CE210487 - ADC with Breathing LED

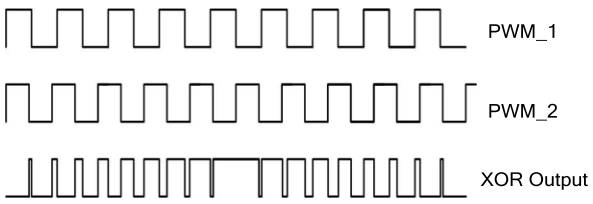
Objective

This code example demonstrates how to measure an input voltage using the CapSense_ADC Component on a PSoC® 4000S device. The ADC output value is used to control the breathing rate of an RGB LED using the Smart IO Component. The ADC value is also sent over I2C to KitProg2, which is accessible on a PC.

Overview

This code example demonstrates the use of the CapSense_ADC Component to measure an input voltage on any I/O pin. The example also shows how to implement a breathing LED using the Smart IO Component. The breathing LED effect is implemented by XORing two pulse-width modulation (PWM) signals that have slightly different frequencies (Figure 1). There are four levels of breathing rates and three different color LEDs. Depending on the ADC result, a specific LED and breathing rate is chosen. The ADC result is sent over I^2C to a host PC running Cypress's Bridge Control Panel (BCP) program.

Figure 1. LED Breathing Effect by XORing Two PWM Signals



Requirements

Tool: PSoC Creator™ 4.0 or later versions

Programming Language: C (ARM® GCC 4.9.3)

Associated Parts: All PSoC 4000S parts

Related Hardware: CY8CKIT-041-40XX PSoC 4 S-Series Pioneer Kit

Design

Figure 2 and Figure 3 show the PSoC Creator schematics of this code example. This example uses the CapSense_ADC, EZI2C Slave, PWM, Pin, Clock, and Smart IO Components.

Figure 2. TopDesign – CapSense_ADC

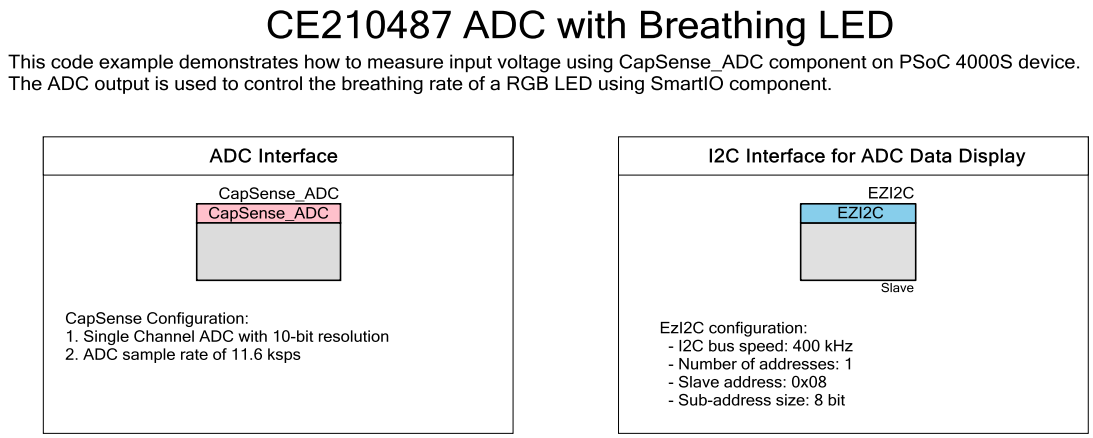
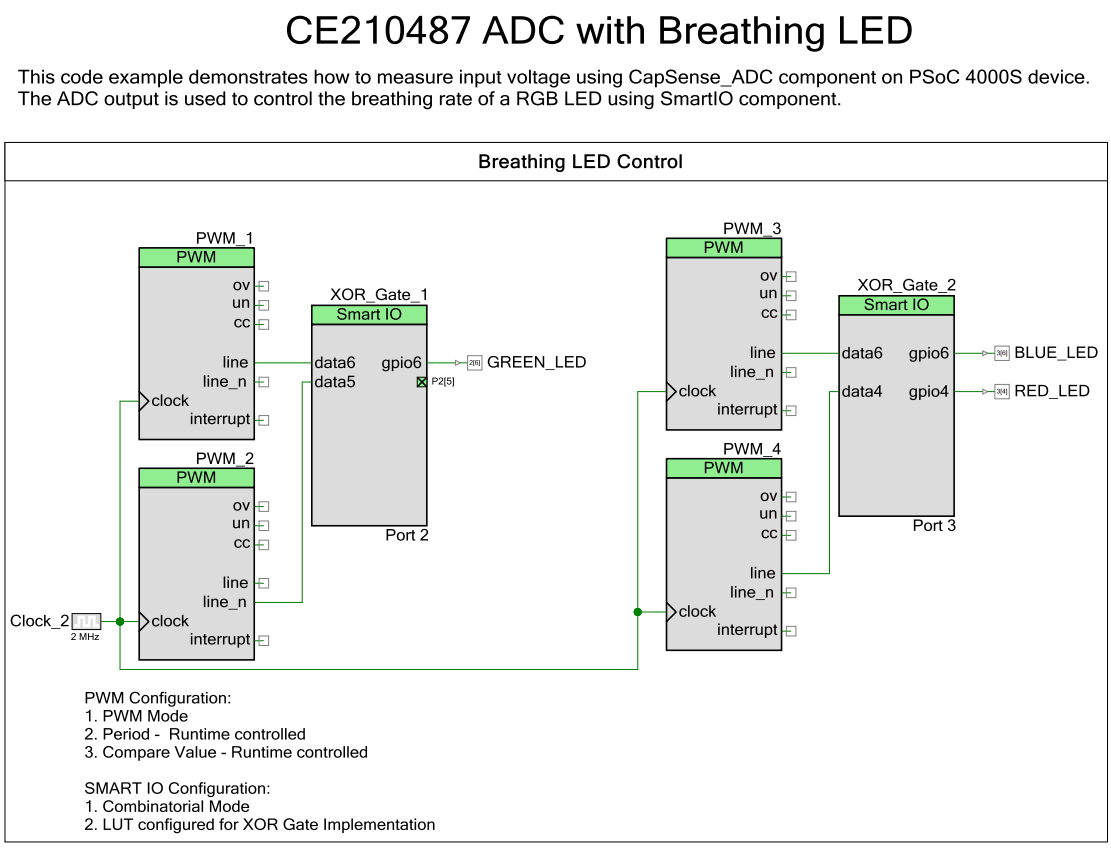


Figure 3. TopDesign – Breathing LED



The CapSense_ADC Component is used to measure the voltage across a potentiometer.

