CE210489 – PSoC 4 Low Power CapSense Proximity

Objective

This code example demonstrates a low-power proximity sensor and CapSense® tuner using the PSoC Creator™ CapSense Component with PSoC® 4.

Requirements

Tool: PSoC Creator 4.2

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: All PSoC 4 family devices that have Serial Communication Blocks and CapSense capabilities

Related Hardware: CY8CKIT-042-BLE-A Bluetooth® Low Energy 4.2 Compliant Pioneer Kit

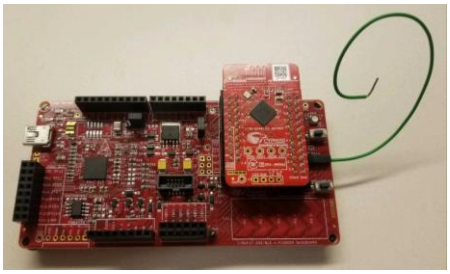
Overview

This code example contains a PSoC Creator project that shows how to use a proximity sensor with low power. The proximity sensor is part of the CapSense Component within PSoC Creator. The proximity value is read by the device and is then used to control an LED. As you move closer to the proximity sensor, a red LED gets brighter; as you move farther away the LED gets dimmer. The code example includes two scanning modes, a fast scan mode that scans 200 times a second and a low scan mode that puts the device into deep sleep and only scans 3 times a second. When in slow scan mode, the device saves power by switching in and out of deep sleep. If the sensor is active (an object is within range), then the device switches to fast scanning mode. If the sensor is inactive (no object is within range) for three seconds, then the device switches into slow scanning mode and goes into Deep Sleep while waiting for the next scan.

Hardware Setup

1. Create a loop with one of the wire jumpers found in the PSoC kit and attach to pin P2[0] as seen in Figure 1.

Figure 1. Wire loop for Proximity Detection (Pioneer kit 042-BLE-A example)



Software Setup

There is no software setup

Operation

1. Plug the CY8CKIT-042-BLE-A kit board into your computer's USB port.
2. Build the project and program it into the PSoC 4 device. Choose **Debug** > **Program**. For more information on device programming, see *PSoC Creator Help.*
3. Move a hand slowly closer to the wire loop and confirm that the LED grows brighter.
4. Move a hand slowly away from the wire loop and confirm that the LED gets dimmer.

5. Right-click the CapSense Component and select **Launch Tuner**. Click **Connect**, select **I2C**, and then click **Start**. Ensure that the data rate is set to **100kHz**. Go to the **Graph View** tab. Confirm that as you move closer to the sensor the raw count increases, and as you move away from the sensor the raw count decreases. For more information, check the CapSense datasheet under Related Documents.

Design and Implementation

The CapSense Component uses capacitive sensing to return a raw count value. The raw count can be used to determine how close something is to the sensor. In this code example the raw count is sent to the CapSense tuner by I2C and graphed in real time. The closer something is to the sensor the higher the raw count, the farther something is from the sensor the lower the raw count. The raw count is also mapped to a PWM controlling an LED. As something moves closer the LED grows brighter. As something moves farther away the LED grows dimmer. This can be seen in the flowchart in Figure 2.

