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.

CE214025 – Trackpad with Color Gamut

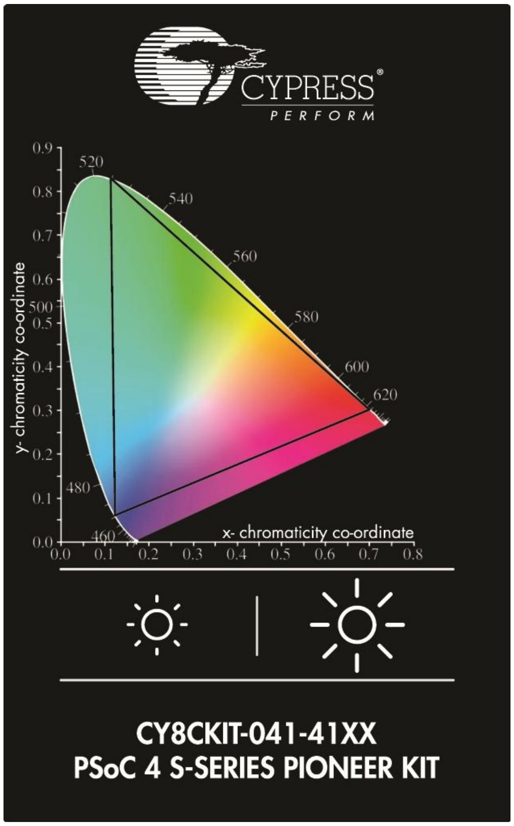
Objective

This code example implements a CapSense®-based trackpad as a user interface to input the required color for the color mixing algorithm.

Overview

This code example implements a CapSense-based trackpad as a user interface. The trackpad has the CIE 1931 color gamut (Figure 1) imprinted; user inputs (touch coordinates) are converted to the corresponding color coordinates. The RGB LED on the board is used to illustrate the chosen color by modulating associated signal densities. The brightness of the RGB LED is controlled by using the two CapSense buttons.

Figure 1. CIE 1931 Color Space Chromaticity Diagram



Requirements

Tool: PSoC Creator™ 4.0 or later versions

Programming Language: C (ARM® GCC 4.9.3)

Associated Parts: All PSoC® 4100S parts

Related Hardware: CY8CKIT-041-41XX PSoC 4100S Pioneer Kit

Design

Figure 2 and Figure 3 show the PSoC Creator schematics of this code example. The code example uses the CapSense, EZI2C Slave, PWM, and Pins Components.

The CapSense Component is configured to scan a 7 x 7 trackpad widget and two button widgets. The trackpad touch coordinates are mapped to the CIE 1931 color space in the firmware and are provided as inputs to the color mixing algorithm. See the Appendix for more details on the CIE 1931 color space and color mixing theory. The two button widgets are used for controlling the RGB LED brightness. The EZI2C Slave Component is used to monitor the sensor data on a PC using the CapSense Tuner available in the PSoC Creator integrated design environment (IDE).

The PWM Component controls the intensity of the RGB LED by driving a pseudo-random PWM signal. The pseudo-random PWM is used to reduce the peak electromagnetic radiation at any specific frequency. The period of the pseudo-random PWM signal is ~6.5 ms (65535/10 MHz). This period value avoids the human eye's ability to sense LED flicker at low intensity levels. The color-mixing process updates the compare value of the pseudo-random PWM to generate the requested color.

Figure 2. TopDesign – CapSense and EZI2C

This code example shows how to implement a trackpad as a user interface to input the color from the color gamut. It also shows how to implement the color mixing functionality and display the resulting color on the RGB LED.

