CE225409 – PSoC 4700S MagSense COIL

Objective

This code example demonstrates using the PSoC® Creator™ MagSense™ Component in PSoC 4700S series devices to interface and process the sensor data from inductive sensing circuits on the CY8CKIT-148-COIL kit.

Overview

This code example demonstrates how to use each of the coil types on the CY8CKIT-148-COIL Inductive Sensing Coil Breakout Board with the CY8CKIT-148 Inductive Sensing Evaluation Kit. Five projects are included:

- 6-Segment Slider
- Linear Encoder
- Linear Target
- Rotary Encoder
- Proximity Sensor

Each project is designed for a specific sensor. Custom sensors that follow the guidelines in Appendix A of AN219207 – Inductive Sensing Design Guide may also be used with a few changes to Component settings.

Requirements

Tool: PSoC Creator 4.2

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: PSoC 4700S

Related Hardware: CY8CKIT-148 PSoC 4700S Inductive Sensing Evaluation Kit, CY8CKIT-148-COIL Inductive Sensing Coil Breakout Board

Hardware Setup

This code example requires both the CY8CKIT-148 kit and the CY8CKIT-148-COIL kit.

See the Hardware Setup section for each project for specific hardware setup instructions.

Software Setup

Note: Ensure that you have the latest versions of the PSoC 4700S Device and the MagSense Component in PSoC Creator. Follow these steps:

1. In PSoC Creator, select **Tools** > **Find New Devices** > **Install**.
2. When installation completes, select **Tools** > **Find New Components** > **Select All** > **Install Checked Components**.
3. Close and reopen PSoC Creator.
4. Right-click each project and choose **Reload**. This step only applies if you opened the code example workspace before performing step 1.

MagSense Tuner

Each project included in the code example workspace is compatible with the MagSense Tuner. The tuner enables real time viewing of MagSense sensor data. To use the Tuner, follow the steps:

1. Right-click the project that you want to program into the CY8CKIT-148 kit and choose **Set As Active Project**

2. Choose **Debug** > **Program** to build the project and program it into the PSoC device. For more information on device programming, see PSoC Creator Help.
3. Launch the Tuner: right-click the MagSense Component in the schematic (TopDesign.cysch) of the active project and choose **Launch Tuner**.
4. In the Tuner window, select **Tools** > **Tuner Communication Setup**. Under **Ports**, select the **KitProg – I2C** device. Ensure that the Port Configuration matches the settings in Figure 1.
5. In the Tuner window, select **Connect** followed by **Start**.

Figure 1. Tuner Communication Settings

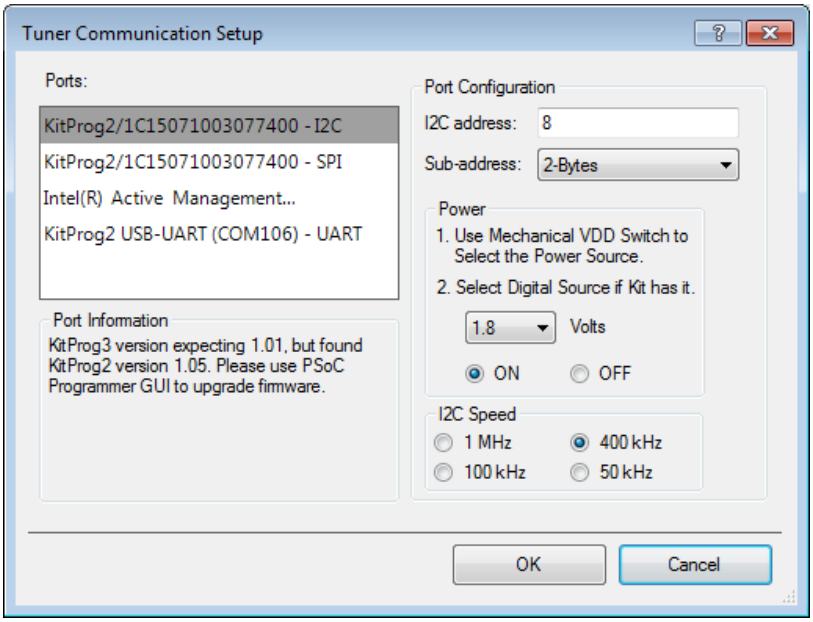
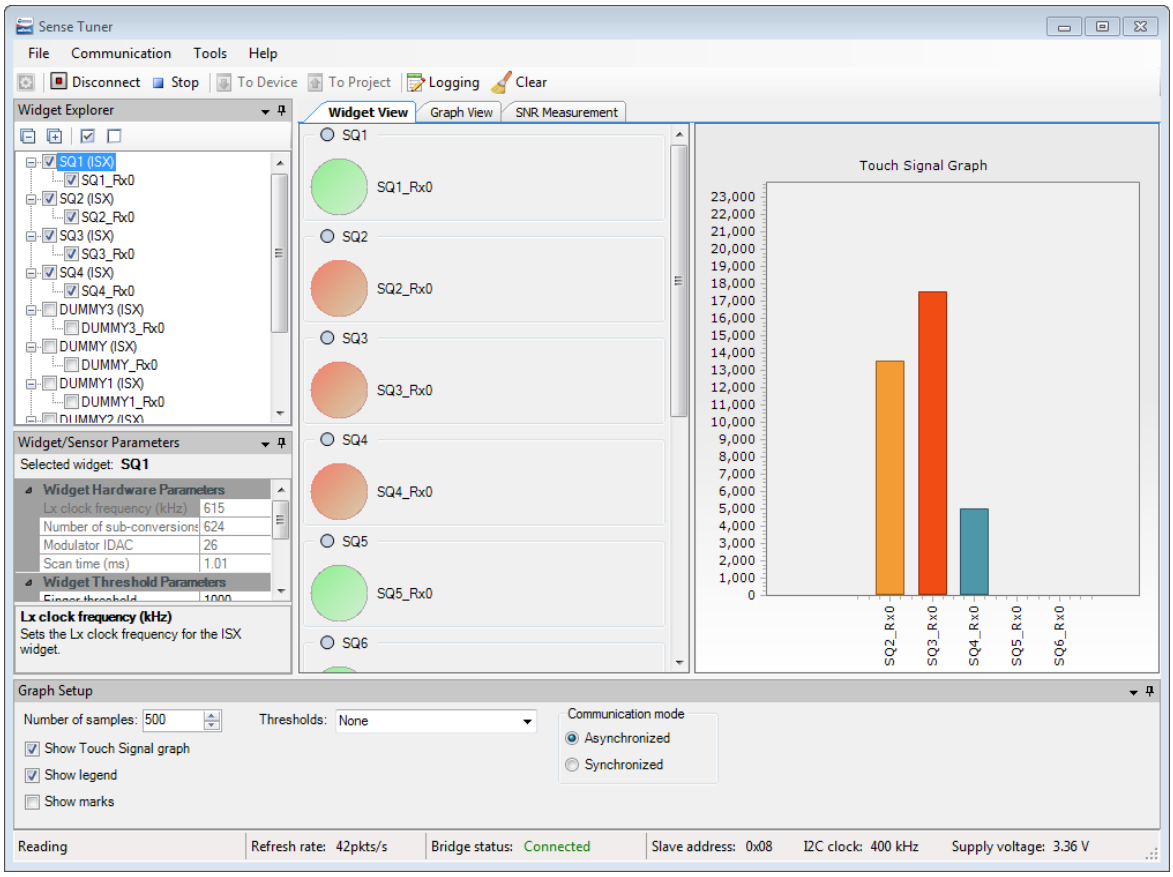


