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.

CE220331 – PSoC 6 MCU with BLE Connectivity: BLE with User Interface (RTOS)

Objective

This code example demonstrates interfacing PSoC® 6 MCU with Bluetooth Low Energy (BLE) Connectivity (PSoC 6 MCU) with user interface functions such as an RGB LED and touch sensors based on self and mutual capacitance (CapSense® CSD and CSX). These functions provide bi-directional BLE connectivity between the PSoC 6 MCU and a PC running the CySmart™ BLE Host Emulation tool or a mobile device running the CySmart mobile application.

Overview

This code example demonstrates interfacing PSoC 6 MCU with an RGB LED with color and intensity control and touch buttons based on mutual capacitance (CSX), and touch-slider-based on self-capacitance (CSD). This code example also shows connectivity between the PSoC 6 BLE (acting as a Peripheral and GATT Server) and a PC running the CySmart BLE Host Emulation tool or a mobile device running the CySmart mobile application (acting as a Central and GATT Client). Custom BLE services are used for CapSense touch sensing and LED control.

In more detail:

- RGB LED color and intensity control using configurable digital blocks of PSoC 6 MCU
- CapSense slider and buttons
- PSoC 6 MCU's ability to simultaneously scan touch sensors based on self-capacitance and mutual-capacitance
- BLE connectivity
  - Advertisement and connection with a Central device
  - Three custom services (CapSense Slider, CapSense Button, and RGB LED)
  - Data transfer over BLE using notifications, read, and write

This code example assumes that you are familiar with the PSoC 6 MCU and the PSoC Creator™ Integrated Design Environment (IDE). If you are new to PSoC 6 MCU, see the application note AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity.

This code example uses FreeRTOS. See PSoC 6 101: Lesson 1-4 FreeRTOS training video to learn how to create a PSoC 6 FreeRTOS project with PSoC Creator. Visit the FreeRTOS website for documentation and API references of FreeRTOS.

Requirements

Tool: PSoC Creator 4.2; Peripheral Driver Library (PDL) 3.0.1

Programming Language: C (Arm® GCC 5.4.1)

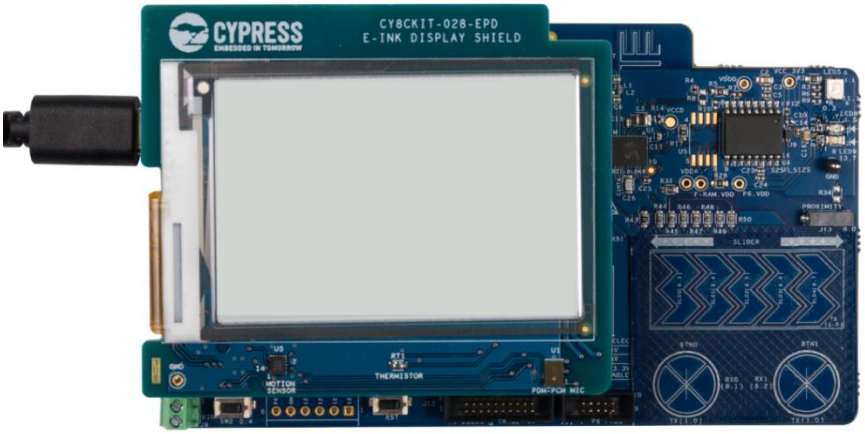
Associated Parts: All PSoC 6 MCUs with BLE Connectivity

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Hardware Setup

Plug in the E-INK display shield on to the Pioneer Board as Figure 1 shows.

Figure 1. Hardware Setup



Set the switches and jumpers on the Pioneer Board as shown in Table 1.

Table 1. Switch and Jumper Selection

Switch/Jumper	Position	Location
SW5	3.3 V	Front
SW6	PSoC 6 BLE	Back
SW7	V_{DDD} / KitProg2	Back
J8	Installed	Back

Software Setup

Install the CY8CKIT-62-BLE PSoC 6 BLE Pioneer Kit software, which contains all the required software to evaluate this code example. No additional software setup is required.

Operation

You can verify the code example using either of these methods: the CySmart BLE Host Emulation Tool or BLE Dongle on a PC or the CySmart mobile application.

Note: For this code example, the CapSense Button service is not available in the CySmart iOS app. Use the host emulation tool or the Android app to evaluate this service.

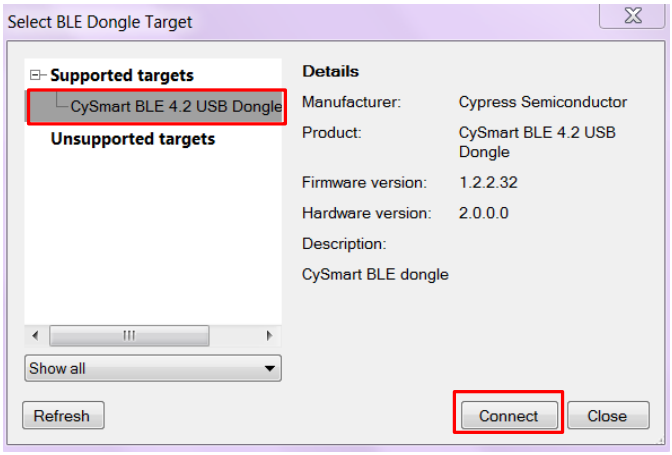
CySmart BLE Host Emulation Tool

To verify the *CE220331_BLE_UI_RTOS* code example using the CySmart BLE host emulation tool, follow these steps:

Note: See the CySmart BLE host emulation tool documentation to learn how to use the tool.

1. Connect the BLE Dongle to one of the USB ports on the computer.
2. Start the CySmart BLE host emulation tool on the computer by going to **Start** > **All Programs** > **Cypress** > **CySmart <version>** > **CySmart <version>**. You will see a list of BLE Dongles connected to it. If no dongle is found, click **Refresh**. Select the BLE Dongle and click **Connect**.

Figure 2. Connect to BLE Dongle



3. Power the Pioneer Board through the USB connector **J10**.
4. Program the Pioneer Board with the *CE220331_BLE_UI_RTOS* project. See the Pioneer Kit guide for details on how to program firmware into the device.

   After programming, the E-INK display will refresh and show the instructions to use this project and BLE will start advertising. The advertising timeout is configured to be 20 seconds. The orange LED (**LED8**) remains ON during this period to indicate the BLE advertising state as Figure 3 shows. Table 2 lists all the LED indications corresponding to BLE states.

