# Code Example for SmartSense™ User Module in PSoC® 1

**Project Name**: CE85976
**Programming Language**: C
**Associated Part Families**: CY8C22x45, CY8C21x45
**Software Version**: PSoC® Designer™ 5.4
**Related Hardware**: CY3280-22x45 UCC Board,CY3240-I2USB Bridge or MiniProg3, MiniProg1 or MiniProg3
**Author**: Tejender Sheoran

## Project Objective

This project demonstrates the operation of the SmartSense2X User Module (UM) of the PSoC 1 device.

## Overview

The proper operation of capacitive sensors requires you to manually tune several parameters depending on the parasitic capacitance ($C_P$). The SmartSense UM improves and expands the basic architecture of the CSD UM. A key feature is SmartSense Auto-Tuning. The SmartSense UM optimizes the parasitic capacitance operational parameters during runtime based on the $C_P$ of each sensor; you do not need to manually tune when you use SmartSense. This code example demonstrates how to use the SmartSense User Module. We use the EzI2Cs User Module to send CapSense® parameters, such as raw counts, difference counts, baselines of each button, and centroid position of slider, to the $I^2C$ master.

## User Module List and Placement

The following table lists the user modules used in this project and the hardware resources occupied by each user module.

| User Module | Placement |
|---|---|
| EzI2Cs | No analog or digital block is used (A dedicated $I^2C$ hardware resource is used) |
| SmartSense2X | Capsense0, ACE02 (SmartSense2X CMP) |
| LED_0 | P2[5] |
| LED_1 | P2[7] |
| LED_2 | P0[1] |
| LED_3 | P0[3] |
| LED_4 | P1[2] |

## User Module Parameter Settings

The following tables show user module parameter settings for each of the user modules used in the project.

| EzI2Cs | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| Slave_Addr | 0 | This parameter assigns the Slave address. The value assigned can be any value from 0 to 127 (decimal) |
| Address_Type | Static | Slave address is not dynamically varied. |
| ROM_Registers | Disable | ROM read is not used in this example. |
| I2C_Clock | 100K Standard | Master and slave are communicating at a speed of 100 KHz during Byte transfer phase. |
| I2C_Pin | P[1]0-P[1]1 | P[1]1 → SCLK<br>P[1]0 → SDATA |

| SmartSense2X | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| Sensors Autoreset | Disabled | |
| Debounce | 3 | Sensor is ON only when it is active for 3 scans |
| Background Scanning | Enabled | Relieve the CPU when scanning is in progress |
| Buttons | 5 | SW0- P3[1] |
| | | SW1- P1[3] |
| | | SW2- P3[3] |
| | | SW3- P2[1] |
| | | SW4- P2[3] |
| Buttons Sensitivity | 0.4(Med-Low) | |
| Sliders | 1 | |
| Slider Sensors Count | 10 | S1(0)-P1[4] |
| | | S1(1)-P0[6] |
| | | S1(2)-P0[4] |
| | | S1(3)-P0[2] |
| | | S1(4)-P2[6] |
| | | S1(5)-P2[4] |
| | | S1(6)-P2[2] |
| | | S1(7)-P2[0], |
| | | S1(8)-P3[2], |
| | | S1(9)-P3[0] |
| Slider Resolution | 50 | |
| Diplex | False | |
| Slider Sensitivity Level | 0.3(Medium) | |
| Radial Sliders | 0 | |
| Proximity Count | 0 | |
| Shield Electrode Output | GOO[6] to P1[6] | |
| Modulator Capacitor Pin | P0[5] | |
| Threshold Setting | Auto tune based on Sensitivity | |

| LED | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| Drive | Active Low | |
| LED_0 | P2[5] | |
| LED_1 | P2[7] | |
| LED_2 | P0[1] | |
| LED_3 | P0[3] | |
| LED_4 | P1[3] | |