Figure 2. The MagSense Tuner With Sensor Signals Shown



6-Segment Slider Project

This project uses six square sensors to detect the position of a metal target as it moves across each sensor. The position of the target controls the brightness of an LED on the CY8CKIT-148 kit board.

Hardware Setup

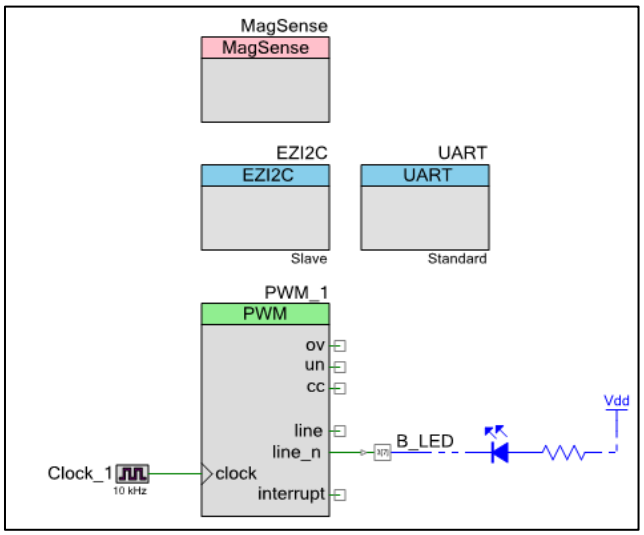
1. Connect the FPC cable from header **J3** on the CY8CKIT-148-COIL board to header **J6** on the CY8CKIT-148 kit board.

Operation

1. Plug the CY8CKIT-148 kit board into your computer's USB port.
2. In the PSoC Creator Workspace, right-click the 6-Segment Slider project and choose **Set As Active Project**.
3. Choose **Debug** > **Program** to build the project and program it into the PSoC device. For more information on device programming, see PSoC Creator Help.
4. Launch a terminal emulator such as PuTTY or Tera Term and select the COM port for the KitProg device. Configure the terminal for 115200 baud, 8-bit data, no parity, 1 stop bit, and no flow control.
5. Slide Metal Target 1 over each sensor and observe how the brightness of LED2 on CY8CKIT-148 kit board changes as the target slides over each coil. The UART terminal will display the PWM duty cycle as the target moves.
6. Follow steps 3-5 under MagSense Tuner in the Software Setup section to launch the Tuner and observe real time sensor data.

Design and Implementation

Figure 3. 6-Segment Slider Project Schematic



This example uses six 10-mm square coils to determine the location of a metal target over the coil array. Each coil is implemented as a button-type widget in firmware. The settings for each button are identical and have been tuned for optimal sensor performance. For more information on tuning sensors, see AN219207 – Inductive Sensing Design Guide.

The code example firmware processes the data from the tuned sensors. A centroid algorithm is used to convert the sensor data for each of the six sensors into positional data. This algorithm returns values in the range 0–100, which represent the position of the target and the duty cycle for the PWM Component.

The PWM Component accepts the centroid data as arguments to the `PWM_WriteCompare()` API function. The brightness of the LED then changes based on the PWM duty cycle.

Components and Settings

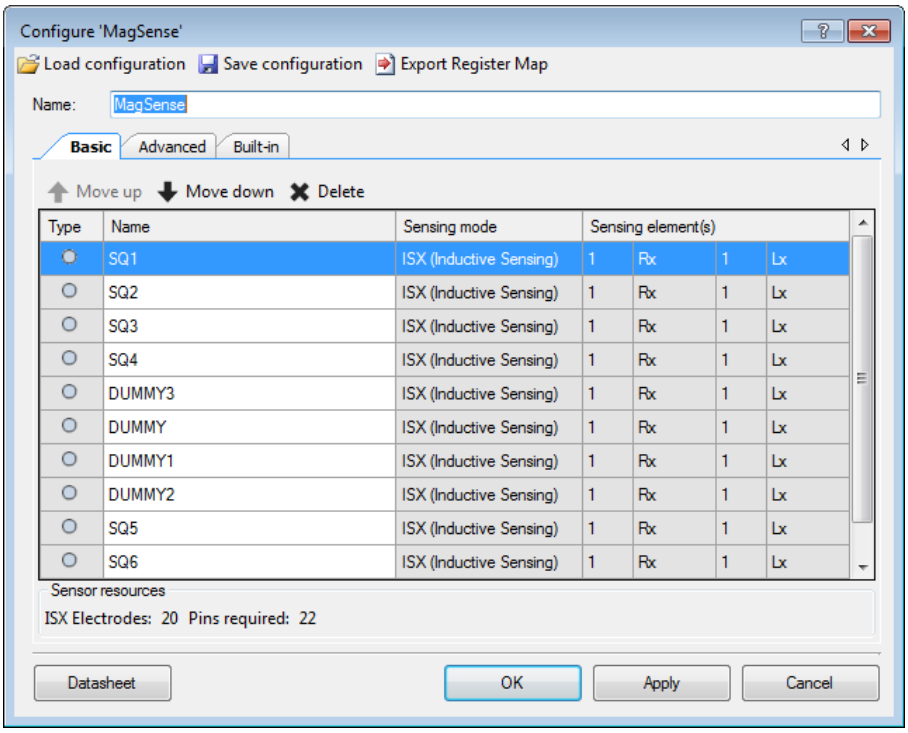
Table 1 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 1. PSoC Creator Components