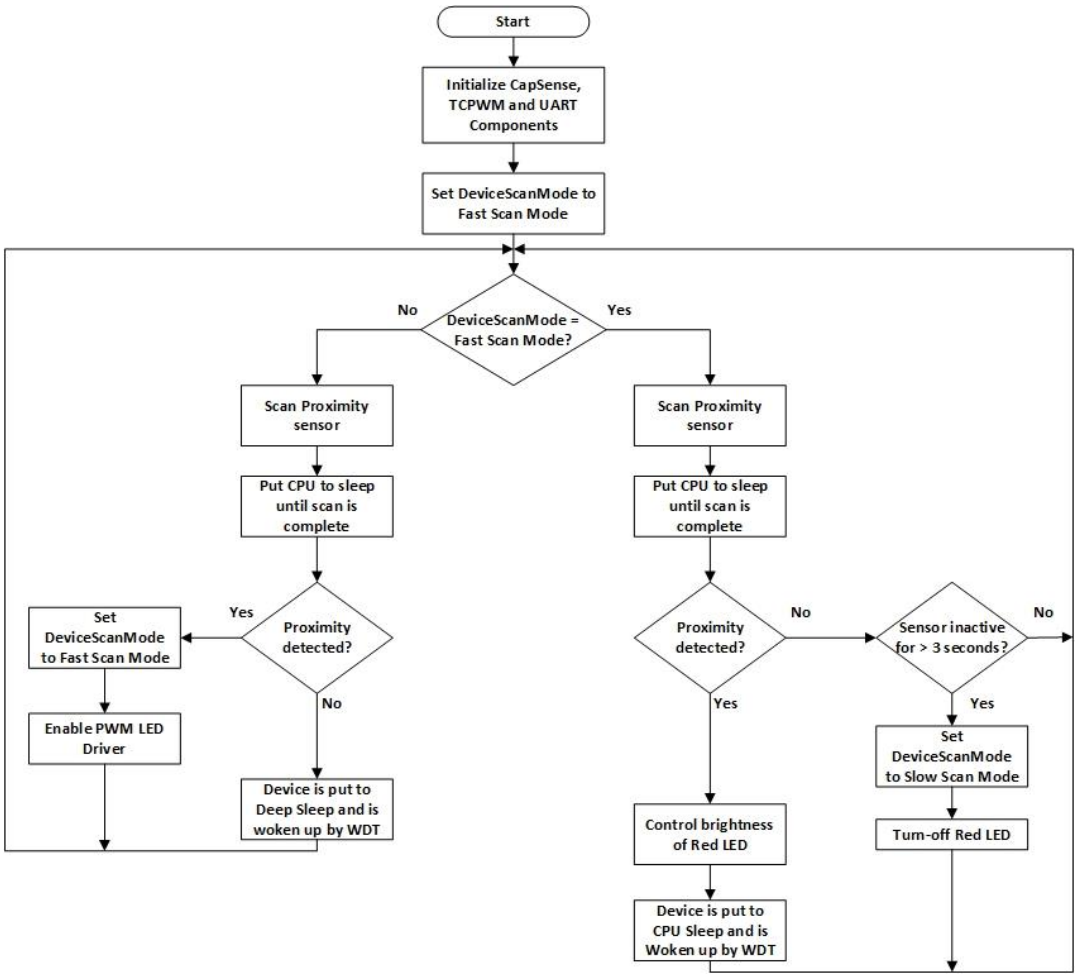
CapSense Proximity

In the CE210489_Low_Power_CapSense_Proximity code example, the following functions are performed:

1. Initialize and start all hardware components.
2. The watch dog timer is set up to keep track of time for both slow and fast scanning modes and wakes the device up when in deep sleep. The functions to control the watchdog timer vary depending on the PSoC 4 device used. This code example is set up to use the watch dog timer for the different types of PSoC 4 devices.
3. Enter a state machine with state SENSOR_SCAN.
    a. I2C transfers all CapSense scan data to the CapSense tuner. No data is sent the first iteration because the sensors have not been scanned or processed.
    b. Scans all CapSense sensors.
4. Enter a state of WAIT_FOR_SCAN_COMPLETE.
    a. Put the CPU to sleep and wait until the scan is complete.
5. Enter a state of PROCESS_DATA
    a. Process all CapSense widgets (sensors that are used to make up a CapSense interface).
    b. If the device is in fast scan mode, check if the widget is active. If the widget is active, scale the proximity sensor value to the PWM Period value to power the LED and reset the soft counter.
    c. If the widget is not active the soft counter adds one. The LED is turned off. When the soft counter reaches 150 the device is set to slow scan mode, this means that the sensor has been inactive for an extended period. The device will be put into deep sleep and only wake up three times a second to scan the widget. This will then save power.
    d. If in slow scan mode, check to see if the widget is active. If the widget is active the device is set to fast scan mode.
6. Enter a state of SLEEP
    a. Go into Deep Sleep if the device is in slow scan mode.
    b. Go into Sleep if the device is in fast scan mode.
    c. The device wakes up when an interrupt occurs. There are two interrupt sources, the first is the I2C to ensure connection between the tuner and the device is stable. The second interrupt source comes from the watch dog timer and the frequency is set by the scan mode.
7. The state is then set to SENSOR_SCAN and the state machine repeats from step 3.