Note: A dummy CapSense button widget is added to the CapSense_ADC Component because it requires at least one CapSense sensor to be configured in order to use the ADC.

As shown in Figure 1, an LED breathing effect is generated by XORing two PWM signals that have slightly different frequencies with a duty cycle of 50%. The XOR operation is implemented using the Smart IO Component. To drive the Green LED, two PWM Components and one Smart IO Component is used; for the Red and Blue LEDs, another set of two PWM Components and a Smart IO Component is used.

The Smart IO Component is configured in combinatorial mode with the Look-Up-Table (LUT) configured for an XOR gate implementation.

The clock input to all the PWM Components is 2 MHz. One of the PWM inputs to the XOR gate has a fixed frequency of 100 Hz (2 MHz/20000). The second PWM input frequency is varied depending on the ADC result. Output frequencies of the second PWM are shown below:

2 MHz/19950 = 100.25 Hz

2 MHz/19800 = 101.01 Hz

2 MHz/19230 = 104.00 Hz

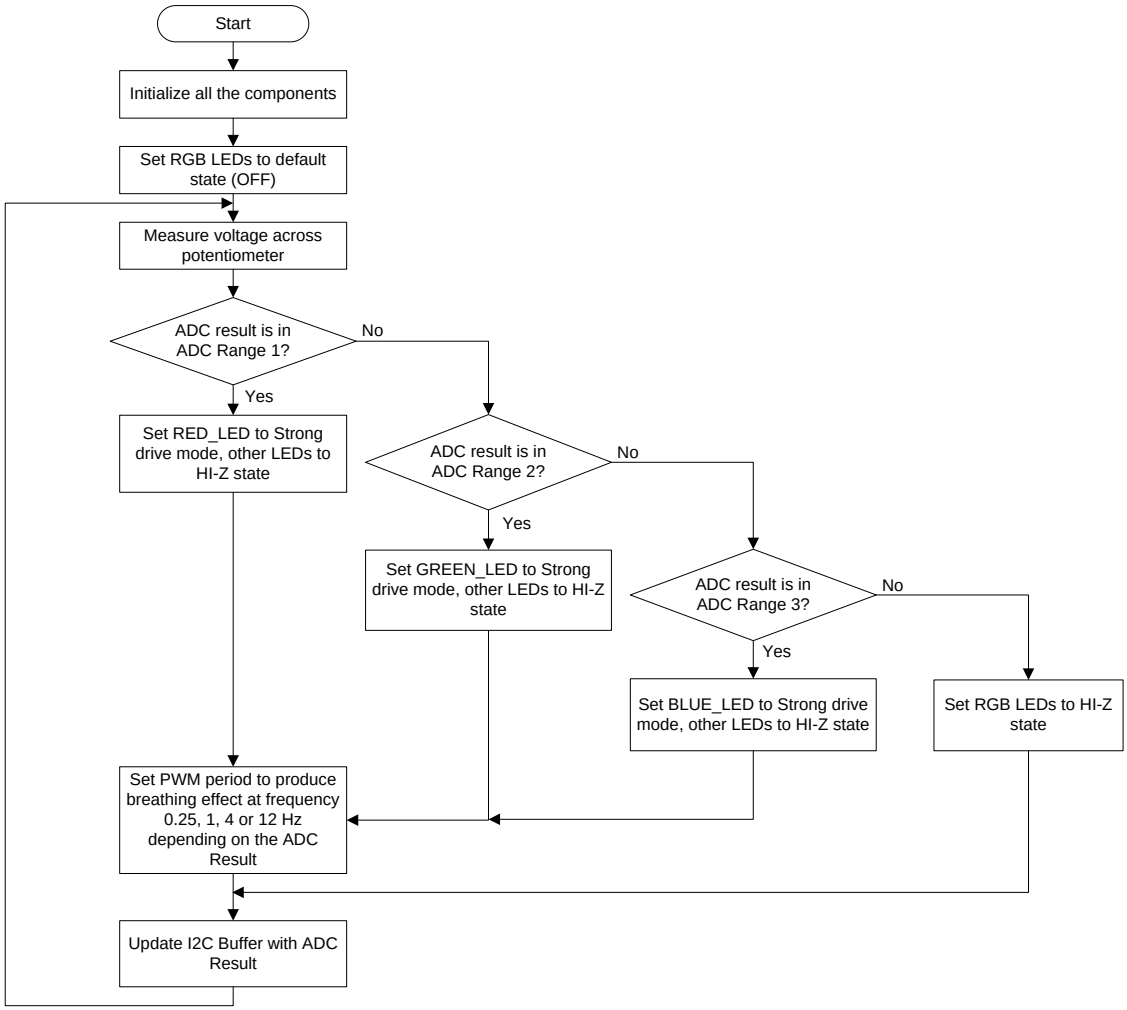
2 MHz/17857 = 112.00 Hz

Routing these PWM signals through an XOR gate yields an output signal with a gradually changing duty cycle. Driving an LED with this signal results in a “breathing” effect, where the LED gradually gets alternately brighter and dimmer. The rate of change is proportional to the difference between the PWM output frequencies.

