Please note that Cypress is an Infineon Technologies Company.

The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers

Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE225691 – PSoC 4 CapSense Proximity

Objective

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

Requirements

Tool: PSoC Creator 4.3

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: All PSoC 4 family devices that have a CapSense Component

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

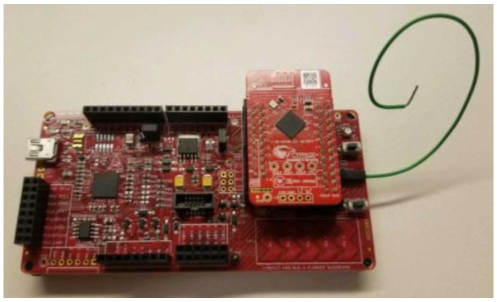
Overview

This code example contains a PSoC Creator project that uses a proximity sensor from the CapSense Component. An LED is controlled by the proximity sensor. As you move closer to the proximity sensor, a red LED gets brighter; as you move farther away the LED gets dimmer. The range of the proximity sensor depends on the size and shape of the wire loop made in the Hardware Setup section.

Hardware Setup

1. Create a loop with one of the wire jumpers found in the PSoC kit and attach it 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 **400 kbps**. 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 15-bit value called a diff value. The PWM is set up to take a 16-bit value, so the CapSense diff value is scaled to a 16-bit value and shifted up by 500. This value is then used to control the PWM to make the LED brighter or dimmer.

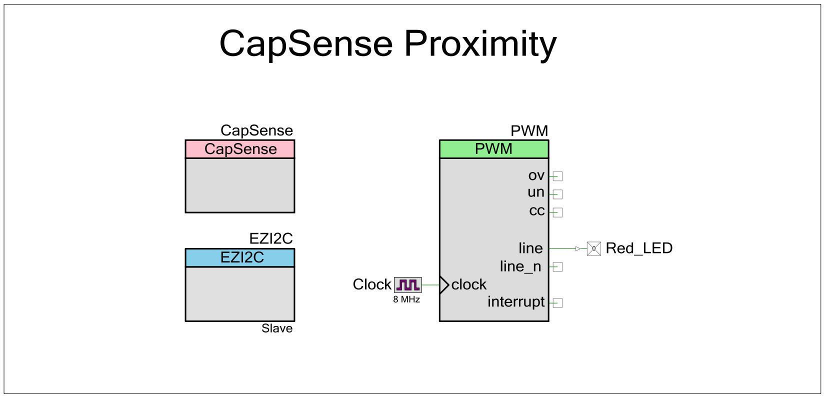
CapSense Proximity

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

1. Initialize and start all hardware components.
2. Link the EZI2C to the CapSense data structure.
3. Scan the proximity sensor.
4. Process the diff value by scaling it to the PWM max compare value. The diff value returns the difference between the raw count and the baseline. The baseline stays around 85% of the maximum raw count, which means that the returned maximum diff value is about 15% of the raw count. The value is then multiplied by 25 to scale to the PWM value and shifted by 500 so that at low diff values the duty cycle is high enough to turn the LED ON.
5. Change the PMW duty cycle according to the proximity sensor value, changing the brightness of the LED.
6. Send all data to the CapSense tuner.
7. Scan the proximity sensor and return to step 4.

Figure 2 shows the top-design of the CapSense_Proximity Creator Project.