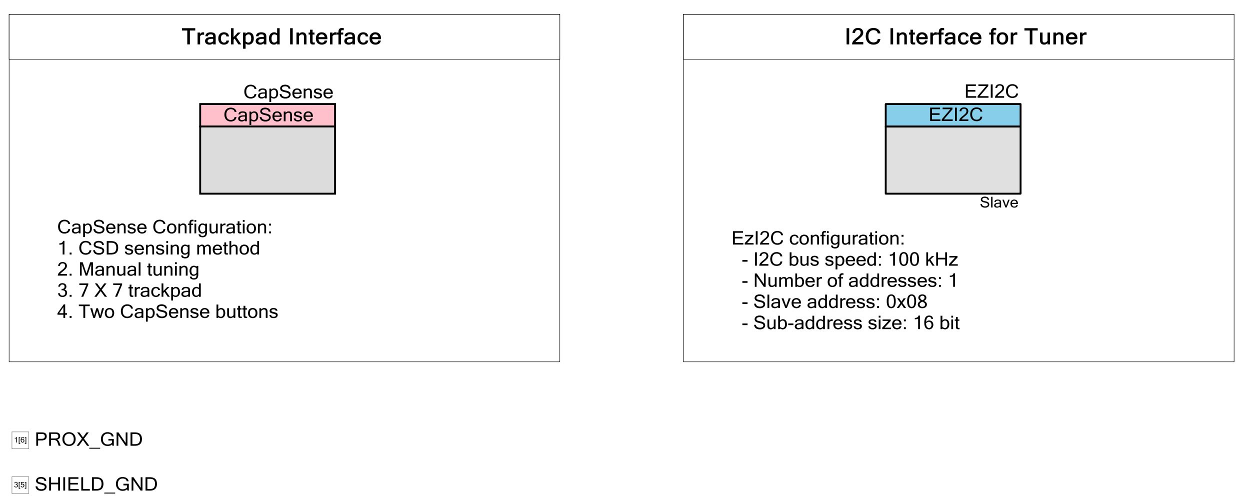


Figure 3. TopDesign – RGB LED Drive

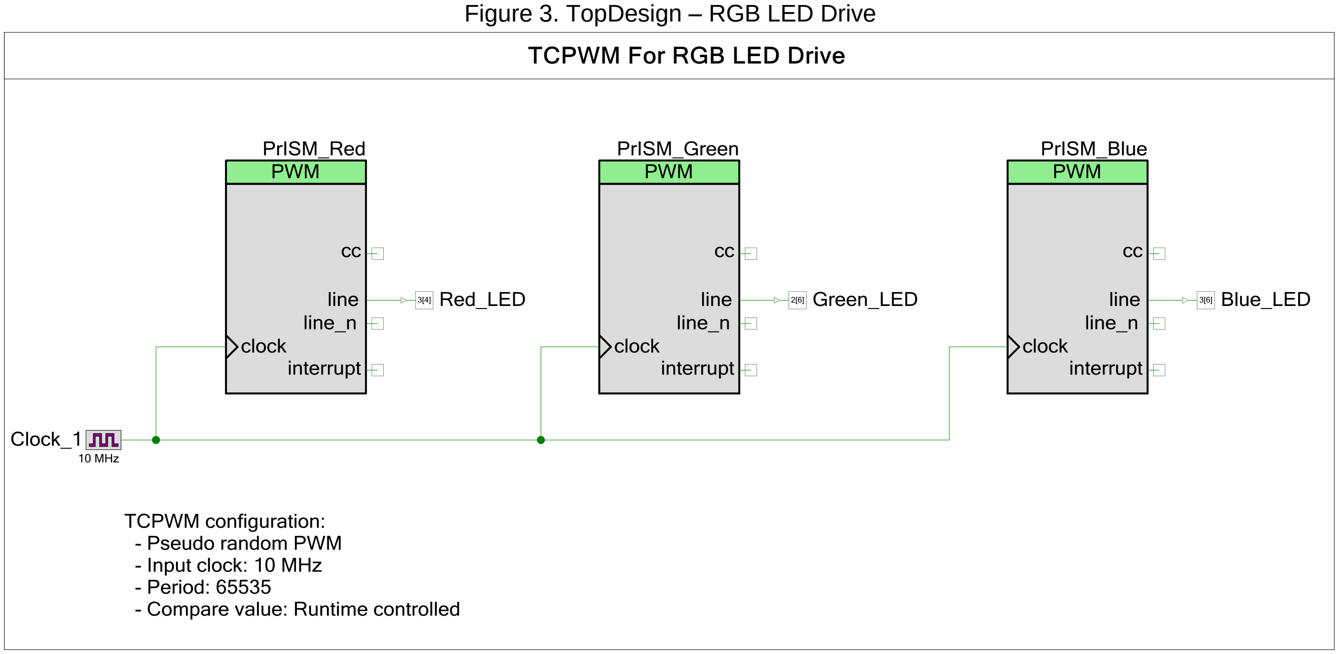
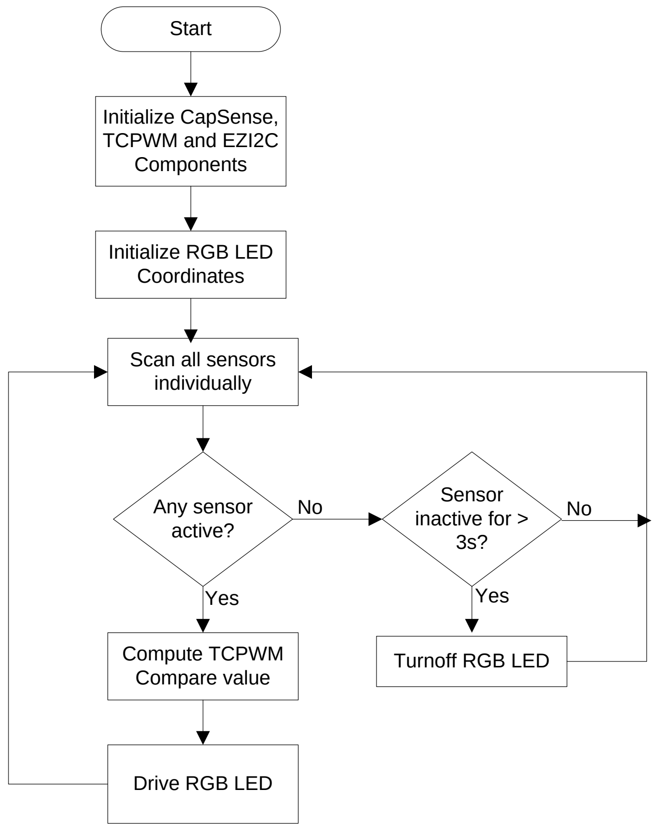


Figure 4. Firmware Flow Chart



Design Considerations

This code example is designed to run on the CY8CKIT-041-41XX PSoC 4100S Pioneer Kit with the PSoC 4100S device. To port the design to other PSoC 4 devices and kits, you must change the target device in the Device Selector, change the pin assignments in the *.cydwr* settings, and retune the CapSense sensors. For the tuning procedure, see the PSoC 4 CapSense Design Guide.

Notes:

1. The color response of the onboard RGB LED is similar to the selected color; the RGB LED might not show the true color for all color combinations because of the limited lumens of the RGB LED.
2. Because the RGB LED current depends on the kit operating voltage, the color response is optimum when the kit is operated at 5 V.

Hardware Setup

The code example works with the default settings on the CY8CKIT-041-41XX PSoC 4100S Pioneer Kit. If the settings are different from the default values, see the “Switches Default Position” table in the kit guide to learn how to reset them to the default settings.

Software Setup

This code example does not require any special software considerations.

Components

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

Table 1. PSoC Creator Components

Component	Instance Name	Version	Hardware Resources
CapSense	CapSense	v3.10	CSD and 18 GPIO pins
EZI2C Slave (SCB mode)	EZI2C	v3.20	SCB, 2 GPIO Pins
PWM (TCPWM mode)	PriSm_Red, PriSm_Green, PriSm_Blue	v2.10	1 TCPWM each
Digital Output Pin	Red_LED, Green_LED, Blue_LED, PROX_GND, SHIELD_GND	v2.20	1 GPIO pin each
Clock	Clock_1	v2.20	1 Clock Divider

Parameter Settings

CapSense

Figure 5 through Figure 9 show the CapSense Component settings that are changed from the default values. See the CapSense Component datasheet for additional information.

