EFM32®

... the world's most energy friendly microcontrollers

Pulse Counter

AN0024 - Application Note

Introduction

This application note describes how to configure and use the different modes in the EFM32 Pulse Counter, select clock sources and use the available interrupts to achieve high energy efficiency.

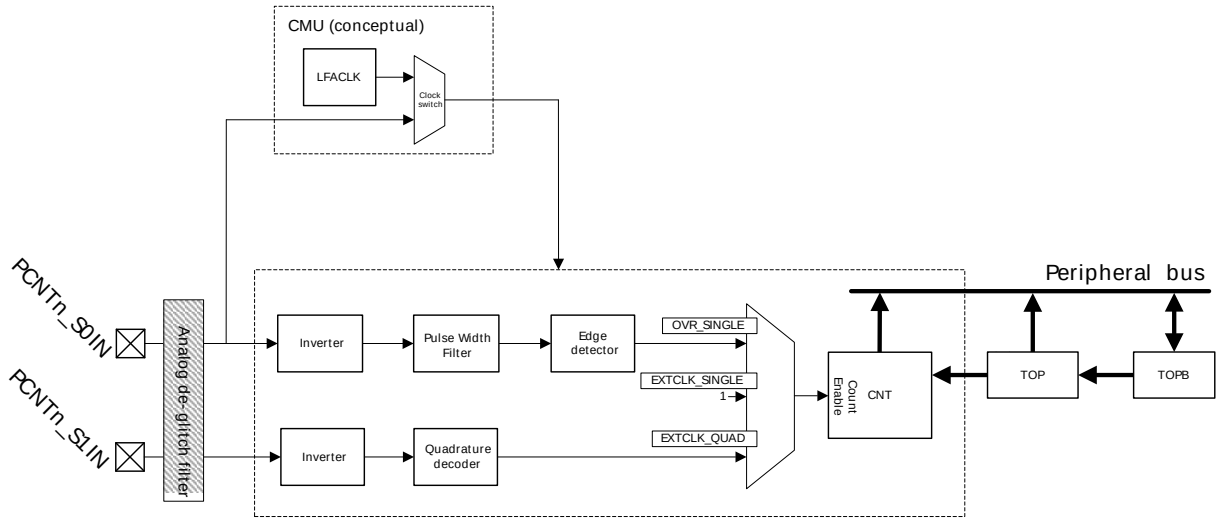
This application note includes:

- This PDF document
- Source files (zip)
  - Example C-code
  - Multiple IDE projects

1 Overview

The Pulse Counter (PCNT) can be used for counting incoming pulses on a single input or to decode quadrature encoded inputs. It can run from the internal LFACLK (EM0-EM2) while counting pulses on the PCNTn_S0IN pin or using this pin as an external clock source (EM0-EM3) that runs both the PCNT counter and register access.

Figure 1.1. Pulse Counter Overview



The peripheral incorporates an 8-bit up/down-counter to keep track of the incoming pulses or rotations. It is possible to generate an interrupt after a specific number of pulses (or rotations) and there is also a change in direction interrupt available (quadrature decoder mode only). This eliminates the need for timing or IO interrupts and CPU processing to measure pulse widths.

2 Pulse Counter Modes

The PCNT can operate in single oversampling mode (OVSSINGLE), externally clocked single input counter mode (EXTCLKSINGLE) and externally clocked quadrature decoder mode (EXTCLKQUAD).

- In OVSSINGLE mode the input on PCNTn_S0IN is sampled by the LFACLK and PCNTn_S1IN is ignored
- In EXTCLKSINGLE the PCNTn_S0IN input clocks both the PCNT counter and register access. PCNTn_S1IN is also ignored in this mode
- In EXTCLKQUAD the PCNTn_S0IN input clocks the register access as well and is used to sample PCNTn_S1IN in order to decode the quadrature signal

2.1 Functions

The operational mode and general initialization can be set using the following function from the emlib:

```
void PCNT_Init(PCNT_TypeDef *pcnt, PCNT_Init_TypeDef *init)
```

Using this function the user will be able to configure the following parameters:

- Operational mode
- Initial counter value (maximum of 0xFF)
- Initial top value (maximum of 0xFF)
- Polarity of incoming edge
- Count up or down
- Filter enable

If the user only wants to change the operational mode the function `void PCNT_Enable(PCNT_TypeDef *pcnt, PCNT_Mode_TypeDef mode)` can be used. It does not do any configuration, only sets the operational mode. This is normally only required after initialization is done and if not done as part of the initialization or if requiring to disable/reenable the PCNT.