## Global Resources

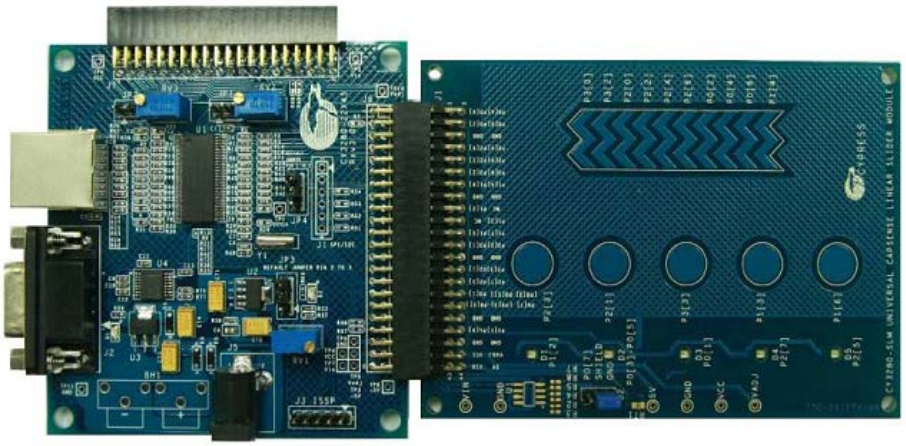
| Important Global Resources | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| Power Setting (VCC/Sys.Clock Frequency) | 5.0 V/24 MHz | Sys.Clock = 24-MHz clock source accuracy is trimmed for 5-V power setting. |
| CPU Clock | SysClk/2 | Sets the CPU frequency to 12 MHz |
| 32K_Select | Internal | |
| PLL_Mode | Disable | |

All other global resources are left at their default, as they are not specific to this project.

## Hardware Connections

1. Connect the CY3280-SLM Universal CapSense Linear Slider Module board to the parent board as shown in Figure 1.

   Figure 1. Connecting the CY3280-SLM Universal CapSense Linear Slider Module Board to the Parent Board



2. Place shunts on pins 2 and 3 of JP3 and pins 1 and 2 of JP4 (default setting).
3. Connect your computer to the CapSense test board's in-system serial programming (ISSP) connector (J3) using the MiniProg1 or MiniProg3 and a USB cable. If you are using the MiniProg1 or MiniProg3 for the first time, install the driver using the following steps before proceeding:
   a. When the **Found New Hardwar**e Wizard opens, select the **Install the software automatically (Recommended)** option and click **Next**.
   b. A warning message may tell you the software you are trying to install has not passed Windows Logo testing. Click **Continue Anyway** each time it appears.
   c. When the installation is complete, click **Finish**.
4. Click **Start** > **All Programs** > **Cypress** > **PSoC Programmer <version xxx>** > **PSoC Programmer <version xxx>**.
5. Click **File Load**, navigate to and open the *CY3280_22x45_SmartSense_Example_Project.hex* file.
6. From the **Device Family** menu, select CY8C22X45.
7. From the **Device** menu, select CY8C22545-24AXI.
8. Click **Program**. **"Programming Succeeded..."** appears in the Actions pane when programming is complete.

## Operation

Upon program execution, all hardware settings from the device configuration are loaded into the device and the code is executed.

The following operations are performed in *main.c*.