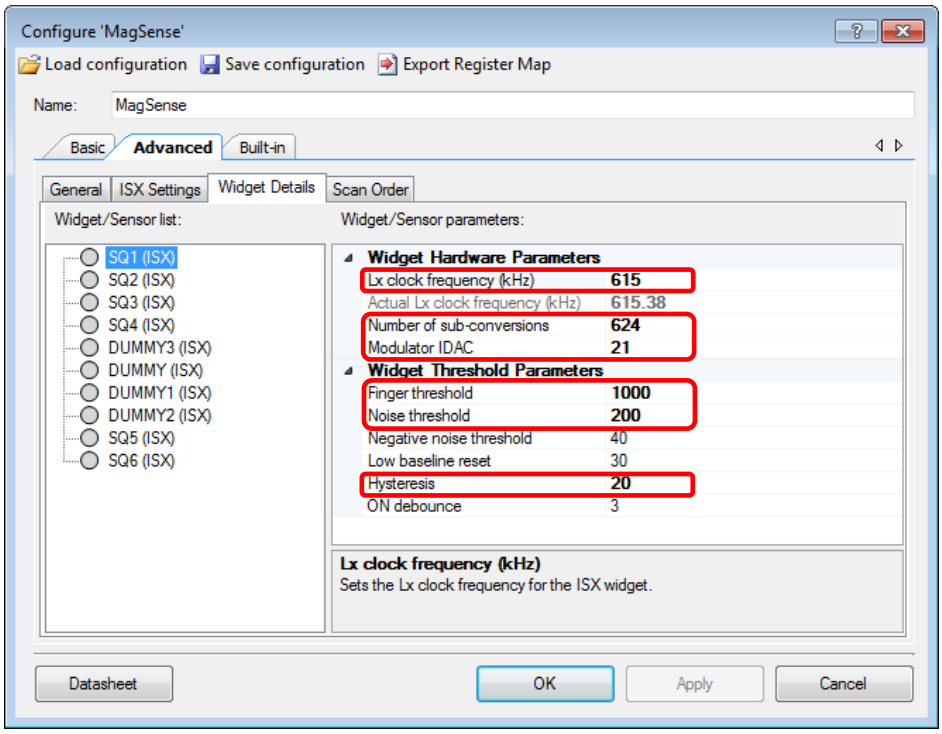
Component	Instance Name	Purpose	Non-default Settings
MagSense	MagSense	Enables inductive sensing	See Figure 4.
EZI2C Slave (SCB mode)	EZI2C	Enables sensor tuner	**Data rate (kbps):** 400 **Sub-address size (bits):** 16
PWM (TCPWM mode)	PWM_1	Drives the LED	**Period:** 99
Clock	Clock_1	Drives the PWM	**Frequency:** 10 kHz
Digital Output Pin	B_LED	I/O pin connected to LED	None
UART (SCB mode)	UART	Enable visual display of LED brightness	None

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

Figure 4. MagSense Component Settings



Note: The MagSense Component requires use of an entire port before pins on any other port can be used for inductive sensing. Because of this and the available I/O pins on CY8CKIT-148 kit, dummy sensors DUMMY-DUMMY3 are required to occupy the remainder of port 2. This enables SQ5 and SQ6 to use pins on port 3.



Linear Encoder Project

This project uses a rectangular coil which linearly changes in area to create a linearly increasing sensor signal as a metal target slides across the coil. The sensor data is used to control the brightness of an LED as the metal target moves across the sensor.

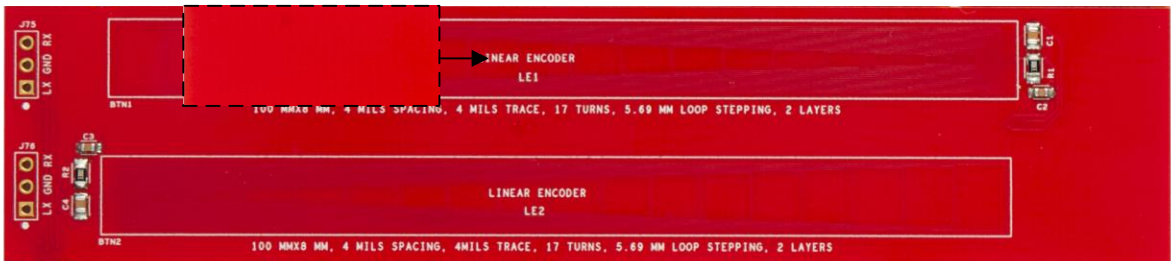
Hardware Setup

1. Connect the FPC cable from header **J2** on the CY8CKIT-148-COIL kit board to header **J6** on the CY8CKIT-148 kit board.

Operation

1. Plug the CY8CKIT-148 kit board into your computer's USB port.
2. In the PSoC Creator Workspace, right-click the **LinearEncoder** project and choose **Set As Active Project**.
3. Choose **Debug** > **Program** to build the project and program it into the PSoC device. For more information on device programming, see PSoC Creator Help.
4. Launch a terminal emulator such as PuTTY or Tera Term and select the COM port for the KitProg device. Configure the terminal for 115200 baud, 8-bit data, no parity, 1 stop bit, and no flow control.
5. Place Metal Target 3 on either LE1 or LE2 and slide it along the length of the coil multiple times so that firmware can detect the sensor signal range. Firmware uses the stronger signal between LE1 and LE2, enabling you to use either coil.

