram, Internal Clock Divided by 4**
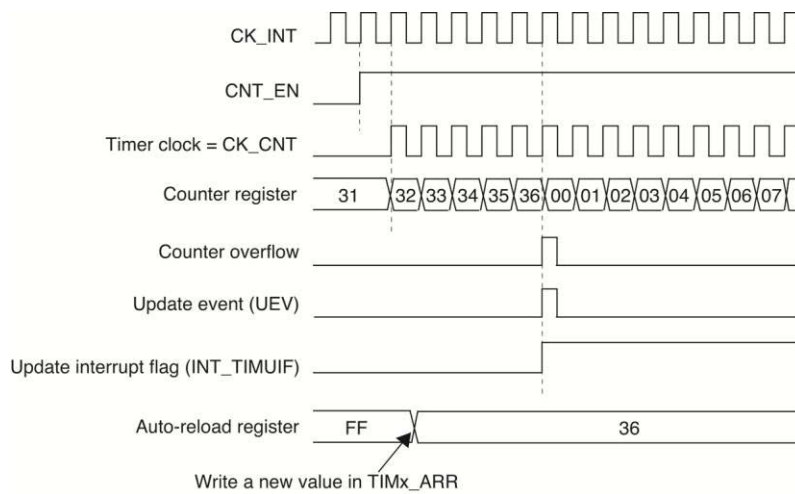
Figure 10.5. Counter Timing Diagram, Update Event when TIM_ARBE = 0 (TIMx_ARR Not Buffered)
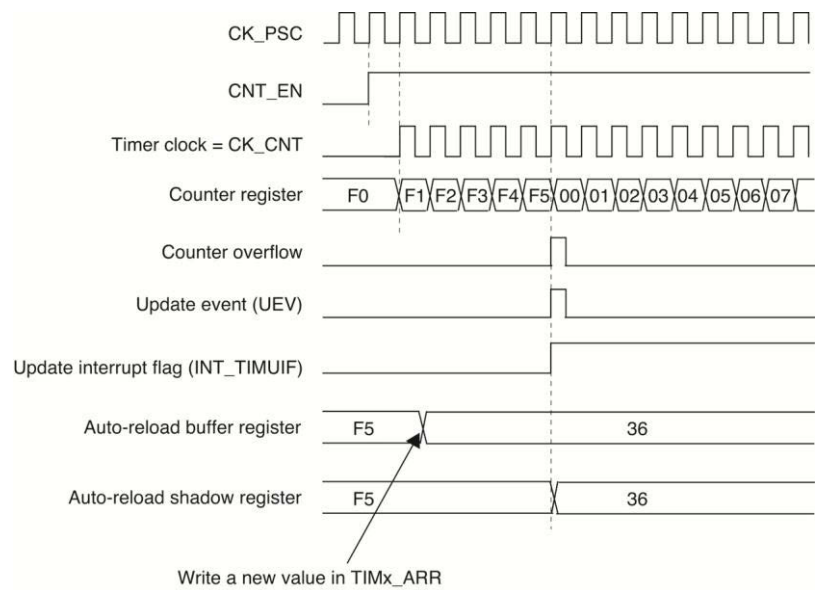
Figure 10.6. Counter Timing Diagram, Update Event when TIM_ARBE = 1 (TIMx_ARR Buffered)

10.3.2.2. Down-Counting Mode

In down-counting mode, the counter counts from the auto-reload value (contents of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

A UEV can be generated at each counter underflow, by setting the TIM_UG bit in the TIMx_EGR register, or by using the slave mode controller. Software can disable the UEV by setting the TIM_UDIS bit in the TIMx_CR1 register, to avoid updating the shadow registers while writing new values in the buffer registers. No UEV occurs until the TIM_UDIS bit is written to 0. However, the counter restarts from the current auto-reload value, whereas the prescaler's counter restarts from 0, but the prescale rate doesn't change.

In addition, if the TIM_URS bit in the TIMx_CR1 register is set, setting the TIM_UG bit generates a UEV, but without setting the INT_TIMUIF flag. Thus no interrupt request is sent. This avoids generating both update and capture interrupts when clearing the counter on the capture event.

When a UEV occurs, the update flag (the INT_TIMUIF bit in the INT_TIMxFLAG register) is set (unless TIM_URS is 1) and the following registers are updated:

- The prescaler shadow register is reloaded with the buffer value (contents of the TIMx_PSC register).
- The auto-reload active register is updated with the buffer value (contents of the TIMx_ARR register). The auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

Figure 10.7 and Figure 10.8 show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.
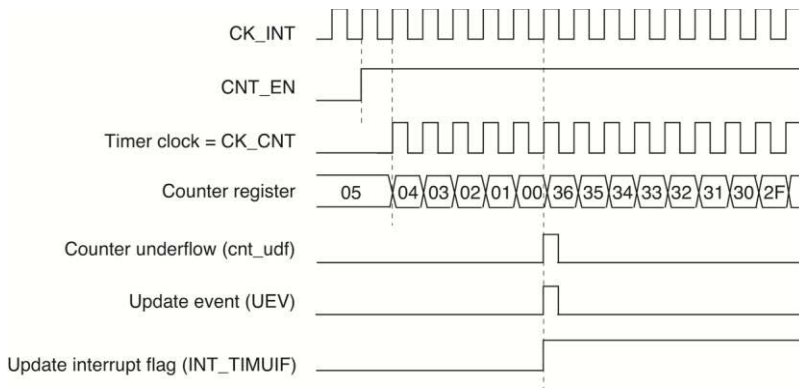
Figure 10.7. Counter Timing Diagram, Internal Clock Divided by 1
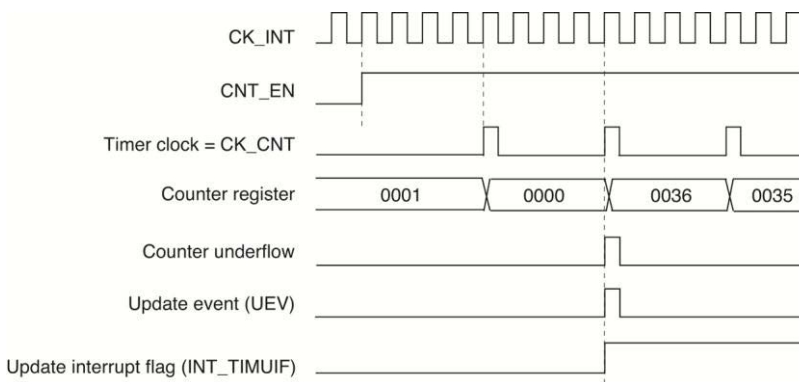
Figure 10.8. Counter Timing Diagram, Internal Clock Divided by 4

10.3.2.3. Center-Aligned Mode (Up/Down Counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register) – 1 and generates a counter overflow event, then counts from the autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

In this mode, the direction bit (TIM_DIR in the TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The UEV can be generated at each counter overflow and at each counter underflow. Setting the TIM_UG bit in the TIMx_EGR register by software or by using the slave mode controller also generates a UEV. In this case, the both the counter and the prescaler's counter restart counting from 0.

Software can disable the UEV by setting the TIM_UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values in the buffer registers. Then no UEV occurs until the TIM_UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the TIM_URS bit in the TIMx_CR1 register is set, setting the TIM_UG bit generates a UEV, but without setting the INT_TIMUIF flag. Thus no interrupt request is sent. This avoids generating both update and capture interrupt when clearing the counter on the capture event.

When a UEV occurs, the update flag (the INT_TIMUIF bit in the INT_TIMxFLAG register) is set (unless TIM_URS is 1) and the following registers are updated:

- The prescaler shadow register is reloaded with the buffer value (contents of the TIMx_PSC register).
- The auto-reload active register is updated with the buffer value (contents of the TIMx_ARR register). If the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one. The counter is loaded with the new value.

Figures 10.9, 10.10, and 10.11 show some examples of the counter behavior for different clock frequencies.

Figure 10.9. Counter Timing Diagram, Internal Clock Divided by 1, TIMx_ARR = 0x6
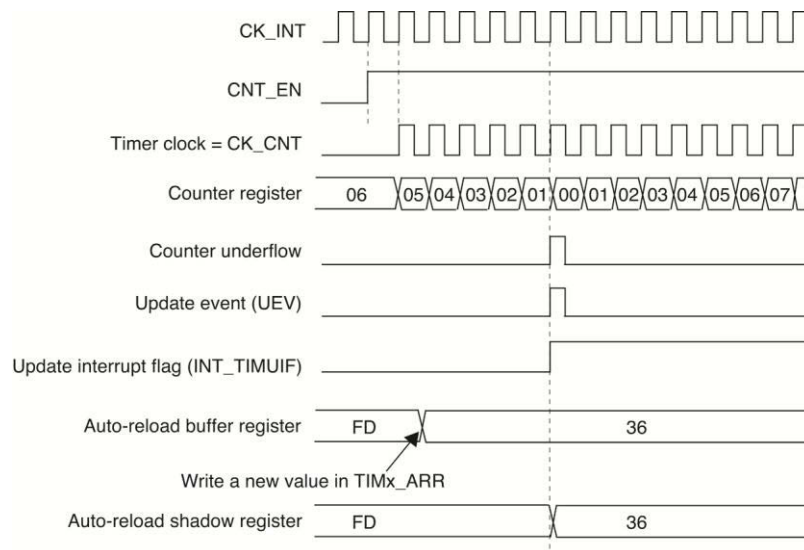
Figure 10.10. Counter Timing Diagram, Update Event with TIM_ARBE = 1 (Counter Underflow)
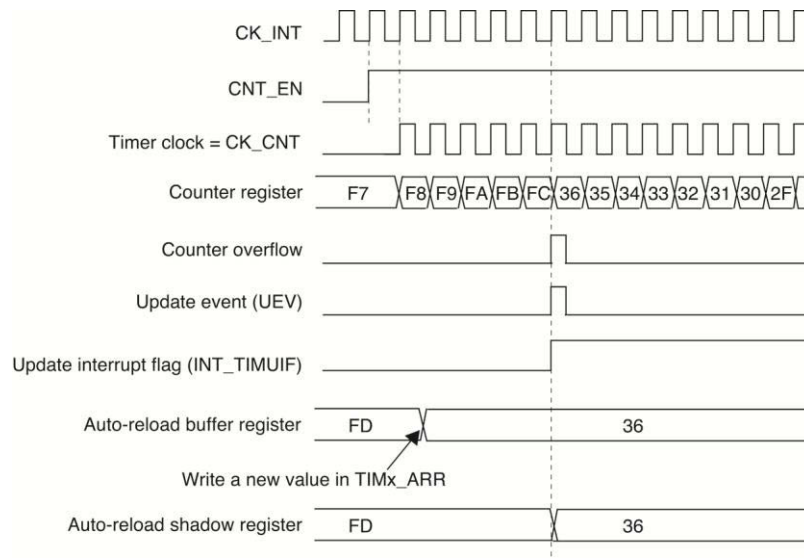
Figure 10.11. Counter Timing Diagram, Update Event with TIM_ARBE = 1 (Counter Overflow)

10.3.3. Clock Selection

The counter clock can be provided by the following clock sources:

- Internal clock (PCLK)
- External clock mode 1: external input pin (TIy)
- External clock mode 2: external trigger input (ETR)
- Internal trigger input (ITR0): using the other timer as prescaler. Refer to "10.3.14.1. Using One Timer as Prescaler for the Other Timer" on page 191 for more details.

10.3.3.1. Internal Clock Source (CK_INT)

The internal clock is selected when the slave mode controller is disabled (TIM_SMS = 000 in the TIMx_SMCR register). In this mode, the TIM_CEN, TIM_DIR (in the TIMx_CR1 register), and TIM_UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software, except for TIM_UG, which remains cleared automatically. As soon as the TIM_CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 10.12 shows the behavior of the control circuit and the up-counter in normal mode, without prescaling.
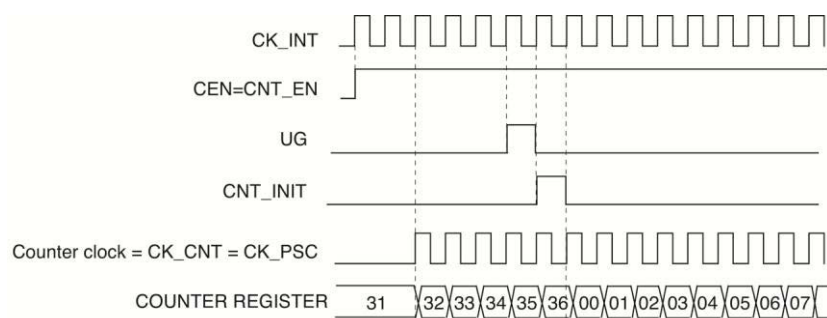
Figure 10.12. Control Circuit in Normal Mode, Internal Clock Divided by 1

10.3.3.2. External Clock Source Mode 1

This mode is selected when TIM_SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input. Figure 10.13 shows the registers and signals used in the example that follows.

Figure 10.13. TI2 External Clock Connection Example

For example, to configure the up-counter to count in response to a rising edge on the TI2 input, use the following procedure:

- Configure channel 2 to detect rising edges on the TI2 input: Write TIM_CC2S = 01 in the TIMx_CCMR1 register.
- Configure the input filter duration: Write the TIM_IC2F bits in the TIMx_CCMR1 register (if no filter is needed, keep TIM_IC2F = 0000).

Note: The capture prescaler is not used for triggering, so it does not need to be configured.

- Select rising edge polarity: Write TIM_CC2P = 0 in the TIMx_CCER register.
- Configure the timer in external clock mode 1: Write TIM_SMS = 111 in the TIMx_SMCR register.
- Select TI2 as the input source: Write TIM_TS = 110 in the TIMx_SMCR register.
- Enable the counter: Write TIM_CEN = 1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the INT_TIMTIF flag is set. The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on the TI2 input. The relationship between rising edges on TI2 and the resulting counter clocks is shown in Figure 10.14.

Figure 10.14. Control Circuit in External Clock Mode 1

10.3.3.3. External Clock Source Mode 2

This mode is selected by writing TIM_ECE = 1 in the TIMx_SMCR register. The counter can count at each rising or falling edge on the external trigger input ETR.

The TIM_EXTRIGSEL bits in the TIMx_OR register select a clock signal that drives ETR, as shown in Table 10.2.

Table 10.2. TIM_EXTRIGSEL Clock Signal Selection

TIM_EXTRIGSEL Bits	Clock Signal Selection
00	PCLK (peripheral clock). When running from the 24 MHz crystal oscillator, the PCLK frequency is 12 MHz. When the 12 MHz RC oscillator is in use, the frequency is 6 MHz.
01	Calibrated 1 kHz internal RC oscillator
10	Optional 32.786 kHz clock
11	TIMxCLK pin. If the TIM_CLKMSKEN bit in the TIMx_OR register is set, this signal is AND'ed with the TIMxMSK pin providing a gated clock input.

Figure 10.15 gives an overview of the external trigger input block.
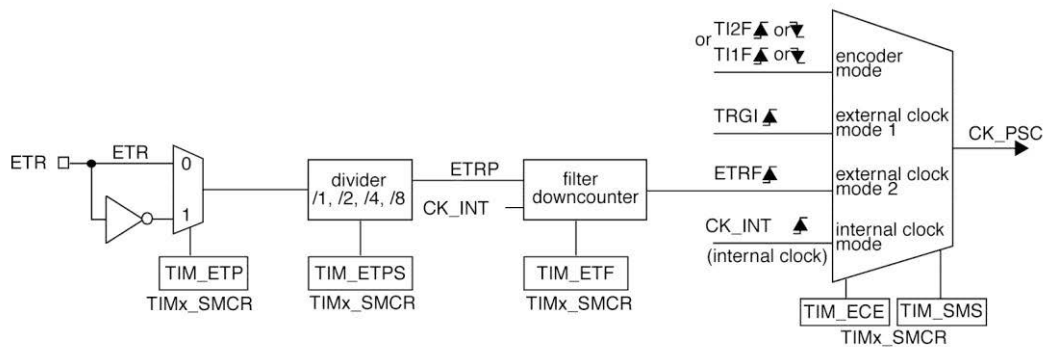
Figure 10.15. External Trigger Input Block

For example, to configure the up-counter to count each 2 rising edges on ETR, use the following procedure:

- Since no filter is needed in this example, write TIM_ETF = 0000 in the TIMx_SMCR register.
- Set the prescaler: Write TIM_ETPS = 01 in the TIMx_SMCR register.
- Select rising edge detection on ETR: WriteTIM_ETP = 0 in the TIMx_SMCR register.
- Enable external clock mode 2: Write TIM_ECE = 1 in the TIMx_SMCR register.
- Enable the counter: Write TIM_CEN = 1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges. The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 10.16 illustrates counting every two rising edges of ETR using external clock mode 2.
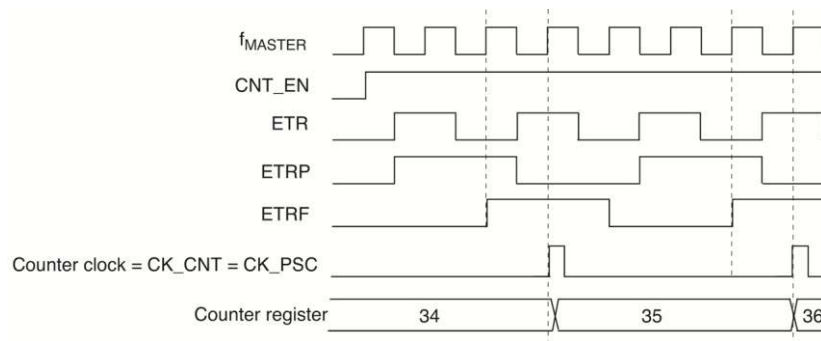
Figure 10.16. Control Circuit in External Clock Mode 2

10.3.4. Capture/Compare Channels

Each capture/compare channel is built around a capture/compare register including a shadow register, an input stage for capture with digital filter, multiplexing and prescaler, and an output stage with comparator and output control.

Figure 10.17 gives an overview of the input stage of one capture/compare channel. The input stage samples the corresponding TIy input to generate a filtered signal (TIyF). Then an edge detector with polarity selection generates a signal (TIyFPy) which can be used either as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICyPS).
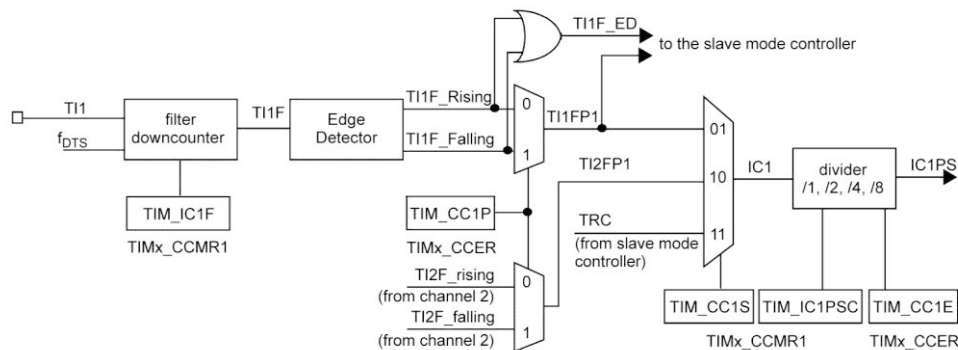
Figure 10.17. Capture/Compare Channel (Example: Channel 1 Input Stage)

The output stage generates an intermediate reference signal, OCyREF, which is only used internally. OCyREF is always active high, but it may be inverted to create the output signal, OCy, that controls a GPIO output. Figure 10.18 shows the basic elements of a capture/compare channel.
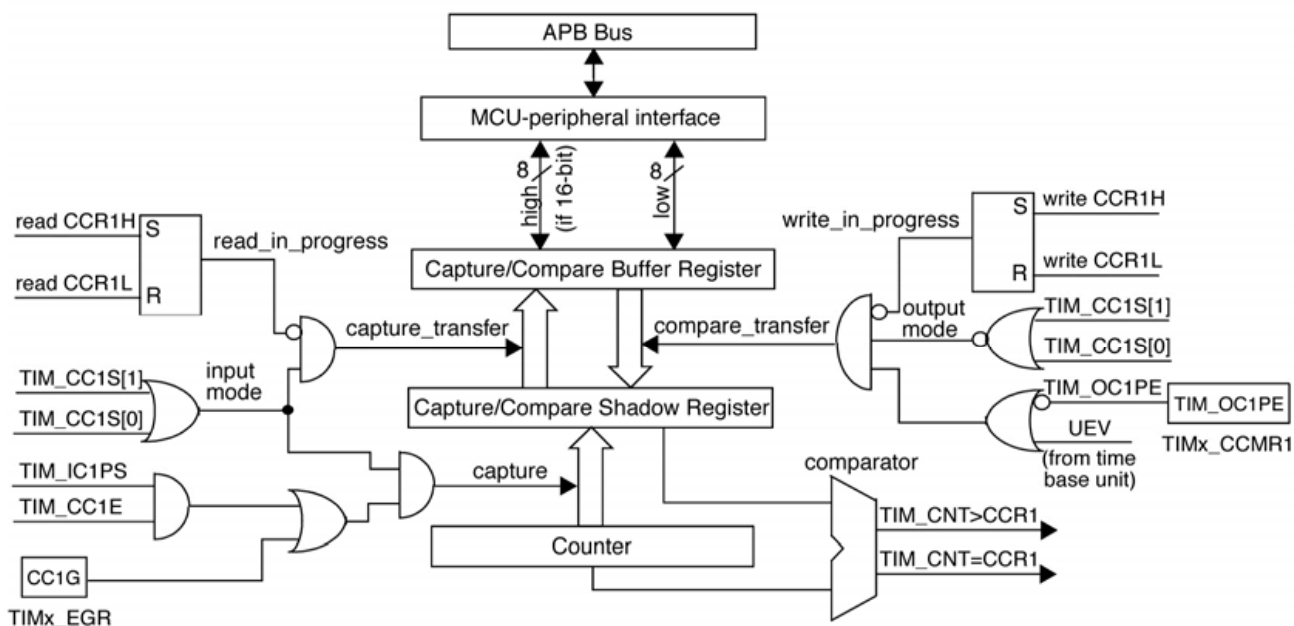
Figure 10.18. Capture/Compare Channel 1 Main Circuit

Figure 10.19 show details of the output stage of a capture/compare channel.
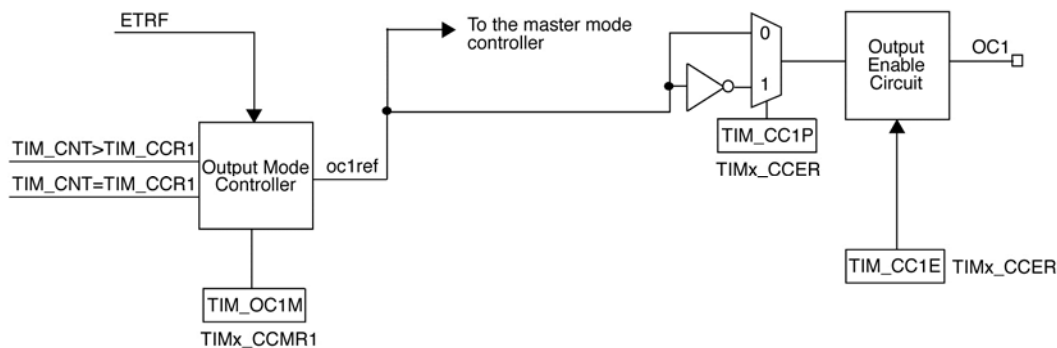
Figure 10.19. Output Stage of Capture/Compare Channel (Channel 1)

The capture/compare block is made of a buffer register and a shadow register. Writes and reads always access the buffer register.