Figure 2. CapSense_Proximity Top Design Schematic



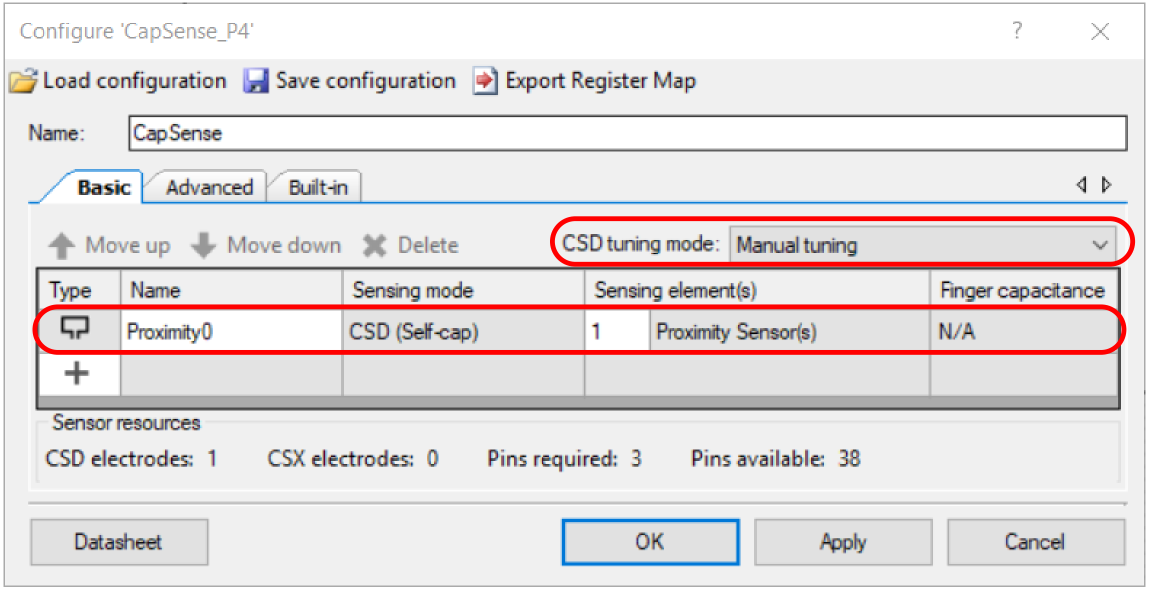
Components and Settings

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 For CSD settings, see Figure 5 For Widget details, see Figure 6
EZI2C	EZI2C	Transmits data from the selected kit to the tuner	Under **EZI2C Basic** change the **Data Rate** to 400 kbps, and change the **Sub-Address Size** to 16
PWM	PWM	Controls the duty cycle of the Red LED	Under PWM change the Interrupt **On terminal count** to OFF**,** and change the **Compare value** to 0

Figure 3. Proximity Settings



Configure 'CapSense_P4'

Name: CapSense

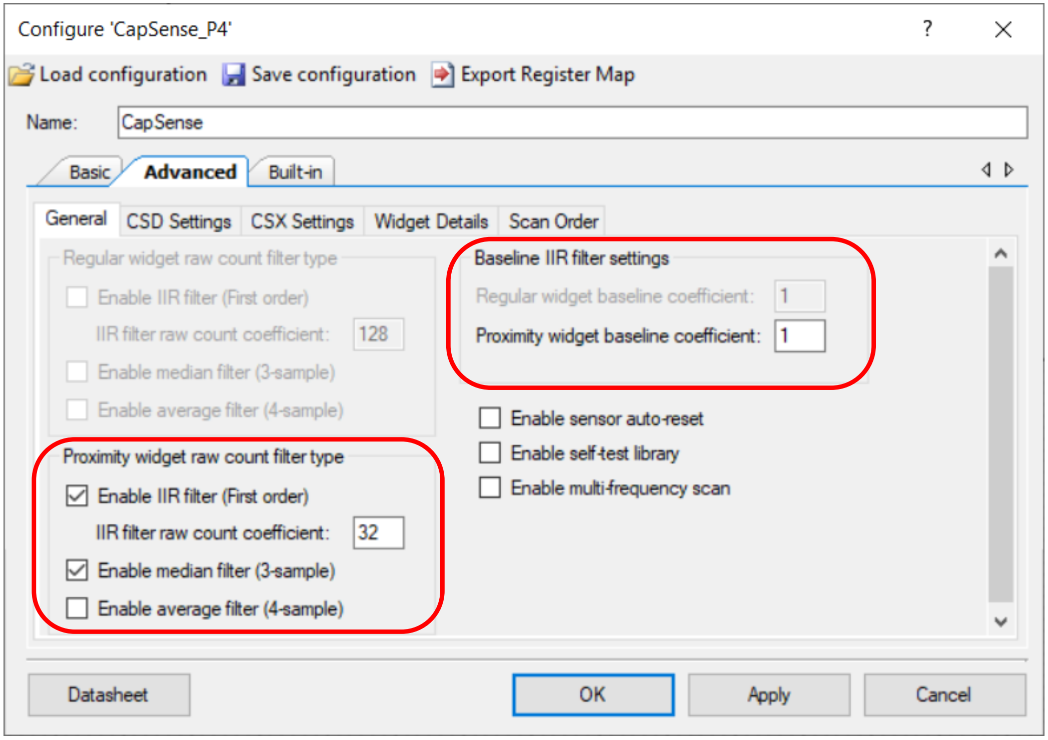
CSD tuning mode: Manual tuning

Type	Name	Sensing mode	Sensing element(s)		Finger capacitance
	Proximity0	CSD (Self-cap)	1	Proximity Sensor(s)	N/A
+					