Figure 3. BLE Advertising

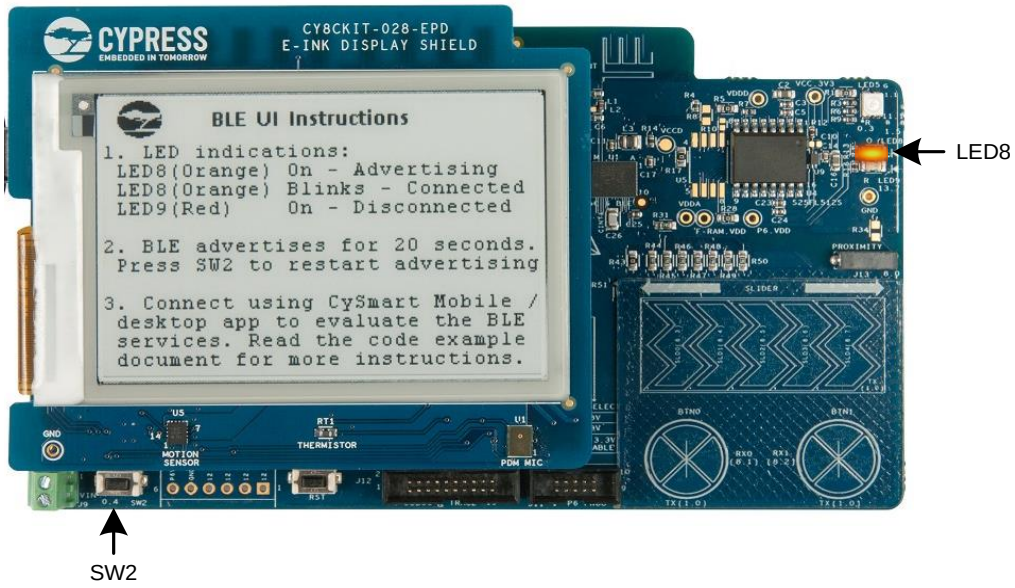
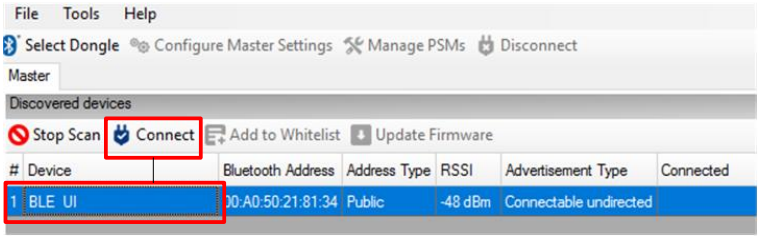


Table 2. LED Indications

BLE State	LED Indication	
Advertising	LED8 (Orange)	ON
	LED9 (Red)	OFF
Connected	LED8 (Orange)	Toggling continuously
	LED9 (Red)	OFF
Disconnected	LED8 (Orange)	OFF
	LED9 (Red)	Blinks once
Idle (following advertisement timeout or disconnection)	LED8 (Orange)	OFF
	LED9 (Red)	OFF

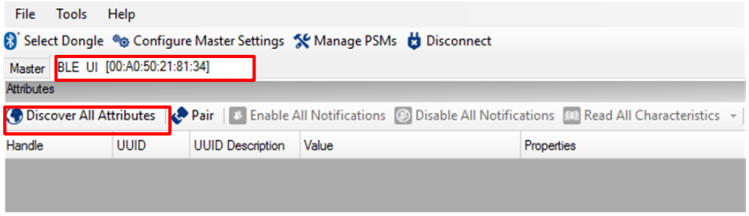
5. If the BLE advertisement has timed out (**LED8** is OFF), press **SW2** to restart advertisement.
6. On the CySmart host emulation tool, click **Start Scan** to see the list of available BLE Peripheral devices. Double-click the **BLE UI** device to connect, or click **BLE UI** and then click **Connect**. A successful connection is indicated by **LED8** continuously blinking at half second intervals.

Figure 4. Connect to BLE Slider and LED Peripheral



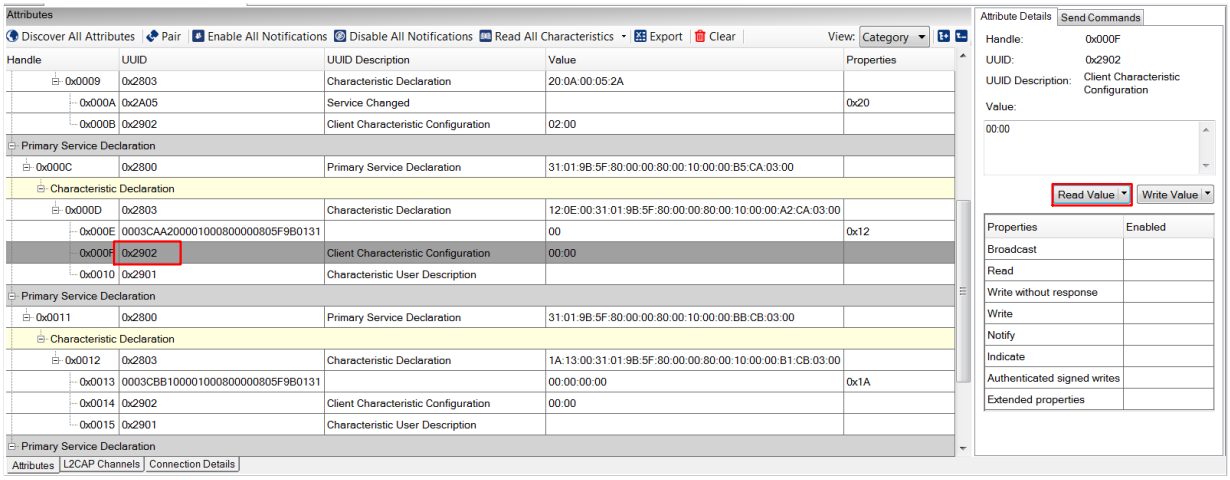
7. Click **Discover All Attributes** to find all attributes supported.

Figure 5. Discover All Attributes



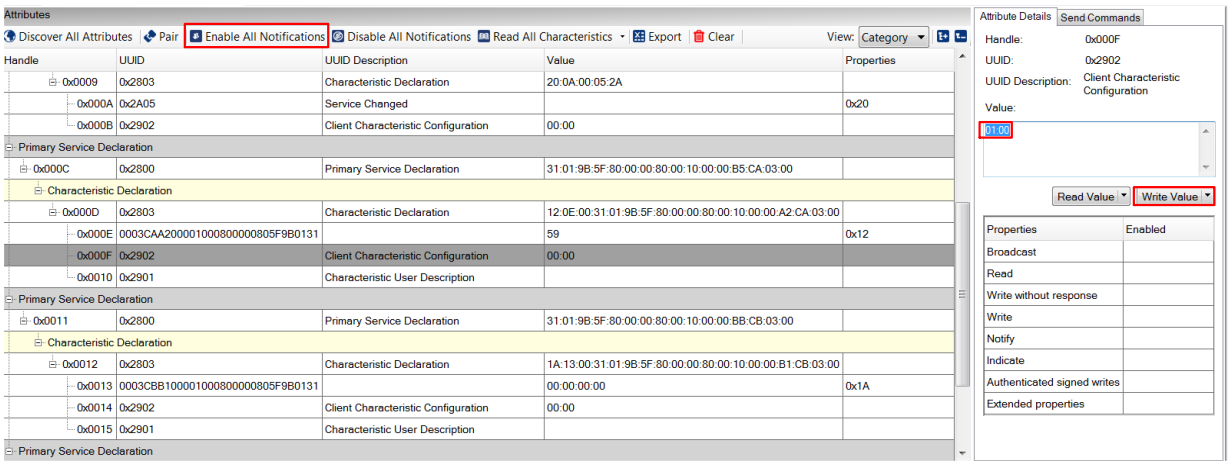
8. Locate the attribute **Client Characteristic Configuration** descriptor (UUID 0x2902) under the CapSense Slider characteristic (**UUID 0x0003CAA2000010008000000805F9B0131**). Click **Read Value** to read the existing Client Characteristic Configuration Descriptor (CCCD) value as shown in Figure 6.

Figure 6. Read CCCD for CapSense Slider Characteristic



9. Modify the **Value** field of the CCCD to '01:00' and click **Write Value**. This enables the notifications on the CapSense slider characteristic. Alternatively, you can press **Enable All Notifications** to enable the notifications for all services.