In capture mode, captures are first written to the shadow register, then copied into the buffer register.

In compare mode, the content of the buffer register is copied into the shadow register which is compared to the counter

10.3.5. Input Capture Mode

In input capture mode, a capture/compare register (TIMx_CCRy) latches the value of the counter after a transition is detected by the corresponding ICy signal. When a capture occurs, the corresponding INT_TIMCCyIF flag in the INT_TIMxFLAG register is set, and an interrupt request is sent if enabled.

If a capture occurs when the INT_TIMCCyIF flag is already high, then the missed capture flag INT_TIMMISSCCyIF in the INT_TIMxMISS register is set. INT_TIMCCyIF can be cleared by software writing a 1 to its bit or reading the captured data stored in the TIMx_CCRy register. To clear the INT_TIMMISSCCyIF bit, write a 1 to it.

The following example shows how to capture the counter value in the TIMx_CCR1 when the TI1 input rises.

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the TIM_CC1S bits to 01 in the TIMx_CCMR1 register. As soon as TIM_CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the required input filter duration with respect to the signal connected to the timer, when the input is one of the TIy (ICyF bits in the TIMx_CCMR1 register). Consider a situation in which, when toggling, the input signal is unstable during at most 5 internal clock cycles. The filter duration must be longer than these 5 clock cycles. The transition on TI1 can be validated when 8 consecutive samples with the new level have been detected (sampled at PCLK frequency). To do this, write the TIM_IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel: Write the TIM_CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler: In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write the TIM_IC1PSC bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register: Set the TIM_CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the INT_TIMCC1IF bit in the INT_TIMxCFG register.
- When an input capture occurs:
    - The TIMx_CCR1 register gets the value of the counter on the active transition.
    - INT_TIMCC1IF flag is set (capture/compare interrupt flag). The missed capture/compare flag INT_TIMMISSCC1IF in INT_TIMxMISS is also set if another capture occurs before the INT_TIMCC1IF flag is cleared.
    - An interrupt may be generated if enabled by the INT_TIMCC1IF bit.

To detect missed captures reliably, read captured data in TIMxCCRy before checking the missed capture/compare flag. This sequence avoids missing a capture that could happen after reading the flag and before reading the data.

Note: Software can generate IC interrupt requests by setting the corresponding TIM_CCyG bit in the TIMx_EGR register.

10.3.6. PWM Input Mode

This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICy signals are mapped on the same TIy input.
- These two ICy signals are active on edges with opposite polarity.
- One of the two TIyFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, to measure the period in the TIMx_CCR1 register and the duty cycle in the TIMx_CCR2 register of the PWM applied on TI1, use the following procedure depending on CK_INT frequency and prescaler value:

- Select the active input for TIMx_CCR1: write the TIM_CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1, used both for capture in the TIMx_CCR1 and counter clear, by writing the TIM_CC1P bit to 0 (active on rising edge).
- Select the active input for TIMx_CCR2by writing the TIM_CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in the TIMx_CCR2) by writing the TIM_CC2P bit to 1 (active on falling edge).
- Select the valid trigger input by writing the TIM_TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode by writing the TIM_SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures by writing the TIM_CC1E and TIM_CC2E bits to 1 in the TIMx_CCER register.

Figure 10.20 illustrates this example.
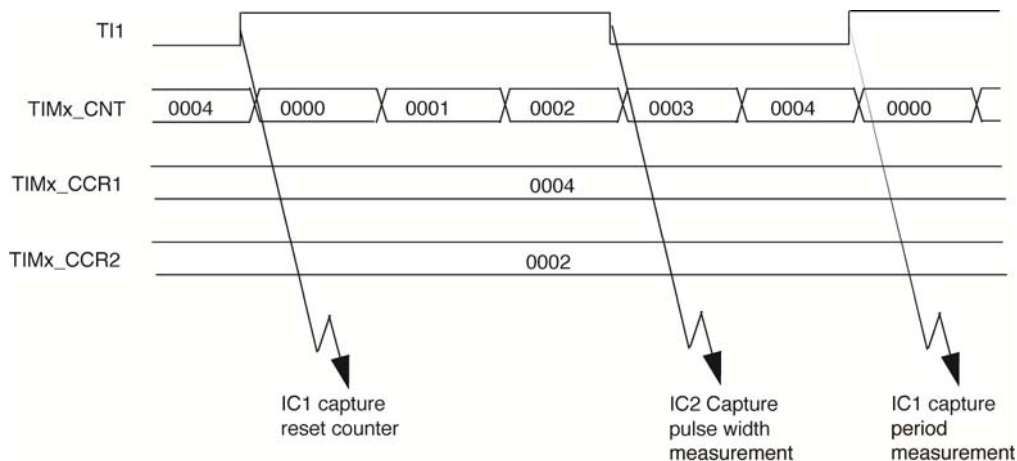
Figure 10.20. PWM Input Mode Timing

10.3.7. Forced Output Mode

In output mode (CCyS bits = 00 in the TIMx_CCMR1 register), software can force each output compare signal (OCyREF and then OCy) to an active or inactive level independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCyREF/OCy) to its active level, write 101 in the TIM_OCyM bits in the corresponding TIMx_CCMR1 register. OCyREF is forced high (OCyREF is always active high) and OCy gets the opposite value to the TIM_CCyP polarity bit. For example, TIM_CCyP = 0 defines OCy as active high, so when OCyREF is active, OCy is also set to a high level.

The OCyREF signal can be forced low by writing the TIM_OCyM bits to 100 in the TIMx_CCMR1 register.

The comparison between the TIMx_CCRy shadow register and the counter is still performed and allows the INT_TIMxCCRyIF flag to be set. Interrupt requests can be sent accordingly. This is described in "10.3.8. Output Compare Mode" .

10.3.8. Output Compare Mode

This mode is used to control an output waveform or to indicate when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

■ Assigns the corresponding output pin to a programmable value defined by the output compare mode (the TIM_OCyM bits in the TIMx_CCMR1 register) and the output polarity (the TIM_CCyP bit in the TIMx_CCER register). The output can be frozen (TIM_OCyM = 000), be set active (TIM_OCyM = 001), be set inactive (TIM_OCyM = 010), or can toggle (TIM_OCyM = 011) on the match.

■ Sets a flag in the interrupt flag register (the INT_TIMCCyIF bit in the INT_TIMxFLAG register).

■ Generates an interrupt if the corresponding interrupt mask is set (the TIM_CCyIF bit in the INT_TIMxCFG register).

The TIMx_CCRy registers can be programmed with or without buffer registers using the TIM_OCyBE bit in the TIMx_CCMR1 register.

In output compare mode, the UEV has no effect on OCyREF or the OCy output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in one pulse mode).

Procedure:

1. Select the counter clock (internal, external, and prescaler).

2. Write the desired data in the TIMx_ARR and TIMx_CCRy registers.

3. Set the INT_TIMCCyIF bit in INT_TIMxCFG if an interrupt request is to be generated.

4. Select the output mode. For example, you must write TIM_OCyM = 011, TIM_OCyBE = 0, TIM_CCyP = 0

---

and TIM_CCyE = 1 to toggle the OCy output pin when TIMx_CNT matches TIMx_CCRy, TIMx_CCRy buffer is not used, OCy is enabled and active high.

5. Enable the counter: Set the TIM_CEN bit in the TIMx_CR1 register.

To control the output waveform, software can update the TIMx_CCRy register at any time, provided that the buffer register is not enabled (TIM_OCyBE = 0). Otherwise TIMx_CCRy shadow register is updated only at the next UEV. An example is given in Figure 10.21.
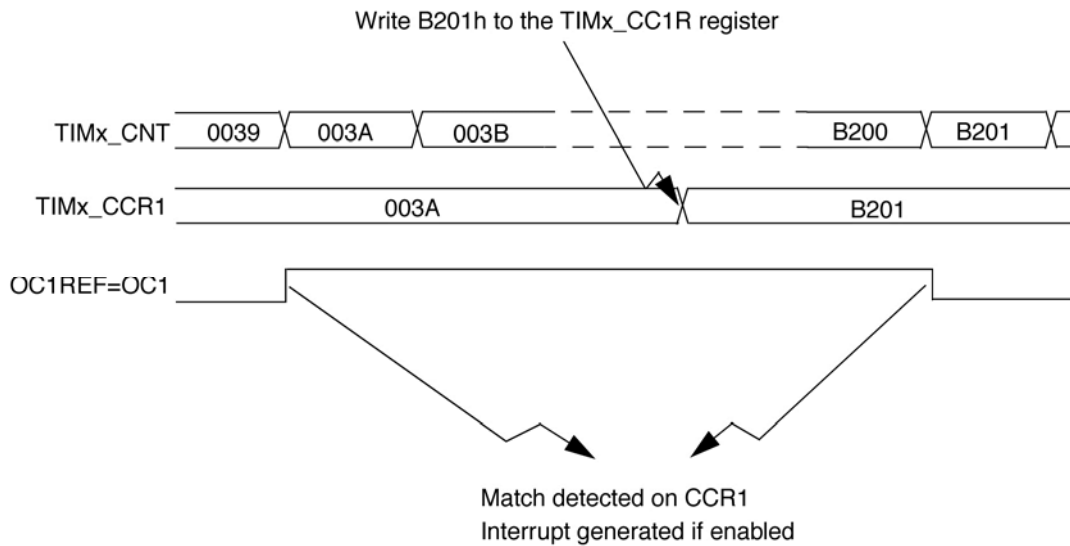
Figure 10.21. Output Compare Mode, Toggle on OC1

10.3.9. PWM Mode

Pulse width modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register, and a duty cycle determined by the value of the TIMx_CCRy register.

PWM mode can be selected independently on each channel (one PWM per OCy output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the TIM_OCyM bits in the TIMx_CCMR1 register. The corresponding buffer register must be enabled by setting the TIM_OCyBE bit in the TIMx_CCMR1 register. Finally, in up-counting or center-aligned mode the auto-reload buffer register must be enabled by setting the TIM_ARBE bit in the TIMx_CR1 register.

Because the buffer registers are only transferred to the shadow registers when a UEV occurs, before starting the counter initialize all the registers by setting the TIM_UG bit in the TIMx_EGR register.

OCy polarity is software programmable using the TIM_CCyP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCy output is enabled by the TIM_CCyE bit in the TIMx_CCER register. Refer to the TIMx_CCER register description in the Registers section for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRy are always compared to determine whether TIMx_CCRy ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRy, depending on the direction of the counter. The OCyREF signal is asserted only:

- When the result of the comparison changes, or
- When the output compare mode (TIM_OCyM bits in the TIMx_CCMR1 register) switches from the "frozen" configuration (no comparison, TIM_OCyM = 000) to one of the PWM modes (TIM_OCyM = 110 or 111).

This allows software to force a PWM output to a particular state while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the TIM_CMS bits in the TIMx_CR1 register.

10.3.9.1. PWM Edge-Aligned Mode: Up-Counting Configuration

Up-counting is active when the TIM_DIR bit in the TIMx_CR1 register is low. Refer to "10.3.2.1. Up-Counting Mode" on page 170.

The following example uses PWM mode 1. The reference PWM signal OCyREF is high as long as TIMx_CNT < TIMx_CCRy, otherwise it becomes low. If the compare value in TIMx_CCRy is greater than the auto-reload value in TIMx_ARR, then OCyREF is held at 1. If the compare value is 0, then OCyREF is held at 0. Figure 10.22 shows some edge-aligned PWM waveforms in an example, where TIMx_ARR = 8.
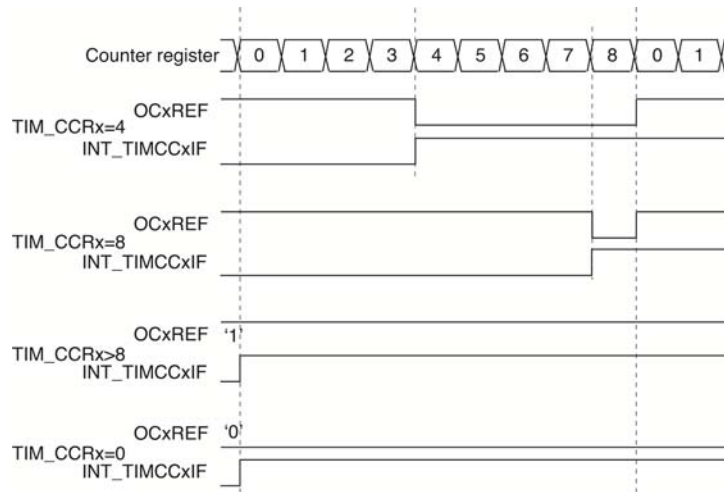
Figure 10.22. Edge-Aligned PWM Waveforms (ARR = 8)

10.3.9.2. PWM Edge-Aligned Mode: Down-Counting Configuration

Down-counting is active when the TIM_DIR bit in the TIMx_CR1 register is high. Refer to "10.3.2.2. Down-Counting Mode" on page 172 for more information.

In PWM mode 1, the reference signal OCyREF is low as long as TIMx_CNT > TIMx_CCRy, otherwise it becomes high. If the compare value in TIMx_CCRy is greater than the auto-reload value in TIMx_ARR, then OCyREF is held at 1. Zero-percent PWM is not possible in this mode.

10.3.9.3. PWM Center-Aligned Mode

Center-aligned mode is active except when the TIM_CMS bits in the TIMx_CR1 register are 00 (all configurations where TIM_CMS is non-zero have the same effect on the OCyREF/OCy signals). The compare flag is set when the counter counts up, when it counts down, or when it counts up and down, depending on the TIM_CMS bits configuration. The direction bit (TIM_DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the "10.3.2.3. Center-Aligned Mode (Up/Down Counting)" on page 173 for more information.

Figure 10.23 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8
- PWM mode is the PWM mode 1
- The output compare flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for TIM_CMS = 01 in the TIMx_CR1 register
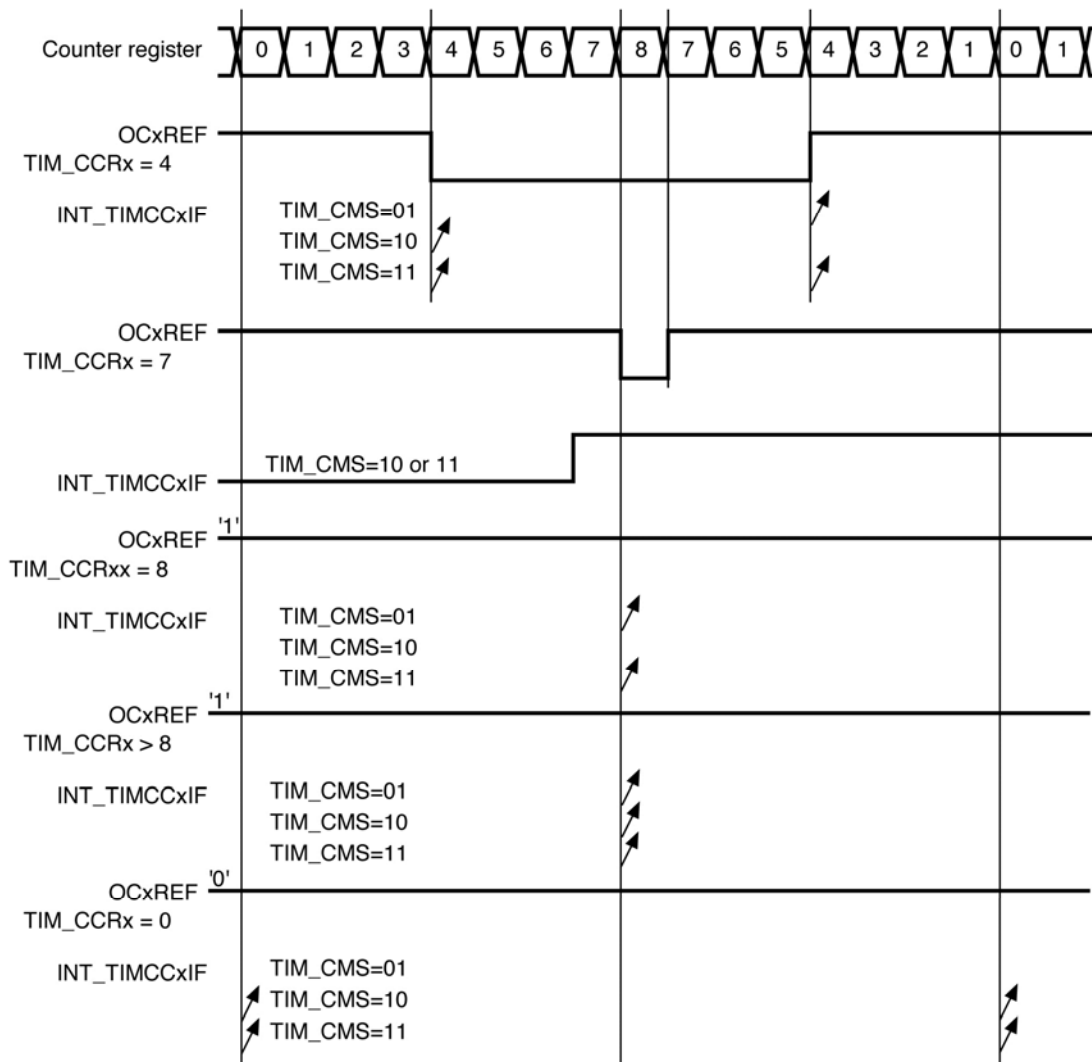
Figure 10.23. Center-Aligned PWM Waveforms (ARR = 8)

Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. This means that the counter counts up or down depending on the value written in the TIM_DIR bit in the TIMx_CR1 register. The TIM_DIR and TIM_CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
    - The direction is not updated when the value written to the counter that is greater than the auto-reload value (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
    - The direction is updated when 0 or the TIMx_ARR value is written to the counter, but no UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the TIM_UG bit in the TIMx_EGR register) just before starting the counter, and not to write the counter while it is running.

10.3.10. One-Pulse Mode

One-pulse mode (OPM) is a special case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. Select OPM by setting the TIM_OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

In up-counting: TIMx_CNT < TIMx_CCRy ≤ TIMx_ARR (in particular, 0 < TIMx_CCRy),

In down-counting: TIMx_CNT > TIMx_CCRy.

For example, to generate a positive pulse on OC1 with a length of tPULSE and after a delay of tDELAY as soon as a rising edge is detected on the TI2 input pin, using TI2FP2 as trigger 1:

- Map TI2FP2 on TI2: Write TIM_IC2S = 01 in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge. Write TIM_CC2P = 0 in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI): Write TIM_TS = 110 in the TIMx_SMCR register.
- Use TI2FP2 to start the counter: Write TIM_SMS to 110 in the TIMx_SMCR register (trigger mode).
- The OPM waveform is defined: Write the compare registers, taking into account the clock frequency and the counter prescaler.
  - The tDELAY is defined by the value written in the TIMx_CCR1 register.
  - The tPULSE is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- To build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value:
  - Enable PWM mode 2: Write TIM_OC1M = 111 in the TIMx_CCMR1 register.
  - Optionally, enable the buffer registers: Write TIM_OC1BE = 1 in the TIMx_CCMR1 register and TIM_ARBE in the TIMx_CR1 register. In this case, also write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the TIM_UG bit, and wait for external trigger event on TI2. TIM_CC1P is written to 0 in this example.

In the example, the TIM_DIR and TIM_CMS bits in the TIMx_CR1 register should be low.