Figure 5. Operation of the Linear Encoder Project

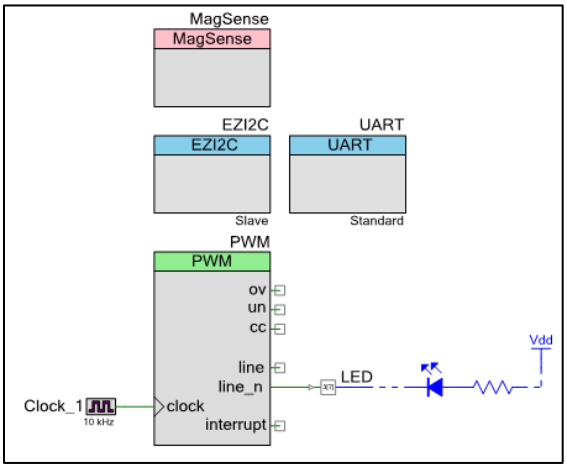


6. Observe LED2 on the CY8CKIT-148 kit board changing as the target slides over the coil.
7. Follow steps 3-5 under MagSense Tuner in the Software Setup section to launch the Tuner and observe real time sensor data.

Note: Any metal target may be used with this project, however, this will change the sensor data range as well as the project performance. Reset the CY8CKIT-148 kit after removing your old target and before using the new metal target.

Design and Implementation

Figure 6. Linear Encoder Schematic



This project uses the shape of the coil to determine the position of a target over the coil. However, unlike the 6-Segment Slider project, there is only a single coil that is designed to have a linear change in performance as a target moves across it. Firmware receives the data from the MagSense Component and updates the PWM Compare value on the kit board in response. The sensor data is scaled down in firmware to fit into a range of values between 0–100. This value is passed as the argument to the `PWM_WriteCompare()` API function. The brightness of the LED then changes based on the PWM duty cycle.

Components and Settings

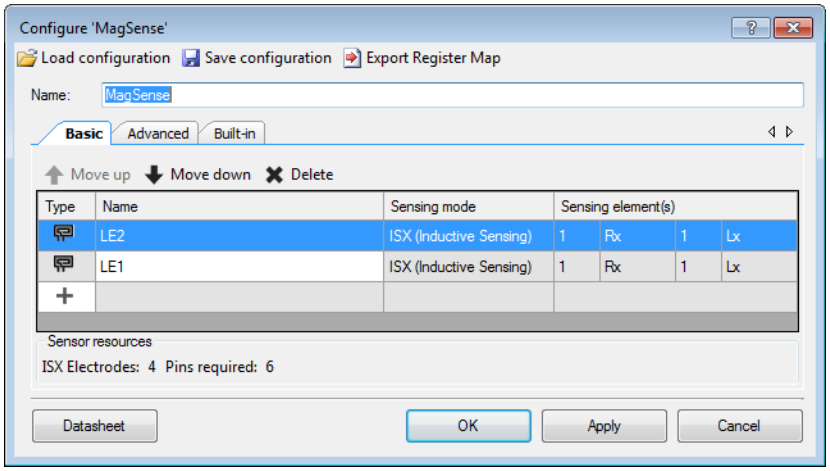
Table 2 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

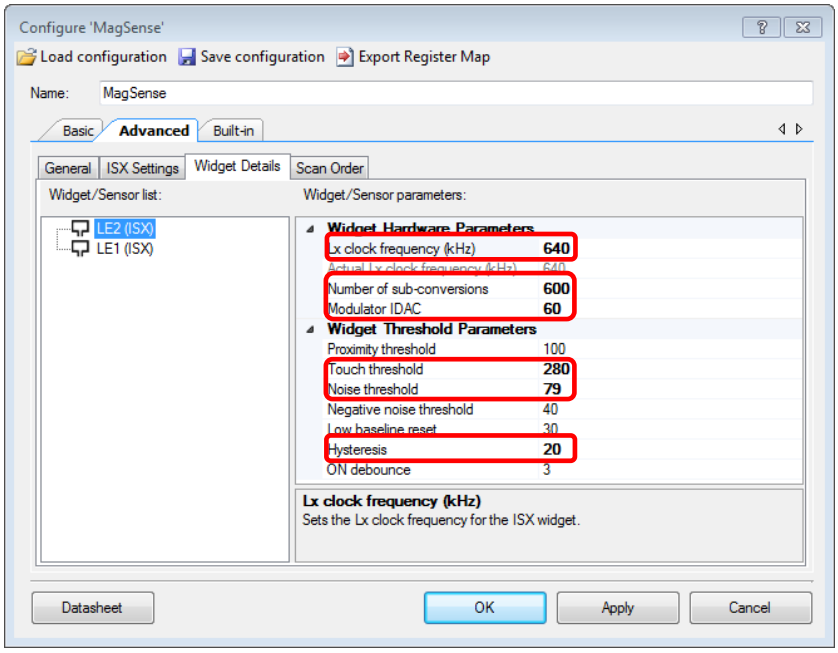
Table 2. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
MagSense	MagSense	Enables inductive sensing	See Figure 7
EZI2C Slave (SCB mode)	EZI2C	Enables sensor Tuner	**Data rate (kbps):** 400 **Sub-address size (bits):** 16
Digital Output Pin	LED	Supports connections to LEDs	None
PWM (TCPWM mode)	PWM	Drives the LED	**Period:** 99
Clock	Clock_1	Drives the PWM	**Frequency:** 10 kHz
UART (SCB mode)	UART	Enable visual display of LED brighness	None

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

Figure 7. MagSense Component Settings





Linear Target Project

This project uses a square coil with a metal target to generate a linearly changing signal. The metal target consists of a triangular copper area which increases in width linearly. As the metal target slides over the edge of the square coil, a linearly changing signal is produced which controls the brightness of an LED.