Figure 7. Write CCCD to Enable Notifications



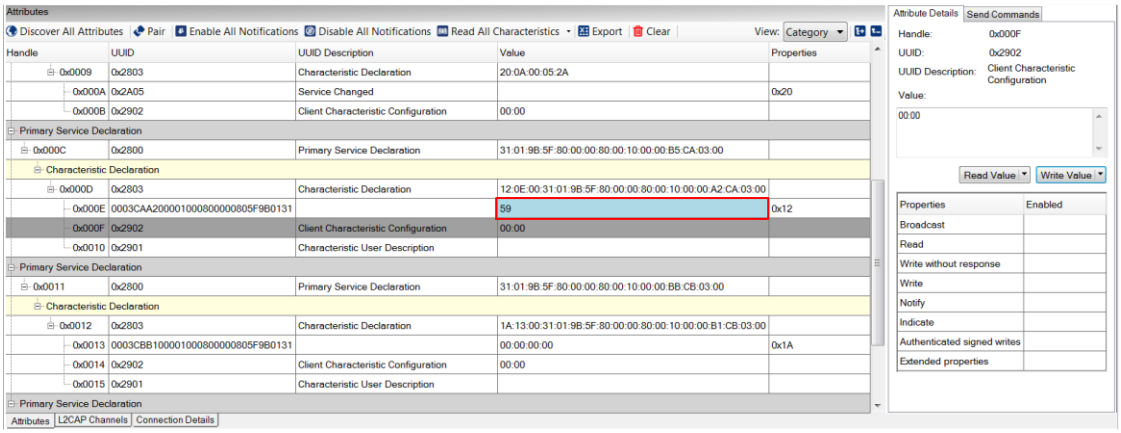
10. Swipe your finger on the CapSense slider on the Pioneer Board as shown in Figure 8, and see the notification values in the CapSense Slider value field as shown in Figure 9.

Note: The sensor auto-reset feature is enabled for CapSense sensors. Pressing and holding a CapSense sensor for more than three seconds will reset the sensor to the inactive state. This feature will prevent stuck-on sensors under rapidly changing environments – see the CapSense Component datasheet for additional information.

Figure 8. CapSense Slider

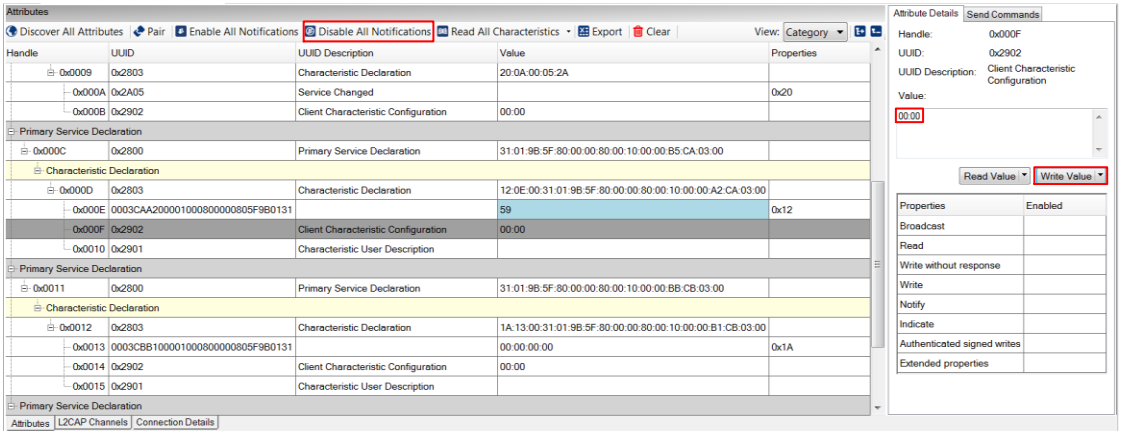


Figure 9. CapSense Slider Notification Received



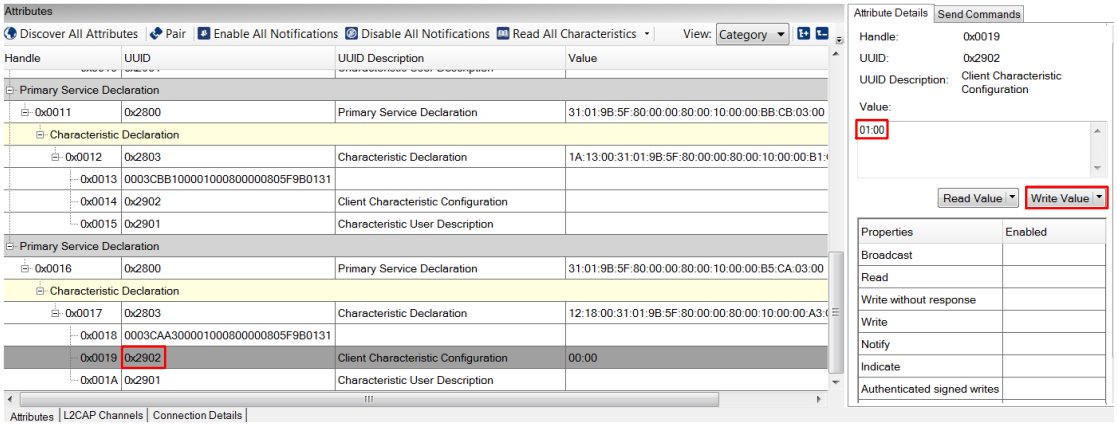
11. To disable notifications, modify the **Value** field of the **Client Characteristic Configuration** descriptor to '00:00' and click **Write Value**. Alternatively, you can press **Disable All Notifications** to disable the notifications of all services.

Figure 10. Disable Notifications



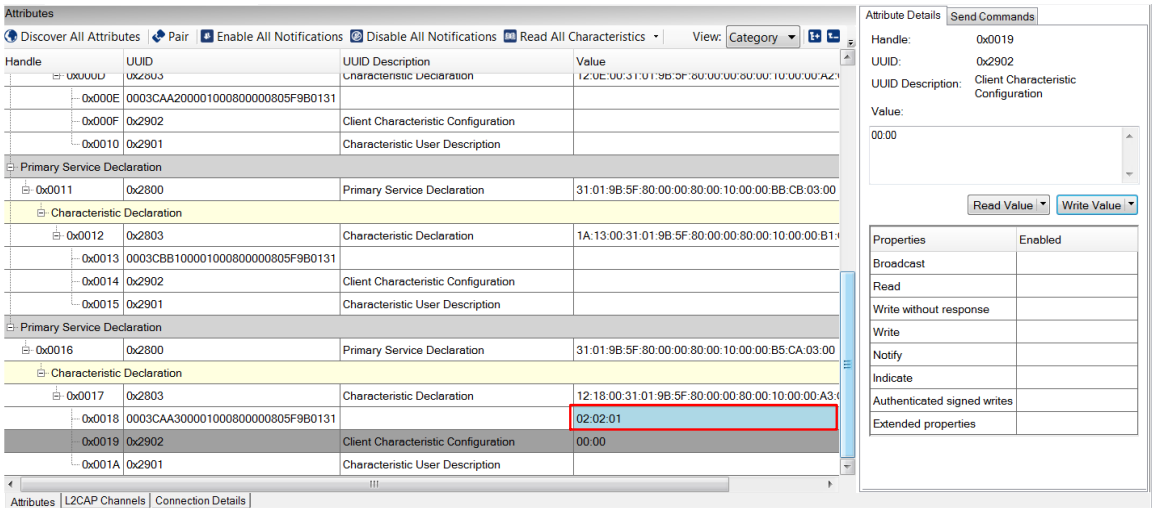
12. Locate the attribute **Client Characteristic Configuration** descriptor (UUID 0x2902) under CapSense Button characteristic (**UUID 0x0003CAA3000010008000000805F9B0131**), read the value and enable the notification as described in steps 8 and 9.