Since only one pulse is desired, software should set the TIM_OPM bit in the TIMx_CR1 register to stop the counter at the next UEV (when the counter rolls over from the auto-reload value back to 0).
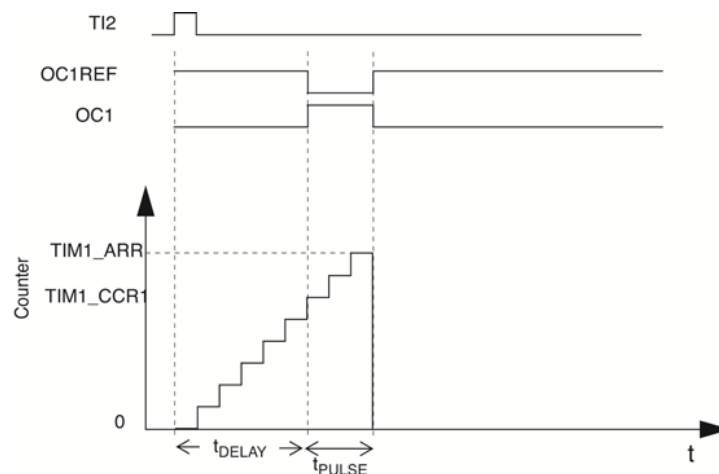
Figure 10.24 illustrates this example.

Figure 10.24. Example of One Pulse Mode

10.3.10.1. A Special Case: OCy Fast Enable

In one-pulse mode, the edge detection on the TIy input sets the TIM_CEN bit, which enables the counter. Then the comparison between the counter and the compare value toggles the output. However, several clock cycles are needed for this operation, and it limits the minimum delay (tDELAY min) achievable.

To output a waveform with the minimum delay, set the TIM_OCyFE bit in the TIMx_CCMR1 register. Then OCyREF and OCy are forced in response to the stimulus, without taking the comparison into account. Its new level is the same as if a compare match had occurred. TIM_OCyFE acts only if the channel is configured in PWM mode 1 or 2.

10.3.11. Encoder Interface Mode

To select encoder interface mode, write TIM_SMS = 001 in the TIMx_SMCR register to count only TI2 edges, TIM_SMS = 010 to count only TI1 edges, and TIM_SMS = 011 to count both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the TIM_CC1P and TIM_CC2P bits in the TIMx_CCER register. If needed, program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder (see Table 10 3). Assuming that it is enabled (the TIM_CEN bit in the TIMx_CR1 register = 1), the counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1 = TI1 if not filtered and not inverted, TI2FP2 = TI2 if not filtered and not inverted.) The timer input logic evaluates the sequence of the two inputs' values, and from this generates both count pulses and the direction signal. Depending on the sequence, the counter counts up or down, and hardware modifies the TIM_DIR bit in the TIMx_CR1 register accordingly. The TIM_DIR bit is calculated at each transition on any input (TI1 or TI2), whether the counter is counting on TI1 only, TI2 only, or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to TIMx_ARR or TIMx_ARR down to 0 depending on the direction), so TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, and trigger output features continue to work as normal.

In this mode the counter is modified automatically following the speed and the direction of the incremental encoder, and therefore its contents always represent the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. Table 10.3 summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 10.3. Counting Direction versus Encoder Signals

Active Edges	Level on Opposite Signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 Signal		TI2FP2 Signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally used to convert an encoder's differential outputs to digital signals, and this greatly increases noise immunity. If a third encoder output indicates the mechanical zero (or index) position, it may be connected to an external interrupt input and can trigger a counter reset.

SILICON LABS

Figure 10.25 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated for when both inputs are used for counting. This might occur if the sensor is positioned near one of the switching points. This example assumes the following configuration:

- TIM_CC1S = 01 (TIMx_CCMR1 register, IC1FP1 mapped on TI1).
- TIM_CC2S = 01 (TIMx_CCMR2 register, IC2FP2 mapped on TI2).
- TIM_CC1P = 0 (TIMx_CCER register, IC1FP1 non-inverted, IC1FP1 = TI1).
- TIM_CC2P = 0 (TIMx_CCER register, IC2FP2 non-inverted, IC2FP2 = TI2).
- TIM_SMS = 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- TIM_CEN = 1 (TIMx_CR1 register, counter is enabled).
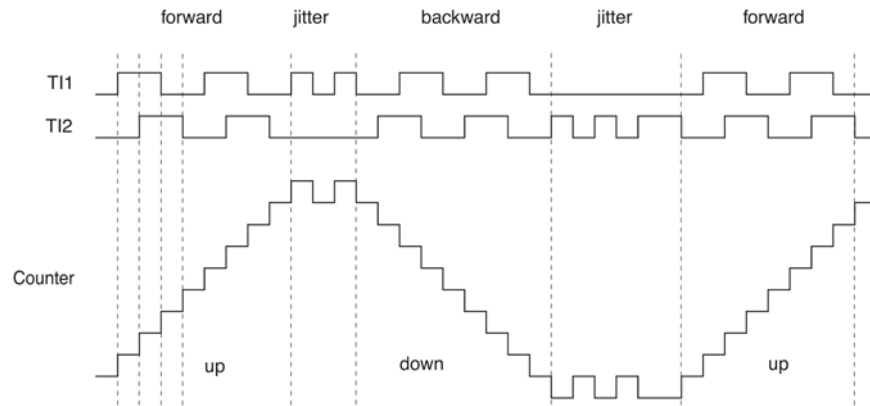
Figure 10.25. Example of Counter Operation in Encoder Interface Mode

Figure 10.26 gives an example of counter behavior when IC1FP1 polarity is inverted (same configuration as above except TIM_CC1P = 1).
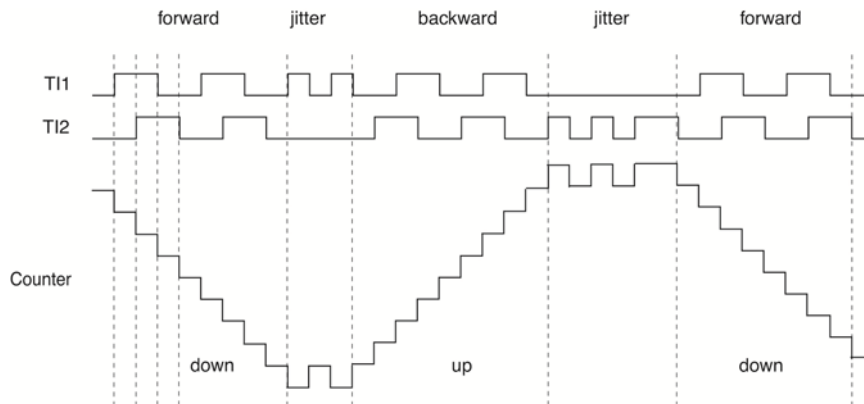
Figure 10.26. Example of Encoder Interface Mode with IC1FP1 Polarity Inverted

The timer configured in encoder interface mode provides information on a sensor's current position. To obtain dynamic information (speed, acceleration/deceleration), measure the period between two encoder events using a second timer configured in capture mode. The output of the encoder that indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. Do this by latching the counter value into a third input capture register. (In this case the capture signal must be periodic and can be generated by another timer).

10.3.12. Timer Input XOR Function

The TIM_TI1S bit in the TIM1_CR2 register allows the input filter of channel 1 to be connected to the output of a XOR gate that combines the three input pins TIMxC2 to TIMxC4.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is especially useful to interface to Hall effect sensors.

10.3.13. Timers and External Trigger Synchronization

The timers can be synchronized with an external trigger in several modes: reset mode, gated mode, and trigger mode.

10.3.13.1. Slave Mode: Reset Mode

Reset mode reinitializes the counter and its prescaler in response to an event on a trigger input. Moreover, if the TIM_URS bit in the TIMx_CR1 register is low, a UEV is generated. Then all the buffered registers (TIMx_ARR, TIMx_CCRy) are updated.

In the following example, the up-counter is cleared in response to a rising edge on the TI1 input:

- Configure the channel 1 to detect rising edges on TI1:
  - Configure the input filter duration. In this example, no filter is required so TIM_IC1F = 0000.
  - The capture prescaler is not used for triggering, so it is not configured.
  - The TIM_CC1S bits select the input capture source only, TIM_CC1S = 01 in the TIMx_CCMR1 register.
  - Write TIM_CC1P = 0 in the TIMx_CCER register to validate the polarity, and detect rising edges only.
- Configure the timer in reset mode: Write TIM_SMS = 100 in the TIMx_SMCR register.
- Select TI1 as the input source by writing TIM_TS = 101 in the TIMx_SMCR register.
- Start the counter: Write TIM_CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until the TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (the INT_TIMTIF bit in the INT_TIMxFLAG register) and an interrupt request can be sent if enabled (depending on the INT_TIMTIF bit in the INT_TIMxCFG register).

Figure 10.27 shows this behavior when the auto-reload register TIMx_ARR = 0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on the TI1 input.
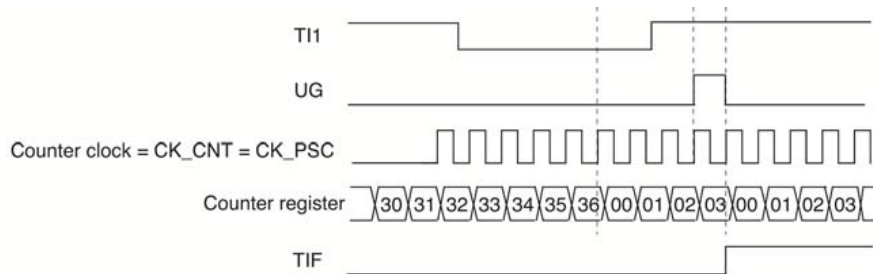

Figure 10.27. Control Circuit in Reset Mode

10.3.13.2. Slave Mode: Gated Mode

In gated mode the counter is enabled depending on the level of a selected input.

In the following example, the up-counter counts only when the TI1 input is low:

- Configure channel 1 to detect low levels on TI1:
  - Configure the input filter duration. In this example, no filter is required, so TIM_IC1F = 0000.
  - The capture prescaler is not used for triggering, so it is not configured.
  - The TIM_CC1S bits select the input capture source only, TIM_CC1S = 01 in the TIMx_CCMR1 register.
  - Write TIM_CC1P = 1 in the TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode: Write TIM_SMS = 101 in the TIMx_SMCR register.
- Select TI1 as the input source by writing TIM_TS = 101 in the TIMx_SMCR register.
- Enable the counter: Write TIM_CEN = 1 in the TIMx_CR1 register. In gated mode, the counter does not start if TIM_CEN = 0, regardless of the trigger input level.

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The INT_TIMTIF flag in the INT_TIMxFLAG register is set when the counter starts and when it stops. The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on the TI1 input.

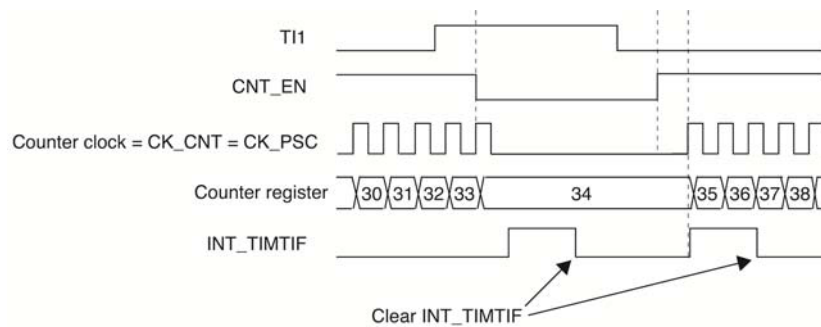Figure 10.28 shows the counter in gated mode with counting enabled when TI1 is low.

Figure 10.28. Control Circuit in Gated Mode

10.3.13.3. Slave Mode: Trigger Mode

In trigger mode the counter starts in response to an event on a selected input.

In the following example, the up-counter starts in response to a rising edge on the TI2 input:

- Configure channel 2 to detect rising edges on TI2:
  - Configure the input filter duration. In this example, no filter is required so TIM_IC2F = 0000.
  - The capture prescaler is not used for triggering, so it is not configured.
  - The TIM_CC2S bits select the input capture source only, TIM_CC2S = 01 in the TIMx_CCMR1 register.
  - Write TIM_CC2P = 0 in the TIMx_CCER register to validate the polarity and detect high level only.
- Configure the timer in trigger mode: Write TIM_SMS = 110 in the TIMx_SMCR register.
- Select TI2 as the input source by writing TIM_TS = 110 in the TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the INT_TIMTIF flag is set. The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on the TI2 input.

Figure 10.29 illustrates the example in which the counter is started by a rising edge on TI2.
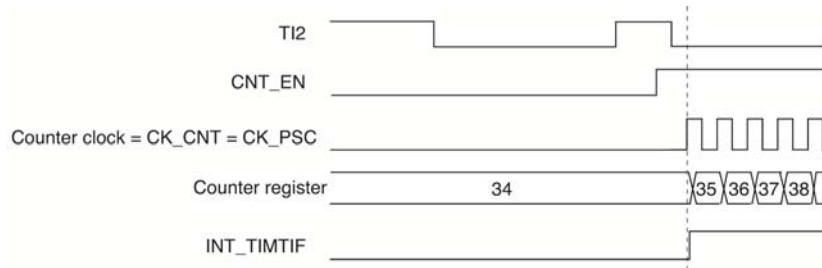
Figure 10.29. Control Circuit in Trigger Mode

10.3.13.4. Slave Mode: External Clock Mode 2 +Trigger Mode

External clock mode 2 can be used in combination with another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is not recommended to select ETR as TRGI through the TIM_TS bits of TIMx_SMCR register.

In the following example, shown in Figure 10.30, the up-counter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

- Configure the external trigger input circuit: Program the TIMx_SMCR register as follows:
  - TIM_ETF = 0000: no filter.
  - TIM_ETPS = 00: prescaler disabled.
  - TIM_ETP = 0: detection of rising edges on ETR and TIM_ECE = 1 to enable the external clock mode 2.
- Configure the channel 1 to detect rising edges on TI, as follows:
  - TIM_IC1F = 0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - TIM_CC1S = 01 in the TIMx_CCMR1 register to select only the input capture source.
  - TIM_CC1P = 0 in the TIMx_CCER register to validate the polarity (and detect rising edge only).
- Configure the timer in trigger mode: WriteTIM_SMS = 110 in the TIMx_SMCR register.
- Select TI1 as the input source by writing TIM_TS = 101 in the TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the INT_TIMTIF flag. The counter then counts on ETR rising edges. The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.
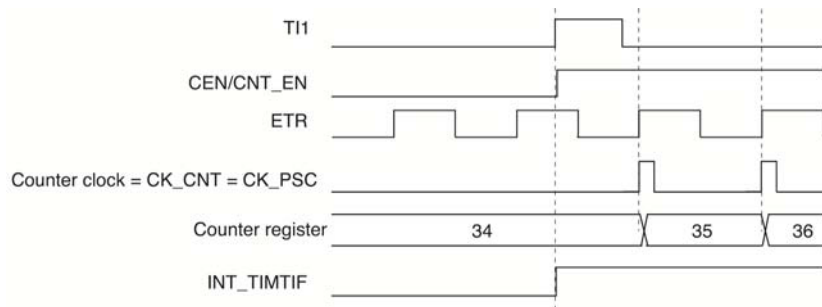

Figure 10.30. Control Circuit in External Clock Mode 2 + Trigger Mode

10.3.14. Timer Synchronization

The two timers can be linked together internally for timer synchronization or chaining. A timer configured in master mode can reset, start, stop or clock the counter of the other timer configured in slave mode.

Figure 10.31 presents an overview of the trigger selection and the master mode selection blocks.

10.3.14.1. Using One Timer as Prescaler for the Other Timer

For example, to configure Timer 1 to act as a prescaler for Timer 2:

- Configure Timer 1 in master mode so that it outputs a periodic trigger signal on each UEV. Writing TIM_MMS = 010 in the TIM1_CR2 register causes a rising edge to be output on TRGO each time a UEV is generated.
- To connect the TRGO output of Timer 1 to Timer 2, configure Timer 2 in slave mode using ITR0 as an internal trigger. Write TIM_TS = 100 in the TIM2_SMCR register.
- Put the slave mode controller in external clock mode 1: Write TIM_SMS = 111 in the TIM2_SMCR register. This causes Timer 2 to be clocked by the rising edge of the periodic Timer 1 trigger signal, which corresponds to the Timer 1 counter overflow.
- Finally, enable both timers: Set their respective TIM_CEN bits in the TIMx_CR1 register.

Note: If OCy is selected on Timer 1 as trigger output (TIM_MMS = 1xx), its rising edge is used to clock the counter of Timer 2.
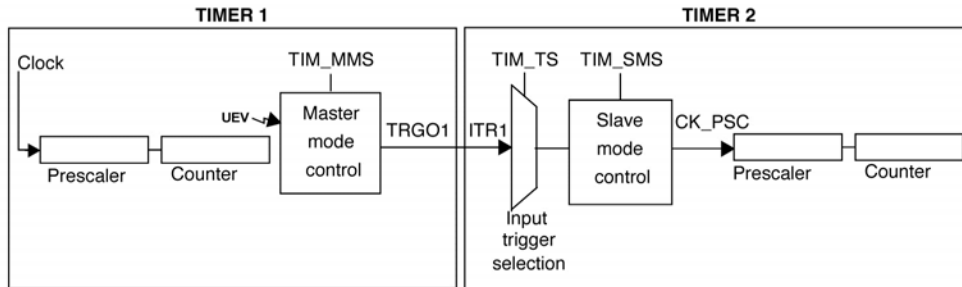

Figure 10.31. Master/Slave Timer Example

10.3.14.2. Using One Timer to Enable the Other Timer

In this example, shown in Figure 10.32, the enable of Timer 2 is controlled with the output compare 1 of Timer 1. Timer 2 counts on the divided internal clock only when OC1REF of Timer 1 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT (fCK_CNT = fCK_INT /3).

- Configure Timer 1 in master mode to send its Output Compare Reference (OC1REF) signal as trigger output: Write TIM_MMS = 100 in the TIM1_CR2 register.
- Configure the Timer 1 OC1REF waveform (TIM1_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1: Write TIM_TS = 000 in the TIM2_SMCR register.
- Configure Timer 2 in gated mode: Write TIM_SMS = 101 in the TIM2_SMCR register.
- Enable Timer 2: Write 1 in the TIM_CEN bit in the TIM2_CR1 register.
- Start Timer 1: Write 1 in the TIM_CEN bit in the TIM1_CR1 register.

Note: The counter 2 clock is not synchronized with counter 1. This mode only affects the Timer 2 counter enable signal.
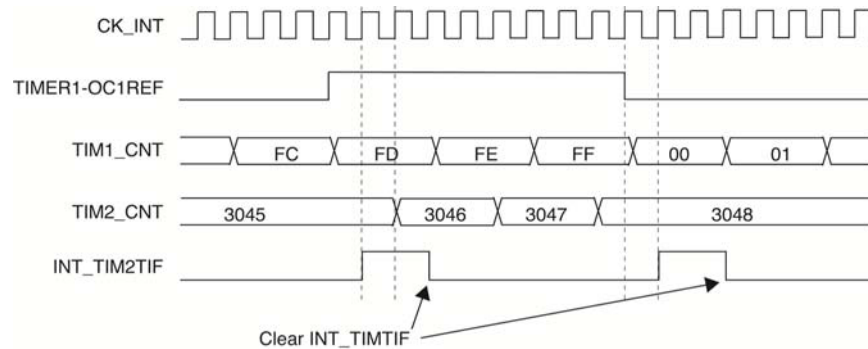

Figure 10.32. Gating Timer 2 with OC1REF of Timer 1

In the example in Figure 10.32, the Timer 2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting Timer 1, then writing the desired value in the timer counters. The timers can easily be reset by software using the TIM_UG bit in the TIMx_EGR registers.

The next example, shown in Figure 10.33, synchronizes Timer 1 and Timer 2. Timer 1 is the master and starts from 0. Timer 2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. Timer 2 stops when Timer 1 is disabled by writing 0 to the TIM_CEN bit in the TIM1_CR1 register:

- Configure Timer 1 in master mode to send its Output Compare Reference (OC1REF) signal as trigger output: Write TIM_MMS = 100 in the TIM1_CR2 register)
- Configure the Timer 1 OC1REF waveform (TIM1_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1: Write TIM_TS = 000 in the TIM2_SMCR register.
- Configure Timer 2 in gated mode: Write TIM_SMS = 101 in the TIM2_SMCR register.
- Reset Timer 1: Write 1 in the TIM_UG bit (TIM1_EGR register.
- Reset Timer 2 by writing 1 in the TIM_UG bit (TIM2_EGR register).
- Initialize Timer 2 to 0xE7: Write 0xE7 in the Timer 2 counter (TIM2_CNTL).
- Enable Timer 2: Write 1 in the TIM_CEN bit in the TIM2_CR1 register.
- Start Timer 1: Write 1 in the TIM_CEN bit in the TIM1_CR1 register.
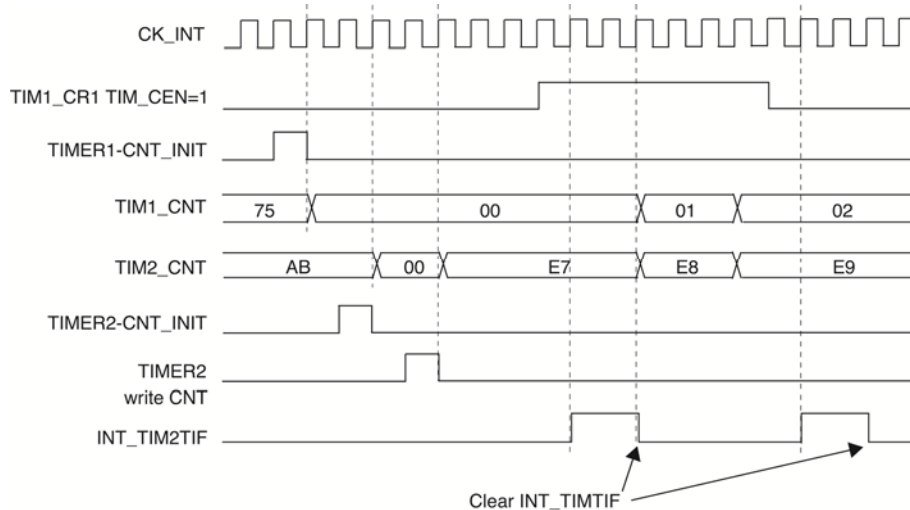- Stop Timer 1: Write 0 in the TIM_CEN bit in the TIM1_CR1 register.