The EZI2C Slave Component is used to send the ADC result to the PC. The I2C-to-USB Bridge (KitProg2) on the kit is used to transfer the data from the PSoC I^2C interface to the PC's USB interface. The Cypress Bridge Control Panel program can be used to view the ADC result on the PC.

Figure 4 shows the firmware flowchart of this code example.

Figure 4. Firmware Flowchart



Design Considerations

This code example is designed to run on CY8CKIT-041-40XX with the PSoC 4000S device. To port the design to other PSoC 4 devices and kits, you must change the target device in Device Selector, and change the pin assignments in the cydwr settings.

Note: Because the forward voltage of the Blue and Green LEDs is 3.0 V, they will not turn ON if the kit is configured for operation below 3.3 V.

Hardware Setup

The code example works with the default settings on the CY8CKIT-041-40XX PSoC 4 S-Series Pioneer kit. If the settings are different from the default values, see the “Switches Default Position” table in the kit guide to reset to the default settings.

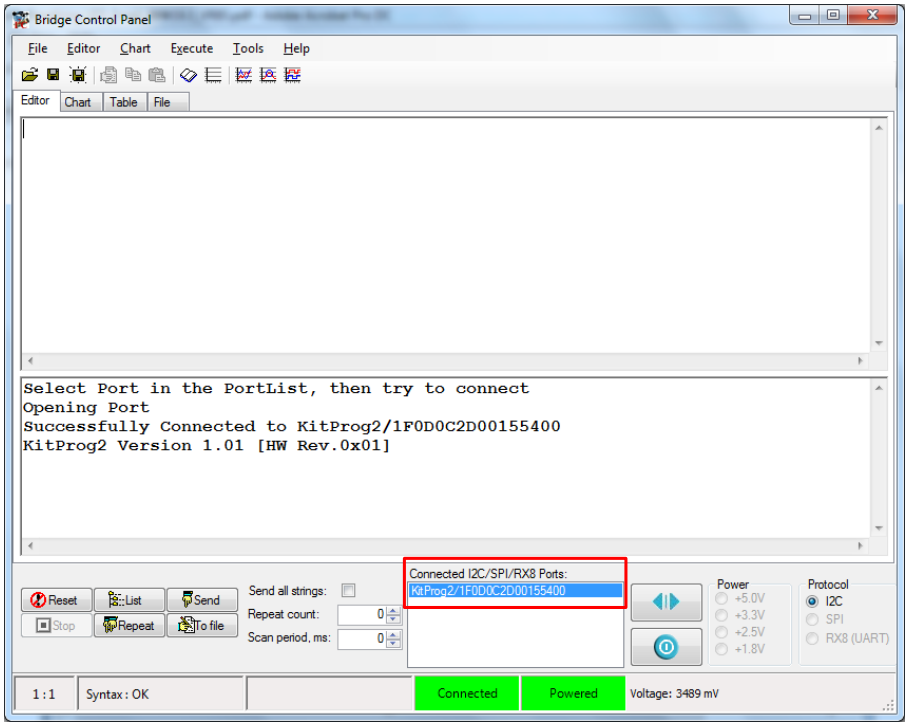
Software Setup

This section describes how to set up the Cypress Bridge Control Panel (BCP) program for viewing the ADC result sent over I^2C.

The BCP software is installed automatically as part of the kit software installation. Follow these steps to configure the BCP software:

1. Open BCP (**Start** > **All Programs** > **Cypress** > **Bridge Control Panel <version>** > **Bridge Control Panel <version>**).
2. Select **KitProg2/<serial number>** under **Connected I2C/SPI/RX8 Ports** (see Figure 5). Note that the PSoC 4 S-Series Pioneer Kit must be connected to the USB port of your computer.

Figure 5. Bridge Control Panel



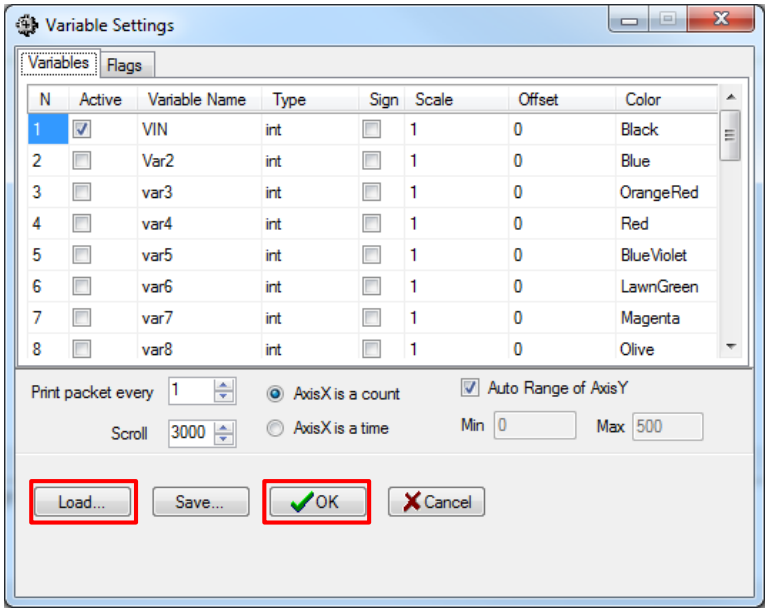
3. Go to **Tools** > **Protocol Configuration**, navigate to the **I2C** tab, and set the **I2C speed** to ‘400 kHz’. Click **OK**.
4. Select **File** > **Open File** and open the *CE210487 ADC with Breathing LED.iic* file from the following path:
   <Install_Directory>\CY8CKIT-041-40XX PSoC 4 S-Series Pioneer Kit\<version>\Firmware\PSoC 4000S\BCP Command\.

5. Select **Chart** > **Variable Settings** and **Load** the *CE210487 ADC with Breathing LED.ini* file from the following folder. Click **OK** (see Figure 6).

   <Install_Directory>\CY8CKIT-041-40XX PSoC 4 S-Series Pioneer Kit\<version>\Firmware\PSoC 4000S\BCP Command\.