Figure 11. Enable CapSense Button Notification



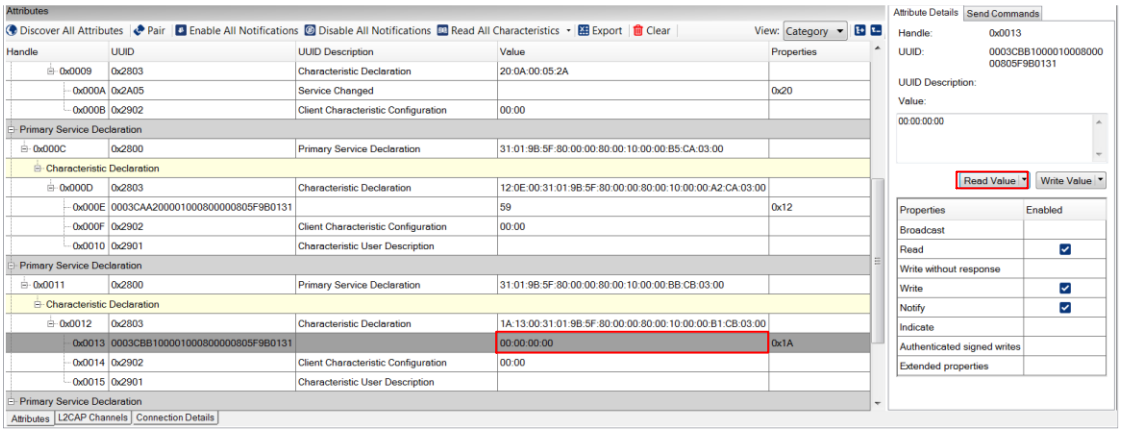
13. Touch the CapSense buttons on the Pioneer Board, and see the notification values in the CapSense Button **Value** field, as shown in Figure 12. The LSB (byte 0) indicates the button mask – 0 when no buttons are active, 1 when BTN0 is active, 2 when BTN1 is active, and 3 when both buttons are active. See step 11 for instructions to disable notifications.

Figure 12. CapSense Button Notification Received



14. Locate the **RGB LED Control** characteristic (**UUID 0x0003CBB1-0000-1000-8000-00805F9B0131**). Click **Read Value** to read the existing 4-byte onboard RGB LED color information, as shown in Figure 13. The four bytes indicate red, green, blue, and the overall intensity, respectively.

Figure 13. Read RGB LED Control Characteristic Value



15. Modify the four bytes of data in the **Value** field to **FF:00:00:FF** and click **Write Value**, as shown in Figure 14. You will see the corresponding change in the color (Red) and intensity (full intensity) of the RGB LED on the Pioneer Board as shown in Figure 15.

Figure 14. Write RGB LED Control Characteristic Value

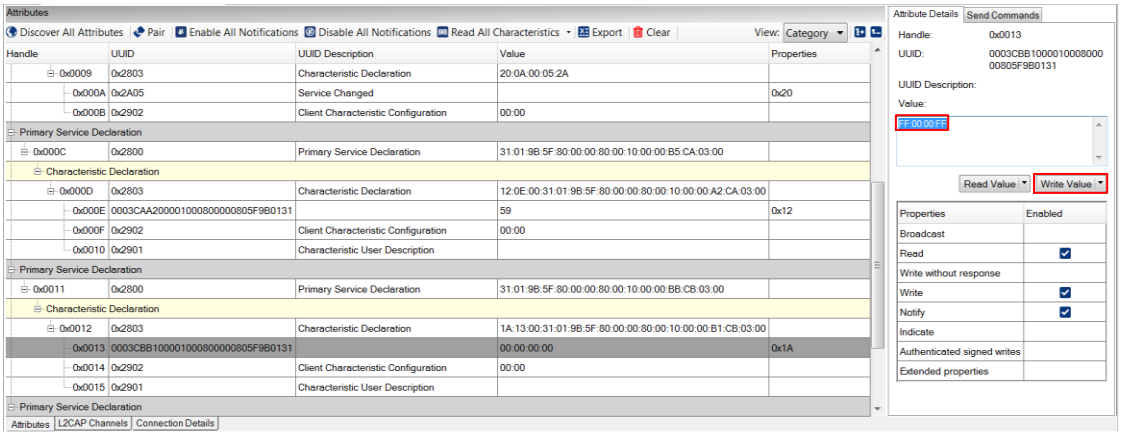
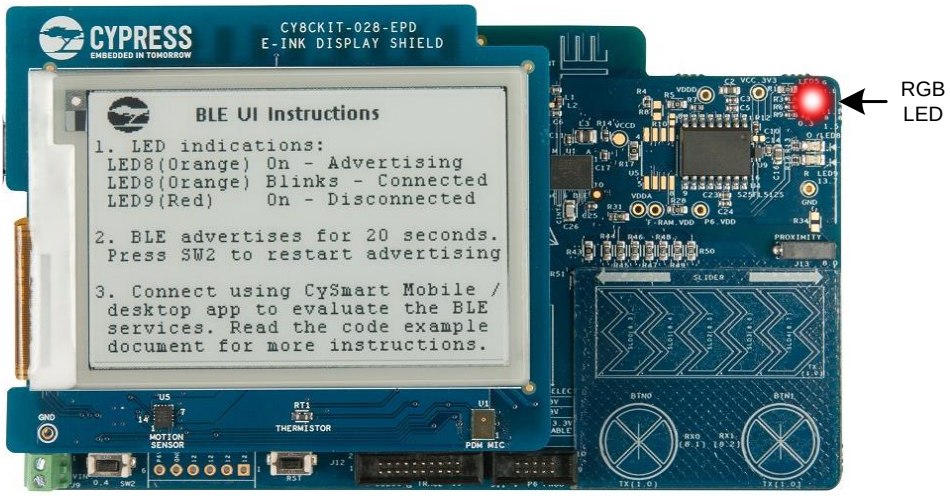


Figure 15. RGB LED Control with BLE



16. To disconnect from the device, click **Disconnect**, as shown in Figure 16. The red LED (**LED9**) will turn ON for three seconds to indicate a disconnect event. Press **SW2** to restart advertisement, if required.

Figure 16. Disconnect from the Device

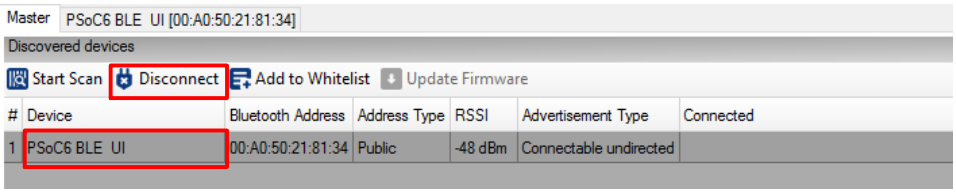
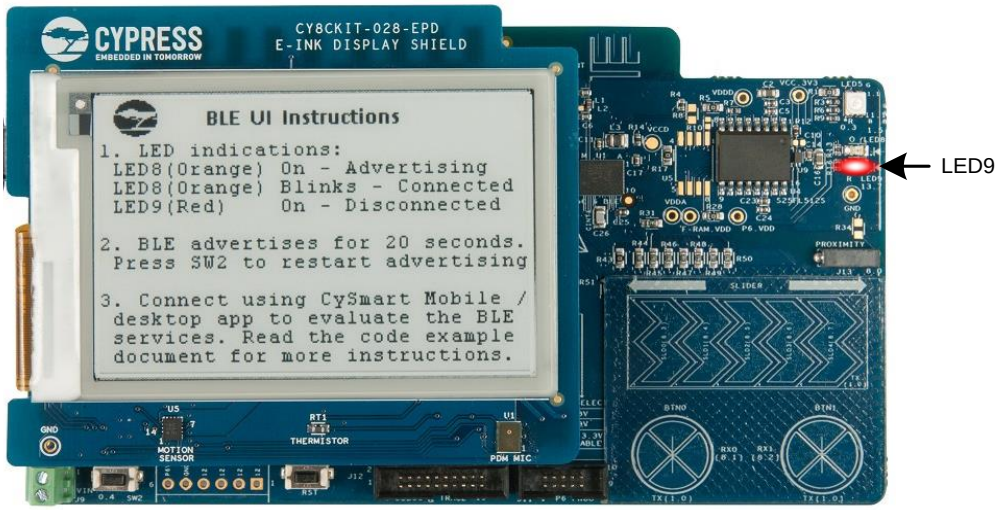