2.2 Clock Sources

There are two selectable clock sources for the PCNT, the 32kHz LFA clock or an external clock through PCNTn_S0IN pin. Clock selection is done using the CMU_PCNTCTRL register in the CMU module and the LE interface clock must also be enabled in addition to the peripheral clock. The external clock is used to clock the counter in EXTCLKSINGLE mode and to sample PCNTn_S1IN in EXTCLKQUAD mode.

When changing clock source the PCNT Clock Domain Reset bit (RSTEN in the PCNTn_CTRL register) should be set. If changing to an external source the clock pin has to be enabled as input in the GPIO module prior to deasserting RSTEN. Changing clock source without enabling RSTEN results in undefined behaviour. The clock source can be selected using the initialization function for the PCNT in the emlib `void PCNT_Init(PCNT_TypeDef *pcnt, PCNT_Init_TypeDef *init)` which also handles automatically the RSTEN assertion.

The pins used by the PCNT must be properly configured as inputs in the GPIO module and routed in the PCNT using the ROUTE register. Configuring the pins is covered in AN0012 GPIO.

2.3 Single Input Oversampling Mode

Single input oversampling mode can also be enabled by writing OVSSINGLE (0x1) to the MODE bitfield in the PCNTn_CTRL register. The PCNTn_S0IN is the only observed input in this mode and is sampled

by the LFACLK. The number of detected positive or negative edges on this pin appears in PCNTn_CNT. The counter can count either up or down and the digital pulse width filter is available in this mode and can be enabled using the FILT bit in PCNTn_CTRL. Count direction and edge polarity of the incoming signal can be changed using CNTDIR and EDGE bits in PCNTn_CTRL.

2.3.1 Software Examples

There is one software example named `pcnt_count` that demonstrates OVSSINGLE mode and can be used for both STKs and DK. In this example the pulses are generated by making a connection between PC4(PC13 on the Tiny Gecko STK) and PC5 (P4.7 and P4.8 on the DK protoboard). PC4 is configured as input with a pull down to tie it to ground and avoid having a floating voltage. PC5 is configured as output high so when making a connection between them PC4 will have a high level also and thus generating a pulse. Due to mechanical friction and bouncing each contact can generate more than one pulse. The overflow interrupts are used in this example to update the LCD with the number of pulses.

The top value is initially set to 10 and then is increased by 10 on each overflow. The counter gets the previous top value because it goes to 0 after an overflow. The LCD will update when a number of pulses multiple of 10 is generated. The counting direction change interrupt is not used in this example.

2.4 Externally Clocked Single Input Counter Mode

Externally clocked single input counter mode can also be enabled by writing EXTCLKSINGLE (0x2) to the MODE bitfield in the PCNTn_CTRL register. The external pin clock source must be configured in the CMU_PCNTCTRL register in the CMU. Positive edges on the PCNTn_S0IN are used to clock the counter and PCNTn_S1IN is ignored in this mode. Given that the LFACLK is not used the PCNT can operate in EM3. The counter can be configured to count down or up using the DIR bit in PCNTn_CTRL register, but the digital pulse width filter is not available. An alternative is using the analog glitch suppression filter in the GPIO pads to remove some unwanted noise. That configuration has to be done in the GPIO module when enabling the pins as inputs for the PCNT. The EDGE bit in PCNTn_CTRL has no effect in this mode where only positive signal edges are counted.