Figure 5. CapSense Component – Basic Configuration

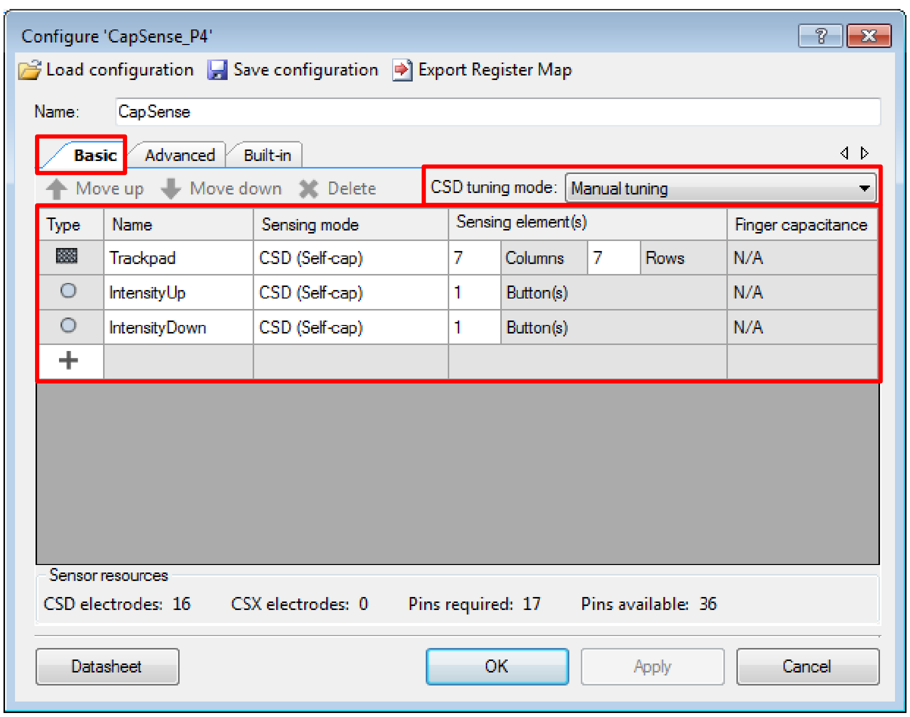


Figure 6. CapSense Component – Advanced Tab General Settings

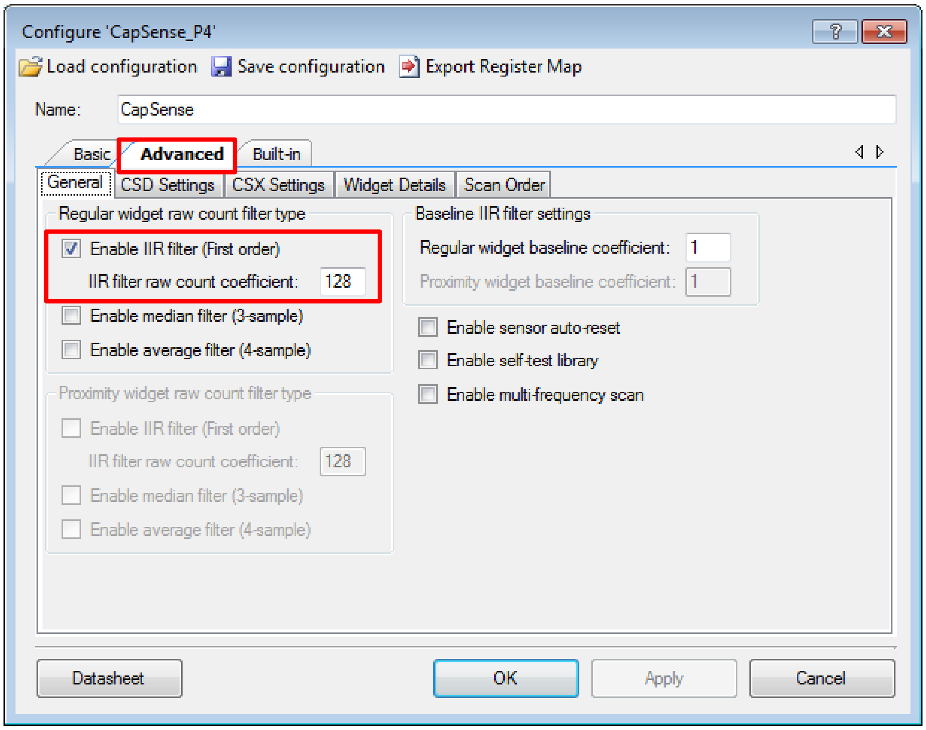