Figure 17. Disconnect Indication



CySmart Mobile Application

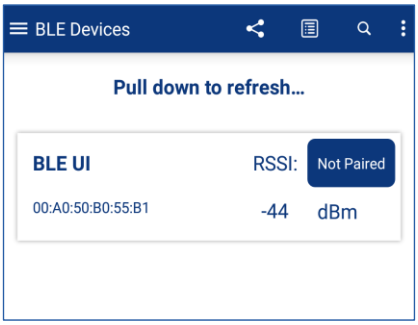
To verify this code example using the CySmart mobile application (see the CySmart Mobile App webpage), follow these steps:

Note: For this project, the CapSense Button service is not available in the CySmart iOS app. Use the Android app to evaluate this service.

1. Install the CySmart app.
2. Power the Pioneer Board through the USB connector **J10**.
3. Program the Pioneer Board with the *CE220331_BLE_UI_RTOS* project. See the Pioneer Kit guide for details on how to program firmware into the device.

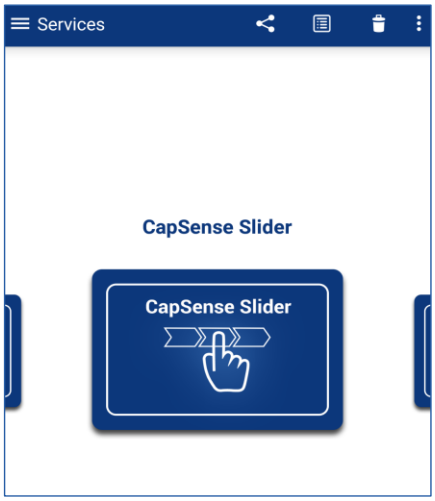
   After programming, the E-INK display refreshes and shows the instructions to use this project; BLE starts advertising. The advertising timeout is configured to be 20 seconds. The orange LED (**LED8**) remains ON during this period to indicate the BLE advertising state.
4. If the BLE advertisement has timed out (**LED8** is OFF), press **SW2** to restart advertisement. See the figures in the earlier section for LED and switch locations.
5. Open the CySmart app on the mobile device. If Bluetooth is not enabled on the device, the application will prompt you to enable it.
6. After Bluetooth is enabled, the CySmart mobile application automatically searches for available devices and lists them. Select the **BLE UI** peripheral as shown in Figure 18. A successful connection is indicated by **LED8** continuously blinking at half-second intervals.

Figure 18. BLE UI Peripheral



7. When connected, the CySmart mobile application lists the services supported by the device. Scroll and select the CapSense Slider icon, as shown in Figure 19.

Figure 19. CapSense Slider Service Page



8. Swipe your finger on the CapSense slider on the Pioneer Board and see a similar response on the CapSense Slider page in the CySmart application (see Figure 20).

Figure 20. CapSense Slider



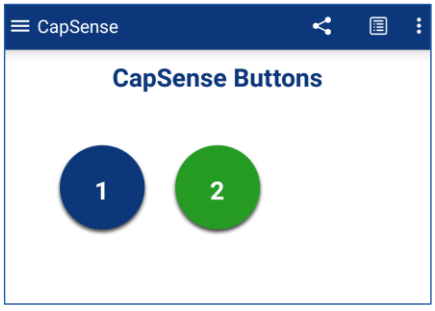
9. Press the back button to return to the service selection page. Scroll and tap on the CapSense Button service.

Figure 21. CapSense Button Service



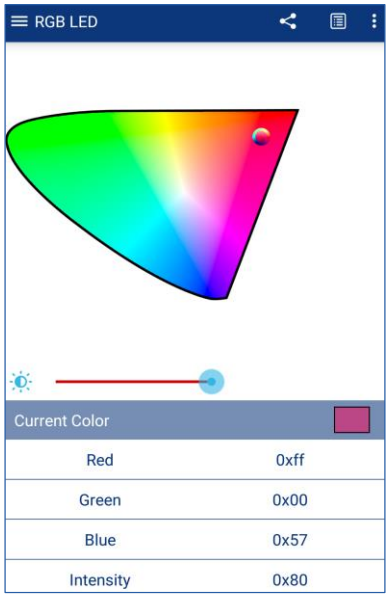
10. Touch CapSense buttons on the Pioneer Board and see a similar response on the CapSense Button page in the CySmart application.

Figure 22. CapSense Buttons



11. Press the back button to return to the service selection page. Scroll and tap on the RGB LED service.
12. On the RGB LED service page, select a color on the color gamut to see a similar color response on the Pioneer Board RGB LED. The slider below the color gamut controls the intensity of the RGB LED color.

Figure 23. RGB LED Control with CySmart Mobile Application



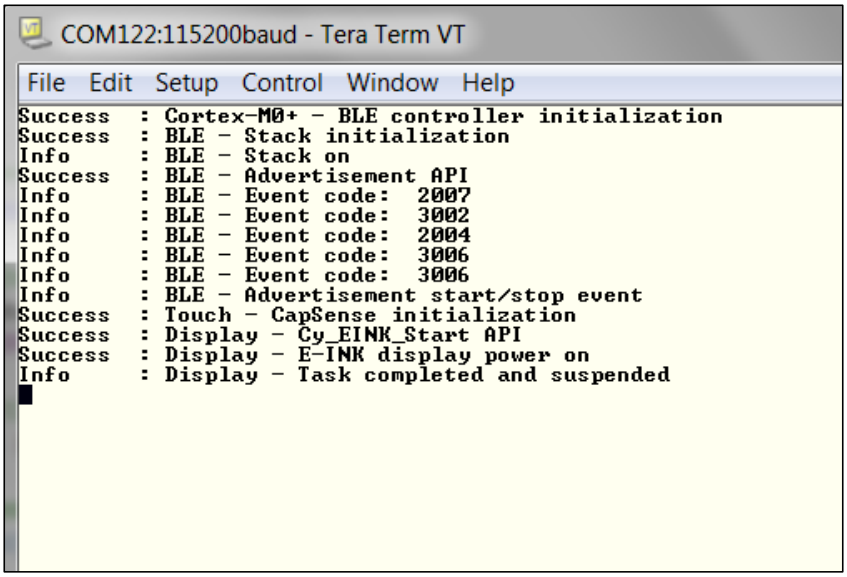
On the service selection page, there is also a “GATT DB” selection, which allows you to examine the GATT database directly. From this page, you can read and write characteristics and enable and disable notifications.