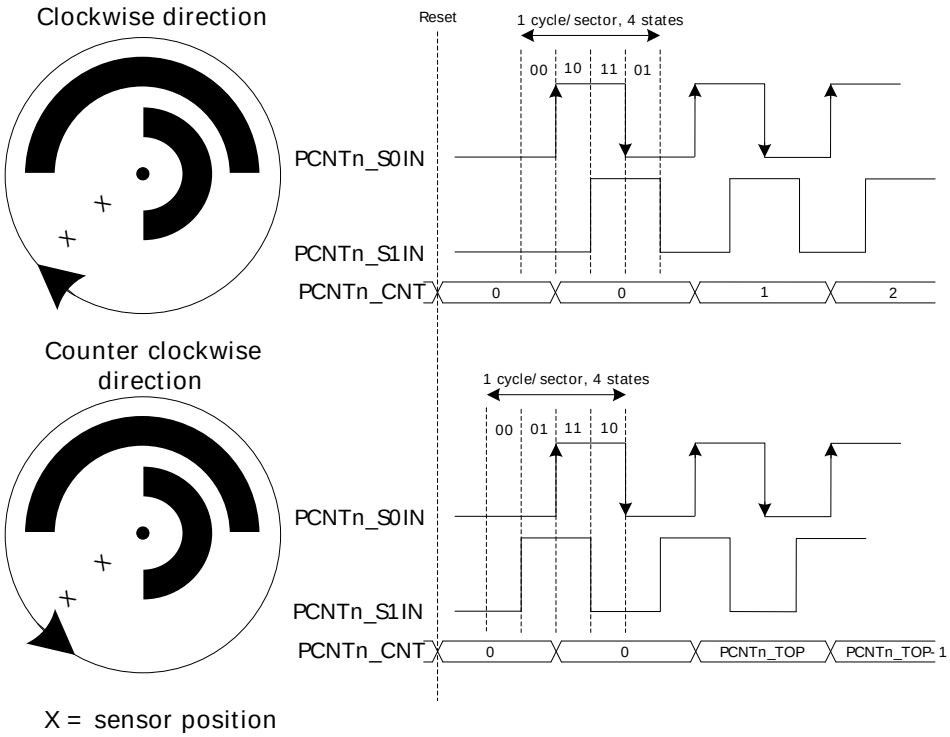
2.4.1 Software Examples

The software example from Section 2.3.1 (p. 4) can also be used to demonstrate this mode. To do so the `mode` parameter in `pcntInit` structure should be changed from `pcntModeOvsSingle` to `pcntModeExtSingle`. The only difference to the previous example is that the PCNT only starts counting on the 4th pulse. This is due to the required synchronization described in Section 4.2 (p. 8)

2.5 Externally Clocked Quadrature Decoder Mode

This mode is enabled by writing EXTCLKQUAD (0x3) to the MODE field in PCNT_CTRL register. The external pin clock source must be configured in the CMU_PCNTCTRL register in the CMU. Both edges of the PCNTn_S0IN pin are used to sample PCNTn_S1IN in order to decode the quadrature code. The LFACLK is not used in this mode enabling operation down to EM3. The direction of the counter can be inverted using the EDGE bit in PCNTn_CTRL register and the digital pulse width filter is not available in this mode. A quadrature coded signal contains information about the relative speed and direction of a rotating shaft as illustrated by Figure 2.1 (p. 5)

Figure 2.1. Quadrature Decoding



If PCNTn_S0IN leads PCNTn_S1IN in phase, the direction is clockwise, and if it lags in phase the direction is counter-clockwise. Although the direction is automatically detected it may be inverted by writing 1 to the EDGE bit in PCNTn_CTRL register. The counter direction may be read from the DIR bit in the PCNTn_STATUS register.

2.5.1 Software Examples

There is one software example named `pcnt_quad` that demonstrates EXTCLKQUAD mode and can be used for both STK and DK. To simulate the quadrature encoded signal pins PC4 and PC5 (P4.7 and P4.8 on the DK protoboard) are configured as inputs for the PCNT inputs and pins PC6 and PC7 (P4.9 and P4.10) are configured as output high. These should be connected to the input pins to generate the quadrature coded signal in one of two different cycles:

For the Tiny Gecko STK the PCNT inputs are on PC13 and PC14. Exchange PC4, PC5 with PC13, PC14 in the following two cycles to generate the quadrature signals.

The following cycle will make the PCNT count up:

- Connect PC4 to PC6
- Connect PC5 to PC7 and keep PC4 and PC6 connected
- Disconnect PC4 from PC6
- Disconnect PC5 from PC7

The next cycle will make the PCNT count down:

- Connect PC5 to PC7
- Connect PC4 to PC6 and keep PC5 and PC7 connected
- Disconnect PC5 from PC7
- Disconnect PC4 from PC6

The counter only starts counting on the 4th cycle. This is due to the required synchronization described in Section 4.2 (p. 8). In this example both counter direction (UP or DOWN) and value are displayed on the LCD. When the direction changes a direction change interrupt is generated and the LCD is updated with the new direction and current counter value. The LCD has START and 0 written at the beginning and is only updated after a direction change.