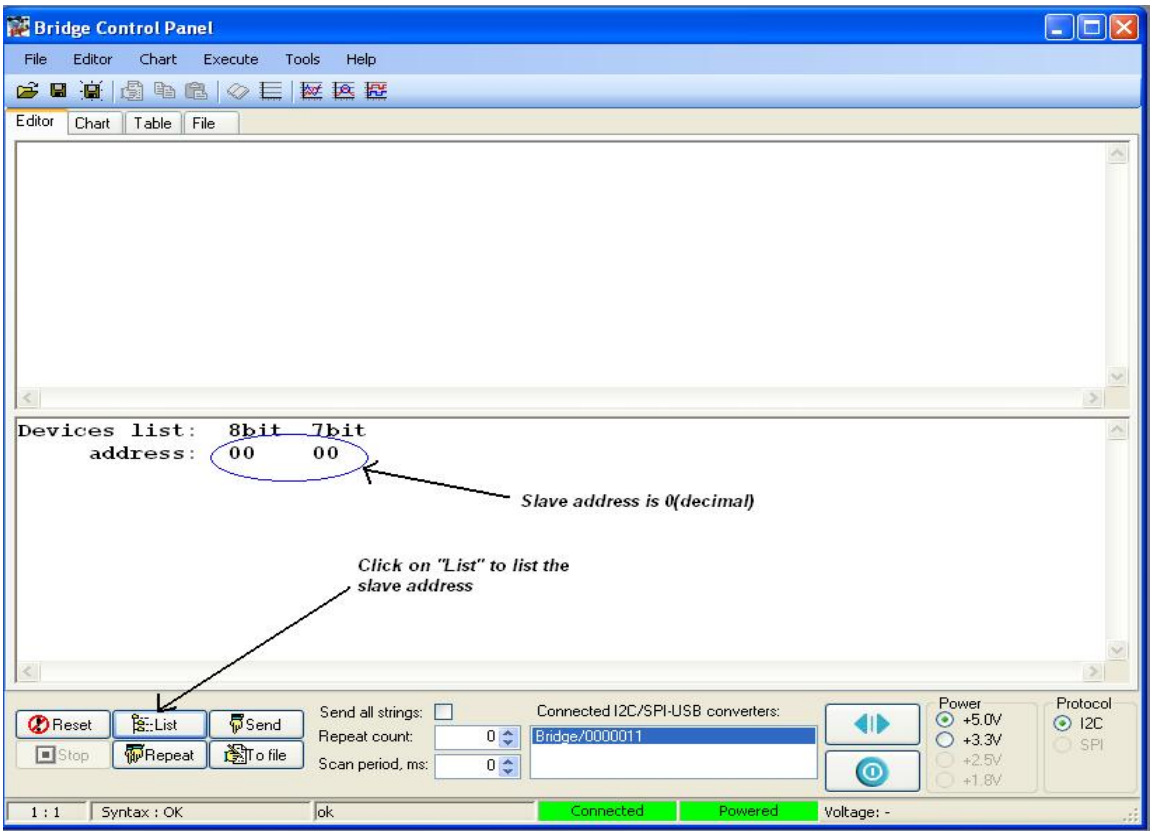
1. Enable Global interrupts.
2. Set EzI2Cs buffer - exposes 5 bytes of RAM locations to the $I^2C$ master using EzI2Cs_SetRamBuffer API and starts EzI2Cs.
3. Start SmartSense2X and initialize baselines.
4. Set scanningFlag to scan first sensor.
5. If scanningFlag is scan first sensor, initialize capSensorNo to 0 (first sensor index), scan it, and set scanningFlag to first sensor scan done.
6. If scanningFlag is first sensor scan done and scanning is over, clear the scan complete flag, update sensor baseline, and check if sensor is active.
7. Increment capSensorNo.
8. If capsensorNo is less than total sensors, scan the next sensor; if not, get slider centroid, control LEDs and set scanningFlag to scan first sensor.
9. Load the fresh data (rawcount, baseline, diff, and centroid) in the $I^2C$ buffer.
10. Go to step 5.

## Testing the Project

1. Touch any button; the corresponding LED turns ON. When the finger is removed, the LED turns OFF.
2. Touch the slider. As the finger moves upwards, the number LEDs that are ON will increase. When the last segment of the slider is touched, all the LEDs are turned ON.
3. To test the SmartSense working, use the CY3240-I2USB Bridge or MiniProg3 with associated Bridge Control Panel software as the master. Make external jumper connections in the slave side as explained in the section Hardware Connections.
4. Connect the CY3240-I2USB Bridge or MiniProg3 between the PSoC device (using the ISSP header) and the PC (using the USB port). The following settings are required to enter communication commands in the Bridge Control Panel software.
    a. Open the Bridge Control Panel software from the path Start > All Programs > Cypress > Bridge Control Panel 1.6 > Bridge Control Panel 1.6.
    b. Select $I^2C$ Speed of 100 K in the path Tools > Protocol Configuration > I2C in the Bridge Control Panel software.
    c. Select +5 V in the ‘Power Selection’ panel, which is the window below the ‘Results’ window, and click on the ‘Toggle Power’ button (This powers the target device with +5 V).