13. If the CySmart app is closed, or Bluetooth is turned OFF, the red LED (**LED9**) will turn ON for three seconds to indicate a disconnect event. Press **SW2** to restart the advertisement, if required.

Viewing Debug Messages

This code example allows you to view debug messages from various tasks and functions using a serial port terminal emulator such as Tera Term or HyperTerminal as Figure 24 shows.

Figure 24. Viewing Debug Messages using UART



This feature is disabled by default for higher performance and power efficiency. To enable the UART debug feature, right-click the UART_DEBUG Component in the TopDesign schematic and select “Enable”. In addition, you should set `UART_DEBUG_ENABLE` macro in the *uart_debug.h* to “true”.

After re-building the projects and re-programming PSoC 6 MCU with the updated project, set up a serial port terminal emulator with these settings to view the debug information:

- Baud rate : 115200
- Data size : 8-bit
- Parity : None
- Stop : 1-bit
- Flow Control : None

Design and Implementation

The BLE profile in this code example consists of three BLE custom services: CapSense Slider, CapSense Button, and RGB LED. The two CapSense services consist of custom characteristics that are used to send data as notifications to the GATT client device. The notification data consists of the finger location read by the CapSense Component on the slider and the ON/OFF status of the two CapSense buttons. These characteristics support notification, which allows the GATT server to send data to the connected client device whenever new data is available. The RGB LED service consists of one custom characteristic called RGB LED Control. This characteristic supports three operations (read, write, and notify) through which the connected GATT client device can read data as well as write a new value to the characteristic. This data has four single-byte values indicating red, green, blue, and the intensity to control the onboard RGB LED. The properties for the custom service/characteristics are configured in the BLE Component under the **GATT Settings** tab. As an example, Figure 25 shows the configuration of the CapSense Slider Service.

Figure 25. BLE CapSense Slider Service Configuration

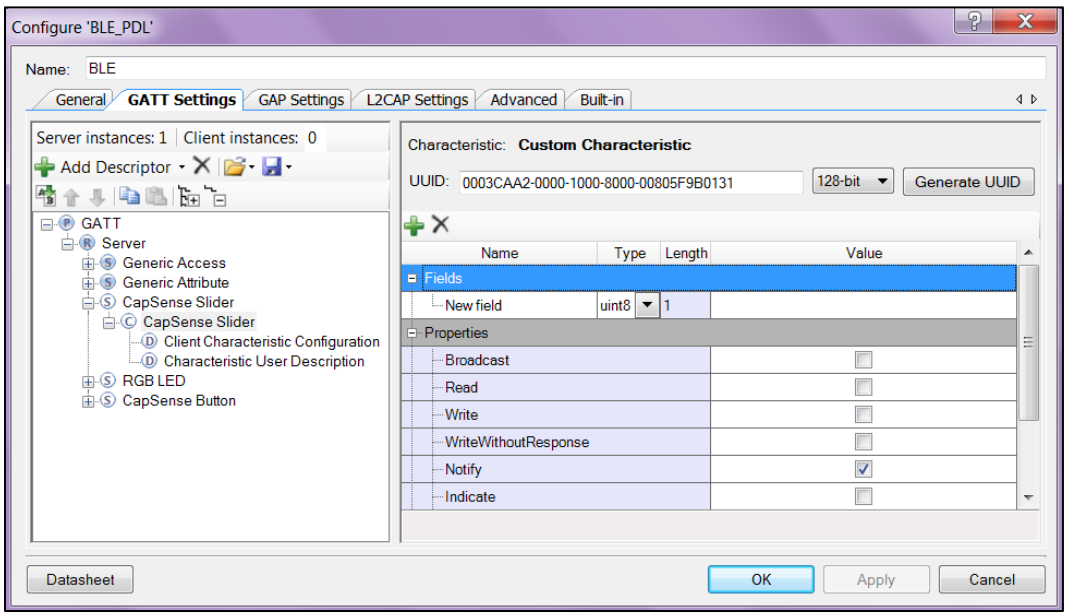


Figure 26, Figure 26, Figure 28 and Figure 29 show the TopDesign schematic of this code example.

The 5-element CapSense CSD slider (self-capacitance), and two CapSense CSX buttons (mutual-capacitance) are scanned with SmartSense™ auto-tuning. See AN85951 - PSoC 4 and PSoC 6 MCU CapSense Design Guide for details of CapSense touch sensing technology and to design capacitive touch sensing applications with PSoC 6 MCU.

Three TCPWM Components operating in Pseudo Random PWM mode are used to drive the RGB LED. The Pseudo Random PWM signal density is modified to display the required color and intensity per the data received over BLE.

The E-INK display shows the instructions to use this code example at startup and is then turned OFF to save power. E-INK displays consume no power to retain the display. For more details on E-INK display, see the code example CE218136 – PSoC 6 MCU E-INK Display with CapSense (RTOS).