Figure 10.33. Gating Timer 2 with Enable of Timer 1

10.3.14.3. Using One Timer to Start the Other Timer

In this example (see Figure 10.34), the enable of Timer 2 is set with the UEV of Timer 1. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as Timer 1 generates the UEV.

When Timer 2 receives the trigger signal its TIM_CEN bit is automatically set and the counter counts until 0 is written to the TIM_CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}$ /3).

- Configure Timer 1 in master mode to send its UEV as trigger output: WriteTIM_MMS = 010 in the TIM1_CR2 register.
- Configure the Timer 1 period (TIM1_ARR register).
- Configure Timer 2 to get the input trigger from Timer 1: Write TIM_TS = 000 in the TIM2_SMCR register.
- Configure Timer 2 in trigger mode. Write TIM_SMS = 110 in the TIM2_SMCR register.
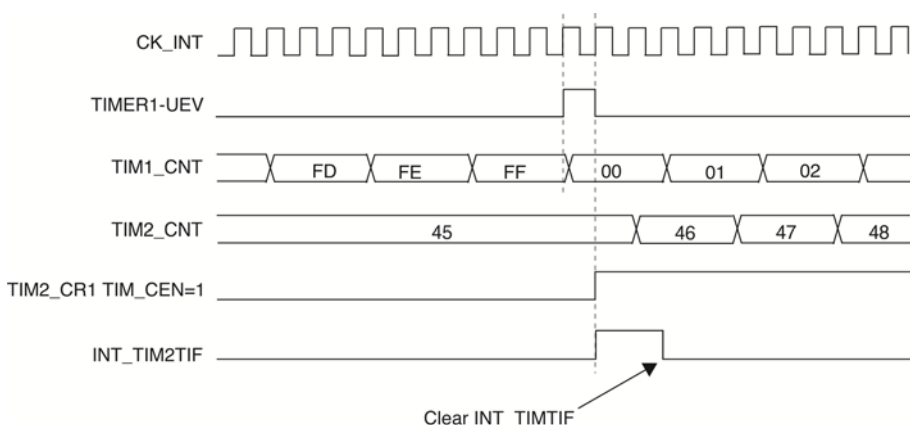- Start Timer 1: Write 1 in the TIM_CEN bit in theTIM1_CR1 register.

Figure 10.34. Triggering Timer 2 with Update of Timer 1

As in the previous example, both counters can be initialized before starting counting. Figure 10.35 shows the behavior with the same configuration shown in Figure 10.34, but in trigger mode instead of gated mode (TIM_SMS = 110 in the TIM2_SMCR register).
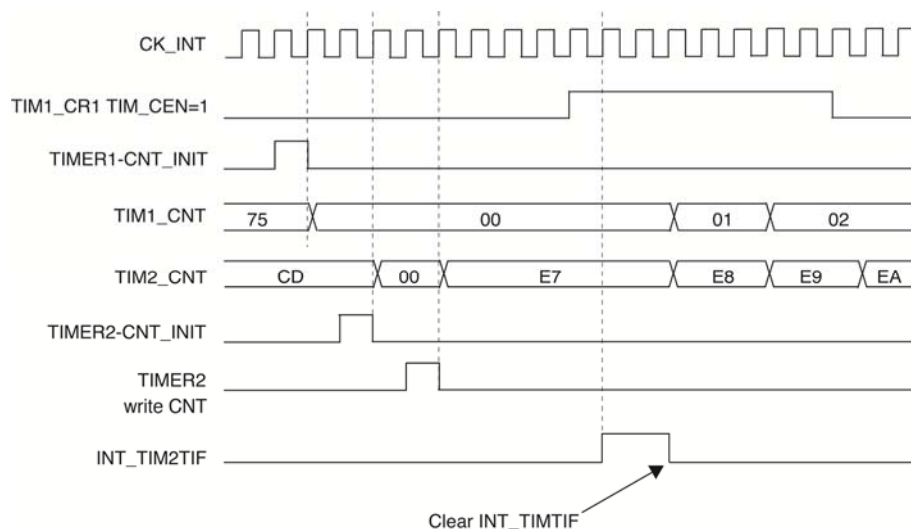
Figure 10.35. Triggering Timer 2 with Enable of Timer 1

10.3.14.4. Starting both Timers Synchronously in Response to an External Trigger

This example sets the enable of Timer 1 when its TI1 input rises, and the enable of Timer 2 with the enable of Timer 1. To ensure the counters are aligned, Timer 1 must be configured in master/slave mode (slave with respect to TI1, master with respect to Timer 2):

- Configure Timer 1 in master mode to send its Enable as trigger output: Write TIM_MMS = 001 in the TIM1_CR2 register.
- Configure Timer 1 slave mode to get the input trigger from TI1: Write TIM_TS = 100 in the TIM1_SMCR register.
- Configure Timer 1 in trigger mode: Write TIM_SMS = 110 in the TIM1_SMCR register.
- Configure the Timer 1 in master/slave mode: Write TIM_MSM = 1 in the TIM1_SMCR register.
- Configure Timer 2 to get the input trigger from Timer 1: Write TIM_TS = 000 in the TIM2_SMCR register.
- Configure Timer 2 in trigger mode: Write TIM_SMS = 110 in the TIM2_SMCR register.

When a rising edge occurs on TI1 (Timer 1), both counters start counting synchronously on the internal clock and both timers' INT_TIMTIF flags are set. Figure 10.36 shows this in operation.

Note: In this example both timers are initialized before starting by setting their respective TIM_UG bits. Both counters starts from 0, but an offset can be inserted between them by writing any of the counter registers (TIMx_CNT). The master/slave mode inserts a delay between CNT_EN and CK_PSC on Timer 1.
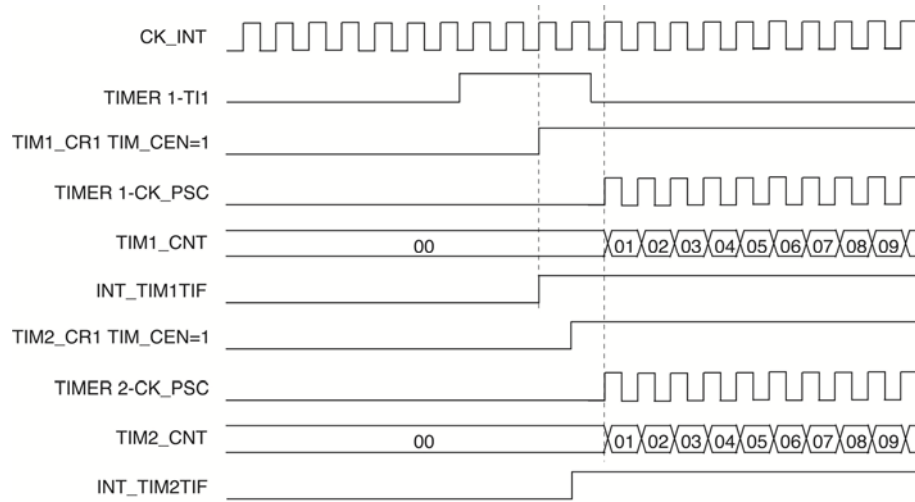
Figure 10.36. Triggering Timer 1 and 2 with Timer 1 TI1 Input

10.3.15. Timer Signal Descriptions

Table 10.4. Timer Signal Descriptions

Signal	Internal/ External	Description
CK_INT	Internal	Internal clock source: connects to EM358x peripheral clock (PCLK) in internal clock mode.
CK_PSC	Internal	Input to the clock prescaler.
ETR	Internal	External trigger input (used in external timer mode 2): a clock selected by TIM_EXTRIG-SEL in TIMx_OR.
ETRF	Internal	External trigger: ETRP after filtering.
ETRP	Internal	External trigger: ETR after polarity selection, edge detection and prescaling.
ICy	External	Input capture or clock: TIy after filtering and edge detection.
ICyPS	Internal	Input capture signal after filtering, edge detection and prescaling: input to the capture register.
ITR0	Internal	Internal trigger input: connected to the other timer's output, TRGO.
OCy	External	Output compare: TIMxCy when used as an output. Same as OCyREF but includes possible polarity inversion.
OCyREF	Internal	Output compare reference: always active high, but may be inverted to produce OCy.
PCLK	External	Peripheral clock connects to CK_INT and used to clock input filtering. Its frequency is 12 MHz if using the 24 MHz crystal oscillator and 6 MHz if using the 12 MHz RC oscillator.
TIy	Internal	Timer input: TIMxCy when used as a timer input.
TIyFPy	Internal	Timer input after filtering and polarity selection.
TIMxCy	Internal	Timer channel at a GPIO pin: can be a capture input (ICy) or a compare output (OCy).
TIMxCLK	External	Clock input (if selected) to the external trigger signal (ETR).

Table 10.4. Timer Signal Descriptions (Continued)

Signal	Internal/ External	Description
TIMxMSK	External	Clock mask (if enabled) AND'ed with the other timer's TIMxCLK signal.
TRGI	Internal	Trigger input for slave mode controller.

10.4. Interrupts

Each timer has its own top-level NVIC interrupt. Writing 1 to the INT_TIMx bit in the INT_CFGSET register enables the TIMx interrupt, and writing 1 to the INT_TIMx bit in the INT_CFGCLR register disables it. Chapter 3, Interrupt System, describes the interrupt system in detail.

Several kinds of timer events can generate a timer interrupt, and each has a status flag in the INT_TIMxFLAG register to identify the reason(s) for the interrupt:

- INT_TIMTIF – set by a rising edge on an external trigger, either edge in gated mode
- INT_TIMCCRyIF – set by a channel y input capture or output compare event
- INT_TIMUIF – set by a UEV

Clear bits in INT_TIMxFLAG by writing a 1 to their bit position. When a channel is in capture mode, reading the TIMx_CCRy register will also clear the INT_TIMCCRyIF bit.

The INT_TIMxCFG register controls whether or not the INT_TIMxFLAG bits actually request a top-level NVIC timer interrupt. Only the events whose bits are set to 1 in INT_TIMxCFG can do so.

If an input capture or output compare event occurs and its INT_TIMMISSCCyIF is already set, the corresponding capture/compare missed flag is set in the INT_TMRxMISS register. Clear a bit in the INT_TMRxMISS register by writing a 1 to it.

SILICON LABS

10.5. Registers

Note: Substitute "1" or "2" for "x" in the following detailed descriptions.

Register 10.1. TIMx_CR1
TIM1_CR1: Timer 1 Control Register 1
TIM2_CR1: Timer 2 Control Register 1

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

TIM1_CR1: Address: 0x4000F000   Reset: 0x0
TIM2_CR1: Address: 0x40010000   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_ARBE	[7]	RW	Auto-Reload Buffer Enable. 0: TIMx_ARR register is not buffered. 1: TIMx_ARR register is buffered.
TIM_CMS	[6:5]	RW	Center-aligned Mode Selection. 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (TIM_DIR). 01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of configured output channels (TIM_CCyS=00 in TIMx_C-CMRy register) are set only when the counter is counting down. 10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of configured output channels (TIM_CCyS=00 in TIMx_C-CMRy register) are set only when the counter is counting up. 11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of configured output channels (TIM_CCyS=00 in TIMx_C-CMRy register) are set both when the counter is counting up or down. **Note:** Software may not switch from edge-aligned mode to center-aligned mode when the counter is enabled (TIM_CEN = 1).
TIM_DIR	[4]	RW	Direction. 0: Counter used as up-counter. 1: Counter used as down-counter.
TIM_OPM	[3]	RW	One Pulse Mode. 0: Counter does not stop counting at the next UEV. 1: Counter stops counting at the next UEV (and clears the bit TIM_CEN).

Bitname	Bitfield	Access	Description
TIM_URS	[2]	RW	Update Request Source. 0: When enabled, update interrupt requests are sent as soon as registers are updated (counter overflow/underflow, setting the TIM_UG bit, or update generation through the slave mode controller). 1: When enabled, update interrupt requests are sent only when the counter reaches overflow or underflow.
TIM_UDIS	[1]	RW	Update Disable. 0: A UEV is generated as soon as a counter overflow occurs, a software update is generated, or a hardware reset is generated by the slave mode controller. Shadow registers are then loaded with their buffer register values. 1: A UEV is not generated and shadow registers keep their value (TIMx_ARR, TIMx_PSC, TIMx_CCRy). The counter and the prescaler are reinitialized if the TIM_UG bit is set or if a hardware reset is received from the slave mode controller.
TIM_CEN	[0]	RW	Counter Enable. 0: Counter disabled. 1: Counter enabled. Note: External clock, gated mode and encoder mode can work only if the TIM_CEN bit has been previously set by software. Trigger mode sets the TIM_CEN bit automatically through hardware.

Register 10.2. TIMx_CR2
TIM1_CR2: Timer 1 Control Register 2
TIM2_CR2: Timer 2 Control Register 2

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	TIM_TI1S	TIM_MMS			0	0	0	0

TIM1_CR2: Address: 0x4000F004   Reset: 0x0
TIM2_CR2: Address: 0x40010004   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_TI1S	[7]	RW	TI1 Selection. 0: TI1M (input of the digital filter) is connected to TI1 input. 1: TI1M is connected to the TI_HALL inputs (XOR combination).
TIM_MMS	[6:4]	RW	Master Mode Selection. This selects the information to be sent in master mode to a slave timer for synchronization using the trigger output (TRGO). 000: Reset - the TIM_UG bit in the TMRx_EGR register is trigger output. If the reset is generated by the trigger input (slave mode controller configured in reset mode), then the signal on TRGO is delayed compared to the actual reset. 001: Enable - counter enable signal CNT_EN is trigger output. This mode is used to start both timers at the same time or to control a window in which a slave timer is enabled. The counter enable signal is generated by either the TIM_CEN control bit or the trigger input when configured in gated mode. When the counter enable signal is controlled by the trigger input there is a delay on TRGO except if the master/slave mode is selected (see the TIM_MSM bit description in TMRx_SMCR register). 010: Update - UEV is trigger output. This mode allows a master timer to be a prescaler for a slave timer. 011: Compare Pulse. The trigger output sends a positive pulse when the TIM_CC1IF flag is to be set (even if it was already high) as soon as a capture or a compare match occurs. 100: Compare - OC1REF signal is trigger output. 101: Compare - OC2REF signal is trigger output. 110: Compare - OC3REF signal is trigger output. 111: Compare - OC4REF signal is trigger output.

Register 10.3. TIMx_SMCR
TIM1_SMCR: Timer 1 Slave Mode Control Register
TIM2_SMCR: Timer 2 Slave Mode Control Register

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

TIM1_SMCR: Address: 0x4000F008   Reset: 0x0
TIM2_SMCR: Address: 0x40010008   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_ETP	[15]	RW	External Trigger Polarity. This bit selects whether ETR or the inverse of ETR is used for trigger operations. 0: ETR is non-inverted, active at a high level or rising edge. 1: ETR is inverted, active at a low level or falling edge.
TIM_ECE	[14]	RW	External Clock Enable. This bit enables external clock mode 2. 0: External clock mode 2 disabled. 1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal. **Notes:** **1.** Setting the TIM_ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (TIM_SMS=111 and TIM_TS=111). **2.** It is possible to use this mode simultaneously with the following slave modes: reset mode, gated mode and trigger mode. TRGI must not be connected to ETRF in this case (the TIM_TS bits must not be 111). **3.** If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input will be ETRF.
TIM_ETPS	[13:12]	RW	External Trigger Prescaler. External trigger signal ETRP frequency must be at most 1/4 of CK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful with fast external clocks. 00: ETRP prescaler off. 01: Divide ETRP frequency by 2. 10: Divide ETRP frequency by 4. 11: Divide ETRP frequency by 8.

SILICON LABS

Bitname	Bitfield	Access	Description
TIM_ETF	[11:8]	RW	External Trigger Filter. This defines the frequency used to sample the ETRP signal, Fsampling, and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N events are needed to validate a transition on the output: 0000: Fsampling=PCLK, no filtering. 0001: Fsampling=PCLK, N=2. 0010: Fsampling=PCLK, N=4. 0011: Fsampling=PCLK, N=8. 0100: Fsampling=PCLK/2, N=6. 0101: Fsampling=PCLK/2, N=8. 0110: Fsampling=PCLK/4, N=6. 0111: Fsampling=PCLK/4, N=8. 1000: Fsampling=PCLK/8, N=6. 1001: Fsampling=PCLK/8, N=8. 1010: Fsampling=PCLK/16, N=5. 1011: Fsampling=PCLK/16, N=6. 1100: Fsampling=PCLK/16, N=8. 1101: Fsampling=PCLK/32, N=5. 1110: Fsampling=PCLK/32, N=6. 1111: Fsampling=PCLK/32, N=8. **Note:** PCLK is 12 MHz when the EM358x is using the 24 MHz crystal oscillator, and 6 MHz if using the 12 MHz RC oscillator.
TIM_MSM	[7]	RW	Master/Slave Mode. 0: No action. 1: The effect of an event on the trigger input (TRGI) is delayed to allow exact synchronization between the current timer and the slave (through TRGO). It is useful for synchronizing timers on a single external event.
TIM_TS	[6:4]	RW	Trigger Selection. This bit field selects the trigger input used to synchronize the counter. 000 : Internal Trigger 0 (ITR0). 100 : TI1 Edge Detector (TI1F_ED). 101 : Filtered Timer Input 1 (TI1FP1). 110 : Filtered Timer Input 2 (TI2FP2). 111 : External Trigger input (ETRF). **Note:** These bits must be changed only when they are not used (when TIM_SMS = 000) to avoid detecting spurious edges during the transition.

Bitname	Bitfield	Access	Description
TIM_SMS	[2:0]	RW	Slave Mode Selection. When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input. 000: Slave mode disabled. If TIM_CEN = 1 then the prescaler is clocked directly by the internal clock. 001: Encoder mode 1. Counter counts up/down on TI1FP1 edge depending on TI2FP2 level. 010: Encoder mode 2. Counter counts up/down on TI2FP2 edge depending on TI1FP1 level. 011: Encoder mode 3. Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input. 100: Reset Mode. Rising edge of the selected trigger signal (TRGI) >reinitializes the counter and generates an update of the registers. 101: Gated Mode. The counter clock is enabled when the trigger signal (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both starting and stopping the counter are controlled. 110: Trigger Mode. The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only starting the counter is controlled. 111: External Clock Mode 1. Rising edges of the selected trigger (TRGI) clock the counter. **Note:** Gated mode must not be used if TI1F_ED is selected as the trigger input (TIM_TS = 100). TI1F_ED outputs 1 pulse for each transition on TI1F, whereas gated mode checks the level of the trigger signal.

Register 10.4. TIMx_EGR
TIM1_EGR: Timer 1 Event Generation Register
TIM2_EGR: Timer 2 Event Generation Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	TIM_TG	0	TIM_CC4G	TIM_CC3G	TIM_CC2G	TIM_CC1G	TIM_UG

TIM1_EGR: Address: 0x4000F014   Reset: 0x0
TIM2_EGR: Address: 0x40010014   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_TG	[6]	W	Trigger Generation. 0: Does nothing. 1: Sets the TIM_TIF flag in the INT_TIMxFLAG register.
TIM_CC4G	[4]	W	Capture/Compare 4 Generation. 0: Does nothing. 1: If CC4 configured as output channel: The TIM_CC4IF flag is set. If CC4 configured as input channel: The TIM_CC4IF flag is set. The INT_TIMMISSCC4IF flag is set if the TIM_CC4IF flag was already high. The current value of the counter is captured in TMRx_CCR4 register.
TIM_CC3G	[3]	W	Capture/Compare 3 Generation. 0: Does nothing. 1: If CC3 configured as output channel: The TIM_CC3IF flag is set. If CC3 configured as input channel: The TIM_CC3IF flag is set. The INT_TIMMISSCC3IF flag is set if the TIM_CC3IF flag was already high. The current value of the counter is captured in TMRx_CCR3 register.

SILICON LABS

Bitname	Bitfield	Access	Description
TIM_CC2G	[2]	W	Capture/Compare 2 Generation. 0: Does nothing. 1: If CC2 configured as output channel: The TIM_CC2IF flag is set. If CC2 configured as input channel: The TIM_CC2IF flag is set. The INT_TIMMISSCC2IF flag is set if the TIM_CC2IF flag was already high. The current value of the counter is captured in TMRx_CCR2 register.
TIM_CC1G	[1]	W	Capture/Compare 1 Generation. 0: Does nothing. 1: If CC1 configured as output channel: The TIM_CC1IF flag is set. If CC1 configured as input channel: The TIM_CC1IF flag is set. The INT_TIMMISSCC1IF flag is set if the TIM_CC1IF flag was already high. The current value of the counter is captured in TMRx_CCR1 register.
TIM_UG	[0]	W	Update Generation. 0: Does nothing. 1: Re-initializes the counter and generates an update of the registers. This also clears the prescaler counter but the prescaler ratio is not affected. The counter is cleared if center-aligned mode is selected or if TIM_DIR=0 (up-counting), otherwise it takes the auto-reload value (TMR1_ARR) if TIM_DIR=1 (down-counting).

Register 10.5. TIMx_CCMR1
TIM1_CCMR1: Timer 1 Capture/Compare Mode Register 1
TIM2_CCMR1: Timer 2 Capture/Compare Mode Register 1

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	TIM_OC2M			TIM_OC2BE	TIM_OC2FE	TIM_CC2S	
	TIM_IC2F				TIM_IC2PSC			
Bit	7	6	5	4	3	2	1	0
Name	0	TIM_OC1M			TIM_OC1BE	TIM_OC1FE	TIM_CC1S	
	TIM_IC1F				TIM_IC1PSC			

TIM1_CCMR1: Address: 0x4000F018   Reset: 0x0
TIM2_CCMR1: Address: 0x40010018   Reset: 0x0

Timer channels can be programmed as inputs (capture mode) or outputs (compare mode). The direction of channel y is defined by TIM_CCyS in this register.