5. Click on **List** in the window that appears below the **Results** window. This lists the slave address in the **Results** window.

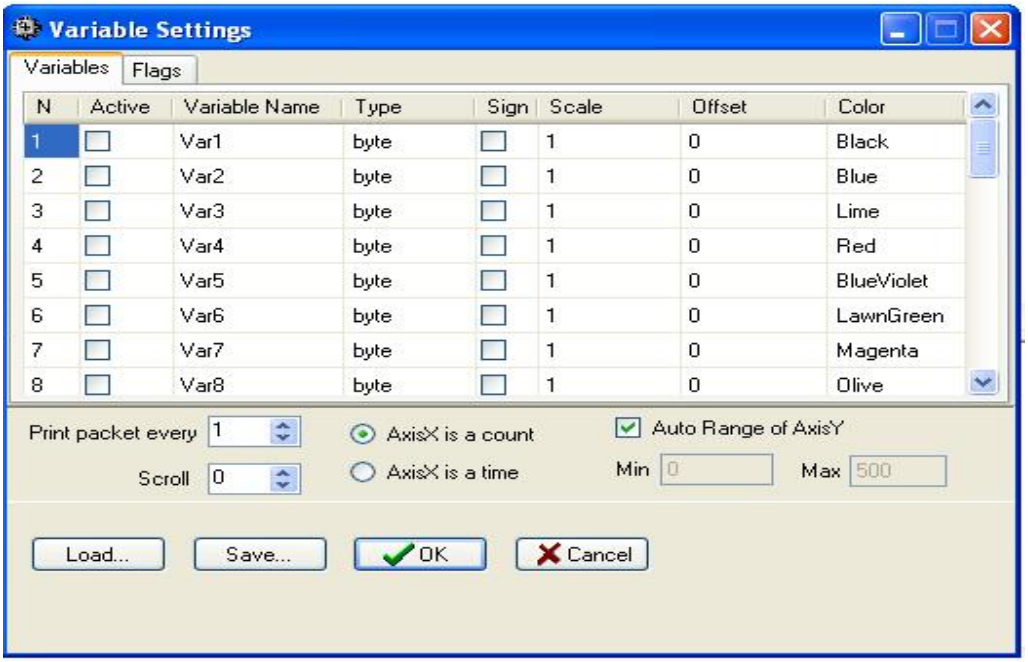
Figure 2. Bridge Control Panel Showing the slave Address



To set the variables in the Bridge Control Panel software, follow these steps:

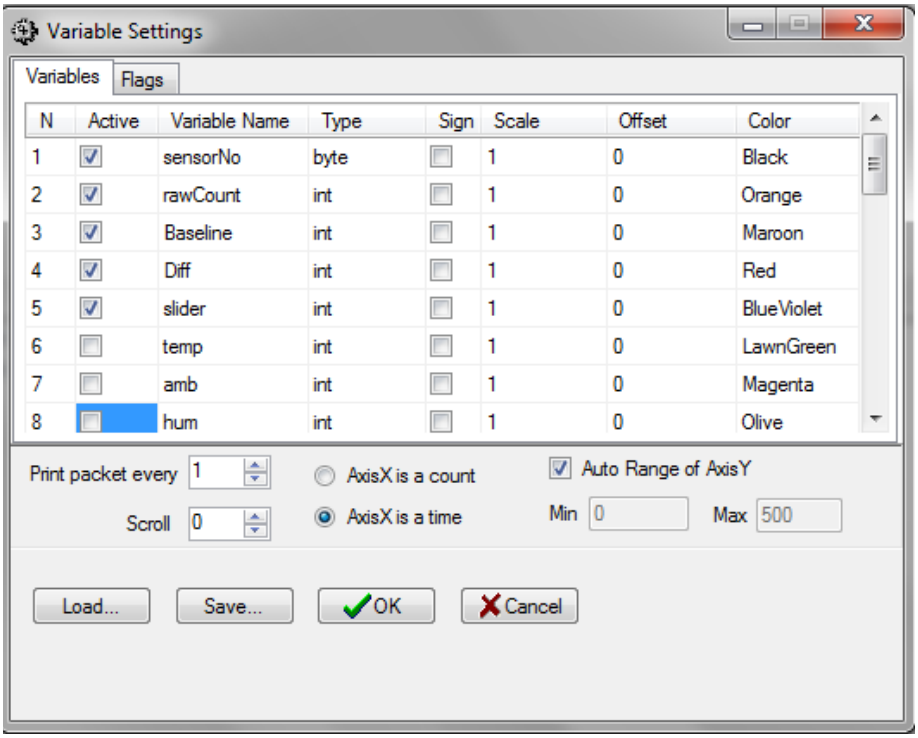
1. Click on **Chart** and select **Variable Settings**. The following window opens.