Figure 26. TopDesign Schematic: User Interface

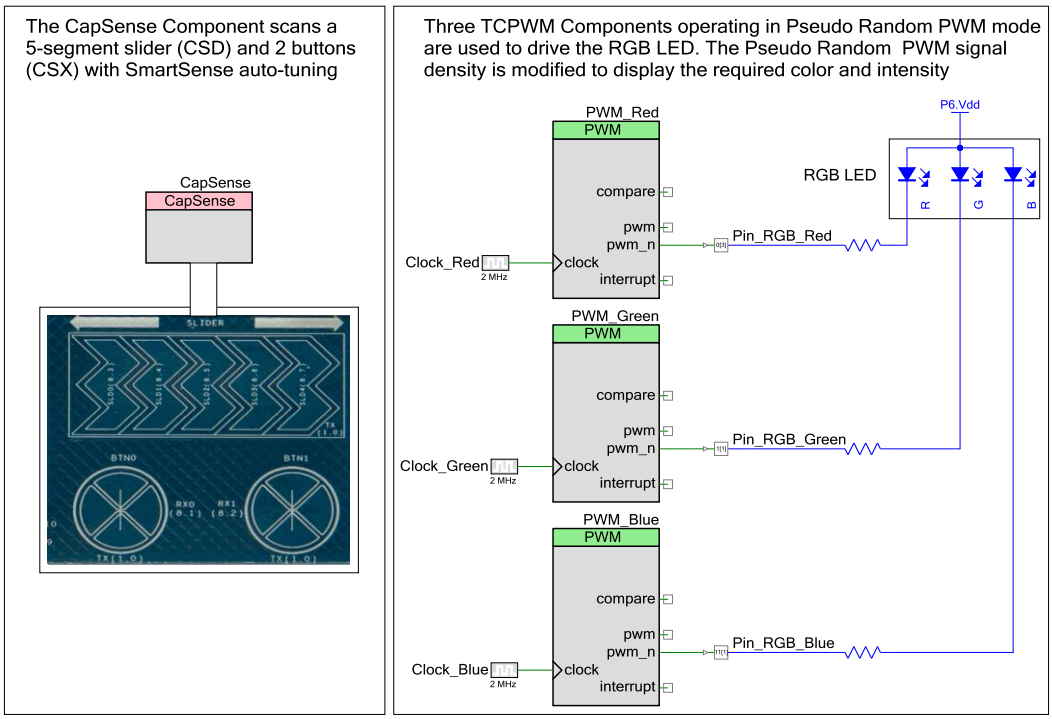


Figure 27. TopDesign Schematic: BLE

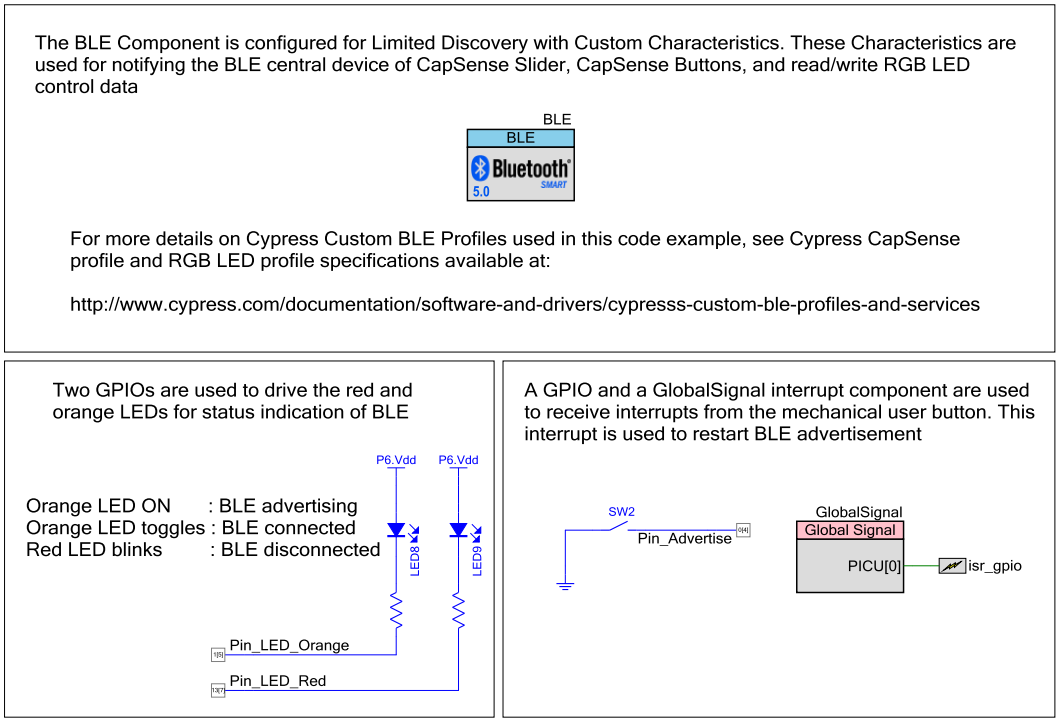


Figure 28. TopDesign Schematic: E-INK Display

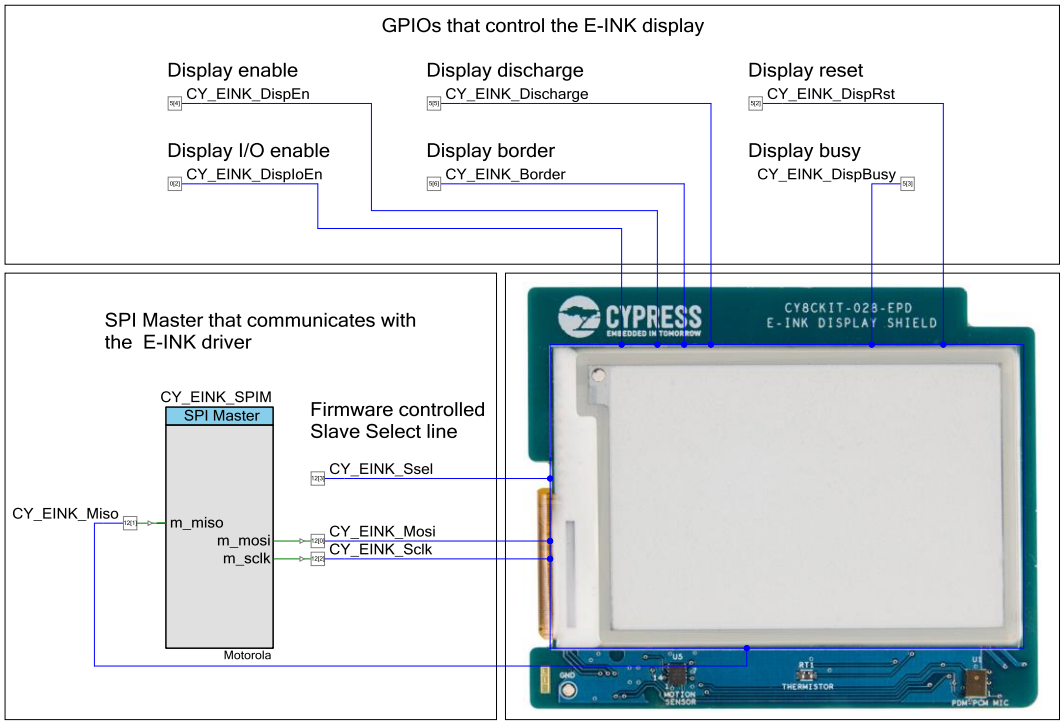
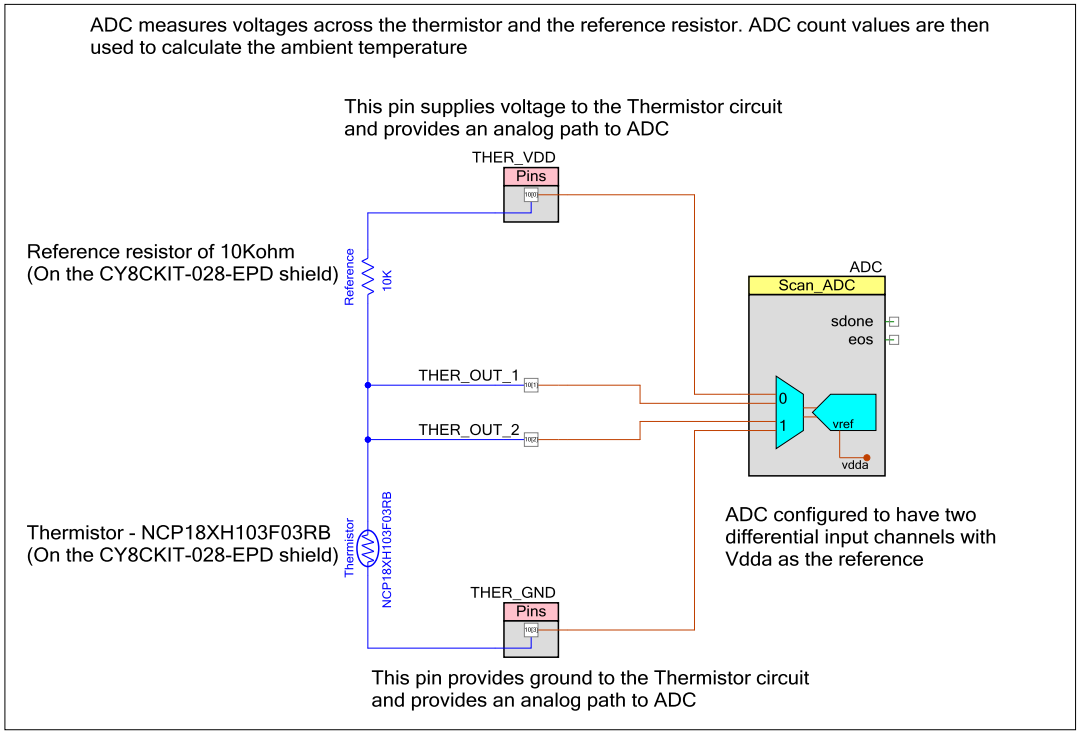


Figure 29. TopDesign Schematic: Temperature Compensation for E-INK Display



The code example consists of the following files:

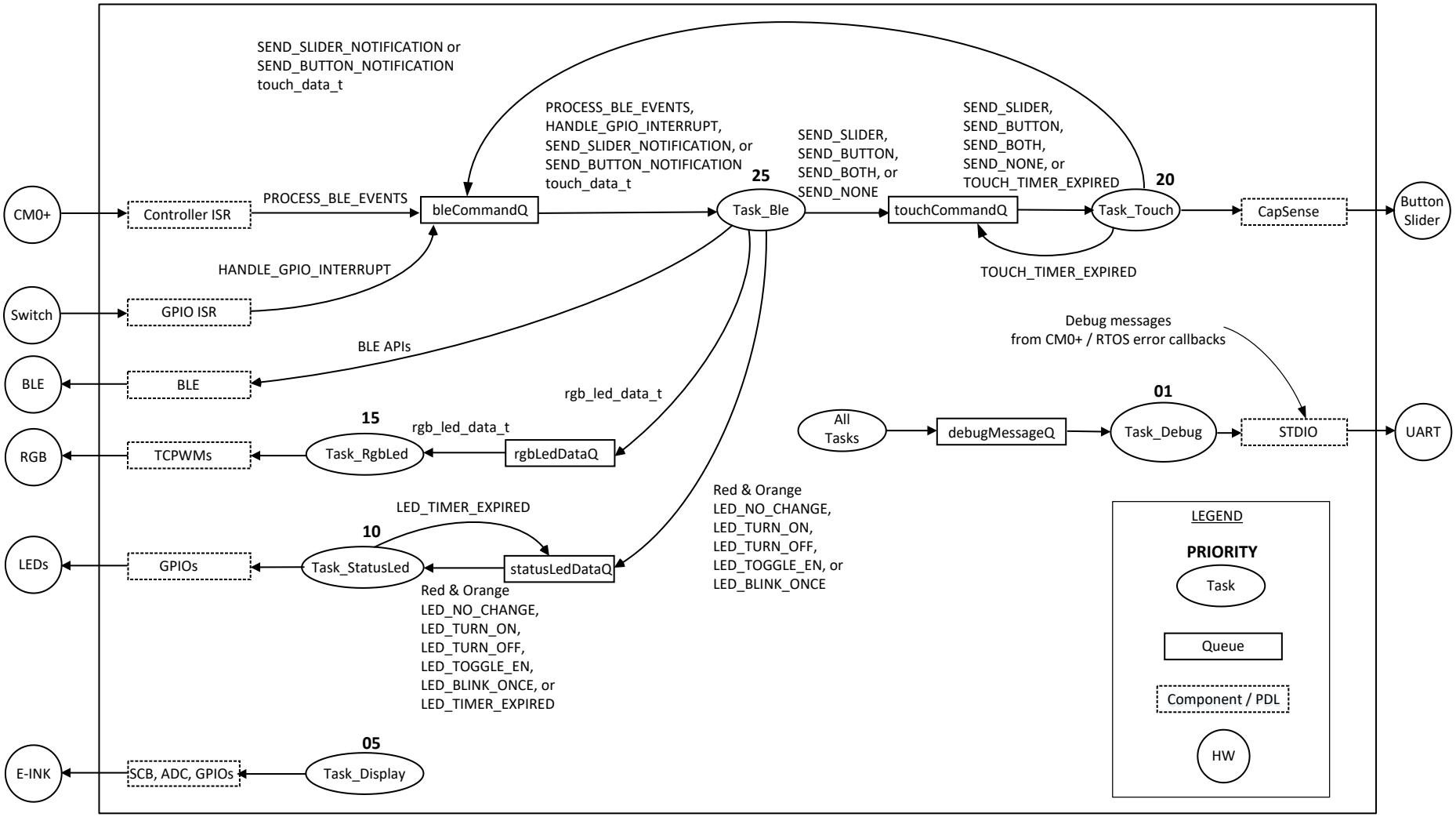
- *FreeRTOSConfig.h* contains the FreeRTOS settings and configuration. Non-default settings are explained with in-line comments.
- *main_cm4.c* contains the main function, which is the entry point and execution of the firmware application. The main function sets up user tasks and then starts the RTOS scheduler.
- *main_cm0p.c* contains functions that start up the BLE controller, start up the CM4, and continuously service BLE stack events.
- *ble_task.c/.h* contain the task and associated functions that handle BLE communication and operation.
- *ble_custom_service_config.h* contains the macros and datatypes used for the three custom BLE services
- *touch_task.c/h* contain the task that scan CapSense sensors and process the data.
- *rgb_led_task.c/.h* contain the task that initialize and control the RGB LED and intensity.
- *status_led_task.c/h* – contain the task that controls status LED indications.
- *display_task.c/.h* contain the task that initialize the E-INK display and show the instructions to use code example at startup[1].
- *uart_debug.c/h* contain the task and functions that enable UART based debug message printing.
- *screen_contents.c/h* contain the text and background images used by the display module.
- *temperature_eink.c/h* contain functions that measure ambient temperature for E-INK display compensation

See the corresponding header/source files for more details.

Figure 30 shows the RTOS firmware flow of this code example.

[1] For a detailed list of files included in the E-INK Library, see the code example, CE218136 – PSoC 6 MCU E-INK Display with CapSense (RTOS)

Figure 30. RTOS Firmware Flow



Components

Table 3. List of PSoC Creator Components

Component	Instance Name	Purpose
BLE	BLE	The BLE Component is configured for Limited Discovery with custom characteristics. These characteristics are used for notifying the BLE Central device of CapSense Slider, CapSense Buttons, and read/write RGB LED control data.
CapSense	CapSense	The CapSense Component scans a 5-segment slider (CSD) and two buttons (CSX) with SmartSense auto-tuning.
Digital Output Pin	Pin_LED_Red Pin_LED_Orange	These GPIOs are configured as firmware controlled digital output pins that control status LEDs.
	Pin_RGB_Red Pin_RGB_Blue Pin_RGB_Green	These GPIOs are configured as digital output pins with hardware connections. These pins route PWM signals to RGB LED.
Digital Input Pin	Pin_Advertise	This pin is configured as a digital input pin that is used to generate interrupts when the user button (**SW2**) is pressed.
Global Signal Reference	GlobalSignal	The global signal component is configured to extract interrupts from **Pin_Advertise** pin.
PWM	PWM_Red PWM_Blue PWM_Green	These three TCPWMs are configured in PWM mode to control the color of the RGB LED.
UART	DEBUG_UART	UART is used to transmit debug information to a terminal (disabled by default)

Note: See the code example CE218136 – PSoC 6 MCU E-INK Display with CapSense (RTOS) for more details on components used by E-INK library and temperature compensation.

See the PSoC Creator project for more details of PSoC Component configurations and design wide resource settings.

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project
AN85951 – PSoC 4 and PSoC 6 MCU CapSense Design Guide	Describes how to design Capacitive touch sensing applications with PSoC 6 MCU
AN215656 – PSoC 6 MCU: Dual-Core CPU system Design	Describes the dual-core CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-core design
AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project	Describes how to import the code generated by PSoC Creator into your preferred IDE
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
Bluetooth Low Energy	Supports BLE connectivity.
CapSense	Supports touch sensing
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kit Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	
Training Videos	
PSoC 6 101: Lesson 1-4 FreeRTOS	

Document History

Document Title: CE220331 – PSoC 6 MCU with BLE Connectivity: BLE with User Interface (RTOS)

Document Number: 002-20331

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6086612	NIDH	03/02/2018	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 Forums | 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, 2018. 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. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. 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. 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.