The other bits in this register have different functions in input and in output modes. The TIM_OC* fields only apply to a channel configured as an output (TIM_CCyS = 0), and the TIM_IC* fields only apply to a channel configured as an input (TIM_CCyS > 0).

Bitname	Bitfield	Access	Description
TIM_OC2M	[14:12]	RW	Output Compare 2 Mode. (Applies only if TIM_CC2S = 0.) Define the behavior of the output reference signal OC2REF from which OC2 derives. OC2REF is active high whereas OC2''s active level depends on the TIM_CC2P bit. 000: Frozen - The comparison between the output compare register TIMx_CCR2 and the counter TIMx_CNT has no effect on the outputs. 001: Set OC2REF to active on match. The OC2REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 2 (TIMx_CCR2) 010: Set OC2REF to inactive on match. OC2REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 2 (TIMx_CCR2). 011: Toggle - OC2REF toggles when TIMx_CNT = TIMx_CCR2. 100: Force OC2REF inactive. 101: Force OC2REF active. 110: PWM mode 1 - In up-counting, OC2REF is active as long as TIMx_CNT < TIMx_CCR2, otherwise OC2REF is inactive. In down-counting, OC2REF is inactive if TIMx_CNT > TIMx_CCR2, otherwise OC2REF is active. 111: PWM mode 2 - In up-counting, OC2REF is inactive if TIMx_CNT < TIMx_CCR2, otherwise OC2REF is active. In down-counting, OC2REF is active if TIMx_CNT > TIMx_CCR2, otherwise it is inactive. **Note:** In PWM mode 1 or 2, the OC2REF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.
TIM_OC2BE	[11]	RW	Output Compare 2 Buffer Enable. (Applies only if TIM_CC2S = 0.) 0: Buffer register for TIMx_CCR2 is disabled. TIMx_CCR2 can be written at anytime, the new value is used by the shadow register immediately. 1: Buffer register for TIMx_CCR2 is enabled. Read/write operations access the buffer register. TIMx_CCR2 buffer value is loaded in the shadow register at each UEV. **Note:** The PWM mode can be used without enabling the buffer register only in one pulse mode (TIM_OPM bit set in the TIMx_CR2 register), otherwise the behavior is undefined.
TIM_OC2FE	[10]	RW	Output Compare 2 Fast Enable. (Applies only if TIM_CC2S = 0.) This bit speeds the effect of an event on the trigger in input on the OC2 output. 0: OC2 behaves normally depending on the counter and TIM_CCR2 values even when the trigger is ON. The minimum delay to activate OC2 when an edge occurs on the trigger input is 5 clock cycles. 1: An active edge on the trigger input acts like a compare match on the OC2 output. OC2 is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate OC2 output is reduced to 3 clock cycles. TIM_OC2FE acts only if the channel is configured in PWM 1 or PWM 2 mode.

Bitname	Bitfield	Access	Description
TIM_IC2F	[15:12]	RW	Input Capture 1 Filter. (Applies only if TIM_CC2S > 0.) This defines the frequency used to sample the TI2 input, Fsampling, and the length of the digital filter applied to TI2. The digital filter requires N consecutive samples in the same state before being output. 0000: Fsampling=PCLK, no filtering. 0001: Fsampling=PCLK, N=2. 0010: Fsampling=PCLK, N=4. 0011: Fsampling=PCLK, N=8. 0100: Fsampling=PCLK/2, N=6. 0101: Fsampling=PCLK/2, N=8. 0110: Fsampling=PCLK/4, N=6. 0111: Fsampling=PCLK/4, N=8. 1000: Fsampling=PCLK/8, N=6. 1001: Fsampling=PCLK/8, N=8. 1010: Fsampling=PCLK/16, N=5. 1011: Fsampling=PCLK/16, N=6. 1100: Fsampling=PCLK/16, N=8. 1101: Fsampling=PCLK/32, N=5. 1110: Fsampling=PCLK/32, N=6. 1111: Fsampling=PCLK/32, N=8. **Note:** PCLK is 12 MHz when using the 24 MHz crystal oscillator, and 6 MHz using the 12 MHz RC oscillator.
TIM_IC2PSC	[11:10]	RW	Input Capture 1 Prescaler. (Applies only if TIM_CC2S > 0.) 00: No prescaling, capture each time an edge is detected on the capture input. 01: Capture once every 2 events. 10: Capture once every 4 events. 11: Capture once every 8 events.
TIM_CC2S	[9:8]	RW	Capture / Compare 2 Selection. This configures the channel as an output or an input. If an input, it selects the input source. 00: Channel is an output. 01: Channel is an input and is mapped to TI2. 10: Channel is an input and is mapped to TI1. 11: Channel is an input and is mapped to TRGI. This mode requires an internal trigger input selected by the TIM_TS bit in the TIMx_SMCR register. **Note:** TIM_CC2S may be written only when the channel is off (TIM_CC2E = 0 in the TIMx_CCER register).
TIM_OC1M	[6:4]	RW	Output Compare 1 Mode. (Applies only if TIM_CC1S = 0.) See TIM_OC2M description above.
TIM_OC1BE	[3]	RW	Output Compare 1 Buffer Enable. (Applies only if TIM_CC1S = 0.) See TIM_OC2BE description above.
TIM_OC1FE	[2]	RW	Output Compare 1 Fast Enable. (Applies only if TIM_CC1S = 0.) See TIM_OC2FE description above.

SILICON LABS

Bitname	Bitfield	Access	Description
TIM_IC1F	[7:4]	RW	Input Capture 1 Filter. (Applies only if TIM_CC1S > 0.) See TIM_IC2F description above.
TIM_IC1PSC	[3:2]	RW	Input Capture 1 Prescaler. (Applies only if TIM_CC1S > 0.) See TIM_IC2PSC description above.
TIM_CC1S	[1:0]	RW	Capture / Compare 1 Selection. This configures the channel as an output or an input. If an input, it selects the input source. 00: Channel is an output. 01: Channel is an input and is mapped to TI1. 10: Channel is an input and is mapped to TI2. 11: Channel is an input and is mapped to TRGI. This requires an internal trigger input selected by the TIM_TS bit in the TIM_SMCR register. **Note:** TIM_CC1S may be written only when the channel is off (TIM_CC1E = 0 in the TIMx_CCER register).

Register 10.6. TIMx_CCMR2
TIM1_CCMR2: Timer 1 Capture/Compare Mode Register 2
TIM2_CCMR2: Timer 2 Capture/Compare Mode Register 2

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	TIM_OC4M			TIM_OC4BE	TIM_OC4FE	TIM_CC4S	
	TIM_IC4F				TIM_IC4PSC			
Bit	7	6	5	4	3	2	1	0
Name	0	TIM_OC3M			TIM_OC3BE	TIM_OC3FE	TIM_CC3S	
	TIM_IC3F				TIM_IC3PSC			

TIM1_CCMR2: Address: 0x4000F01C   Reset: 0x0
TIM2_CCMR2: Address: 0x4001001C   Reset: 0x0

Timer channels can be programmed as inputs (capture mode) or outputs (compare mode). The direction of channel y is defined by TIM_CCyS in this register.

The other bits in this register have different functions in input and in output modes. The TIM_OC* fields only apply to a channel configured as an output (TIM_CCyS = 0), and the TIM_IC* fields only apply to a channel configured as an input (TIM_CCyS > 0).

Bitname	Bitfield	Access	Description
TIM_OC4M	[14:12]	RW	Output Compare 4 Mode. (Applies only if TIM_CC4S = 0.) Define the behavior of the output reference signal OC4REF from which OC4 derives. OC4REF is active high whereas OC4's active level depends on the TIM_CC4P bit. 000: Frozen - The comparison between the output compare register TIMx_CCR4 and the counter TIMx_CNT has no effect on the outputs. 001: Set OC4REF to active on match. The OC4REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 4 (TIMx_CCR4) 010: Set OC4REF to inactive on match. OC4REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 4 (TIMx_CCR4). 011: Toggle - OC4REF toggles when TIMx_CNT = TIMx_CCR4. 100: Force OC4REF inactive. 101: Force OC4REF active. 110: PWM mode 1 - In up-counting, OC4REF is active as long as TIMx_CNT < TIMx_CCR4, otherwise OC4REF is inactive. In down-counting, OC4REF is inactive if TIMx_CNT > TIMx_CCR4, otherwise OC4REF is active. 111: PWM mode 2 - In up-counting, OC4REF is inactive if TIMx_CNT < TIMx_CCR4, otherwise OC4REF is active. In down-counting, OC4REF is active if TIMx_CNT > TIMx_CCR4, otherwise it is inactive. **Note:** In PWM mode 1 or 2, the OC4REF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.
TIM_OC4BE	[11]	RW	Output Compare 4 Buffer Enable. (Applies only if TIM_CC4S = 0.) 0: Buffer register for TIMx_CCR4 is disabled. TIMx_CCR4 can be written at anytime, the new value is used by the shadow register immediately. 1: Buffer register for TIMx_CCR4 is enabled. Read/write operations access the buffer register. TIMx_CCR4 buffer value is loaded in the shadow register at each UEV. **Note:** The PWM mode can be used without enabling the buffer register only in one pulse mode (TIM_OPM bit set in the TIMx_CR2 register), otherwise the behavior is undefined.
TIM_OC4FE	[10]	RW	Output Compare 4 Fast Enable. (Applies only if TIM_CC4S = 0.) This bit speeds the effect of an event on the trigger in input on the OC4 output. 0: OC4 behaves normally depending on the counter and TIM_CCR4 values even when the trigger is ON. The minimum delay to activate OC4 when an edge occurs on the trigger input is 5 clock cycles. 1: An active edge on the trigger input acts like a compare match on the OC4 output. OC4 is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate OC4 output is reduced to 3 clock cycles. TIM_OC4FE acts only if the channel is configured in PWM 1 or PWM 2 mode.

Bitname	Bitfield	Access	Description
TIM_IC4F	[15:12]	RW	Input Capture 4 Filter. (Applies only if TIM_CC4S > 0.) This defines the frequency used to sample the TI4 input, Fsampling, and the length of the digital filter applied to TI4. The digital filter requires N consecutive samples in the same state before being output. 0000: Fsampling=PCLK, no filtering. 0001: Fsampling=PCLK, N=2. 0010: Fsampling=PCLK, N=4. 0011: Fsampling=PCLK, N=8. 0100: Fsampling=PCLK/2, N=6. 0101: Fsampling=PCLK/2, N=8. 0110: Fsampling=PCLK/4, N=6. 0111: Fsampling=PCLK/4, N=8. 1000: Fsampling=PCLK/8, N=6. 1001: Fsampling=PCLK/8, N=8. 1010: Fsampling=PCLK/16, N=5. 1011: Fsampling=PCLK/16, N=6. 1100: Fsampling=PCLK/16, N=8. 1101: Fsampling=PCLK/32, N=5. 1110: Fsampling=PCLK/32, N=6. 1111: Fsampling=PCLK/32, N=8. **Note:** PCLK is 12 MHz when using the 24 MHz crystal oscillator, and 6 MHz using the 12 MHz RC oscillator.
TIM_IC4PSC	[11:10]	RW	Input Capture 4 Prescaler. (Applies only if TIM_CC4S > 0.) 00: No prescaling, capture each time an edge is detected on the capture input. 01: Capture once every 2 events. 10: Capture once every 4 events. 11: Capture once every 8 events.
TIM_CC4S	[9:8]	RW	Capture / Compare 4 Selection. This configures the channel as an output or an input. If an input, it selects the input source. 00: Channel is an output. 01: Channel is an input and is mapped to TI4. 10: Channel is an input and is mapped to TI3. 11: Channel is an input and is mapped to TRGI. This mode requires an internal trigger input selected by the TIM_TS bit in the TIMx_SMCR register. **Note:** TIM_CC4S may be written only when the channel is off (TIM_CC4E = 0 in the TIMx_CCER register).
TIM_OC3M	[6:4]	RW	Output Compare 3 Mode. (Applies only if TIM_CC3S = 0.) See TIM_OC4M description above.
TIM_OC3BE	[3]	RW	Output Compare 3 Buffer Enable. (Applies only if TIM_CC3S = 0.) See TIM_OC4BE description above.
TIM_OC3FE	[2]	RW	Output Compare 3 Fast Enable. (Applies only if TIM_CC3S = 0.) See TIM_OC4FE description above.

SILICON LABS

Bitname	Bitfield	Access	Description
TIM_IC3F	[7:4]	RW	Input Capture 3 Filter. (Applies only if TIM_CC3S > 0.) See TIM_IC4F description above.
TIM_IC3PSC	[3:2]	RW	Input Capture 3 Prescaler. (Applies only if TIM_CC3S > 0.) See TIM_IC4PSC description above.
TIM_CC3S	[1:0]	RW	Capture / Compare 3 Selection. This configures the channel as an output or an input. If an input, it selects the input source. 00: Channel is an output. 01: Channel is an input and is mapped to TI3. 10: Channel is an input and is mapped to TI4. 11: Channel is an input and is mapped to TRGI. This requires an internal trigger input selected by the TIM_TS bit in the TIM_SMCR register. **Note:** TIM_CC3S may be written only when the channel is off (TIM_CC3E = 0 in the TIMx_CCER register).

Register 10.7. TIMx_CCER
TIM1_CCER: Timer 1 Capture/Compare Enable Register
TIM2_CCER: Timer 2 Capture/Compare Enable Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	TIM_CC4P	TIM_CC4E	0	0	TIM_CC3P	TIM_CC3E
Bit	7	6	5	4	3	2	1	0
Name	0	0	TIM_CC2P	TIM_CC2E	0	0	TIM_CC1P	TIM_CC1E

TIM1_CCER: Address: 0x4000F020   Reset: 0x0
TIM2_CCER: Address: 0x40010020   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_CC4P	[13]	RW	Capture/Compare 4 output Polarity. If CC4 is configured as an output channel: 0: OC4 is active high. 1: OC4 is active low. If CC4 configured as an input channel: 0: IC4 is not inverted. Capture occurs on a rising edge of IC4. When used as an external trigger, IC4 is not inverted. 1: IC4 is inverted. Capture occurs on a falling edge of IC4. When used as an external trigger, IC4 is inverted.
TIM_CC4E	[12]	RW	Capture/Compare 4 output Enable. If CC4 is configured as an output channel: 0: OC4 is disabled. 1: OC4 is enabled. If CC4 configured as an input channel: 0: Capture is disabled. 1: Capture is enabled.
TIM_CC3P	[9]	RW	Refer to the CC4P description above.
TIM_CC3E	[8]	RW	Refer to the CC4E description above.
TIM_CC2P	[5]	RW	Refer to the CC4P description above.
TIM_CC2E	[4]	RW	Refer to the CC43 description above.
TIM_CC1P	[1]	RW	Refer to the CC4P description above.
TIM_CC1E	[0]	RW	Refer to the CC4E description above.

Register 10.8. TIMx_CNT
TIM1_CNT: Timer 1 Counter Register
TIM2_CNT: Timer 2 Counter Register

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

TIM1_CNT: Address: 0x4000F024   Reset: 0x0
TIM2_CNT: Address: 0x40010024   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_CNT	[15:0]	RW	Counter value.

Register 10.9. TIMx_PSC
TIM1_PSC: Timer 1 Prescaler Register
TIM2_PSC: Timer 2 Prescaler Register

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

TIM1_PSC: Address: 0x4000F028   Reset: 0x0
TIM2_PSC: Address: 0x40010028   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_PSC	[3:0]	RW	The prescaler divides the internal timer clock frequency. The counter clock frequency CK_CNT is equal to fCK_PSC / (2 ^ TIM_PSC). Clock division factors can range from 1 through 32768. The division factor is loaded into the shadow prescaler register at each UEV (including when the counter is cleared through TIM_UG bit of TMR1_EGR register or through the trigger controller when configured in reset mode).

SILICON LABS

Register 10.10. TIMx_ARR
TIM1_ARR: Timer 1 Auto-Reload Register
TIM2_ARR: Timer 2 Auto-Reload Register

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

TIM1_ARR: Address: 0x4000F02C   Reset: 0xFFFF
TIM2_ARR: Address: 0x4001002C   Reset: 0xFFFF

Bitname	Bitfield	Access	Description
TIM_ARR	[15:0]	RW	TIM_ARR is the value to be loaded in the shadow auto-reload register. The auto-reload register is buffered. Writing or reading the auto-reload register accesses the buffer register. The content of the buffer register is transfered in the shadow register permanently or at each UEV, depending on the auto-reload buffer enable bit (TIM_ARBE) in TMRx_CR1 register. The UEV is sent when the counter reaches the overflow point (or underflow point when down-counting) and if the TIM_UDIS bit equals 0 in the TMRx_CR1 register. It can also be generated by software. The counter is blocked while the auto-reload value is 0.

Register 10.11. TIMx_CCR1
TIM1_CCR1: Timer 1 Capture/Compare Register 1
TIM2_CCR1: Timer 2 Capture/Compare Register 1

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

TIM1_CCR1: Address: 0x4000F034   Reset: 0x0
TIM2_CCR1: Address: 0x40010034   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_CCR	[15:0]	RW	If the CC1 channel is configured as an output (TIM_CC1S = 0): TIM_CCR1 is the buffer value to be loaded in the actual capture/compare 1 register. It is loaded permanently if the preload feature is not selected in the TMR1_CCMR1 register (bit OC1PE). Otherwise the buffer value is copied to the shadow capture/compare 1 register when an UEV occurs. The active capture/compare register contains the value to be compared to the counter TMR1_CNT and signaled on the OC1 output. If the CC1 channel is configured as an input (TIM_CC1S is not 0): CCR1 is the counter value transferred by the last input capture 1 event (IC1).

Register 10.12. TIMx_CCR2
TIM1_CCR2: Timer 1 Capture/Compare Register 2
TIM2_CCR2: Timer 2 Capture/Compare Register 2

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

TIM1_CCR2: Address: 0x4000F038   Reset: 0x0
TIM2_CCR2: Address: 0x40010038   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_CCR	[15:0]	RW	See description in the TIMx_CCR1 register.

Register 10.13. TIMx_CCR3
TIM1_CCR3: Timer 1 Capture/Compare Register 3
TIM2_CCR3: Timer 2 Capture/Compare Register 3

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

TIM1_CCR3: Address: 0x4000F03C   Reset: 0x0
TIM2_CCR3: Address: 0x4001003C   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_CCR	[15:0]	RW	See description in the TIMx_CCR1 register.

Register 10.14. TIMx_CCR4
TIM1_CCR4: Timer 1 Capture/Compare Register 4
TIM2_CCR4: Timer 2 Capture/Compare Register 4

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

TIM1_CCR4: Address: 0x4000F040   Reset: 0x0
TIM2_CCR4: Address: 0x40010040   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_CCR	[15:0]	RW	See description in the TIMx_CCR1 register.

Register 10.15. TIM1_OR: Timer 1: Option Register

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

Timer 1: Address: 0x4000F050   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_ORRSVD	[3]	RW	Reserved: this bit must always be set to 0.
TIM_CLKMSKEN	[2]	RW	Enables TIM1MSK when TIM1CLK is selected as the external trigger: 0 = TIM1MSK not used, 1 = TIM1CLK is ANDed with the TIM1MSK input.
TIM_EXTRIGSEL	[1:0]	RW	Selects the external trigger used in external clock mode 2: 0 = PCLK, 1 = calibrated 1 kHz clock, 2 = 32 kHz reference clock (if available), 3 = TIM1CLK pin.

Register 10.16. TIM2_OR: Timer 2 Option Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TIM_REMAPC4	TIM_REMAPC3	TIM_REMAPC2	TIM_REMAPC1	TIM_ORRSVD	TIM_CLKMSKEN	TIM_EXTRIGSEL	

Address: 0x40010050   Reset: 0x0

Bitname	Bitfield	Access	Description
TIM_REMAPC4	[7]	RW	Selects the GPIO used for TIM2C4: 0 = PA2, 1 = PB4.
TIM_REMAPC3	[6]	RW	Selects the GPIO used for TIM2C3: 0 = PA1, 1 = PB3.
TIM_REMAPC2	[5]	RW	Selects the GPIO used for TIM2C2: 0 = PA3, 1 = PB2.
TIM_REMAPC1	[4]	RW	Selects the GPIO used for TIM2C1: 0 = PA0, 1 = PB1.
TIM_ORRSVD	[3]	RW	Reserved: this bit must always be set to 0.
TIM_CLKMSKEN	[2]	RW	Enables TIM2MSK when TIM2CLK is selected as the external trigger: 0 = TIM2MSK not used, 1 = TIM2CLK is ANDed with the TIM2MSK input.
TIM_EXTRIGSEL	[1:0]	RW	Selects the external trigger used in external clock mode 2: 0 = PCLK, 1 = calibrated 1 kHz clock, 2 = 32 kHz reference clock (if available), 3 = TIM2CLK pin.

Register 10.17. INT_TIMxCFG
INT_TIM1CFG: Timer 1 Interrupt Configuration Register
INT_TIM2CFG: Timer 2 Interrupt Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Name	0	INT_TIMTIF	0	INT_TIMCC4IF	INT_TIMCC3IF	INT_TIMCC2IF	INT_TIMCC1IF	INT_TIMUIF

INT_TIM1CFG: Address: 0x4000A840   Reset: 0x0
INT_TIM2CFG: Address: 0x4000A844   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_TIMTIF	[6]	RW	Trigger interrupt enable.
INT_TIMCC4IF	[4]	RW	Capture or compare 4 interrupt enable.
INT_TIMCC3IF	[3]	RW	Capture or compare 3 interrupt enable.
INT_TIMCC2IF	[2]	RW	Capture or compare 2 interrupt enable.
INT_TIMCC1IF	[1]	RW	Capture or compare 1 interrupt enable.
INT_TIMUIF	[0]	RW	Update interrupt enable.