Figure 3. Variable Settings Window



2. Enter the Variable Name, Type, and Sign as shown in the following screenshot and activate all the variables by checking the box provided.

Figure 4. Variable Settings




3. Click OK.

   **Note** To get the Help document for the Bridge Control Panel software, follow the path **Help** > **Help Contents** in the Bridge Control Panel software.

4. To read a particular sensor's raw count, baseline, and diff value, do the following:

   - First, write the sensor number in the first byte of the $I^2C$ buffer.
   - Next, read the value.

The following screenshot shows the byte values read by the master.

Figure 5. Bridge Control Panel

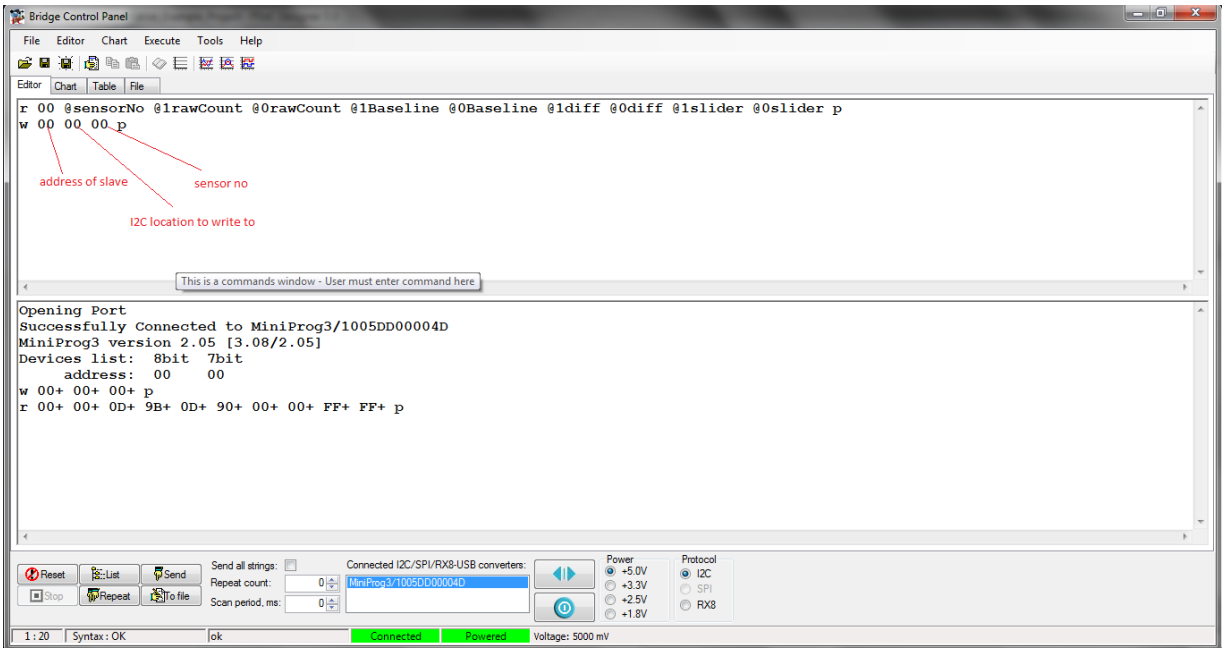
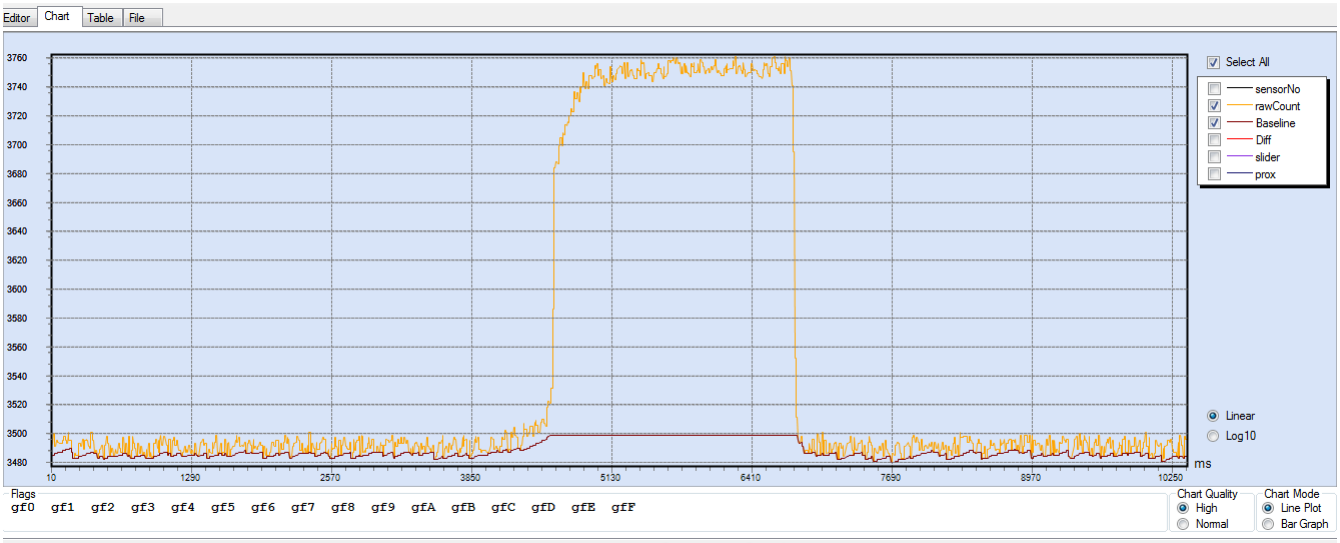