Figure 2 shows the Firmware Flowchart.

Figure 2. Firmware Flowchart



Components and Settings

Table 1 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 1. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
CapSense	CapSense	Gather and process all data from proximity sensor	For Proximity settings, see Figure 3 For General settings, see Figure 4 Under **CSD Settings** change the **Modulator clock frequency** to 12,000 kHz For Widget Details, see Figure 5
EZI2C	EZI2C	Transmits data from the selected kit to the tuner	Under **EZI2C Basic** check the box labeled **Enable wakeup from Deep Sleep Mode**
PWM	PWM	Controls the duty cycle of the Red LED	Under **PWM** deselect **On terminal count**, and set the **Compare** value to 0
Global Signal	WDT	Timer based interrupt to wake up from deep sleep	To use this, **under Design Wide Resources** and **Clocks**, select **LFClk,** see Figure 6

For information on the hardware resources used by a Component, see the Component datasheet.

Figure 3. Proximity Settings

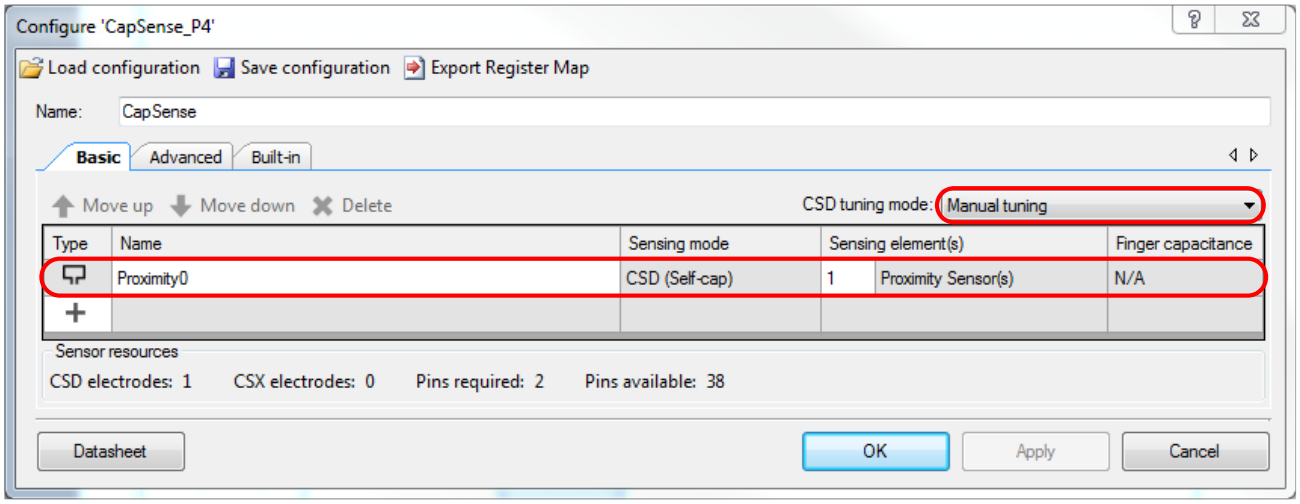


Figure 4. General Settings

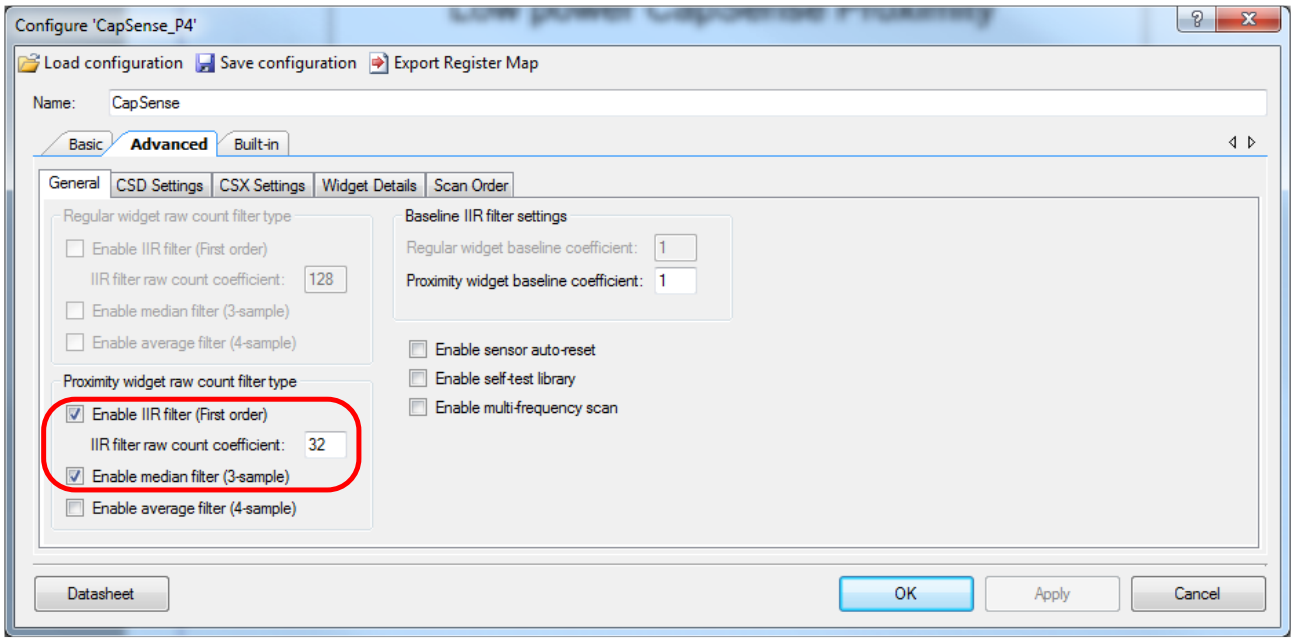


Figure 5. Widget Details

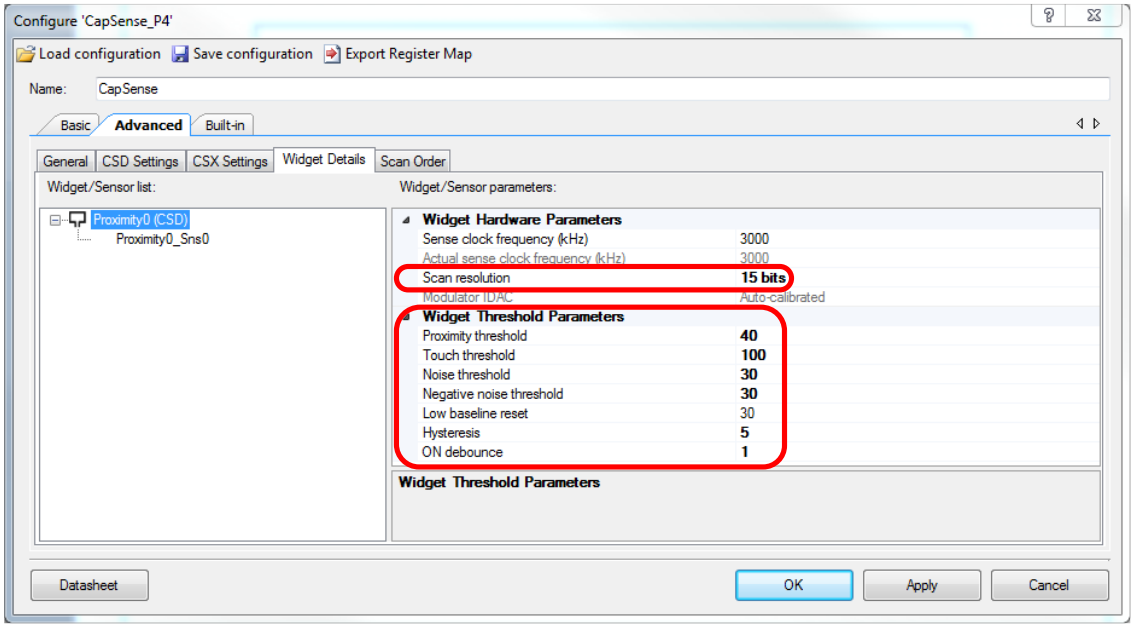
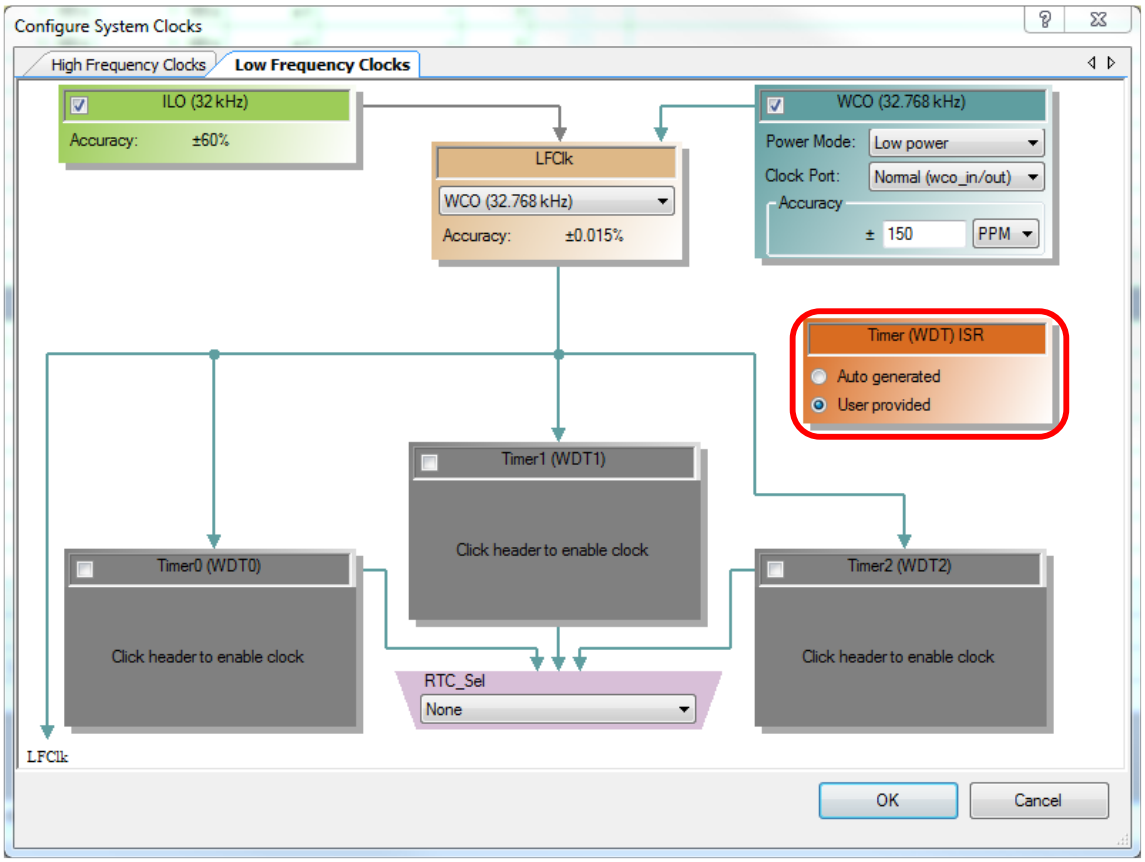


Figure 6. Clocks Setup



Reusing This Example

The kits listed in Table 2 can be used with minimal changes; ensure that:

1. Use one of the five listed kits.
2. Ensure all pins are unlocked in the **Design Wide Resources** tab.
3. Connect the wire loop as seen in Figure 1 to the pin in Table 2.