   This file includes the variable name, its data type, and signs, to represent the data sent over I^2C.

Figure 6. Variable Settings in Bridge Control Panel Software



The BCP software is now ready for reading the ADC result via I^2C interface. See the Operation section for evaluating the code example.

PSoC Creator Components

Table 1 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each Component.

Table 1. List of PSoC Creator Components

Component	Instance Name	Version	Hardware Resources
CapSense_ADC	CapSense_ADC	v3.10	CSD, 3[1] GPIO pin
EZI2C Slave (SCB mode)	EZI2C	v3.20	SCB, 2 GPIO pins
Digital Output Pin	RED_LED, GREEN_LED, BLUE_LED	v2.20	1 GPIO pin each
PWM (TCPWM mode)	PWM_1, PWM_2, PWM_3, PWM_4	v2.10	1 TCPWM each
Smart IO	XOR_Gate_1, XOR_Gate_2	v1.0	1 Smart IO block each, 1[2] GPIO Pin
Clock	Clock_2	v2.20	1 Clock Divider

[1] The CapSense_ADC Component requires at least one sensor to be configured for proper operation. Therefore, one pin is used for sensor and one for CMOD connection.

[2] XOR_Gate_1 consumes one pin P2[5] because the PWM_2 signal is connected through this pin internally.

Parameter Settings

CapSense_ADC

Figure 7 and Figure 8 show the CapSense_ADC Component that is changed from default values. See the CapSense ADC component datasheet for additional information.

Figure 7. CapSense ADC Component Configuration- Enable At least One Sensor

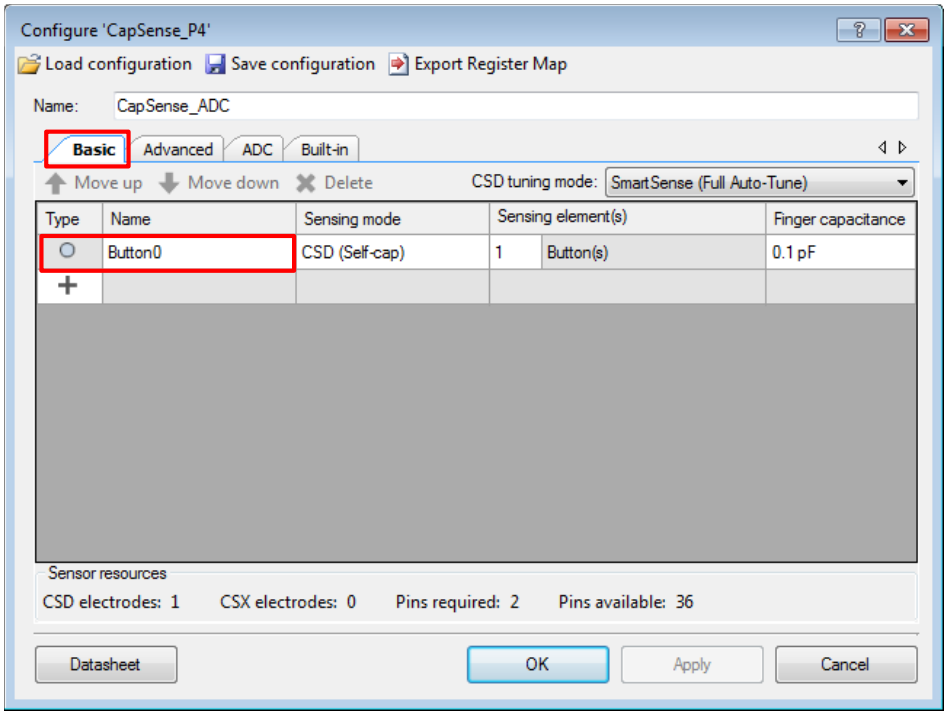
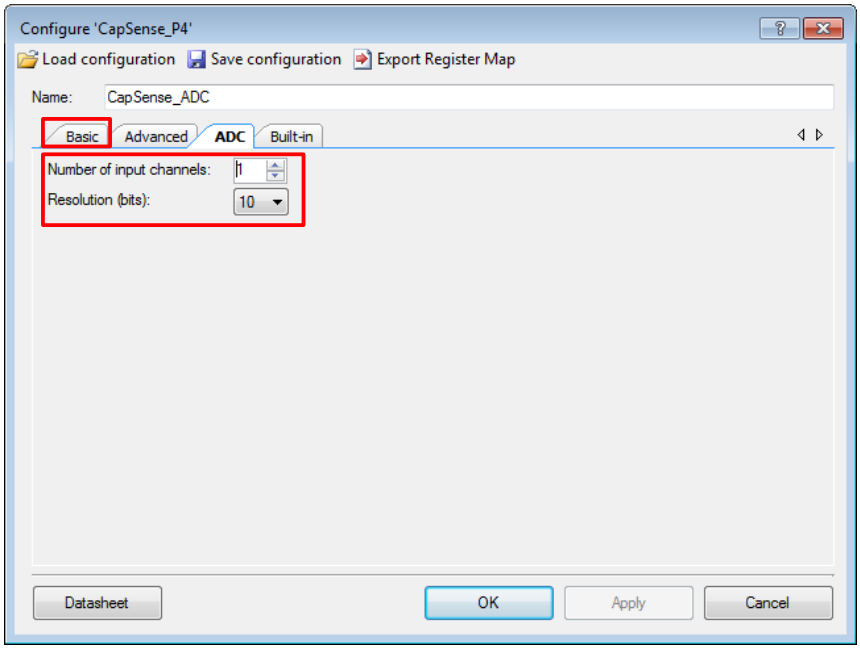