Sensor resources

CSD electrodes: 1 CSX electrodes: 0 Pins required: 3 Pins available: 38

Figure 4. General Settings



Configure 'CapSense_P4'

Name: CapSense

Regular widget raw count filter type

□ Enable IIR filter (First order)

IIR filter raw count coefficient: 128

□ Enable median filter (3-sample)

□ Enable average filter (4-sample)

Proximity widget raw count filter type

☑ Enable IIR filter (First order)

IIR filter raw count coefficient: 32

☑ Enable median filter (3-sample)

□ Enable average filter (4-sample)

Baseline IIR filter settings

Regular widget baseline coefficient: 1

Proximity widget baseline coefficient: 1

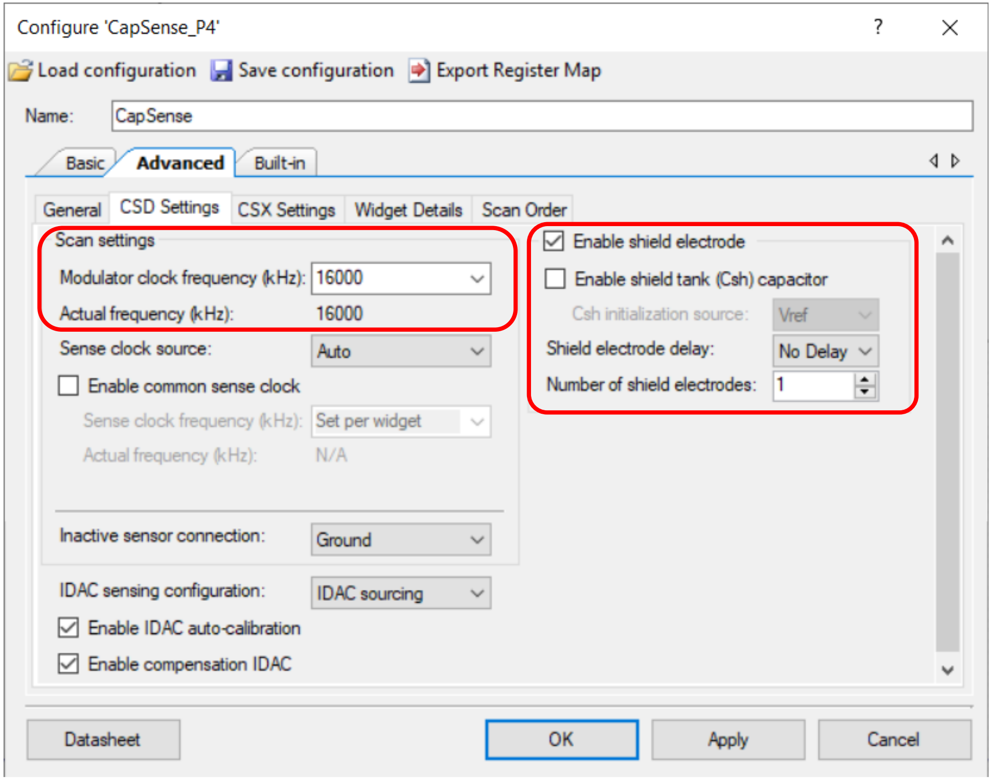
□ Enable sensor auto-reset

□ Enable self-test library

□ Enable multi-frequency scan

Note: The Proximity sensor is sensitive; the filters are used to keep the raw count from jumping a large amount.

Figure 5. CSD Settings



Configure 'CapSense_P4'