Figure 7. CapSense Component – Advanced Tab CSD Settings

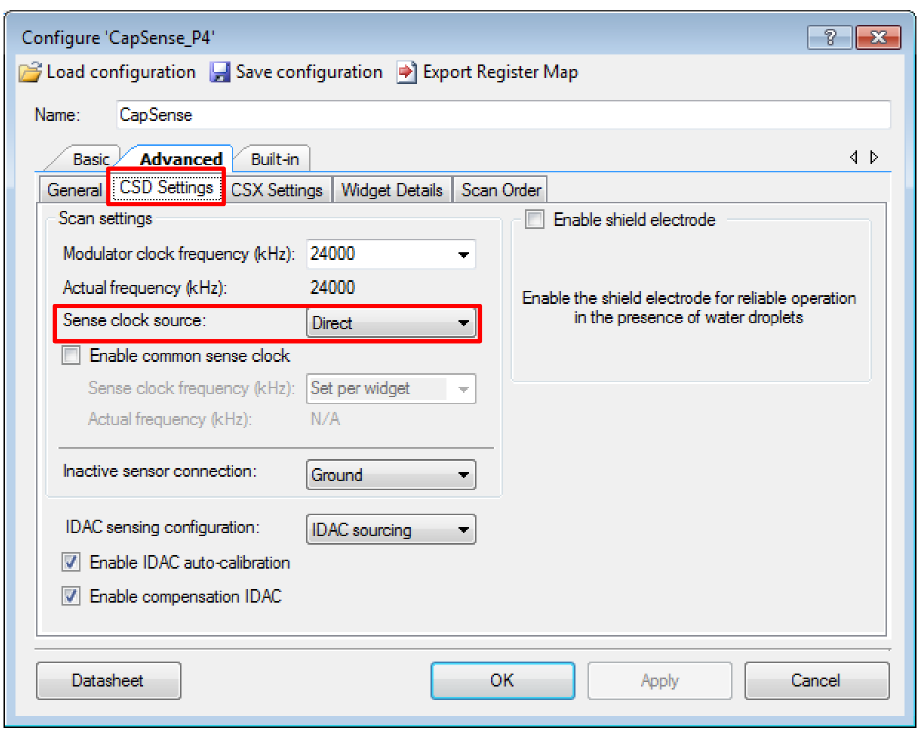


Figure 8. CapSense Component – Advanced Tab Widget Details for Trackpad

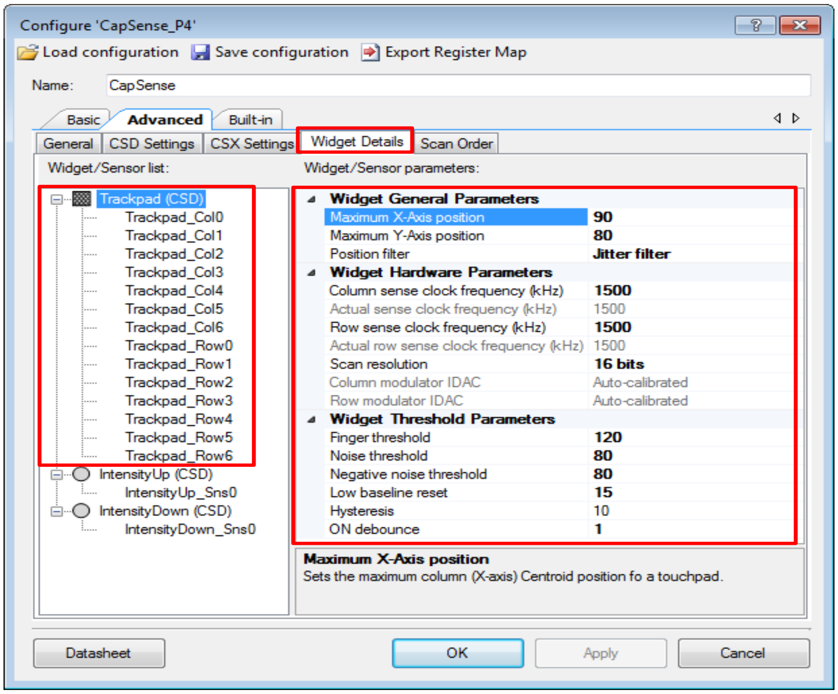
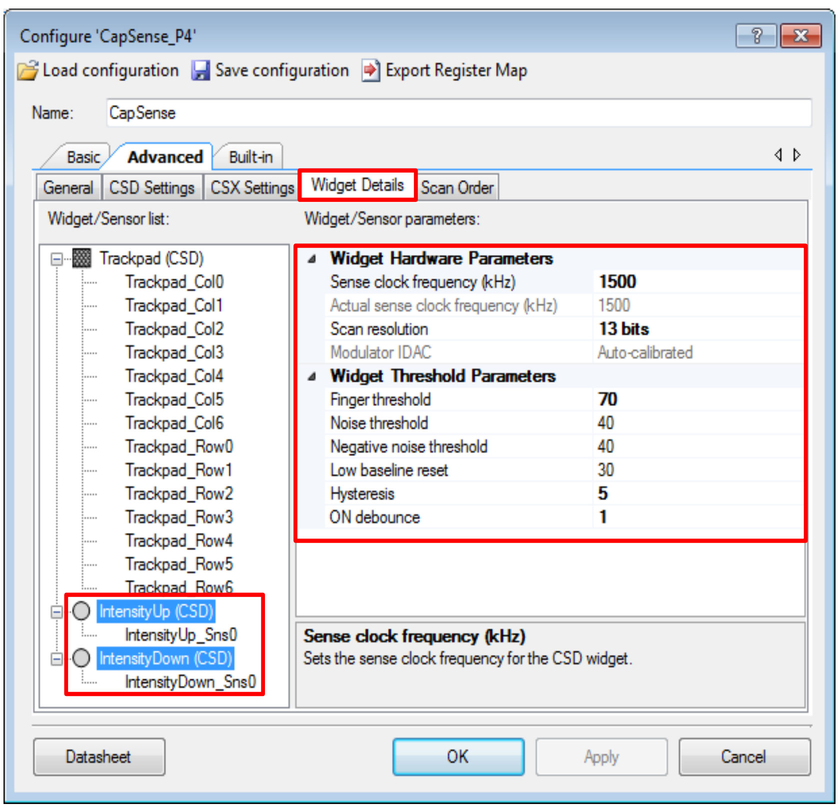