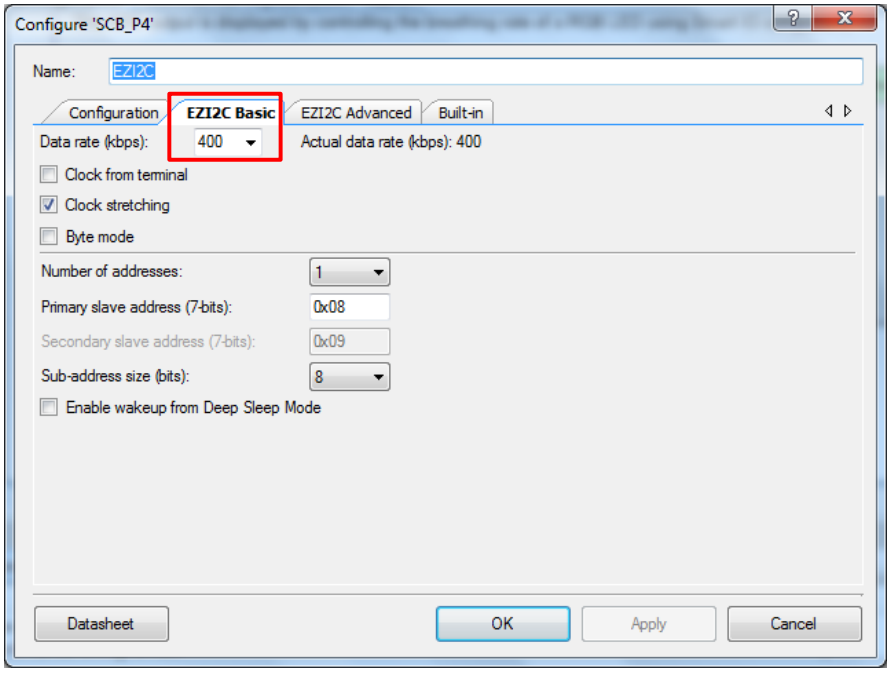
Figure 8. ADC Configuration



EZI2C Slave

Figure 9 shows the EZI2C Slave Component settings that are changed from their default values. See the SCB Component datasheet for additional information.

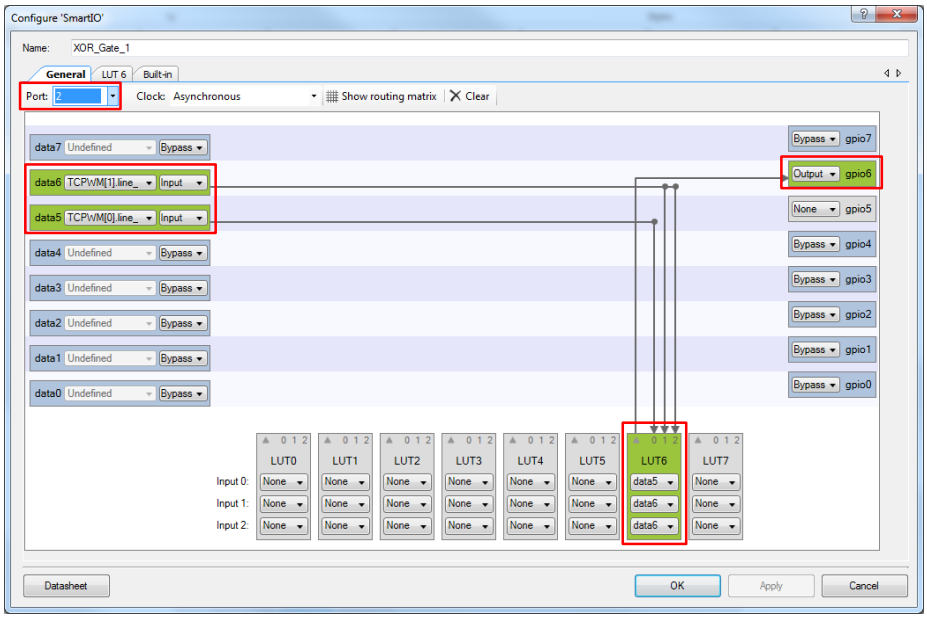
Figure 9. EZI2C Slave Component Configuration



Smart IO

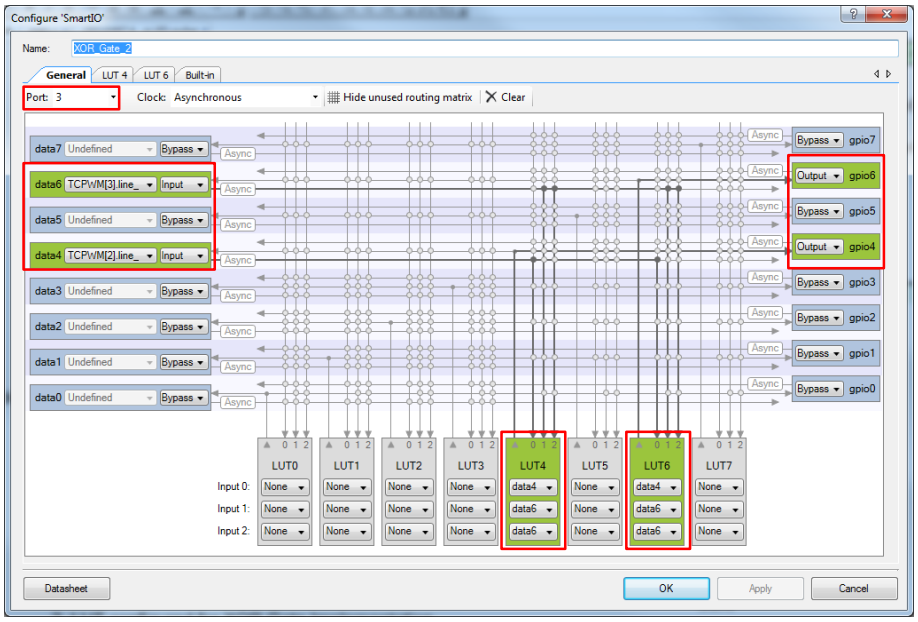
Figure 10 , Figure 11, and Figure 12 show the Smart IO Component settings that are changed from default values. See the Smart IO Component datasheet for additional information.