Hardware Setup

Setup on CY8CKIT-148-COIL

1. Populate header **J73** or **J74** on the CY8CKIT-148-COIL board with female or male pin headers (2.54-mm pitch).

Setup on CY8CKIT-148 Kit Board

1. Populate I/O header **J4** on the CY8CKIT-148 kit board with an 8x2 female or male pin header (2.54-mm pitch).
2. Populate the capacitors labeled **C52**-**C57** on the back side of the board with 10-pF 0603 surface mount capacitors.

General Setup

1. Connect the LX pin of either SQ7 (if you populated header **J73**) or SQ8 (if you populated header **J74**) to pin 2[4] on the CY8CKIT-148 kit board.
2. Connect the RX pin of the same 20-mm square coil to pin 0[4] on the CY8CKIT-148 kit board.
3. Connect the GND pin of the same 20-mm square coil to any GND pin on the CY8CKIT-148 kit board.

Operation

1. Plug the CY8CKIT-148 kit board into your computer's USB port.
2. In the PSoC Creator Workspace, right-click the **LinearTarget** project and choose **Set As Active Project.**
3. Choose **Debug** > **Program** to build the project and program it into the PSoC device. For more information on device programming, see PSoC Creator Help.
4. Launch a terminal emulator such as PuTTY or Tera Term and select the COM port for the KitProg device. Configure the terminal for 115200 baud, 8-bit data, no parity, 1 stop bit, and no flow control.
5. Place Metal Target 2 on the coil and slide the entire target over the coil once to detect the sensor signal range.
6. Observe the LED2 on the CY8CKIT-148 kit board changing as the target slides over the coil.

7. Follow steps 3-5 under MagSense Tuner in the Software Setup section to launch the Tuner and observe real time sensor data.

Note: Any metal target may be used with this project, however, this will change the sensor data range as well as the project performance. Reset the CY8CKIT-148 kit after removing your old target and before using the new metal target.

Design and Implementation

This project is the same as the Linear Encoder Project except that it uses the linear target over a fixed square coil rather than a square target over a linear coil. The theory and firmware implementation are entirely the same otherwise.

Components and Settings

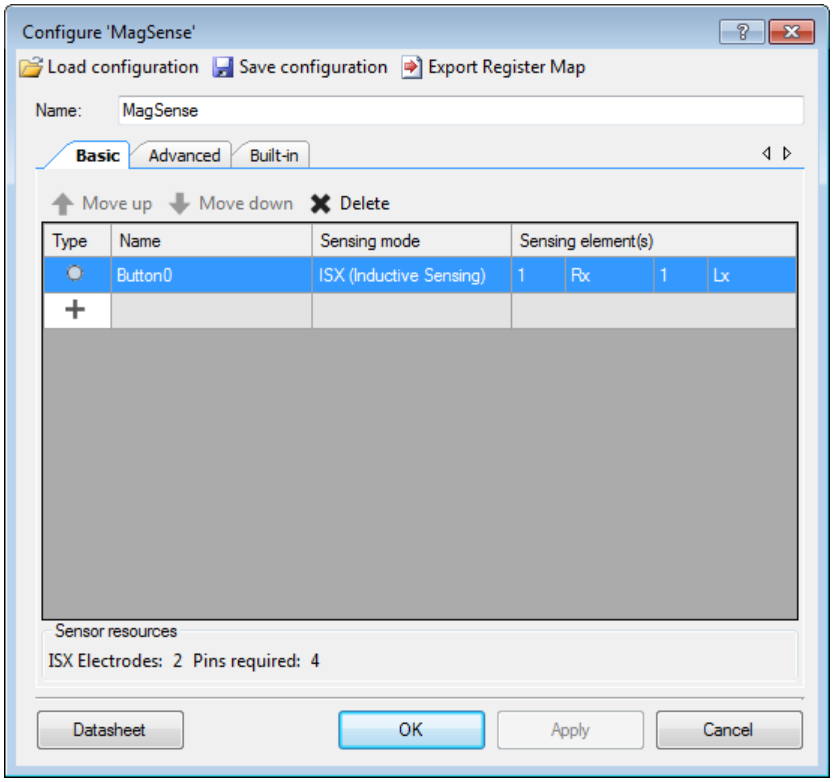
Table 3 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

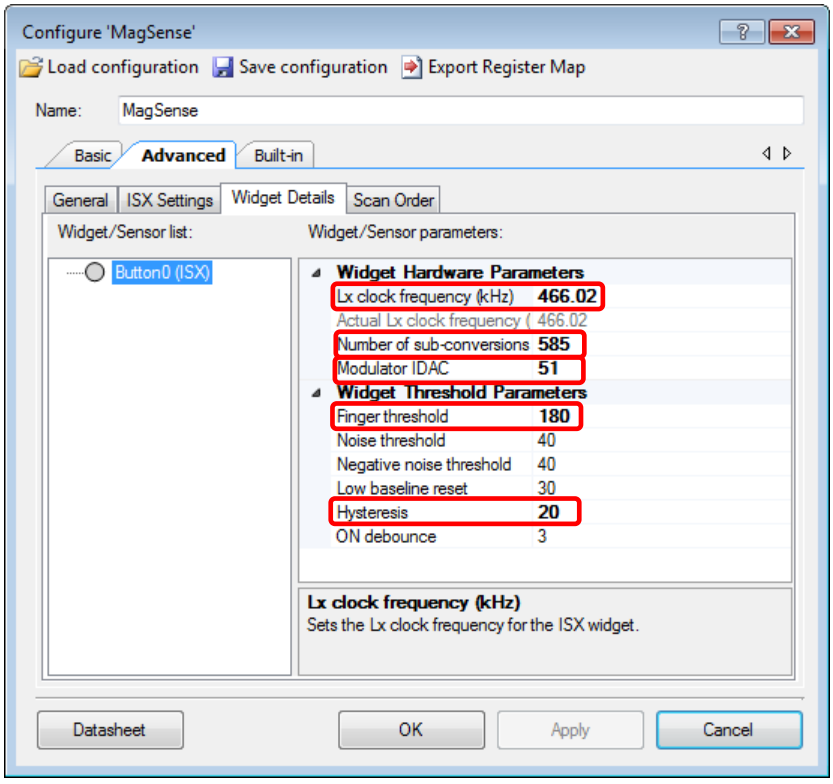
Table 3. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
MagSense	MagSense	Enables inductive sensing	See Figure 8
EZI2C Slave (SCB mode)	EZI2C	Enables sensor tuner	**Data rate (kbps):** 400 **Sub-address size (bits):** 16
Digital Output Pin	LED	Supports connections to LEDs	None
PWM (TCPWM mode)	PWM	Drives the LED	**Period:** 99
Clock	Clock_1	Drives the PWM	**Frequency:** 10 kHz
UART (SCB mode)	UART	Enable visual display of LED brightness	None