Table 2. Proximity Sensing Pin Input

Kit Selection	Part Number	PIN
CY8CKIT-042 PSoC 4 Pioneer Kit	CY8C4245AXI-483	P0[4]
CY8CKIT-042-BLE-A Bluetooth® Low Energy 4.2 Compliant Pioneer Kit	CY8C4248LQI-BL583	P2[0]
CY8CKIT-044 PSoC 4 M-Series Pioneer Kit	CY8C4247AZI-M485	P3[7]
CY8CKIT-041-40XX PSoC 4 S-Series Pioneer Kit	CY8C4045AZI-S413	P1[6]
CY8CKIT-041-41XX PSoC 4100S CapSense Pioneer Kit	CY8C4146AZI-S433	P1[6]

To port the code to a new device, in PSoC Creator, select **Project** > **Device Selector** and change to the target device.

Before porting this example to another device, note the following:

1. Not all PSoC 4 devices have hardware to use PSoC Creator CapSense, I2C, and PWM Components.
2. Pinouts change from device to device. Some pins may need to be moved. See the **Pin Layout** tab in PSoC Creator

In some cases, a resource used by a code example (for example, a Universal Digital Block) is not supported on another device. In that case, the example will not work. If you build the code targeted at such a device, you will get errors. See the device datasheet for information on what a device supports.

Related Documents

For a comprehensive list of PSoC 3, PSoC 4, and PSoC 5LP resources, see KBA86521 in the Cypress community.

Application Notes	
AN79953 – Getting Started with PSoC® 4	Describes PSoC 4 devices and how to build your first PSoC Creator project
AN85951 – PSoC 4 and PSoC 6 MCU CapSense Design Guide	Describes how to tune and use the CapSense Component
Code Examples	
CE225691 – CapSense Proximity Sensor	Shows how to use the CapSense proximity sensor
PSoC Creator Component Datasheets	
CapSense	CapSense Component datasheet for more information
TCPWM	TCPWM Component datasheet for more information
I2C	I2C Component datasheet for more information
Device Documentation	
PSoC 4 Datasheets	PSoC 4 Technical Reference Manuals
Development Kit Documentation	
PSoC 4 Kits	
Tool Documentation	
PSoC Creator	Look in the **Downloads** tab for Quick Start and User Guides

Document History

Document Title: CE210489 – PSoC 4 Low Power CapSense Proximity

Document Number: 002-10489

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5269042	DCHE	07/12/2016	New code example
*A	5524679	SRDS	11/18/2016	Updated template
*B	6525359	NRSH	03/29/2019	Updated hardware components and document. Updated firmware to work with more devices.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2016-2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.
TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress shall have no liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. CYPRESS DOES NOT REPRESENT, WARRANT, OR GUARANTEE THAT CYPRESS PRODUCTS, OR SYSTEMS CREATED USING CYPRESS PRODUCTS, WILL BE FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (collectively, "Security Breach"). Cypress disclaims any liability relating to any Security Breach, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any Security Breach. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. "High-Risk Device" means any device or system whose failure could cause personal injury, death, or property damage. Examples of High-Risk Devices are weapons, nuclear installations, surgical implants, and other medical devices. "Critical Component" means any component of a High-Risk Device whose failure to perform can be reasonably expected to cause, directly or indirectly, the failure of the High-Risk Device, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any use of a Cypress product as a Critical Component in a High-Risk Device. You shall indemnify and hold Cypress, its directors, officers, employees, agents, affiliates, distributors, and assigns harmless from and against all claims, costs, damages, and expenses, arising out of any claim, including claims for product liability, personal injury or death, or property damage arising from any use of a Cypress product as a Critical Component in a High-Risk Device. Cypress products are not intended or authorized for use as a Critical Component in any High-Risk Device except to the limited extent that (i) Cypress's published data sheet for the product explicitly states Cypress has qualified the product for use in a specific High-Risk Device, or (ii) Cypress has given you advance written authorization to use the product as a Critical Component in the specific High-Risk Device and you have signed a separate indemnification agreement.
Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.