Figure 10. Configuration of Smart IO Component for Port2



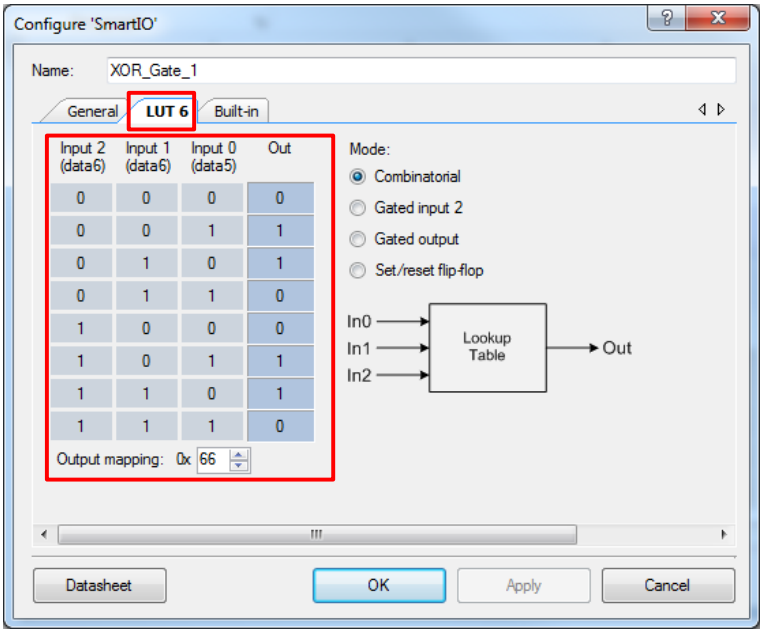
In the above figure, TCPWM0 line_out_compl and TCPWM1 line_out are routed to the LUT6 inputs. The LUT6 output is then routed to a pin to drive the Green LED.

Figure 11. Configuration of Smart IO Component for Port3



In the above figure, TCPWM3 line_out and TCPWM2 line_out are routed to two LUT (LUT4, LUT6) inputs. The LUT outputs are then routed to a pin by the Smart IO Component to drive the Red and Blue LEDs.

Figure 12. LUT[3] Configuration of Port2 and Port 3 Smart IO to Implement XOR Gate



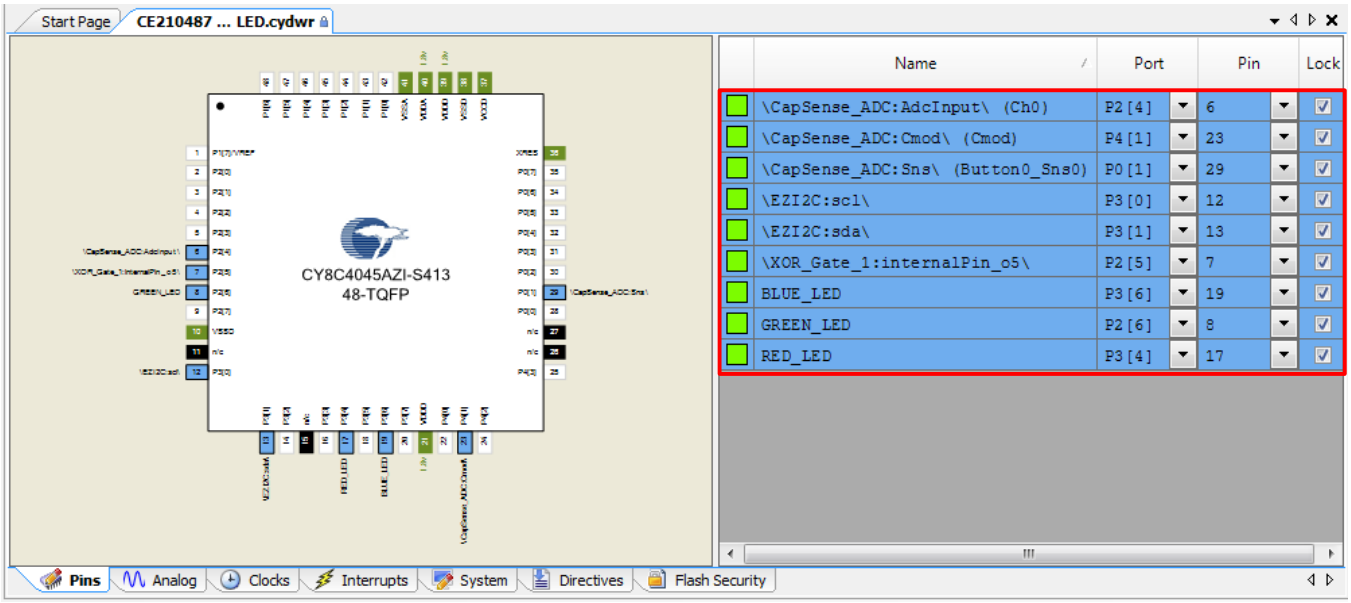
Note that there are only two unique inputs to the LUT (data6 and data5) because Input 2 and Input1 are both data6. Therefore, the output values are set based only on the XOR combination of Input 1 and Input 0. Also, the four rows in which Input 2 and Input 1 are not the same will never occur because those inputs are the same signal.

[3] LUT6 in XOR_GATE1 and LUT4, LUT6 in XOR_GATE2 have the same LUT configuration.

Design-Wide

Figure 13 shows the non-default cydwr settings for the project.

Figure 13. CYDWR Pins Tab Settings



Operation

1. Select the *CE210487 ADC with Breathing LED.cywrk* file in the PSoC Creator Start Page, under **Examples and Kit**s > **Kit**s > **CY8CKIT-041-40XX**. Select a location to save the code example.
2. Build the project (**Build** > **Build CE210487 ADC with Breathing LED**).
3. Connect the PSoC 4 S-Series Pioneer Kit to your computer using the USB cable provided.
4. Program the PSoC 4000S device (**Debug** > **Program**). See the kit guide for details on programming the kit.
5. Vary the input voltage slowly by rotating the potentiometer (R2) wheel counter-clockwise and observe that the RGB LED glows in the following order: Red → Green → Blue. Also, notice that when the input voltage is varied, the breathing rate of the LED increases.