Register 10.18. INT_TIMxFLAG
INT_TIM1FLAG: Timer 1 Interrupt Flag Register
INT_TIM2FLAG: Timer 2 Interrupt Flag Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Name	0	0	0	INT_TIMRSVD				0

Bit	7	6	5	4	3	2	1	0
Name	0	INT_TIMTIF	0	INT_TIMCC4IF	INT_TIMCC3IF	INT_TIMCC2IF	INT_TIMCC1IF	INT_TIMUIF

INT_TIM1FLAG: Address: 0x4000A800   Reset: 0x0
INT_TIM2FLAG: Address: 0x4000A804   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_TIMRSVD	[12:9]	R	May change during normal operation.
INT_TIMTIF	[6]	RW	Trigger interrupt.
INT_TIMCC4IF	[4]	RW	Capture or compare 4 interrupt pending.
INT_TIMCC3IF	[3]	RW	Capture or compare 3 interrupt pending.
INT_TIMCC2IF	[2]	RW	Capture or compare 2 interrupt pending.
INT_TIMCC1IF	[1]	RW	Capture or compare 1 interrupt pending.
INT_TIMUIF	[0]	RW	Update interrupt pending.

Register 10.19. INT_TIMxMISS
INT_TIM1MISS: Timer 1 Missed Interrupt Register
INT_TIM2MISS: Timer 2 Missed Interrupts Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	0	0	0	INT_ TIMMISSCC4IF	INT_ TIMMISSCC3IF	INT_ TIMMISSCC2IF	INT_ TIMMISSCC1IF	0
Bit	7	6	5	4	3	2	1	0
Name	0	INT_TIMMISSRSVD						

INT_TIM1MISS: Address: 0x4000A818   Reset: 0x0
INT_TIM2MISS: Address: 0x4000A81C   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_TIMMISSCC4IF	[12]	RW	Capture or compare 4 interrupt missed.
INT_TIMMISSCC3IF	[11]	RW	Capture or compare 3 interrupt missed.
INT_TIMMISSCC2IF	[10]	RW	Capture or compare 2 interrupt missed.
INT_TIMMISSCC1IF	[9]	RW	Capture or compare 1 interrupt missed.
INT_TIMMISSRSVD	[6:0]	R	May change during normal operation.

11. ADC (Analog to Digital Converter)

The EM358x ADC is a first-order sigma-delta converter with the following features:

- Resolution of up to 14 bits
- Sample times as fast as 5.33 µs (188 kHz)
- Differential and single-ended conversions from six external and four internal sources
- One voltage range (differential): –VREF to +VREF
- Choice of internal or external VREF
- Internal VREF may be output to PB0 or external VREF may be derived from PB0
- Digital offset and gain correction
- Dedicated DMA channel with one-shot and continuous operating modes

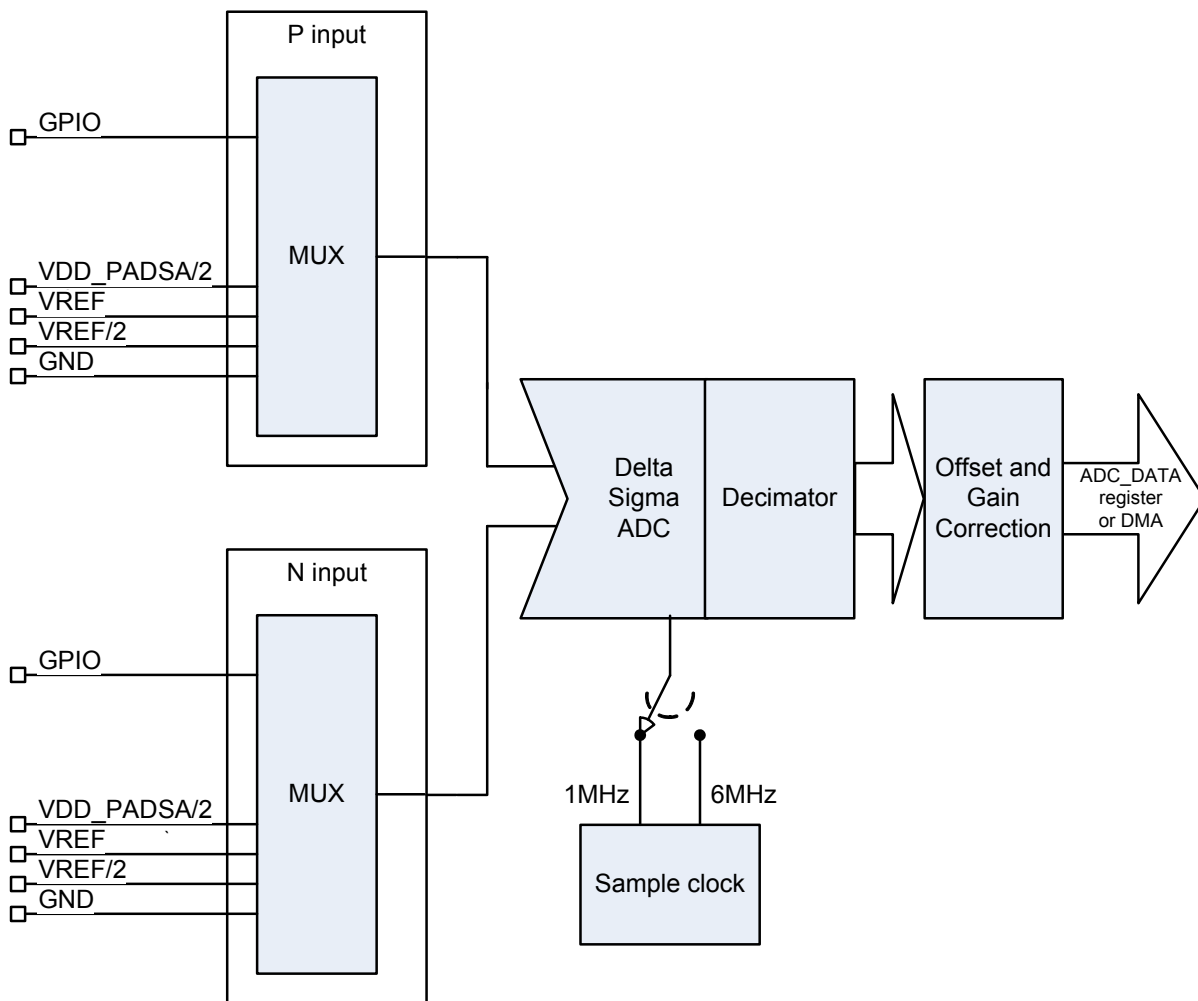Figure 11.1 shows the basic ADC structure.

Figure 11.1. ADC Block Diagram

While the ADC Module supports both single-ended and differential inputs, the ADC input stage always operates in differential mode. Single-ended conversions are performed by connecting one of the differential inputs to VREF/2 while fully differential operation uses two external inputs..

Note: The regulator input voltage, VDD_PADS, cannot be measured using the ADC, but it can be measured through Ember software.

11.1. Setup and Configuration

To use the ADC follow this procedure, described in more detail in the next sections:

- Configure any GPIO pins to be used by the ADC in analog mode.
- Configure the voltage reference (internal or external).
- Set the offset and gain values.
- If using DMA, reset the ADC DMA, define the DMA buffer, and start the DMA in the proper transfer mode.
- If interrupts will be used, configure the top-level and second-level ADC interrupt bits.
- Write the ADC configuration register to define the inputs, sample time, and start the conversions.

11.1.1. GPIO Usage

A GPIO pin used by the ADC as an input or voltage reference must be configured in analog mode by writing 0 to its 4-bit field in the proper GPIO_PxCFGH/L register. Note that a GPIO pin in analog mode cannot be used for any digital functions, and GPIO_PxIN always reads it as 1. Only certain pins can be configured in analog mode. These are listed in Table 11.1.

Table 11.1. ADC GPIO Pin Usage

Analog Signal	GPIO	Configuration control
ADC0 input	PB5	GPIO_PBCFGH[7:4]
ADC1 input	PB6	GPIO_PBCFGH[11:8]
ADC2 input	PB7	GPIO_PBCFGH[15:12]
ADC3 input	PC1	GPIO_PCCFGL[7:4]
ADC4 input	PA4	GPIO_PACFGH[3:0]
ADC5 input	PA5	GPIO_PACFGH[7:4]
VREF input or output	PB0	GPIO_PBCFGL[3:0]

See "7. GPIO (General Purpose Input/Output)" on page 53 for more information about how to configure GPIO.

11.1.2. Voltage Reference

The ADC voltage reference (VREF), may be internally generated or externally sourced from PB0. If internally generated, it may optionally be output on PB0. To output the internal VREF on PB0, the ADC must be enabled (ADC_ENABLE bit set in the ADC_CFG register) and PB0 must be configured in analog mode.

To use an external reference, the Ember software must be called after reset and after waking from deep sleep. PB0 must also be configured in analog mode using GPIO_PBCFGH[3:0]. See the Ember software documentation for more information on using an external reference.

11.1.3. Offset/Gain Correction

When a conversion is complete, the 16-bit converted data is processed in several steps by offset/gain correction hardware:

1. The initial signed ADC conversion result is added to the 16-bit signed (2's complement) value of the ADC offset register (ADC_OFFSET).

2. The offset-corrected data is multiplied by the 16-bit ADC gain register, ADC_GAIN, to produce a 16-bit signed result. If the product is greater than 0x7FFF (32767), or less than 0x8000 (-32768), it is limited to that value and the INT_ADCSAT bit is set in the INT_ADCFLAG register.

3. The offset/gain corrected value is divided by two to produce the final result.

ADC_GAIN is an unsigned scaled 16-bit value: ADC_GAIN[15] is the integer part of the gain factor and ADC_GAIN[14:0] is the fractional part. As a result, ADC_GAIN values can represent gain factors from 0 through $(2 - 2^{-15})$. Although ADC_GAIN can represent a much greater range, its purpose is to correct small gain error, and in practice is loaded with values within a range of about 0.95 to 1.05.

Reset initializes the offset to zero (ADC_OFFSET = 0) and gain factor to one (ADC_GAIN = 0x8000)..

11.1.4. DMA

The ADC DMA channel writes converted data, which incorporates the offset/gain correction, into a DMA buffer in RAM.

The ADC DMA buffer is defined by two registers:

■ ADC_DMABEG is the start address of the buffer and must be even.
■ ADC_DMASIZE specifies the size of the buffer in 16-bit samples, or half its length in bytes.

To prepare the DMA channel for operation, reset it by writing the ADC_DMARST bit in the ADC_DMACFG register, then start the DMA in either linear or auto wrap mode by setting the ADC_DMALOAD bit in the ADC_DMACFG register. The ADC_DMAAUTOWRAP bit in the ADC_DMACFG register selects the DMA mode: 0 for linear mode, 1 for auto wrap mode.

■ In linear mode the DMA writes to the buffer until the number of samples given by ADC_DMASIZE has been output. The DMA then stops and sets the INT_ADCULDFULL bit in the INT_ADCFLAG register. If another ADC conversion completes before the DMA is reset or the ADC is disabled, the INT_ADCOVF bit in the INT_ADCFLAG register is set.
■ In auto wrap mode the DMA writes to the buffer until it reaches the end, then resets its pointer to the start of the buffer and continues writing samples. The DMA transfers continue until the ADC is disabled or the DMA is reset

When the DMA fills the lower and upper halves of the buffer, it sets the INT_ADCULDHALF and INT_ADCULDFULL bits, respectively, in the INT_ADCFLAG register. The current location to which the DMA is writing can also be determined by reading the ADC_DMACUR register.

11.1.5. ADC Configuration Register

The ADC configuration register (ADC_CFG) sets up most of the ADC operating parameters.

11.1.5.1. Input

The analog input of the ADC can be chosen from various sources. The analog input is configured with the ADC_MUXP and ADC_MUXN bits within the ADC_CFG register. Table 11.2 shows the possible input selections.

Table 11.2. ADC Inputs

ADC_MUXn*	Analog Source at ADC	GPIO Pin	Purpose
0	ADC0	PB5	
1	ADC1	PB6	
2	ADC2	PB7	
3	ADC3	PC1	
4	ADC4	PA4	
5	ADC5	PA5	
6	No connection		
7	No connection		
8	GND	Internal connection	Calibration
9	VREF/2	Internal connection	Calibration
10	VREF	Internal connection	Calibration
11	VDD_PADSA/2	Internal connection	Supply monitoring and calibration
12	No connection		
13	No connection		
14	No connection		
15	No connection		
Note: Denotes bits ADC_MUXP or ADC_MUXN in register ADC_CFG.			

Table 11.3 shows the typical configurations of ADC inputs.

Table 11.3. Typical ADC Input Configurations

ADC P Input	ADC N Input	ADC_MUXP	ADC_MUXN	Purpose
ADC0	VREF/2	0	9	Single-ended
ADC1	VREF/2	1	9	Single-ended
ADC2	VREF/2	2	9	Single-ended
ADC3	VREF/2	3	9	Single-ended
ADC4	VREF/2	4	9	Single-ended
ADC5	VREF/2	5	9	Single-ended
ADC1	ADC0	1	0	Differential
ADC3	ADC2	3	2	Differential
ADC5	ADC4	5	4	Differential
GND	VREF/2	8	9	Calibration
VREF	VREF/2	10	9	Calibration
VDD_PADSA/2	VREF/2	11	9	Calibration

SILICON LABS

11.1.5.2. Input Range

The single-ended input range is fixed as 0 V to VREF and the differential input range is fixed as –VREF to +VREF.

11.1.5.3. Sample Time

ADC sample time is programmed by selecting the sampling clock and the clocks per sample.

- The sampling clock may be either 1 MHz or 6 MHz. If the ADC_1MHZCLK bit in the ADC_CFG register is clear, the 6 MHz clock is used; if it is set, the 1 MHz clock is selected. The 6 MHz sample clock offers faster conversion times but the ADC resolution is lower than that achieved with the 1 MHz clock.
- The number of clocks per sample is determined by the ADC_PERIOD bits in the ADC_CFG register. ADC_PERIOD values select from 32 to 4096 sampling clocks in powers of two. Longer sample times produce more significant bits. Regardless of the sample time, converted samples are always 16-bits in size with the significant bits left-aligned within the value.

Table 11.4 shows the options for ADC sample times and the significant bits in the conversion results.

Table 11.4. ADC Sample Times*

ADC_PERIOD	Sample Clocks	Sample Time (µs)		Sample Frequency (kHz)		Significant Bits
		1 MHz Clock	6 MHz Clock	1 MHz Clock	6 MHz Clock	
0	32	32	5.33	31.3	188	7
1	64	64	10.7	15.6	93.8	8
2	128	128	21.3	7.81	46.9	9
3	256	256	42.7	3.91	23.4	10
4	512	512	85.3	1.95	11.7	11
5	1024	1024	170	0.977	5.86	12
6	2048	2048	341	0.488	2.93	13
7	4096	4096	682	0.244	1.47	14

*Note: ADC sample timing is the same whether the EM358x is using the 24 MHz crystal oscillator or the 12 MHz high-speed RC oscillator. This facilitates using the ADC soon after the CPU wakes from deep sleep, before switching to the crystal oscillator.

11.2. Interrupts

The ADC has its own top-level interrupt in the NVIC. The ADC interrupt is enabled by writing the INT_ADC bit to the INT_CFGSET register, and cleared by writing the INT_ADC bit to the INT_CFGCLR register. Chapter 3, Interrupt System, describes the interrupt system in detail.

Five kinds of ADC events can generate an ADC interrupt, and each has a bit flag in the INT_ADCFLAG register to identify the reason(s) for the interrupt:

- INT_ADCOVF – an ADC conversion result was ready but the DMA was disabled (DMA buffer overflow).
- INT_ADCSAT– the gain correction multiplication exceeded the limits for a signed 16-bit number (gain saturation).
- INT_ADCULDFULL – the DMA wrote to the last location in the buffer (DMA buffer full).
- INT_ADCULDHALF – the DMA wrote to the last location of the first half of the DMA buffer (DMA buffer half full).
- INT_ADCDATA – there is data ready in the ADC_DATA register.

Bits in INT_ADCFLAG register may be cleared by writing a 1 to their position. Writing 0 to any bit in the

INT_ADCFLAG register is ineffectual.

The INT_ADCCFG register controls whether or not INT_ADCFLAG register bits actually propagate the ADC interrupt to the NVIC. Only the events whose bits are 1 in the INT_ADCCFG register can do so.

For non-interrupt (polled) ADC operation set the INT_ADCCFG register to zero, and read the bit flags in the INT_ADCFLAG register to determine the ADC status.

Note: Note: When making changes to the ADC configuration it is best to disable the DMA beforehand. If this isn't done it can be difficult to determine at which point the sampled data in the DMA buffer switched from the old configuration to the new configuration. However, since the ADC will be left running, if it completes a conversion after the DMA is disabled, the INT_ADCOVF flag will be set. To prevent these unwanted DMA buffer overflow indications, clear the INT_ADCOVF flag immediately after enabling the DMA, preferably with interrupts off. Disabling the ADC in addition to the DMA is often undesirable because of the additional analog startup time when it is re-enabled.

11.3. Operation

Setting the ADC_EN bit in the ADC_CFG register enables the ADC. Once the ADC is enabled, it performs conversions continuously until it is disabled. If the ADC had previously been disabled, a 21 µs analog startup delay is automatically imposed before the ADC starts conversions. The delay timing is performed in hardware and is simply added to the time until the first conversion result is output.

When the ADC is first enabled, and/or if any change is made to ADC_CFG after it is enabled, the time until a result is output is double the normal sample time. This is because the ADC's internal design requires it to discard the first conversion after startup or a configuration change. This is done automatically and is hidden from software. Switching the system clock between OSCHF and OSC24M also causes the ADC to go through this startup cycle. If the ADC was newly enabled, the analog delay time is added to the doubled sample time.

If the DMA is running when the ADC_CFG register is modified, the DMA does not stop, so the DMA buffer may contain conversion results from both the old and new configurations.

The following procedure illustrates a simple polled method of using the ADC without DMA. This assumes that any GPIOs and the voltage reference have already been configured.

1. Disable all ADC interrupts: Write 0 to the INT_ADCCFG register.

2. Write the desired offset and gain correction values to the ADC_OFFSET and ADC_GAIN registers.

3. Write the desired conversion configuration, with the ADC_EN bit set, to ADC_CFG register.

4. Clear the ADC data flag: Write the INT_ADCDATA bit to INT_ADCFLAG register.

5. Wait until the INT_ADCDATA bit is set in INT_ADCFLAG register, then read the result, as a 16-bit signed variable, from the ADC_DATA register.

The following procedure illustrates a simple polled method of using the ADC with DMA. After completing the procedure, the latest conversion results are available in the location written to by the DMA. This assumes that any GPIOs and the voltage reference have already been configured.

1. Allocate a 16-bit signed variable, for example analogData, to receive the ADC output.
   (Make sure that analogData is half-word aligned – that is, at an even address.)

2. Disable all ADC interrupts: Write 0 to the INT_ADCCFG register.

3. Set up the DMA to output conversion results to the variable, analogData.
   Reset the DMA: Set the ADC_DMARST bit in ADC_DMACFG register.
   Define a one sample buffer: Write analogData's address to the ADC_DMABEG register and set the ADC_DMASIZE register to 1.

4. Write the desired offset and gain correction values to the ADC_OFFSET and ADC_GAIN registers.

5. Start the ADC and the DMA.
   Write the desired conversion configuration, with the ADC_EN bit set, to the ADC_CFG register.
   Clear the ADC buffer full flag: Write the INT_ADCULDFULL bit to the INT_ADCFLAG register.
   Start the DMA in auto wrap mode: Set the ADC_DMAAUTOWRAP and ADC_DMALOAD bits in the ADC_DMACFG register.

6. Wait until the INT_ADCULDFULL bit is set in the INT_ADCFLAG register, then read the result from analogData.

To convert multiple inputs using this approach, repeat steps 4 through 6, loading the desired input configurations to the ADC_CFG register in Step 5. If the inputs can use the same offset/gain correction, just repeat steps 5 and 6.

11.4. Calibration

Sampling of internal connections GND, VREF/2, and VREF allow for offset and gain calibration of the ADC in applications where absolute accuracy is important. Offset error is calculated from the minimum input and gain error is calculated from the full scale input range. Correction using VREF is recommended because VREF is calibrated by the Ember software against VDD_PADSA. The VDD_PADSA regulator is factory-trimmed to 1.80 V ± 20 mV. If better absolute accuracy is required, the ADC can be configured to use an external reference. The ADC calibrates as a single-ended measurement. Differential signals require correction of both their inputs.

The following steps outline the calibration procedure:

1. Calibrate VREF against VDD_PADSA.
2. Determine the ADC gain by sampling independently VREF and GND. Gain is calculated from the slope of these two measurements.
3. Apply gain correction.
4. Determine the ADC offset by sampling GND.
5. Apply offset correction.

Table 11.5 shows the equations used to calculate the gain and offset correction values.

Table 11.5. ADC Gain and Offset Correction Equations

Calibration	Correction Value
Gain	$32768 \times \dfrac{16384}{(N_{VREF} - N_{GND})}$
Offset (after applying gain correction)	$2 \times (57344 - N_{GND})$

Notes:
1. The ADC output is 2s complement. All N are therefore 16-bit 2s complement numbers.
2. Offset is a 16-bit 2s complement number.
3. Gain is a 16-bit number representing a gain of 0 to 65535/32768 in 1/32768 steps. The default value is 32768, corresponding to a gain of 1.
4. N_{GND} is a sampling of ground. Due to the ADC's internal design, VGND does not yield the minimum 16 bit 2s complement value 32768 as the conversion result. Instead, VGND yields a value close to 57344 when the input buffer is not selected. VGND cannot be measured when the input buffer is enabled because it is outside the buffer's input range.
5. N_{VREF} is a sampling of VREF. Due to the ADC's internal design, VREF does not yield the maximum positive 16-bit 2s complement 32767 as the conversion result. Instead, VREF yields a value close to 8192.
6. $N_{VREF/2}$ is a sampling of VREF/2. VREF/2 yields a value close to 0.
7. Offset correction is affected by the gain correction value. Offset correction is calculated after gain correction has been applied.