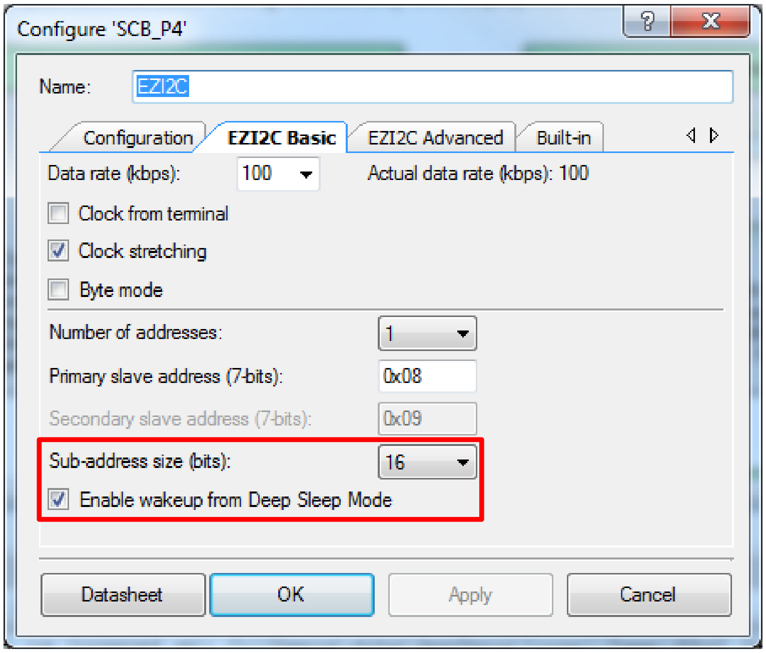
Figure 9 CapSense Component – Advanced Tab Widget Details for Buttons



EZI2C Slave

Figure 10 shows the non-default EZI2C Slave Component settings. See the SCB Component datasheet for additional information.

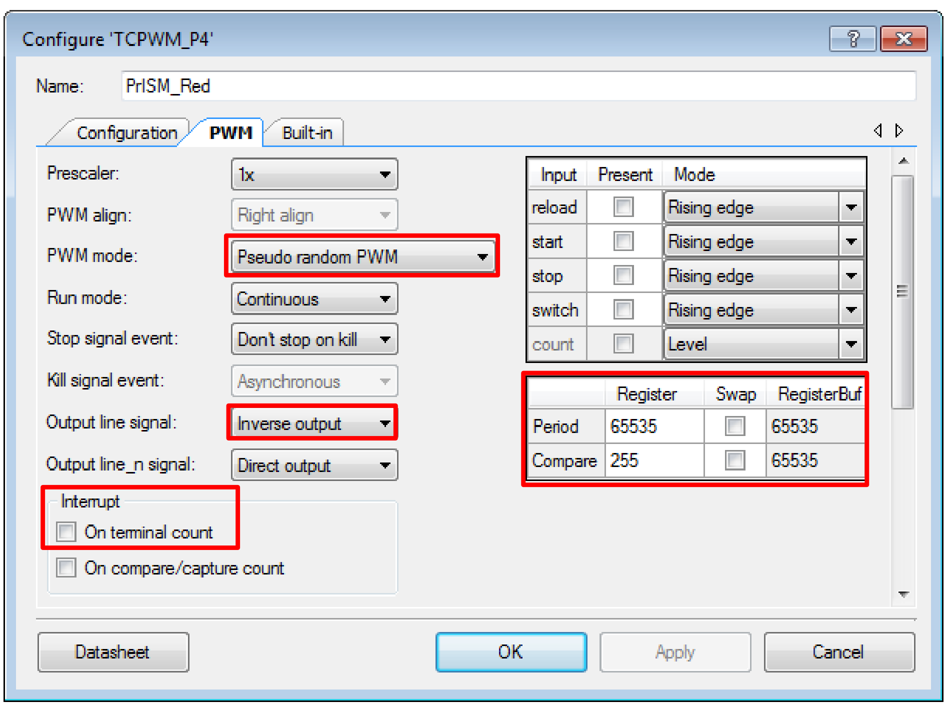
Figure 10. EZI2C Component Basic Tab Configuration



PWM

Figure 11 shows the non-default PWM Component settings. See the TCPWM Component datasheet for additional information.

Figure 11. PWM Component Settings



Design-Wide Resources

Figure 12 and Figure 13 show the non-default .cydwr settings for the project.

Figure 12. .cydwr Pins Tab Settings

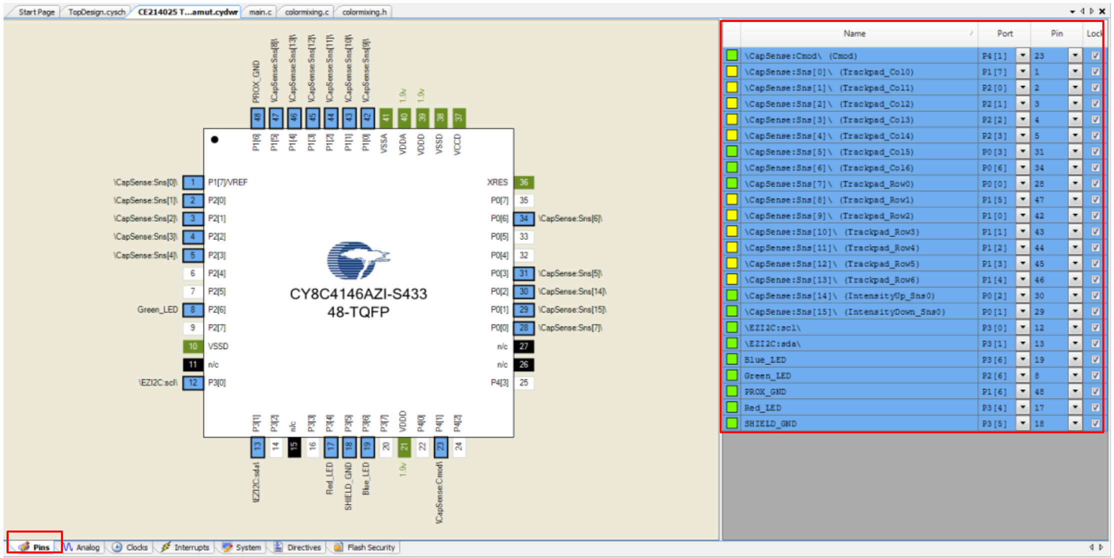
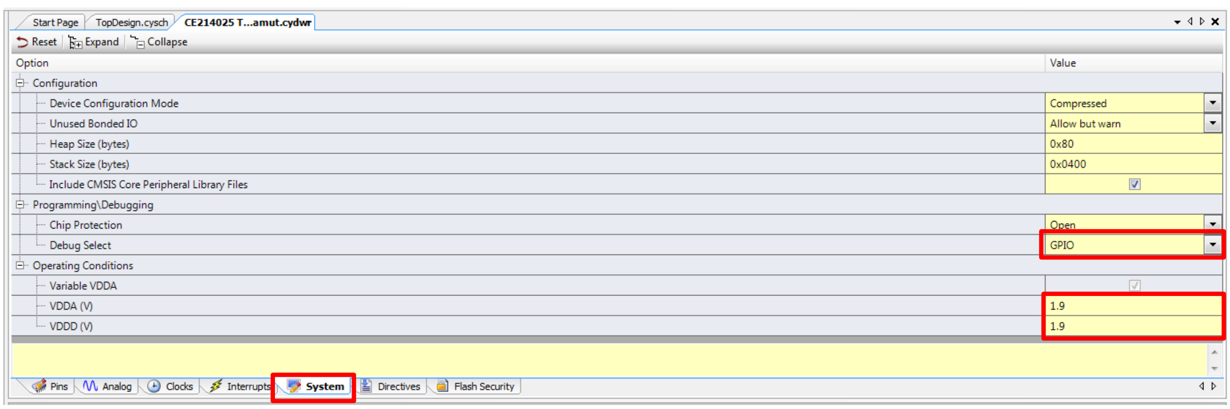