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

Figure 8. MagSense Component Settings





Rotary Encoder Project

This project uses two 120° arc angle sensors and a 119° arc angle metal target to generate linearly changing signals as the target spins over the sensors. The linearly changing signal is translated into rotational position which is displayed over a UART connection.

Hardware Setup

1. Connect the FPC cable from header **J2** on the CY8CKIT-148-COIL kit board to header **J6** on the CY8CKIT-148 kit board.

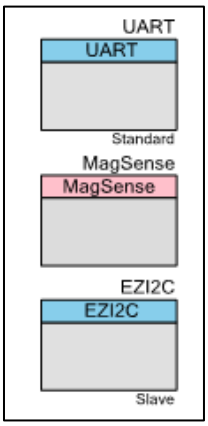
Operation

1. Plug the CY8CKIT-148 kit board into your computer's USB port.
2. In the PSoC Creator Workspace, right-click the Rotary Encoder project and choose **Set As Active Project.**
3. Choose **Debug** > **Program** to build the project and program it into the PSoC device. For more information on device programming, see PSoC Creator Help.
4. Launch a terminal emulator such as PuTTY or Tera Term and select the COM port for the KitProg device. Configure the terminal for 115200 baud, 8-bit data, no parity, 1 stop bit, and no flow control.
5. Place the circular target over the rotary coil and turn the target at least one full rotation. This tunes the system for optimal performance.
6. Continue spinning the target and observe the angle of rotation on the UART terminal.
7. Follow steps 3-5 under MagSense Tuner in the Software Setup section to launch the Tuner and observe real time sensor data.

Note: The metal target may wobble or move slightly during rotation causing small performance glitches. If you are implementing a system with a similar design, it is advised that you firmly affix the target so that it is stable.

Design and Implementation

Figure 9. Rotary Encoder Schematic



This project uses the MagSense Component to determine the rotation angle of an object over inductive sensors. Two sensors each cover 120° of a circle. A metal target with an arc angle of 119° rotates over the sensors. As the target rotates over the first sensor, the sensor's raw counts increase linearly until the target completely covers the sensor. As the target continues to rotate, it begins to cover the second sensor until the second sensor is entirely covered and the first sensor is no longer covered. As the target continues into the final third of the rotation, the second sensor's raw counts decrease linearly from the maximum until the target no longer covers either sensor.

Firmware observes the sensor data from each of the two sensors and converts the data into rotational position based on the following states:

- In State 1, the target moves over the first sensor. This is the point when the target is rotating from 0–120 degrees so the firmware scales the sensor data to fit in that range.
- In State 2, the target moves over the second sensor and off the first sensor. This is the point when the target is rotating from 121–240 degrees. Firmware averages the signals from both sensors to precisely locate the position of the target. This position is scaled to fit into the 121–240 degree range.
- In State 3, the target moves off the second sensor and no longer covers either sensor. This is the state in which the target covers 241–360 degrees. Firmware takes the maximum rotational angle (360) and subtracts a scaled value from the second sensor to create the correct range of angles.

A UART Component is included in the project that prints out the rotational angle of the target as it moves over the sensors. The EZI2C Component enables real time observation of the sensor data through the MagSense Tuner.

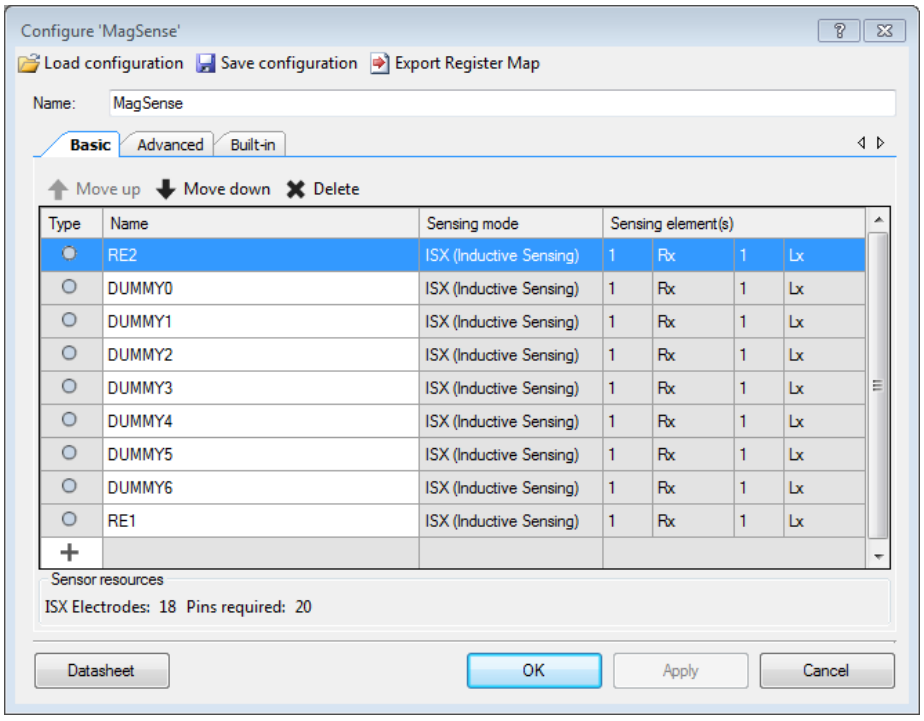
Components and Settings

Table 4 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.