11.5. ADC Key Parameters

Table 11.6 describes the key ADC parameters measured at 25°C and VDD_PADS at 3.0 V, for a sampling clock of 1 MHz. The single-ended measurements were done at f_{input} = 7.7% $f_{Nyquist}$; 0 dBFS level (where full-scale is a 1.2 V p-p swing). The differential measurements were done at f_{input} = 7.7% $f_{Nyquist}$; –6 dBFS level (where full-scale is a 2.4 V p-p swing) and a common mode voltage of 0.6 V.

Table 11.6. ADC Module Key Parameters for 1 MHz Sampling

Parameter	Performance							
ADC_PERIOD	0	1	2	3	4	5	6	7
Conversion Time (µs)	32	64	128	256	512	1024	2048	4096
Nyquist Freq (kHz)	15.6k	7.81k	3.91k	1.95k	977	488	244	122
3 dB Cut-off (kHz)	9.43k	4.71k	2.36k	1.18k	589	295	147	73.7
INL (codes peak)	0.083	0.092	0.163	0.306	0.624	1.229	2.451	4.926
INL (codes RMS)	0.047	0.051	0.093	0.176	0.362	0.719	1.435	2.848
DNL (codes peak)	0.028	0.035	0.038	0.044	0.074	0.113	0.184	0.333
DNL (codes RMS)	0.008	0.009	0.011	0.014	0.019	0.029	0.048	0.079
ENOB (from single-cycle test)	5.6	7.0	8.6	10.1	11.5	12.6	13.0	13.2
SNR (dB) Single-Ended Differential	35 35	44 44	53 53	62 62	70 71	75 77	77 79	77 80
SINAD (dB) Single-Ended Differential	35 35	44 44	53 53	61 62	67 70	69 75	70 76	70 76
SDFR (dB) Single-Ended Differential	59 60	68 69	72 77	72 80	72 81	72 81	72 81	73 81
THD (dB) Single-Ended Differential	–45 –45	–54 –54	–62 –63	–67 –71	–69 –75	–69 –76	–69 –76	–69 –76
ENOB (from SNR) Single-Ended Differential	5.6 5.6	7.1 7.1	8.6 8.6	10.0 10.1	11.3 11.4	12.2 12.5	12.4 12.9	12.5 12.9

*Note: INL and DNL are referenced to a LSB of the Equivalent ADC Bits shown in the last row of Table 11.6. ENOB (effective number of bits) can be calculated from either SNR (signal to non-harmonic noise ratio) or SINAD (signal-to-noise and distortion ratio).

SILICON LABS

Table 11.6. ADC Module Key Parameters for 1 MHz Sampling (Continued)

ENOB (from SINAD) Single-Ended Differential	5.5 5.6	7.0 7.0	8.5 8.5	9.9 10.0	10.9 11.3	11.2 12.1	11.3 12.3	11.3 12.4
Equivalent ADC Bits	7 [15:9]	8 [15:8]	9 [15:7]	10 [15:6]	11 [15:5]	12 [15:4]	13 [15:3]	14 [15:2]

*Note: INL and DNL are referenced to a LSB of the Equivalent ADC Bits shown in the last row of Table 11.6. ENOB (effective number of bits) can be calculated from either SNR (signal to non-harmonic noise ratio) or SINAD (signal-to-noise and distortion ratio).

SILICON LABS

Table 11.7 describes the key ADC parameters measured at 25°C and VDD_PADS at 3.0 V, for a sampling rate of 6 MHz. The single-ended measurements were done at f_{input} = 7.7% $f_{Nyquist}$; 0 dBFS level (where full-scale is a 1.2 V p-p swing). The differential measurements were done at f_{input} = 7.7% $f_{Nyquist}$; –6 dBFS level (where full-scale is a 2.4 V p-p swing) and a common mode voltage of 0.6 V.

Table 11.7. ADC Module Key Parameters for 6 MHz Sampling

Parameter	Performance							
ADC_PERIOD	0	1	2	3	4	5	6	7
Conversion Time (µs)	5.33	10.7	21.3	42.7	85.3	171	341	683
Nyquist Freq (kHz)	93.8k	46.9k	23.4k	11.7k	5.86k	2.93k	1.47k	732
3 dB Cut-off (kHz)	56.6k	28.3k	14.1k	7.07k	3.54k	1.77k	884	442
INL (codes peak)	0.084	0.084	0.15	0.274	0.518	1.057	2.106	4.174
INL (codes RMS)	0.046	0.044	0.076	0.147	0.292	0.58	1.14	2.352
DNL (codes peak)	0.026	0.023	0.044	0.052	0.096	0.119	0.196	0.371
DNL (codes RMS)	0.007	0.009	0.013	0.015	0.024	0.03	0.05	0.082
ENOB (from single-cycle test)	5.6	7.0	8.5	10.0	11.4	12.6	13.1	13.2
SNR (dB) Single-Ended Differential	35 35	44 44	53 53	62 62	70 71	75 77	76 79	77 80
SINAD (dB) Single-Ended Differential	35 35	44 44	53 53	62 62	68 70	71 75	71 77	71 77
SDFR (dB) Single-Ended Differential	60 60	68 69	75 77	75 80	75 80	75 80	75 80	75 80
THD (dB) Single-Ended Differential	–45 –45	–54 –54	–63 –63	–68 –71	–70 –76	–70 –77	–70 –78	–70 –78
ENOB (from SNR) Single-Ended Differential	5.6 5.6	7.1 7.1	8.6 8.6	10.0 10.1	11.4 11.5	12.1 12.5	12.4 12.9	12.5 13.0
*Note: INL and DNL are referenced to a LSB of the Equivalent ADC Bits shown in the last row of Table 11.7. ENOB (effective number of bits) can be calculated from either SNR (signal to non-harmonic noise ratio) or SINAD (signal-to-noise and distortion ratio).								

SILICON LABS

Table 11.7. ADC Module Key Parameters for 6 MHz Sampling (Continued)

ENOB (from SINAD) Single-Ended Differential	5.5 5.6	7.0 7.1	8.5 8.6	9.9 10.1	11.0 11.4	11.4 12.4	11.5 12.8	11.5 13.0
Equivalent ADC Bits	7 [15:9]	8 [15:8]	9 [15:7]	10 [15:6]	11 [15:5]	12 [15:4]	13 [15:3]	14 [15:2]

*Note: INL and DNL are referenced to a LSB of the Equivalent ADC Bits shown in the last row of Table 11.7. ENOB (effective number of bits) can be calculated from either SNR (signal to non-harmonic noise ratio) or SINAD (signal-to-noise and distortion ratio).

Table 11.8 describes the key ADC parameters measured at 25°C and VDD_PADS at 3.0 V, for a sampling clock of 6 MHz. The single-ended measurements were done at f_{input} = 7.7% $f_{Nyquist}$; level = 1.2 V p-p swing centered on 1.5 V. The differential measurements were done at f_{input} = 7.7% $f_{Nyquist}$, level = 1.2 V p-p swing and a common mode voltage of 1.5 V.

Table 11.8. ADC Module Key Parameters for Input Buffer Enabled and 6 MHz Sampling

Parameter	Performance							
ADC_PERIOD	0	1	2	3	4	5	6	7
Conversion Time (µs)	32	64	128	256	512	1024	2048	4096
Nyquist Freq (kHz)	93.8k	46.9k	23.4k	11.7k	5.86k	2.93k	1.47k	732
3 dB Cut-off (kHz)	56.6k	28.3k	14.1k	7.07k	3.54k	1.77k	884	442
INL (codes peak)	0.055	0.032	0.038	0.07	0.123	0.261	0.522	1.028
INL (codes RMS)	0.028	0.017	0.02	0.04	0.077	0.167	0.326	0.65
DNL (codes peak)	0.028	0.017	0.02	0.04	0.077	0.167	0.326	0.65
DNL (codes RMS)	0.01	0.006	0.006	0.007	0.008	0.013	0.023	0.038
ENOB (from single-cycle test)	3.6	5.0	6.6	8.1	9.5	10.7	11.3	11.6
SNR (dB) Single-Ended Differential	23 23	32 32	41 41	50 50	59 59	65 66	67 69	68 71
SINAD (dB) Single-Ended Differential	23 23	32 32	41 41	50 50	58 59	64 66	66 69	66 71
SDFR (dB) Single-Ended Differential	48 48	56 57	65 65	72 74	72 82	72 88	73 88	73 88
THD (dB) Single-Ended Differential	–33 –33	–42 –42	–51 –51	–59 –60	–66 –69	–68 –76	–68 –80	–68 –82
ENOB (from SNR) Single-Ended Differential	3.6 3.6	5.1 5.1	6.6 6.6	8.1 8.1	9.5 9.5	10.5 10.7	10.9 11.3	11 11.5

*Note: INL and DNL are referenced to a LSB of the Equivalent ADC Bits shown in the last row of Table 11.8. ENOB (effective number of bits) can be calculated from either SNR (signal to non-harmonic noise ratio) or SINAD (signal-to-noise and distortion ratio).

SILICON LABS

Table 11.8. ADC Module Key Parameters for Input Buffer Enabled and 6 MHz Sampling (Continued)

ENOB (from SINAD) Single-Ended Differential	3.6 3.6	5.0 5.1	6.5 6.6	8.0 8.0	9.4 9.5	10.3 10.6	10.7 11.3	10.7 11.4
Equivalent ADC Bits	7 [15:9]	8 [15:8]	9 [15:7]	10 [15:6]	11 [15:5]	12 [15:4]	13 [15:3]	14 [15:2]

*Note: INL and DNL are referenced to a LSB of the Equivalent ADC Bits shown in the last row of Table 11.8. ENOB (effective number of bits) can be calculated from either SNR (signal to non-harmonic noise ratio) or SINAD (signal-to-noise and distortion ratio).

Table 11.9 lists other specifications for the ADC not covered in Tables 11.6 and 11.7.

Table 11.9. ADC Specifications*

Parameter	Min	Typ	Max	Units
VREF	1.17	1.2	1.23	V
VREF output current			1	mA
VREF load capacitance			10	nF
External VREF voltage range	1.1	1.2	1.3	V
External VREF input impedance	1			MΩ
Minimum input voltage	0			V
Maximum input voltage			VREF	V
Single-ended signal range	0		VREF	V
Differential signal range	–VREF		+VREF	V
Common mode range	0		VREF	V
Input referred ADC offset	–10		10	mV
Input Impedance: 1 MHz sample clock 6 MHz sample clock Not sampling	1 0.5 10			MΩ

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

SILICON LABS

11.6. Registers

Register 11.1. ADC_DATA: ADC Data Register

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

Address: 0x4000E000   Reset: 0x00000000

Bitname	Bitfield	Access	Description
ADC_DATA_FIELD	[15:0]	R	ADC conversion result. The result is a signed 2s complement value. The significant bits of the value begin at bit 15 regardless of the sample period used.

Register 11.2. ADC_CFG: ADC Configuration Register

Bit	31	30	29	28	27	26	25	24
Name	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
Name	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Name	ADC_PERIOD			ADC_CFGRSVD2		ADC_MUXP		
Bit	7	6	5	4	3	2	1	0
Name	ADC_MUXP	ADC_MUXN				ADC_1MHZCLK	ADC_CFGRSVD	ADC_ENABLE

Address: 0x4000E004   Reset: 0x00001800

Bitname	Bitfield	Access	Description
ADC_PERIOD	[15:13]	RW	ADC sample time in clocks and the equivalent significant bits in the conversion. 0: 32 clocks (7 bits). 1: 64 clocks (8 bits). 2: 128 clocks (9 bits). 3: 256 clocks (10 bits). 4: 512 clocks (11 bits). 5: 1024 clocks (12 bits). 6: 2048 clocks (13 bits). 7: 4096 clocks (14 bits).
ADC_CFGRSVD2	[12:11]	RW	Reserved: these bits must be set to 0.
ADC_MUXP	[10:7]	RW	Input selection for the P channel. 0x0: PB5 pin. 0x1: PB6 pin. 0x2: PB7 pin. 0x3: PC1 pin. 0x4: PA4 pin. 0x5: PA5 pin. 0x8: GND (0 V) (not for high voltage range). 0x9: VREF/2 (0.6 V). 0xA: VREF (1.2 V). 0xB: VDD_PADSA/2 (0.9 V) (not for high voltage range). 0x6, 0x7, 0xC-0xF: Reserved.
ADC_MUXN	[6:3]	RW	Input selection for the N channel. Refer to ADC_MUXP above for choices.
ADC_1MHZCLK	[2]	RW	Select ADC clock: 0 = 6 MHz, 1 = 1 MHz.
ADC_CFGRSVD	[1]	RW	Reserved: this bit must always be set to 0.
ADC_ENABLE	[0]	RW	Enable the ADC: write 1 to enable continuous conversions, write 0 to stop. When the ADC is started the first conversion takes twice the usual number of clocks plus 21 microseconds. If anything in this register is modified while the ADC is running, the next conversion takes twice the usual number of clocks.

SILICON LABS

Register 11.3. ADC_OFFSET: ADC Offset Register

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

Address: 0x4000E008   Reset: 0x0000

Bitname	Bitfield	Access	Description
ADC_OFFSET_FIELD	[15:0]	RW	16-bit signed offset added to the basic ADC conversion result before gain correction is applied.

Register 11.4. ADC_GAIN: ADC Gain Register

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

Address: 0x4000E00C   Reset: 0x8000

Bitname	Bitfield	Access	Description
ADC_GAIN_FIELD	[15:0]	RW	Gain factor that is multiplied by the offset-corrected ADC result to produce the output value. The gain is a 16-bit unsigned scaled integer value with a binary decimal point between bits 15 and 14. It can represent values from 0 to (almost) 2. The reset value is a gain factor of 1.

Register 11.5. ADC_DMACFG: ADC DMA Configuration Register

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

Address: 0x4000E010   Reset: 0x0

Bitname	Bitfield	Access	Description
ADC_DMARST	[4]	W	Write 1 to reset the ADC DMA. This bit auto-clears.
ADC_DMAAUTOWRAP	[1]	RW	Selects DMA mode. 0: Linear mode, the DMA stops when the buffer is full. 1: Auto-wrap mode, the DMA output wraps back to the start when the buffer is full.
ADC_DMALOAD	[0]	RW	Loads the DMA buffer. Write 1 to start DMA (writing 0 has no effect). Cleared when DMA starts or is reset.

Register 11.6. ADC_DMASTAT: ADC DMA Status Register

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

Address: 0x4000E014   Reset: 0x0

Bitname	Bitfield	Access	Description
ADC_DMAOVF	[1]	R	DMA overflow: occurs when an ADC result is ready and the DMA is not active. Cleared by DMA reset.
ADC_DMAACT	[0]	R	DMA status: reads 1 if DMA is active.

Register 11.7. ADC_DMABEG: ADC DMA Begin Address Register

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

Address: 0x4000E018   Reset: 0x20000000

Bitname	Bitfield	Access	Description
ADC_DMABEG	[15:0]	RW	ADC buffer start address. Caution: this must be an even address - the least significant bit of this register is fixed at zero by hardware.

Register 11.8. ADC_DMASIZE: ADC DMA Buffer Size Register

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

Address: 0x4000E01C   Reset: 0x0

Bitname	Bitfield	Access	Description
ADC_DMASIZE_FIELD	[14:0]	RW	ADC buffer size. This is the number of 16-bit ADC conversion results the buffer can hold, not its length in bytes. (The length in bytes is twice this value.)

Register 11.9. ADC_DMACUR: ADC DMA Current Address Register

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

Address: 0x4000E020   Reset: 0x20000000

Bitname	Bitfield	Access	Description
ADC_DMACUR_FIELD	[15:1]	R	Current DMA address: the location that will be written next by the DMA.

Register 11.10. ADC_DMACNT: ADC DMA Count Register

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

Address: 0x4000E024   Reset: 0x0

Bitname	Bitfield	Access	Description
ADC_DMACNT_FIELD	[14:0]	R	DMA count: the number of 16-bit conversion results that have been written to the buffer.

Register 11.11. INT_ADCFLAG: ADC Interrupt Flag Register

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

Address: 0x4000A810   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_ADCOVF	[4]	RW	DMA buffer overflow interrupt pending.
INT_ADCSAT	[3]	RW	Gain correction saturation interrupt pending.
INT_ADCULDFULL	[2]	RW	DMA buffer full interrupt pending.
INT_ADCULDHALF	[1]	RW	DMA buffer half full interrupt pending.
INT_ADCDATA	[0]	RW	ADC_DATA register has data interrupt pending.

Register 11.12. INT_ADCCFG: ADC Interrupt Configuration Register

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

Address: 0x4000A850   Reset: 0x0

Bitname	Bitfield	Access	Description
INT_ADCOVF	[4]	RW	DMA buffer overflow interrupt enable.
INT_ADCSAT	[3]	RW	Gain correction saturation interrupt enable.
INT_ADCULDFULL	[2]	RW	DMA buffer full interrupt enable.
INT_ADCULDHALF	[1]	RW	DMA buffer half full interrupt enable.
INT_ADCDATA	[0]	RW	ADC_DATA register has data interrupt enable.

12. Trace Port Interface Unit (TPIU)

The EM358x integrates the standard ARM® Trace Port Interface Unit (TPIU). The TPIU receives a data stream from the on-chip trace data generated by the standard ARM® Instrument Trace Macrocell (ITM) and ARM® Embedded Trace Macrocell (ETM), buffers the data in a FIFO for the ITM and FIFO for the ETM, formats the data, and serializes the data to be sent off chip through alternate functions of the GPIO. Since the primary function of the TPIU is to provide a bridge between on-chip ARM system debug components and external GPIO, the TPIU itself does not generate data. Figure 12.1 illustrates the three primary components of the TPIU.
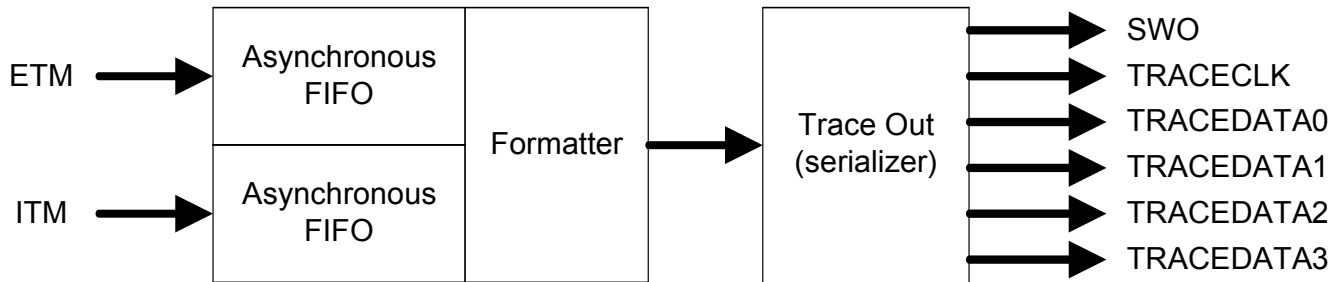
Figure 12.1. TPIU Block Diagram

The TPIU is composed of:

- Two asynchronous FIFOs: The asynchronous FIFOs receive a data stream generated by the ITM and ETM and enables the trace data to be sent off chip at a speed that is not dependent on the speed of the data source.
- Formatter: The formatter inserts source ID signals into the data packet stream so that trace data can be re-associated with its trace source.
- Trace Out: The trace out block serializes the data and sends it off chip by the proper alternate output GPIO functions.

The six pins available to the TPIU are:

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

Since these pins are alternate outputs of GPIO, refer to Chapter 7, GPIO, and in the *Ember EM358x Data Sheet,* Chapter 6, Pin Assignments, for complete pin descriptions and configurations.

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

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

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

13. Instrumentation Trace Macrocell (ITM)

The EM358x integrates the standard ARM® Instrumentation Trace Macrocell (ITM). The ITM is an application-driven trace source that supports printf style debugging to trace software events and emits diagnostic system information from the ARM® Data Watchpoint and Trace (DWT). Software using the ITM generates Software Instrumentation Trace (SWIT). In addition, the ITM provides coarse-grained timestamp functionality. The ITM emits trace information as packets, and these packets are sent to the Trace Port Interface Unit (TPIU). Three sources can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which the packets are output. The three sources, in decreasing order of priority, are:

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

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

For further information on the ITM, contact customer support for the ARM® Cortex™-M3 Technical Reference Manual, the ARM® CoreSightTM Components Technical Reference Manual, the ARM® v7-M Architecture Reference Manual, and the ARM®v7-M Architecture Application Level Reference Manual.

14. Embedded Trace Macrocell (ETM)

The EM358x integrates the standard ARM$^®$ Embedded Trace Macrocell (ETM) version 3.4.

The ETM is a powerful debug component that enables reconstruction of program execution. The ETM is designed as a high-speed, low-power debug tool that only supports instruction trace.

The ETM generates information that trace software tools use to reconstruct the execution of all or part of a program. The ETM can be configured in software to capture only select trace information and only after a specific sequence of conditions.

When the system is running, the ETM collects instruction data, compresses this information and delivers it off-chip in real-time for post processing. The ETM emits trace information to the Trace Port Interface Unit (TPIU). The TPIU combines the other sources of debug data (DWT and ITM), and outputs it to the trace pins

For further information on the ETM, contact customer support for the ARM$^®$ Cortex^{TM}-M3 Technical Reference Manual, the ARM$^®$ CoreSightTM Components Technical Reference Manual, the ARM$^®$ v7-M Architecture Reference Manual, and the ARM$^®$v7-M Architecture Application Level Reference Manual.