   Note: When the potentiometer wheel is rotated, wait for few seconds to observe slow breathing rates.
6. Configure the BCP software as described in the Software Setup section.
7. In the BCP software, click the read command on the **Editor** tab and then click the **Repeat** button, as shown in Figure 14.
8. Click on the Chart tab and rotate the potentiometer wheel to observe the ADC result in the graph view, as shown in Figure 15.

See BCP Help for more information on using the Bridge Control Panel software.

Figure 14. Command for Reading Data from EZI2C Slave Device

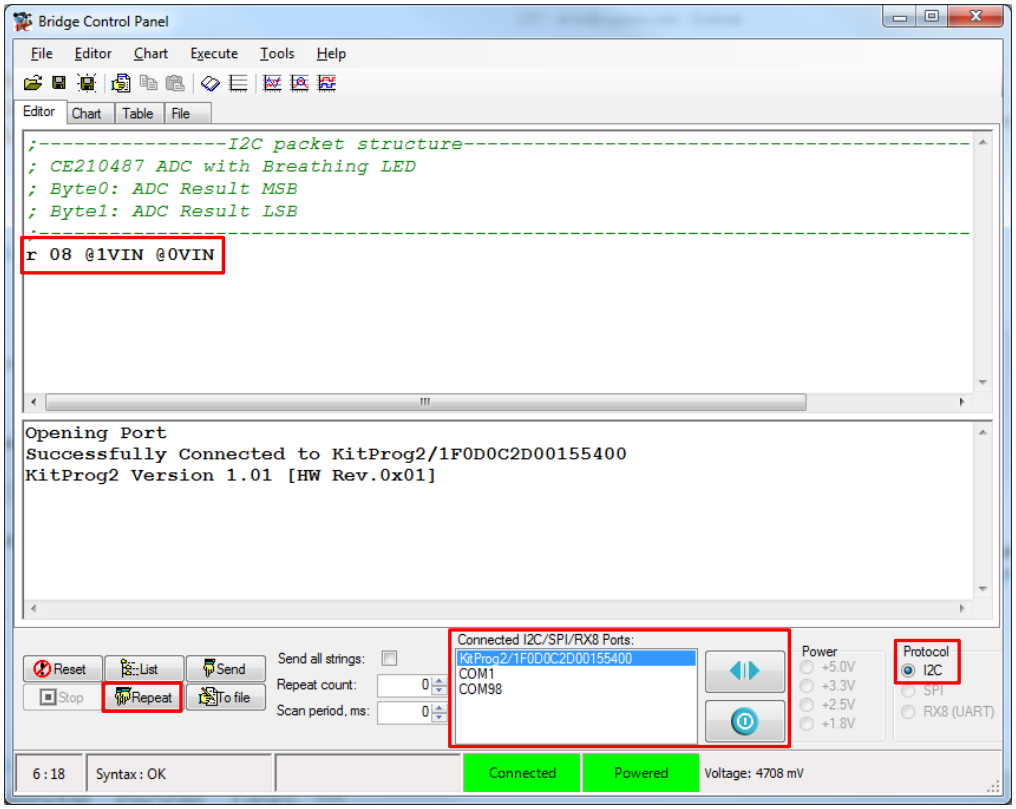
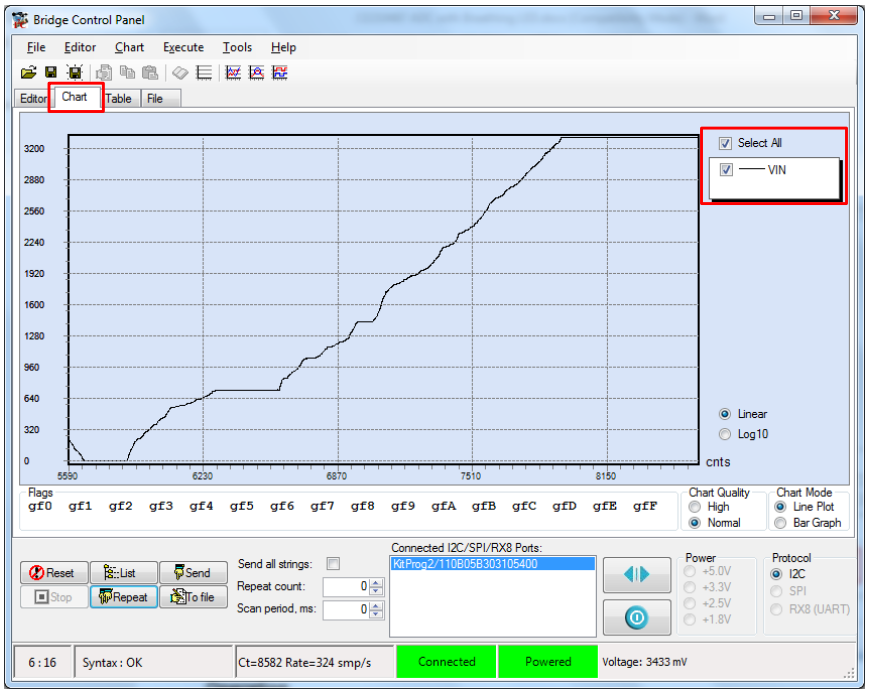


Figure 15. BCP – Graph View of the ADC Data Read from EZI2C Slave



Upgrade Information

The code example is updated to the latest version of PSoC Creator and therefore does not require an upgrade.

Related Documents

Table 2 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and PSoC Creator Component datasheets.