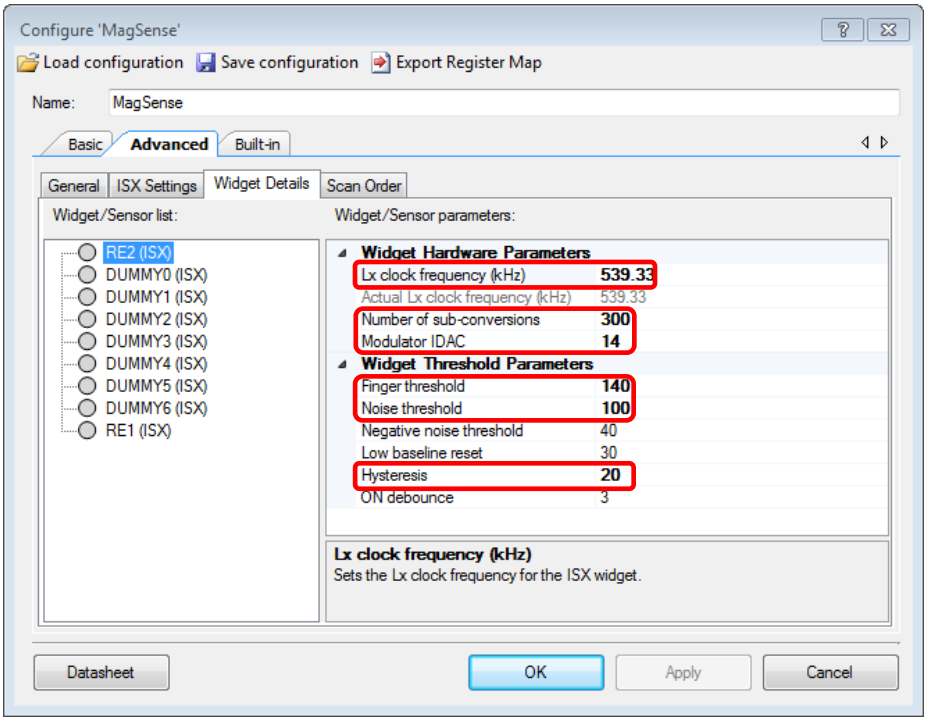
Table 4. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
MagSense	MagSense	Enables inductive sensing	See Figure 10
EZI2C Slave (SCB mode)	EZI2C	Enables sensor Tuner	**Data rate (kbps):** 400 **Sub-address size (bits):** 16
UART	UART	Enables visual feedback of sensor position	None

Figure 10. Rotary Encoder MagSense Settings



Note: The MagSense Component requires use of an entire port before pins on any other port can be used for inductive sensing. Because of this and the available I/O pins on CY8CKIT-148 kit, dummy sensors DUMMY0-DUMMY6 are required to occupy the remainder of port 2. This enables RE1 to use pins on port 3.



Proximity Sensor Project

This project uses the circular coils on the CY8CKIT-148-COIL kit board to demonstrate using MagSense for proximity detection. As a target moves closer to the coils, LED2 on the CY8CKIT-148 kit board increases in brightness. Any of the circular coils (CR1-CR5) may be used with this project.

Hardware Setup

1. If you are using CR5, connect the FPC cable from header **J2** on the COIL kit board to header **J6** on the CY8CKIT-148 kit board. For any other circular coils, see step 2.
2. Populate the 3-pin header of the circular coil you would like to use. Use 2.54-mm pitch pin headers.

Sensor	Header
CR1	J72
CR2	J69
CR3	J53
CR4	J68

3. Populate the capacitors labeled **C52**-**C57** on the back side of the board with 10-pF 0603 surface mount capacitors.
4. Connect the LX, RX, and GND pins of the circular coil to header **J4**, pins 2[7], 1[2] and GND, respectively, on the CY8CKIT-148 kit board.

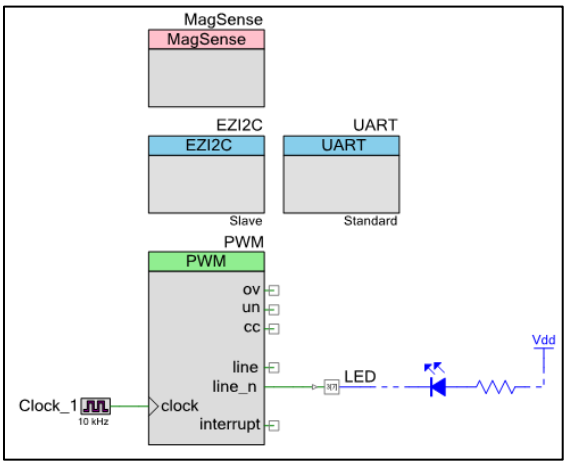
Operation

1. Plug the CY8CKIT-148 kit board into your computer's USB port.
2. In the PSoC Creator Workspace, right-click the Proximity Sensor project and choose **Set As Active Project.**
3. If you are using CR5 proceed to step 4, otherwise set the correct LX clock frequency for your sensor. Right-click the MagSense Component in the project schematic. Choose **Configure** > **Advanced** > **Widget Details** and change the Lx Clock Frequency.
   a. For CR3 or CR4, set the LX clock frequency to 600 kHz.
   b. For CR1 or CR2, set the LX clock frequency to 750 kHz.
   c. For CR5, set the LX clock frequency to 320 kHz.
4. Choose **Debug** > **Program**. Build the project and program it into the PSoC device. For more information on device programming, see PSoC Creator Help.
5. Launch a terminal emulator such as PuTTY or Tera Term and select the COM port for the KitProg device. Configure the terminal for 115200 baud, 8-bit data, no parity, 1 stop bit, and no flow control.
6. Touch any metal target to the coil once to tune the coil. Slowly lift the target from the coil.
7. Observe the LED2 on the CY8CKIT-148 kit board changing as the target moves away from the coil.
8. Follow steps 3-5 under MagSense Tuner in the Software Setup section to launch the Tuner and observe real time sensor data.