Figure 6. Change in rawcounts with finger touch



# Document History

**Document Title:** Code Example for SmartSense™ User Module in PSoC® 1 - CE85976

**Document Number:** 001-85976

| Revision | ECN | Orig. of Change | Submission Date | Description of Change |
|---|---|---|---|---|
| ** | 3897918 | KUK / TEJU | 02/07/2013 | New code example. |
| *A | 4665772 | ASRI | 02/19/2015 | Updated Software Version as “PSoC® Designer™ 5.4” in page 1.<br>Updated to new template.<br>Completing Sunset Review. |
| *B | 4737457 | SSHH | 04/24/2015 | Updated Document Title to read as “Code Example for SmartSense™ User Module in PSoC® 1 - CE85976”.<br>Updated Related Hardware as “CY3280-22x45 UCC Board, CY3240-I2USB Bridge or MiniProg3, MiniProg1 or MiniProg3” in page 1.<br>Replaced "I2CtoUSB Bridge" with "CY3240-I2USB bridge or MiniProg3" in all instances across the document. |

## 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

| | |
|---|---|
| Automotive | cypress.com/go/automotive |
| Clocks & Buffers | cypress.com/go/clocks |
| Interface | cypress.com/go/interface |
| Lighting & Power Control | cypress.com/go/powerpsoc<br>cypress.com/go/plc |
| Memory | cypress.com/go/memory |
| PSoC | cypress.com/go/psoc |
| Touch Sensing | cypress.com/go/touch |
| USB Controllers | cypress.com/go/usb |
| Wireless/RF | cypress.com/go/wireless |

## PSoC® Solutions

psoc.cypress.com/solutions

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

## Cypress Developer Community

Community | Forums |Blogs | Video |Training

## Technical Support

cypress.com/go/support

PSoC and CapSense are registered trademarks of Cypress Semiconductor Corp. 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
http://www.cypress.com/

© Cypress Semiconductor Corporation, 2013-2015. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.