Name: CapSense

Scan settings

Modulator clock frequency (kHz): 16000

Actual frequency (kHz): 16000

Sense clock source: Auto

□ Enable common sense clock

Sense clock frequency (kHz): Set per widget

Actual frequency (kHz): N/A

Inactive sensor connection: Ground

IDAC sensing configuration: IDAC sourcing

☑ Enable IDAC auto-calibration

☑ Enable compensation IDAC

☑ Enable shield electrode

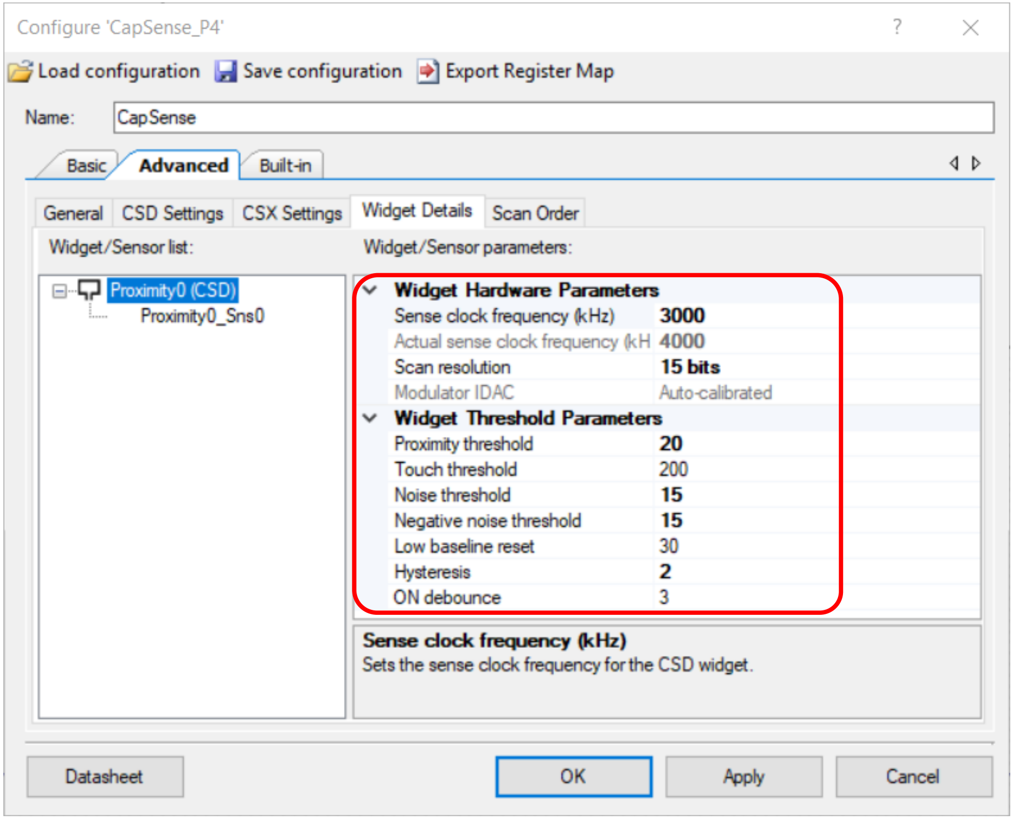
□ Enable shield tank (Csh) capacitor

Csh initialization source: Vref

Shield electrode delay: No Delay

Number of shield electrodes: 1

Figure 6. Widget details



Configure 'CapSense_P4'

Name: CapSense

Widget/Sensor list:

- Proximity0 (CSD)
  - Proximity0_Sns0

Widget/Sensor parameters:

Widget Hardware Parameters	
Sense clock frequency (kHz)	**3000**
Actual sense clock frequency (kH	**4000**
Scan resolution	**15 bits**
Modulator IDAC	Auto-calibrated
Widget Threshold Parameters	
Proximity threshold	**20**
Touch threshold	200
Noise threshold	**15**
Negative noise threshold	**15**
Low baseline reset	30
Hysteresis	**2**
ON debounce	3

Sense clock frequency (kHz)

Sets the sense clock frequency for the CSD widget.

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

Reusing This Example

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

1. Use one of the six 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-040 PSoC 4 Pioneer kit	CY8C4014LQI-422	P2[0]
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 support CapSense, EZI2C, 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

For more information on how to incorporate CapSense with proximity sensing into a design see the two related app notes AN85951 – PSoC 4 and PSoC 6 MCU CapSense Design Guide and AN92239 – Proximity Sensing with CapSense.

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.

For a comprehensive list of PSoC 6 MCU resources, see KBA223067 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
AN92239 – Proximity Sensing with CapSense	Describes how to design a proximity sensor and tune it to achieve a greater proximity sensing distance
Code Examples	
CE210489 – Low Power CapSense Proximity Sensor	Shows how to use the CapSense proximity sensor with low power
PSoC Creator Component Datasheets	
CapSense	CapSense Component datasheet for more information
TCPWM	TCPWM Component datasheet for more information
EZI2C	EZI2C 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: CE225691 – PSoC 4 CapSense Proximity

Document Number: 002-25691

Revision	ECN	Submission Date	Description of Change
**	6418147	2/7/2019	New code example
*A	6896862	6/15/2020	Minor updates to document and code example.

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 | Code Examples | 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
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709

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.