Note: Any metal target may be used with this project, however, this will change the sensor data range as well as the project performance. Reset the CY8CKIT-148 kit after removing your old target and before using the new metal target.

Design and Implementation

Figure 11. The Proximity Sensor Schematic



This project uses a MagSense Component to acquire sensor data from an inductive sensing circuit. As a metal target is brought near the coil, the sensor data increases until the target comes in to contact with the coil. Firmware follows the concepts in the Linear Encoder project used to translate sensor data into the brightness of LED2 on CY8CKIT-148 kit board. See the Linear Encoder Design and Implementation section for more information on the operation of this project.

Components and Settings

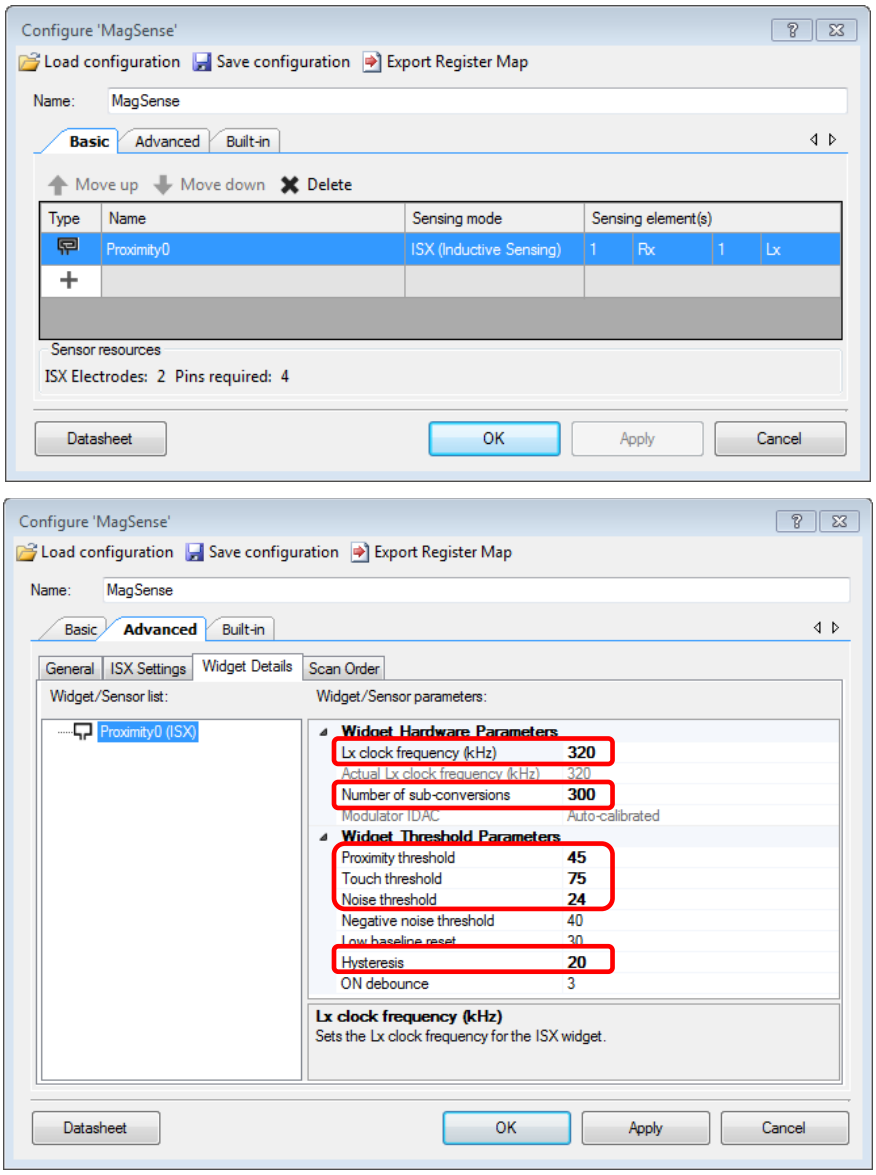
Table 5 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 5. PSoC Creator Components and Settings

Component	Instance Name	Purpose	Non-default Settings
MagSense	MagSense	Enables inductive sensing	See Figure 12
EZI2C Slave (SCB mode)	EZI2C	Enables sensor Tuner	**Data rate (kbps):** 400 **Sub-address size (bits):** 16
Digital Output Pin	LED	Supports connections to LEDs	None
PWM (TCPWM mode)	PWM	Drives the LED	**Period:** 99
Clock	Clock_1	Drives the PWM	**Frequency:** 10 kHz
UART (SCB mode)	UART	Enables visual display of LED brightness	None

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

Figure 12. Proximity Sensor MagSense Component Settings



Reusing these Examples

This example is designed for the CY8CKIT-148-COIL Inductive Sensing Coil Breakout Board. The projects included in this example may also be used with custom coils that have been designed following the guidelines in AN219207 – Inductive Sensing Design Guide. Note that using custom coils will likely require changes to the firmware.

This example is compatible only with PSoC 4700S devices.

Related Documents

Application Notes	
AN219207 – Inductive Sensing Design Guide	Shows how to implement inductive sensing using a compatible PSoC 4 device and tune it for desired performance
PSoC Creator Component Datasheets	
Pins	Supports connection of hardware resources to physical pins
Timer Counter (TCPWM)	Supports fixed-function Timer/Counter implementation
Clock	Supports local clock generation
Interrupt	Supports generating interrupts from hardware signals
Device Documentation	
PSoC® 4: PSoC 4700S Family Datasheet	All PSoC 4700 Family Technical Reference Manuals
Development Kit Documentation	
CY8CKIT-148 PSoC 4700S Inductive Sensing Evaluation Kit	
CY8CKIT-148-COIL Inductive Sensing Coil Breakout Board	

Document History

Document Title: CE225409 – PSoC 4700S MagSense COIL

Document Number: 002-25409

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6408656	BFMC	04/26/2019	New 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 | 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, 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.