Figure 13. .cydwr System Tab Setting



Note: For PSoC 4100S devices, the CapSense V_{REF} voltage is set based on the VDDA setting in the .cydwr tab per Table 2.

Table 2. CapSense V_{REF} Values Based on VDDA Setting

VDDA (V)	V_{REF} (V)
< 2.7	1.2
2.7 to 4.8	2.1
>= 4.8	4.2

If VDDA is set to 1.9 V in the .cydwr tab, V_{REF} is set to 1.2 V. This V_{REF} voltage ensures that the CapSense tuning parameters do not vary with respect to VDDA, thereby avoiding retuning of the sensors.

Operation

1. Select the *CE214025 Trackpad With Color Gamut.cywrk* file on the PSoC Creator Start Page at **Examples and Kits** > **Kits** > **CY8CKIT-041-41XX**. Select a location to save the code example.
2. Build the project (**Build** > **CE214025 Trackpad With Color Gamut**).
3. Connect the PSoC 4100S Pioneer Kit to your computer using the USB cable provided.
4. Program the PSoC 4100S device (**Debug** > **Program**). See the kit guide for details on programming the kit.
5. Move your finger within the color gamut triangle and observe that the color-mixing algorithm modulates the RGB LED to reproduce the selected color.

   Note: The deviation of the reported touch object position from the expected touch object position is equal to a maximum of 2.5 mm for a finger size of 9 mm. Therefore, when a finger is on the color gamut triangle, the reported touch position might fall outside the triangle.
6. Touch the two button sensors to control the RGB LED brightness.

   Note: After reset, if the button sensors are touched before the trackpad, the Red LED will be activated to demonstrate wake-on-touch for buttons. At lower brightness levels, the color reproduced by the RGB LED might look different from the actual selected color.
7. Remove your finger from the kit and notice that the RGB LED is turned OFF after three seconds of delay.
8. Move your finger from the color gamut triangle to outside of the triangle and observe that the RGB LED retains the previous valid color.

The example project supports viewing CapSense data via the CapSense Tuner. For details on how to launch the tuner and read the CapSense data, refer to the CapSense Component Datasheet.

Upgrade Information

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

Related Documents

Table 3 lists the relevant application notes, PSoC Creator Component datasheets, device documentation, and development kit (DVK) documentation.

Table 3. Related Documents

Application Notes		
AN79953	Getting Started with PSoC 4	Describes PSoC 4 and how to build your first PSoC Creator project
AN85951	PSoC 4 and PSoC Analog Coprocessor CapSense Design Guide	Describes PSoC 4 and PSoC Analog Coprocessor CapSense Component tuning
PSoC Creator Component Datasheets		
CapSense	Supports capacitive touch sensing	
EZI2C Slave	Supports I^2C slave operation	
PWM	Supports 16-bit fixed-function pseudo-random PWM implementation	
Pins	Supports connection of hardware resources to physical pins	
Clock	Supports local clock generation	
Device Documentation		
PSoC 4100S Family Datasheet		
PSoC 4100S Family PSoC 4 Architecture Technical Reference Manual		
Development Kit (DVK) Documentation		
CY8CKIT-041-41XX PSoC 4100S 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:

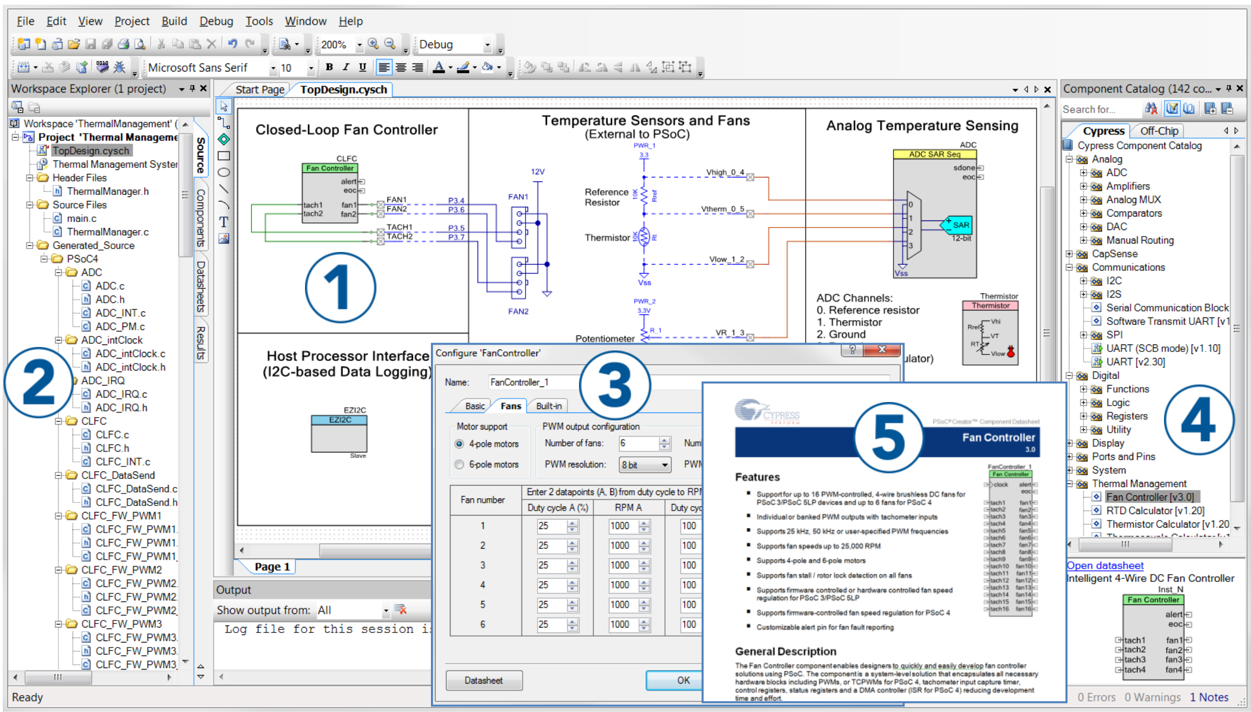
- **Overview:** PSoC Portfolio, PSoC Roadmap
- **Product Selectors:** PSoC 1, PSoC 3, PSoC 4, or PSoC 5LP. In addition, PSoC Creator includes a Device Selector tool.
- **Datasheets** describe and provide electrical specifications for the PSoC 3, PSoC 4, and PSoC 5LP device families.
- **CapSense Design 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 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-041-41XX PSoC 4100S Pioneer Kit is easy-to-use and inexpensive development platform. This kit includes connectors for Arduino™ compatible shields.
  - CY8CKIT-145 is a very low-cost prototyping platform for evaluating PSoC 4 S-Series devices.
- The MiniProg3 device provides an interface for flash programming and debugging.

PSoC Creator

PSoC Creator is a free, Windows-based IDE. It enables concurrent hardware and firmware design of systems based on PSoC 3, PSoC 4, and PSoC 5LP. See Figure 14. With PSoC Creator, you can:

1. Drag and drop Components to build your hardware system design in the main design workspace
2. Co-design 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 14. PSoC Creator Features

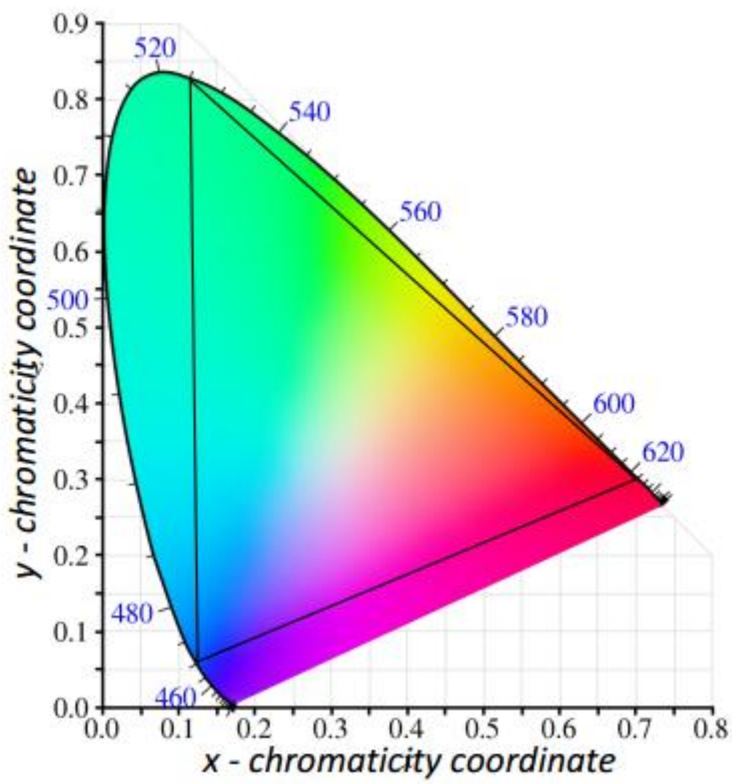


Appendix: CIE 1931 Color Gamut

Color-Mixing Theory

Figure 15 shows the CIE 1931 color chromaticity diagram. The CIE system characterizes colors by luminance parameter "Y" and two color coordinates, "x" and "y," which specify the point on the chromaticity diagram. There are three LEDs: Red, Green, and Blue, plotted in Figure 15. By mixing an appropriate proportion of two colors such as red and blue, all colors along the line that joins red and blue can be generated. Similarly, when blue and green are mixed, all the colors along the blue and green line can be generated. Color mixing these three LEDs can generate any color that lies within this triangle. This area is called the "color gamut."

Figure 15. CIE 1931 Color Space Chromaticity Diagram



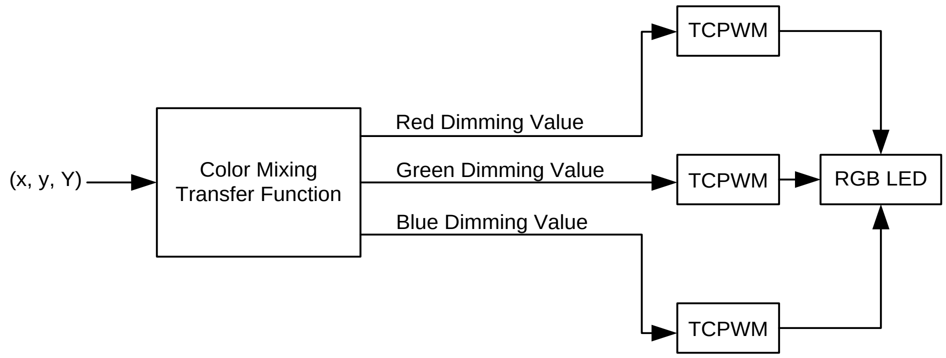
Color Mixing Algorithm

The code example firmware uses the CIE 1931 color space, and any particular color in the CIE 1931 color space is represented with three values, which form a vector (x, y, Y). The x and y values represent the color hue and saturation. Plotting the (x, y) coordinate on the chart in Figure 15 provides a particular shade of color. The colored area represents all visible colors of light, and the white area represents colors that are not visible to the human eye. For example, a (x, y) coordinate of (0.7, 0.7) is not in the colored area and does not represent any visible color.