3 Interrupts

The interrupts generated by PCNT use the PCNTn_INT interrupt vector. Interrupts are set using the PCNTn_IFS register, cleared in the PCNTn_IFC register and read from the PCNTn_IF register. There are also emlib functions to handle these registers:

- `void PCNT_IntEnable(PCNT_TypeDef *pcnt, uint32_t flags)` for enabling interrupts
- `PCNT_IntClear(PCNT_TypeDef *pcnt, uint32_t flags)` for clearing interrupts
- `uint32_t PCNT_IntGet(PCNT_TypeDef *pcnt)` for reading interrupts

The interrupts can also be disabled using the function `void PCNT_IntDisable(PCNT_TypeDef *pcnt, uint32_t flags)`

3.1 Underflow and Overflow Interrupts

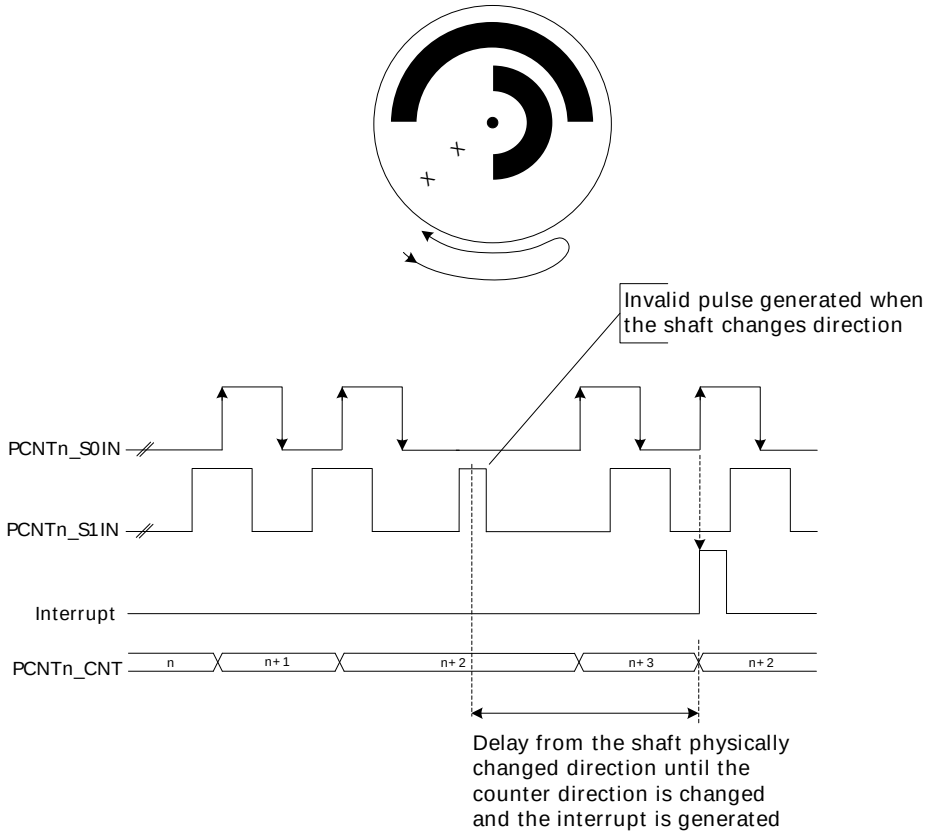
The underflow interrupt flag (UF) is set when the counter counts down from 0. I.e. when the value of the counter is 0 and a new pulse is received. The PCNTn_CNT register is loaded with the PCNTn_TOP value after this event.

The overflow interrupt flag (OF) is set when the counter counts up from the PCNTn_TOP (reload) value. I.e. if PCNTn_CNT = PCNTn_TOP and a new pulse is received. The PCNTn_CNT register is loaded with the value 0 after this event.

3.2 Direction Change Interrupt

The direction change interrupt flag (DIRCNG) is set when the direction of the quadrature code changes. The behavior of this interrupt is illustrated by Figure 3.1 (p. 7)

Figure 3.1. Direction Change Interrupt



4 Register access

4.1 Writing to PCNTn_TOP and PCNTn_CNT

Since the pulse counter is part of the low frequency domain, or externally clocked, and thereby asynchronous to the high frequency domain. All register accesses must be synchronized when writing, please see the reference manual, chapter: *Access to Low Energy Peripherals (Asynchronous Registers)* for more information about this.