15. Data Watchpoint and Trace (DWT)

The EM358x integrates the standard ARM® Data Watchpoint and Trace (DWT). The DWT provides hardware support for profiling and debugging functionality. The DWT offers the following features:

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

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

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

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

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

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

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

16. Flash Patch and Breakpoint (FPB)

The EM358x integrates the standard ARM® Flash Patch and Breakpoint (FPB). The FPB implements hardware breakpoints. The FPB also provides support for remapping of specific instruction or literal locations from flash memory to an address in RAM memory. The FPB contains:

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

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

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

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

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

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

17. Serial Wire and JTAG (SWJ) Interface

The EM358x includes a standard Serial Wire and JTAG (SWJ) Interface. The SWJ is the primary debug and programming interface of the EM358x. The SWJ gives debug tools access to the internal buses of the EM358x, and allows for non-intrusive memory and register access as well as CPU halt-step style debugging. Therefore, any design implementing the EM358x should make the SWJ signals readily available.

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

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

The ARM$^®$ CoreSight^{TM} Debug Access Port (DAP) comprises the Serial Wire and JTAG Interface (SWJ). As illustrated in Figure 17.1, the DAP includes two primary components: a debug port (the SWJ-DP) and an access port (the AHB-AP). The SWJ-DP provides external debug access, while the AHB-AP provides internal bus access. An external debug tool connected to the EM358x's debug pins communicates with the SWJ-DP. The SWJ-DP then communicates with the AHB-AP. Finally, the AHB-AP communicates on the internal bus.
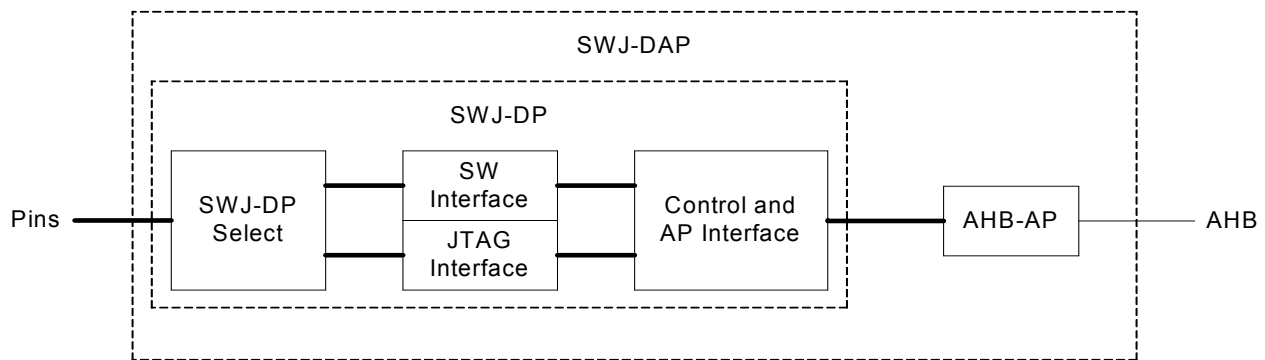

Figure 17.1. SWJ Block Diagram

Serial Wire and JTAG share five pins:

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

Note: The SWJ pins are forced functions, and their corresponding GPIO_PxCFGH/L configurations are overridden when the EM358x resets. An application may reclaim all four of the SWJ GPIOs (PC0, PC2, PC3, and PC4) by disabling the SWJ interface or reclaim just the JTAG specific GPIOs (PC0, PC2, PC3). When using these pins as GPIOs remember that the chip resets in JTAG mode and the forced functions will be active until software reconfigures them.

Since these pins can be repurposed, refer to Section 7.3, Forced Functions, in Chapter 7, GPIO, and, in the *Ember EM358x Data Sheet*, Chapter 6, Pin Assignments, for complete pin descriptions and configurations.

For further information on the SWJ, contact customer support for Application Notes and ARM$^®$ CoreSight^{TM} documentation.

APPENDIX A—REGISTER ADDRESS TABLE

BLOCK	CM_LV	40004000 - 4000403C CM_LV			
Address	**Name**	**Type**	**Reset**	**Description**	
40004038	PERIPHERAL_DISABLE	RW	0	Peripheral Disable Register	
4000403C	RAM_RETAIN	RW	FFFF	RAM Retention Register	

BLOCK	INTERRUPTS	4000A000 - 4000AFFF Interrupts			
Address	**Name**	**Type**	**Reset**	**Description**	
4000A800	INT_TIM1FLAG	RW	0	Timer 1 Interrupt Flag Register	
4000A804	INT_TIM2FLAG	RW	0	Timer 2 Interrupt Flag Register	
4000A808	INT_SC1FLAG	RW	0	Serial Controller 1 Interrupt Flag Register	
4000A80C	INT_SC2FLAG	RW	0	Serial Controller 2 Interrupt Flag Register	
4000A810	INT_ADCFLAG	RW	0	ADC Interrupt Flag Register	
4000A814	INT_GPIOFLAG	RW	0	GPIO Interrupt Flag Register	
4000A818	INT_TIM1MISS	RW	0	Timer 1 Missed Interrupt Register	
4000A81C	INT_TIM2MISS	RW	0	Timer 2 Missed Interrupts Register	
4000A820	INT_MISS	RW	0	Top-Level Missed Interrupts Register	
4000A840	INT_TIM1CFG	RW	0	Timer 1 Interrupt Configuration Register	
4000A844	INT_TIM2CFG	RW	0	Timer 2 Interrupt Configuration Register	
4000A848	INT_SC1CFG	RW	0	Serial Controller 1 Interrupt Configuration Register	
4000A84C	INT_SC2CFG	RW	0	Serial Controller 2 Interrupt Configuration Register	
4000A850	INT_ADCCFG	RW	0	ADC Interrupt Configuration Register	
4000A854	SC1_INTMODE	RW	0	Serial Controller 1 Interrupt Mode Register	
4000A858	SC2_INTMODE	RW	0	Serial Controller 2 Interrupt Mode Register	
4000A860	GPIO_INTCFGA	RW	0	GPIO Interrupt A Configuration Register	
4000A864	GPIO_INTCFGB	RW	0	GPIO Interrupt B Configuration Register	
4000A868	GPIO_INTCFGC	RW	0	GPIO Interrupt C Configuration Register	
4000A86C	GPIO_INTCFGD	RW	0	GPIO Interrupt D Configuration Register	

BLOCK	GPIO	4000B000 - 4000BFFF General Purpose IO			
Address	**Name**	**Type**	**Reset**	**Description**	
4000B000	GPIO_PACFGL	RW	4444	Port A Configuration Register (Low)	
4000B004	GPIO_PACFGH	RW	4444	Port A Configuration Register (High)	
4000B008	GPIO_PAIN	RW	0	Port A Input Data Register	
4000B00C	GPIO_PAOUT	RW	0	Port A Output Data Register	
4000B010	GPIO_PASET	RW	0	Port A Output Set Register	
4000B014	GPIO_PACLR	RW	0	Port A Output Clear Register	
4000B200	GPIO_PBCFGL	RW	4444	Port B Configuration Register (Low)	
4000B204	GPIO_PBCFGH	RW	4444	Port B Configuration Register (High)	
4000B208	GPIO_PBIN	RW	0	Port B Input Data Register	
4000B20C	GPIO_PBOUT	RW	0	Port B Output Data Register	
4000B210	GPIO_PBSET	RW	0	Port B Output Set Register	
4000B214	GPIO_PBCLR	RW	0	Port B Output Clear Register	
4000B400	GPIO_PCCFGL	RW	4444	Port C Configuration Register (Low)	
4000B404	GPIO_PCCFGH	RW	4444	Port C Configuration Register (High)	
4000B408	GPIO_PCIN	RW	0	Port C Input Data Register	
4000B40C	GPIO_PCOUT	RW	0	Port C Output Data Register	
4000B410	GPIO_PCSET	RW	0	Port C Output Set Register	
4000B414	GPIO_PCCLR	RW	0	Port C Output Clear Register	
4000BC00	GPIO_DBGCFG	RW	10	GPIO Debug Configuration Register	
4000BC04	GPIO_DBGSTAT	R	0	GPIO Debug Status Register	
4000BC08	GPIO_PAWAKE	RW	0	Port A Wakeup Monitor Register	
4000BC0C	GPIO_PBWAKE	RW	0	Port B Wakeup Monitor Register	
4000BC10	GPIO_PCWAKE	RW	0	Port C Wakeup Monitor Register	
4000BC20	GPIO_IRQCSEL	RW	F	Interrupt C Select Register	
4000BC24	GPIO_IRQDSEL	RW	10	Interrupt D Select Register	
4000BC28	GPIO_WAKEFILT	RW	0	GPIO Wakeup Filtering Register	

BLOCK	SERIAL			4000C000 - 4000CFFF Serial Controllers	
Address	Name	Type	Reset	Description
4000C000	SC2_RXBEGA	RW	20000000	Receive DMA Begin Address Register A
4000C004	SC2_RXENDA	RW	20000000	Receive DMA End Address Register A
4000C008	SC2_RXBEGB	RW	20000000	Receive DMA Begin Address Register B
4000C00C	SC2_RXENDB	RW	20000000	Receive DMA End Address Register B
4000C010	SC2_TXBEGA	RW	20000000	Transmit DMA Begin Address Register A
4000C014	SC2_TXENDA	RW	20000000	Transmit DMA End Address Register A
4000C018	SC2_TXBEGB	RW	20000000	Transmit DMA Begin Address Register B
4000C01C	SC2_TXENDB	RW	20000000	Transmit DMA End Address Register B
4000C020	SC2_RXCNTA	R	0	Receive DMA Count Register A
4000C024	SC2_RXCNTB	R	0	Receive DMA Count Register B
4000C028	SC2_TXCNT	R	0	Transmit DMA Count Register
4000C02C	SC2_DMASTAT	R	0	Serial DMA Status Register
4000C030	SC2_DMACTRL	RW	0	Serial DMA Control Register
4000C034	SC2_RXERRA	R	0	DMA First Receive Error Register A
4000C038	SC2_RXERRB	R	0	DMA First Receive Error Register B
4000C03C	SC2_DATA	RW	0	Serial Data Register
4000C040	SC2_SPISTAT	R	0	SPI Status Register
4000C044	SC2_TWISTAT	R	0	TWI Status Register
4000C04C	SC2_TWICTRL1	RW	0	TWI Control Register 1
4000C050	SC2_TWICTRL2	RW	0	TWI Control Register 2
4000C054	SC2_MODE	RW	0	Serial Mode Register
4000C058	SC2_SPICFG	RW	0	SPI Configuration Register
4000C060	SC2_RATELIN	RW	0	Serial Clock Linear Prescaler Register
4000C064	SC2_RATEEXP	RW	0	Serial Clock Exponential Prescaler Register
4000C070	SC2_RXCNTSAVED	R	0	Saved Receive DMA Count Register
4000C800	SC1_RXBEGA	RW	20000000	Receive DMA Begin Address Register A
4000C804	SC1_RXENDA	RW	20000000	Receive DMA End Address Register A
4000C808	SC1_RXBEGB	RW	20000000	Receive DMA Begin Address Register B

4000C80C	SC1_RXENDB	RW	20000000	Receive DMA End Address Register B
4000C810	SC1_TXBEGA	RW	20000000	Transmit DMA Begin Address Register A
4000C814	SC1_TXENDA	RW	20000000	Transmit DMA End Address Register A
4000C818	SC1_TXBEGB	RW	20000000	Transmit DMA Begin Address Register B
4000C81C	SC1_TXENDB	RW	20000000	Transmit DMA End Address Register B
4000C820	SC1_RXCNTA	R	0	Receive DMA Count Register A
4000C824	SC1_RXCNTB	R	0	Receive DMA Count Register B
4000C828	SC1_TXCNT	R	0	Transmit DMA Count Register
4000C82C	SC1_DMASTAT	R	0	Serial DMA Status Register
4000C830	SC1_DMACTRL	RW	0	Serial DMA Control Register
4000C834	SC1_RXERRA	R	0	DMA First Receive Error Register A
4000C838	SC1_RXERRB	R	0	DMA First Receive Error Register B
4000C83C	SC1_DATA	RW	0	Serial Data Register
4000C840	SC1_SPISTAT	R	0	SPI Status Register
4000C844	SC1_TWISTAT	R	0	TWI Status Register
4000C848	SC1_UARTSTAT	R	40	UART Status Register
4000C84C	SC1_TWICTRL1	RW	0	TWI Control Register 1
4000C850	SC1_TWICTRL2	RW	0	TWI Control Register 2
4000C854	SC1_MODE	RW	0	Serial Mode Register
4000C858	SC1_SPICFG	RW	0	SPI Configuration Register
4000C85C	SC1_UARTCFG	RW	0	UART Configuration Register
4000C860	SC1_RATELIN	RW	0	Serial Clock Linear Prescaler Register
4000C864	SC1_RATEEXP	RW	0	Serial Clock Exponential Prescaler Register
4000C868	SC1_UARTPER	RW	0	UART Baud Rate Period Register
4000C86C	SC1_UARTFRAC	RW	0	UART Baud Rate Fractional Period Register
4000C870	SC1_RXCNTSAVED	R	0	Saved Receive DMA Count Register

BLOCK	ADC		4000D000 - 4000DFFF Analog to Digital Converter		
Address	**Name**	**Type**	**Reset**	**Description**	
4000E000	ADC_DATA	R	0	ADC Data Register	
4000E004	ADC_CFG	RW	00001800	ADC Configuration Register	
4000E008	ADC_OFFSET	RW	0000	ADC Offset Register	
4000E00C	ADC_GAIN	RW	8000	ADC Gain Register	
4000E010	ADC_DMACFG	RW	0	ADC DMA Configuration Register	
4000E014	ADC_DMASTAT	R	0	ADC DMA Status Register	
4000E018	ADC_DMABEG	RW	20000000	ADC DMA Begin Address Register	
4000E01C	ADC_DMASIZE	RW	0	ADC DMA Buffer Size Register	
4000E020	ADC_DMACUR	R	20000000	ADC DMA Current Address Register	
4000E024	ADC_DMACNT	R	0	ADC DMA Count Register	

BLOCK	TIM1		4000E000 - 4000EFFF General Purpose Timer 1		
Address	**Name**	**Type**	**Reset**	**Description**	
4000F000	TIM1_CR1	RW	0	Timer 1 Control Register 1	
4000F004	TIM1_CR2	RW	0	Timer 1 Control Register 2	
4000F008	TIM1_SMCR	RW	0	Timer 1 Slave Mode Control Register	
4000F014	TIM1_EGR	RW	0	Timer 1 Event Generation Register	
4000F018	TIM1_CCMR1	RW	0	Timer 1 Capture/Compare Mode Register 1	
4000F01C	TIM1_CCMR2	RW	0	Timer 1 Capture/Compare Mode Register 2	
4000F020	TIM1_CCER	RW	0	Timer 1 Capture/Compare Enable Register	
4000F024	TIM1_CNT	RW	0	Timer 1 Counter Register	
4000F028	TIM1_PSC	RW	0	Timer 1 Prescaler Register	
4000F02C	TIM1_ARR	RW	FFFF	Timer 1 Auto-Reload Register	
4000F034	TIM1_CCR1	RW	0	Timer 1 Capture/Compare Register 1	
4000F038	TIM1_CCR2	RW	0	Timer 1 Capture/Compare Register 2	
4000F03C	TIM1_CCR3	RW	0	Timer 1 Capture/Compare Register 3	

| 4000F040 | TIM1_CCR4 | RW | 0 | Timer 1 Capture/Compare Register 4 |
| 4000F050 | TIM1_OR | RW | 0 | Timer 1 Option Register |

BLOCK	TIM2		4000F000 - 4000FFFF General Purpose Timer 2		
Address	**Name**	**Type**	**Reset**	**Description**	
40010000	TIM2_CR1	RW	0	Timer 2 Control Register 1	
40010004	TIM2_CR2	RW	0	Timer 2 Control Register 2	
40010008	TIM2_SMCR	RW	0	Timer 2 Slave Mode Control Register	
40010014	TIM2_EGR	RW	0	Timer 2 Event Generation Register	
40010018	TIM2_CCMR1	RW	0	Timer 2 Capture/Compare Mode Register 1	
4001001C	TIM2_CCMR2	RW	0	Timer 2 Capture/Compare Mode Register 2	
40010020	TIM2_CCER	RW	0	Timer 2 Capture/Compare Enable Register	
40010024	TIM2_CNT	RW	0	Timer 2 Counter Register	
40010028	TIM2_PSC	RW	0	Timer 2 Prescaler Register	
4001002C	TIM2_ARR	RW	FFFF	Timer 2 Auto-Reload Register	
40010034	TIM2_CCR1	RW	0	Timer 2 Capture/Compare Register 1	
40010038	TIM2_CCR2	RW	0	Timer 2 Capture/Compare Register 2	
4001003C	TIM2_CCR3	RW	0	Timer 2 Capture/Compare Register 3	
40010040	TIM2_CCR4	RW	0	Timer 2 Capture/Compare Register 4	
40010050	TIM2_OR	RW	0	Timer 2 Option Register	

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

Document Change List

Revision 0.1 to Revision 0.2

- Information relocated from *Ember EM358 Data Sheet*
- Updated GPT, ADC, and Interrupt registers
- Removed references to serial controllers 3 and 4, and made associated changes to serial controller text.

Revision 0.2 to 0.3

- Introduced EM358x and the 5 variants: EM3581, EM3582, EM3585, EM3586 and EM3588

Revision 0.3 to 0.4

- Addition of EM3587 variant
- Correction of some references to EM358

Revision 0.4 to 1.0

- Update to 1.0 with characterization data for Full Production

Revision 1.0 to 1.1

- Revised for Thread
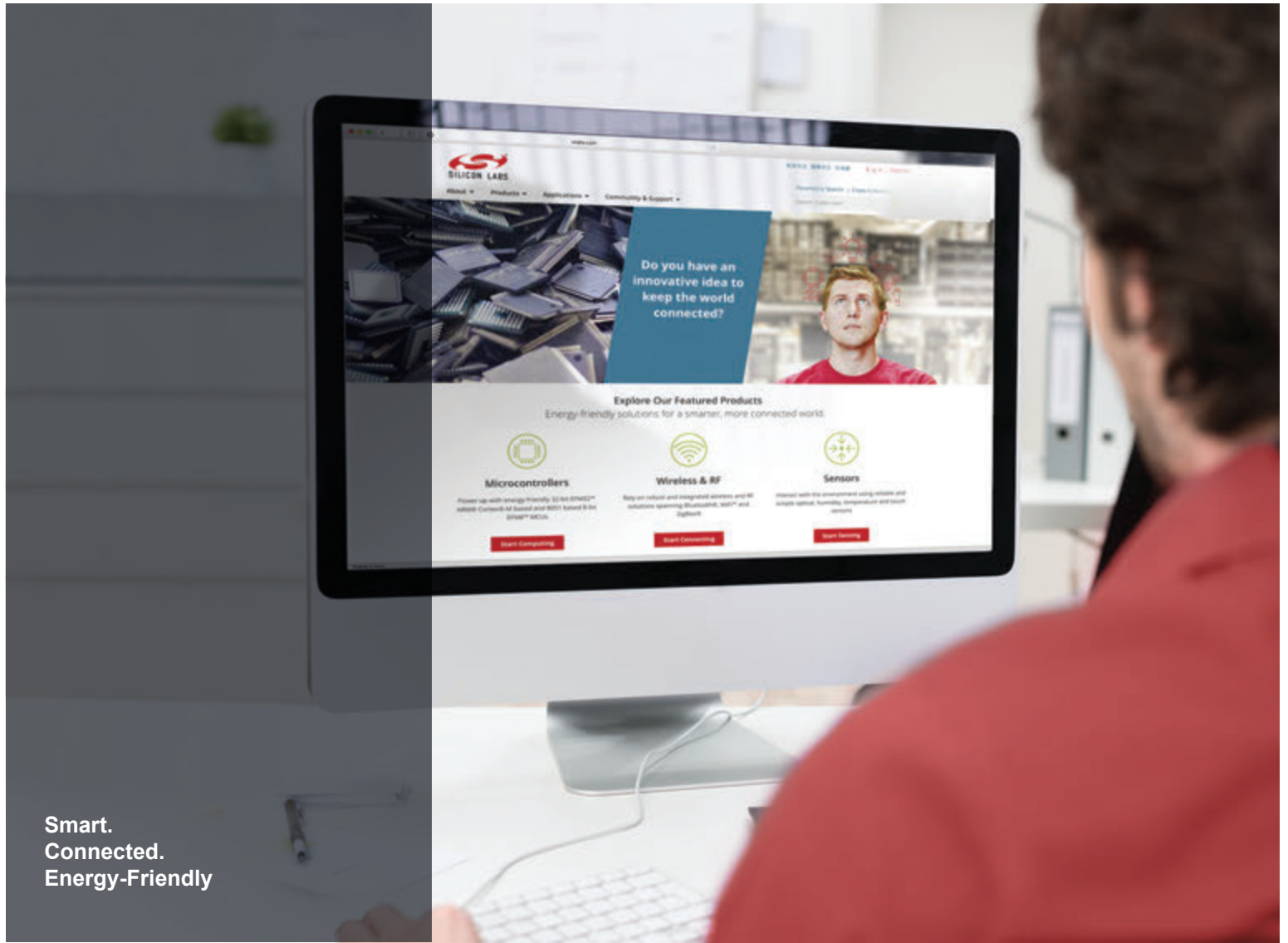
SILICON LABS

**Smart.
Connected.
Energy-Friendly**

Products	Quality	Support and Community
www.silabs.com/products	www.silabs.com/quality	community.silabs.com
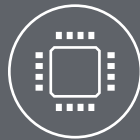
Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com