The third value of the (x, y, Y) vector specifies the luminous flux in lumens. While the (x, y) coordinate is dimensionless, the Y value can have units of lumens or may be expressed as a percentage to signify a relative flux. The Y value cannot be seen in the graph of Figure 15, but it is visualized as a vector orthogonal to the page with a magnitude of Y at some (x, y) coordinate. This (x, y, Y) vector completely describes a light source by denoting its color and its total flux. The firmware must have inputs in (x, y, Y) vector form. The firmware receives color requests in the form of three values. In this particular implementation, the (x, y) coordinate takes the form of two 16-bit unsigned integers, where a value of 10,000 would correspond to an x or y value of 1.0. The Y value is input as an 8-bit unsigned integer that specifies the number of total lumens the mixed color must have. The color-mixing algorithm can then use the values to determine the correct dimming values for the three LEDs that create the required (x, y, Y) color.

Figure 16 shows the inputs of the firmware and the translated outputs. The mathematical functions in this section describe how the three dimming values are obtained from one (x, y, Y) coordinate.

Figure 16. Color-Mixing Process



The first step is the creation of a matrix, as shown in Equation 1. The color subscript (for example, red) denotes the x or y value of the respective Red, Green, or Blue LEDs in the system. The "mix" subscript denotes the x or y value of the input color coordinate request. The lumen output for each LED is obtained from Equation 2.

Equation 1. Color-Mixing Matrix

$$A = \begin{bmatrix} \frac{x_{red} - x_{mix}}{y_{red}} & \frac{x_{green} - x_{mix}}{y_{green}} & \frac{x_{blue} - x_{mix}}{y_{blue}} \\ \frac{y_{red} - y_{mix}}{y_{red}} & \frac{y_{green} - y_{mix}}{y_{green}} & \frac{y_{blue} - y_{mix}}{y_{blue}} \\ 1 & 1 & 1 \end{bmatrix}$$

Equation 2. Computing Lumen Output for Each LED

$$\begin{bmatrix} Y_{red} \\ Y_{green} \\ Y_{blue} \end{bmatrix} = A^{-1} * \begin{bmatrix} 0 \\ 0 \\ Y_{mix} \end{bmatrix}$$

The first mathematical operation takes an inverse of the matrix A, as shown in Equation 3.

Equation 3. Inverse of a Matrix

$$A^{-1} = \frac{1}{\det A}(adj\ A)$$

Finding an inverse of a matrix involves two steps:

1. Finding the determinant of the matrix (det A)
2. Finding the adjoint of the matrix (adj A)

For a 3x3 matrix A, the inverse is given by Equation 5.

Equation 4. 3x3 Matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Equation 5. Inverse of a 3x3 Matrix

$$A^{-1} = \frac{1}{\det A}\begin{bmatrix} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{12} \\ a_{33} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ \begin{vmatrix} a_{23} & a_{21} \\ a_{33} & a_{31} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{11} \\ a_{23} & a_{21} \end{vmatrix} \\ \begin{vmatrix} a_{21} & a_{21} \\ a_{31} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{11} \\ a_{32} & a_{31} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}$$

…where a determinant is given by Equation 6.

Equation 6. Determinant of a 3x3 Matrix

$$\det A = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - (a_{31}a_{22}a_{13} + a_{32}a_{23}a_{11} + a_{33}a_{21}a_{12})$$

Note: The inverse of matrix A is multiplied by a 3x1 matrix (Equation 2), and the first two elements of the 3x1 matrix are zero. Therefore, only the third-column elements are computed for the matrix inverse, A^{-1}.

After the matrix inversion, the next step is to factor in the total flux information of that color. This is done by a matrix multiplication, as shown in Equation 2. The value of Y_{mix} is the number of lumens that the total mixed light output must produce. The resultant Y values of the product are the lumen output of each respective LED that is necessary to create the requested color and flux.

At this point, the math operations give rise to two benefits. If any of the final product's Y values in Equation 2 are negative, it signifies that the requested color coordinate is invalid, and the LEDs in the system cannot create that color. In other words, the requested color is outside the gamut of the LEDs. The second item to check is if any of the product's Y values are larger than the maximum lumen output of any of the three LEDs. If this is the case, then it means that the Y_{mix} input is too large, and the LEDs in the system cannot create that much total flux at the given (x, y) coordinate. The firmware checks to see if either of these conditions occurs. If the requested flux is too large, the firmware scales back the values so that they produce the maximum possible flux at the requested (x, y) coordinate. If the (x, y) coordinate is invalid, the firmware retains the previous correct LED state.

Equation 7 expresses how a dimming value is produced from the Y_{red} value (the same equation would also apply to the other colors). $Y_{max,red}$ is the lumens that the Red LED has if it is not dimmed at all, which is its maximum flux. N is the number of bits of resolution that the hardware dimmers (TCPWM resolution) have. In this system, N is equal to 16. After applying this equation to each color channel, each channel has a unique dimming (compare value of TCPWM) value that is applied to the TCPWM Component for LED dimming.

Equation 7. Computing Dimming Value from LED Lumen Output

$$DimValue_{red} = \frac{Y_{red}}{Y_{\max,red}} * (2^N - 1)$$

Document History

Document Title: CE214025 - Trackpad with Color Gamut

Document Number: 002-14025

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5444073	SRDS / SLAN	11/18/2016	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
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