The emlib includes a function to load both TOP and COUNT registers:

```
void PCNT_CounterTopSet(PCNT_TypeDef *pcnt, uint32_t count, uint32_t top)
```

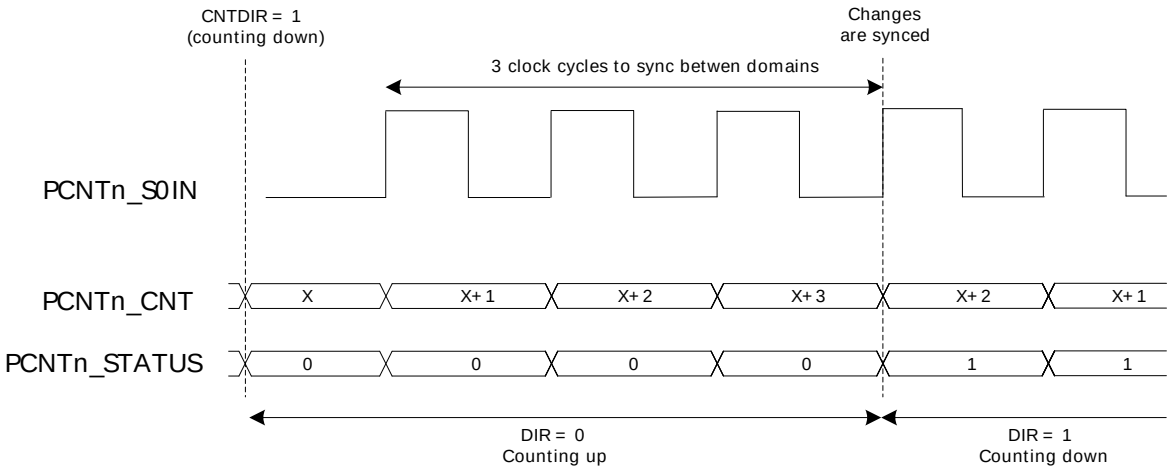
In this function the LTOPBIM command is always issued since it has no effect in revision C but is necessary in revisions A/B. If running the PCNT from an external clock source (EXTCLKSINGLE or EXTCLKQUAD modes) it is advisable not to use this function. It contains several sync cycles that can cause the program to stall if the clock is not present or delay the execution if the clock is slow. Before writing to PCNTn_TOPB, PCNTn_CMD or PCNTn_CTRL a sync check should be done to make sure these registers were synchronized to the low frequency domain.

When using the initialization function `void PCNT_Init(PCNT_TypeDef *pcnt, PCNT_Init_TypeDef *init)` the counter value is not written for the external clock modes (EXTCLKSINGLE and EXTCLKQUAD). The counter value must first be written to TOP and then to CNT and it is necessary to sync the register access between the high and low frequency domains. In External clock modes there is no guarantee that the clock is present so waiting for syncs could cause the program to stall.

4.2 Synchronizing in External Clock Modes

When writing to PCNTn_TOPB, PCNTn_CMD or PCNTn_CTRL registers these need to be synchronized from the high frequency domain to the low frequency domain. The changes will not take effect until the synchronization is done and it takes 3 low frequency clock cycles to synchronize between the two domains. When using the PCNT in an external clock mode (EXTCLKSINGLE or EXTCLKQUAD), it also takes 3 cycles on the external clock to sync the registers. This should be taken into account when the user writes the PCNT registers in run-time. Figure 4.1 (p. 8) illustrates the sync in EXTCLKSINGLE mode after changing the CNTDIR bit in PCNT_CTRL register.

Figure 4.1. Register Sync in EXTCLKSINGLE



The PCNT will need 3 clock cycles (or pulses) on the external clock (PCNTn_S0IN) to sync before the changes take effect. This means that the PCNT counts up for 3 more cycles before starting to count down. In this example the counter will be affected by 6 extra counts (2 x SYNC_CYCLES).