Table 2. Related Documents

Application Notes		
AN79953	Getting Started with PSoC 4	Describes PSoC 4, and how to build your first PSoC Creator project.
PSoC Creator Component Datasheets		
CapSense_ADC	Supports voltage measurement on any I/O pin	
EZI2C Slave	Supports I2C slave operation	
Smart IO	Allows implementing logic gates	
PWM	Supports 16-bit fixed-function PWM implementation	
Pins	Supports connection of hardware resources to physical pins	
Clock	Supports local clock generation	
Device Documentation		
PSoC 4000S Family Datasheet		
PSoC 4000S Family PSoC 4 Architecture Technical Reference Manuals		
Development Kit (DVK) Documentation		
CY8CKIT-041-40XX PSoC 4 S-Series Pioneer Kit		

PSoC Resources

Cypress provides a wealth of data at www.cypress.com to help you to select the right PSoC device for your design, and quickly and effectively integrate the device into your design. For a comprehensive list of resources, see KBA86521, How to Design with PSoC 3, PSoC 4, and PSoC 5LP. The following is an abbreviated list for PSoC 4:

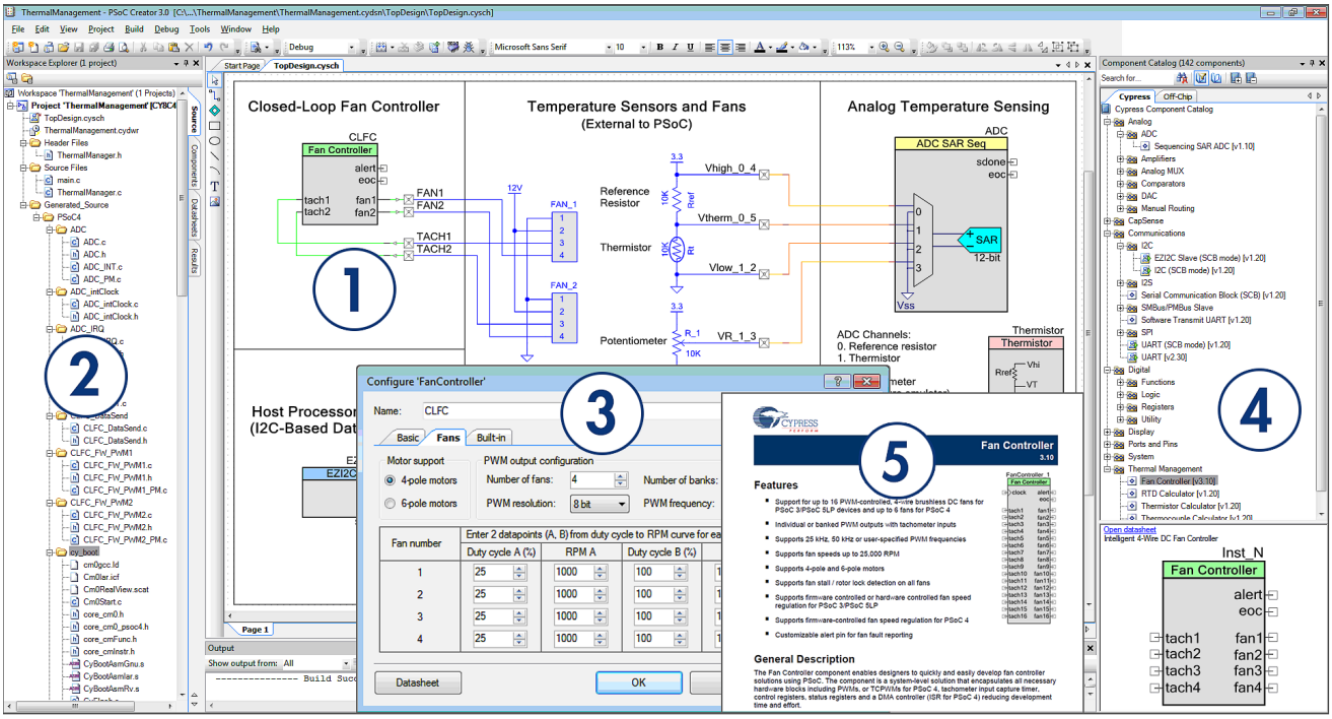
- **Overview:** PSoC Portfolio, PSoC Roadmap
- **Product Selectors:** PSoC 1, PSoC 3, PSoC 4, or PSoC 5LP. In addition, PSoC Creator includes a device selection tool.
- **Datasheets** describe and provide electrical specifications for the PSoC 3, PSoC 4, and PSoC 5LP device families.
- **CapSense Design Guides:** Learn how to design capacitive touch-sensing applications with the PSoC 3, PSoC 4, and PSoC 5LP families of devices.
- **Application Notes** and **Code Examples** cover a broad range of topics, from basic to advanced level. Many of the application notes include code examples.
- **Technical Reference Manuals (TRM)** provide detailed descriptions of the architecture and registers in each of the PSoC 3, PSoC 4, and PSoC 5LP device families.
- **PSoC Training Videos**: These videos provide step-by-step instructions on getting started building complex designs with PSoC.
- **Development Kits:**
  - CY8CKIT-041 PSoC 4 S-Series Pioneer kit is easy-to-use and inexpensive development platform. This kit include connectors for Arduino™ compatible shields.
  - CY8CKIT-145 is a very low-cost prototyping platform for evaluating the PSoC 4 S-Series of devices.
- The MiniProg3 device provides an interface for flash programming and debug.

PSoC Creator

PSoC Creator is a free Windows-based Integrated Design Environment (IDE). It enables concurrent hardware and firmware design of systems based on PSoC 3, PSoC 4, and PSoC 5LP. See Figure 16– with PSoC Creator, you can:

1. Drag and drop Components to build your hardware system design in the main design workspace
2. Codesign your application firmware with the PSoC hardware
3. Configure Components using configuration tools
4. Explore the library of 100+ Components
5. Review Component datasheets

Figure 16. PSoC Creator Features



Document History

Document Title: CE210487 - ADC with Breathing LED

Document Number: 002-10487

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5269276	SSHH	05/12/2016	New code example
*A	5524670	SRDS	11/18/2016	Updated template

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
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corporation. 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

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("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. 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. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, 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 or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

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.