4.2.1 Initializing in External Clock Mode

When initializing the PCNT with one of the external clock modes (EXTCLKSINGLE or EXTCLKQUAD) the counting can start on the third or fourth pulse depending on how the initialization is done. There are two possible initialization sequences.

- Single write to PCNTn_CTRL with chosen configuration and PCNT clock domain reset enable (RSTEN)
- Wait for sync by polling the CTRL bit in the PCNTn_SYNCBUSY register
- Select external clock source
- Release the reset by writing 0 to RSTEN in PCNTn_CTRL
- Write PCNTn_TOPB
- Issue LTOPBIM command in PCNTn_CMD (EFM32G revisions A/B only)

Using this sequence the configuration will be synchronized simultaneously with the RSTEN assertion. When clearing RSTEN the PCNT is already configured to the correct mode. The reset is synchronously released two PCNT clock edges after RSTEN being cleared and thus the PCNT will start counting on the third pulse.

This initialization has however a drawback. The TOP value has to be written after clearing RSTEN. The PCNTn_TOP reset value is restored after clearing RSTEN so writing it before will have no effect. After reset it will take 4 or 3 clock cycles (revisions A/B and revision C and onward EFM32G and other EFM32 parts respectively) to synchronize PCNTn_TOP so when using this sequence the PCNT starts counting with a TOP value of 0xFF. If count down is selected the third pulse (first countable pulse) will make PCNTn_CNT go from 0 to 0xFF because the new PCNTn_TOP value has not been synchronized yet.

The initialization sequence can be done differently to make sure that PCNTn_TOP is synchronized before or at the same time as PCNTn_CTRL after releasing reset.

- Write to PCNTn_CTRL enabling PCNT clock domain reset enable (RSTEN)
- Wait for sync by polling the CTRL bit in the PCNTn_SYNCBUSY register
- Select external clock source
- Release the reset by writing 0 to RSTEN in PCNTn_CTRL
- Write PCNTn_TOPB
- Issue LTOPBIM command in PCNTn_CMD (EFM32G revisions A/B only)
- Wait for PCNTn_TOPB and PCNTn_CMD synchronization (EFM32G revisions A/B only)
- Write PCNTn_CTRL with the chosen configurations

The emlib function `void PCNT_Init(PCNT_TypeDef *pcnt, PCNT_Init_TypeDef *init)` implements this solution but without the wait for synchronization because the state of the external clock is unknown.

If count down is selected PCNTn_CNT will go to the new PCNTn_TOP value instead of 0xFF. However in EFM32G revisions A/B chips there is the need for one more clock cycle as described in Section 4.1 (p. 8) . For these chips the LTOPBIM command has to be issued after writing PCNTn_TOPB and both should be synchronized before writing and synchronizing PCNTn_CTRL. To avoid using the pulses for synchronization the user can configure PCNTn_S0IN as push pull and generate the pulses by software. This makes sure that the first external pulse is counted by the PCNT

5 Revision History

5.1 Revision 1.07

2013-09-03

New cover layout

5.2 Revision 1.06

2013-08-07

Replaced hard coded interrupt flags and bug in interrupt handler.

5.3 Revision 1.05

2013-05-08

Added software projects for ARM-GCC and Atollic TrueStudio.

Removed section about issues with early Gecko revisions, as this is errata material.

5.4 Revision 1.04

2012-11-12

Adapted software projects to new kit-driver and bsp structure.

Added projects for Tiny and Giant Gecko STKs.

5.5 Revision 1.03

2012-04-20

Adapted software projects to new peripheral library naming and CMSIS_V3.

5.6 Revision 1.02

2011-10-21

Updated IDE project paths with new kits directory.

5.7 Revision 1.01

2011-05-18

Updated projects to align with new bsp version.

5.8 Revision 1.00

2010-12-02

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

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

A.2 Trademark Information

Silicon Laboratories Inc., Silicon Laboratories, the Silicon Labs logo, Energy Micro, EFM, EFM32, EFR, logo and combinations thereof, and others are the registered trademarks or trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

B Contact Information

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

Please visit the Silicon Labs Technical Support web page:
http://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.

Table of Contents

List of Figures

silabs.com


ZERO
ARM Cortex-M0+
TINY
ARM Cortex-M3
GECKO
ARM Cortex-M3
LEOPARD
ARM Cortex-M3
GIANT
ARM Cortex-M3
WONDER
ARM Cortex-M4