CE218137 – PSoC 6 MCU with BLE Connectivity: BLE with Proximity (RTOS)

Objective

This code example demonstrates connectivity between the PSoC® 6 MCU with Bluetooth Low Energy (BLE) and CySmart™ BLE host emulation tool or mobile device running the CySmart mobile application, to transfer CapSense® proximity sensing information.

Overview

This code example demonstrates connectivity between the PSoC 6 MCU with BLE Connectivity (PSoC 6 MCU), which acts as a Peripheral and GATT Server device, and CySmart BLE host emulation PC tool or mobile device running the CySmart mobile application (acting as a Central and GATT Client). A custom BLE service is used for the proximity sensor.

In more detail:

- An “always-on” E-INK display that shows the instructions to use the code example. The E-INK display remains ON after a restart, while consuming no power for display retention.
- CapSense proximity sensor
- BLE connectivity
  - Advertisement and connection with any Central device
  - Custom BLE profile and service
  - Data transfer over BLE using notifications

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, you can find introductions in 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

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

Populate **J13** header with a proximity wire provided with the kit. Form a loop with the proximity wire as Figure 1 shows for increased proximity range.

Figure 1. Hardware Setup

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

The code example can be verified using either of these methods: the CySmart BLE Host Emulation Tool and BLE dongle on a PC or the CySmart mobile application.

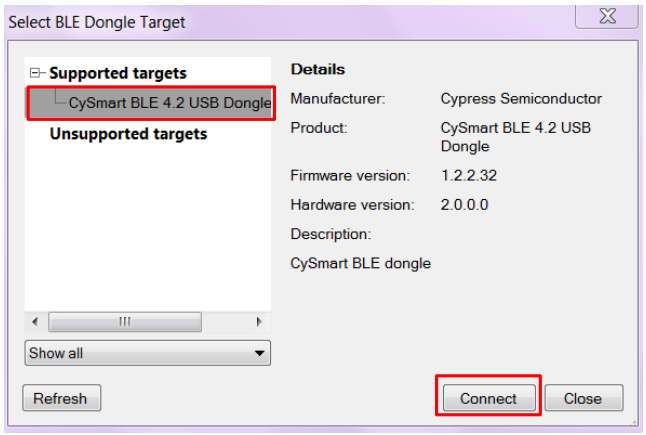
CySmart BLE Host Emulation Tool

To verify the *CE218137_BLE_Proximity* 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



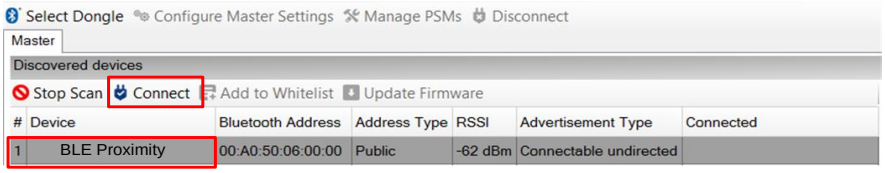
3. Power the Pioneer Board through the USB connector **J10**.
4. Program the Pioneer Board with the *CE218137_BLE_Proximity* project. See the Pioneer Kit guide for details on how to program firmware into the device.

   After programming successfully, the E-INK display will refresh and show the instructions to use this project and the 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.

Figure 3. BLE Advertising

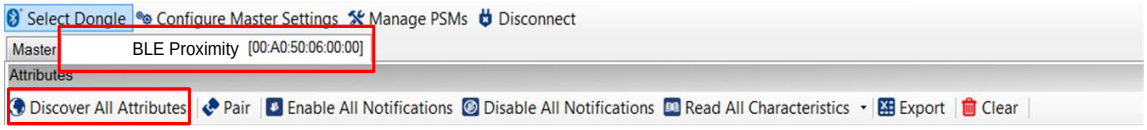
5. If the BLE advertisement has timed out (**LED8** is OFF), press **SW2** to restart advertisement.
6. On the CySmart Host Emulation tool, click **Start Scan** to see the list of available BLE Peripheral devices. Double-click the **BLE Proximity** device to connect, or click **BLE Proximity** and then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

Figure 4. Connect to BLE Proximity Peripheral



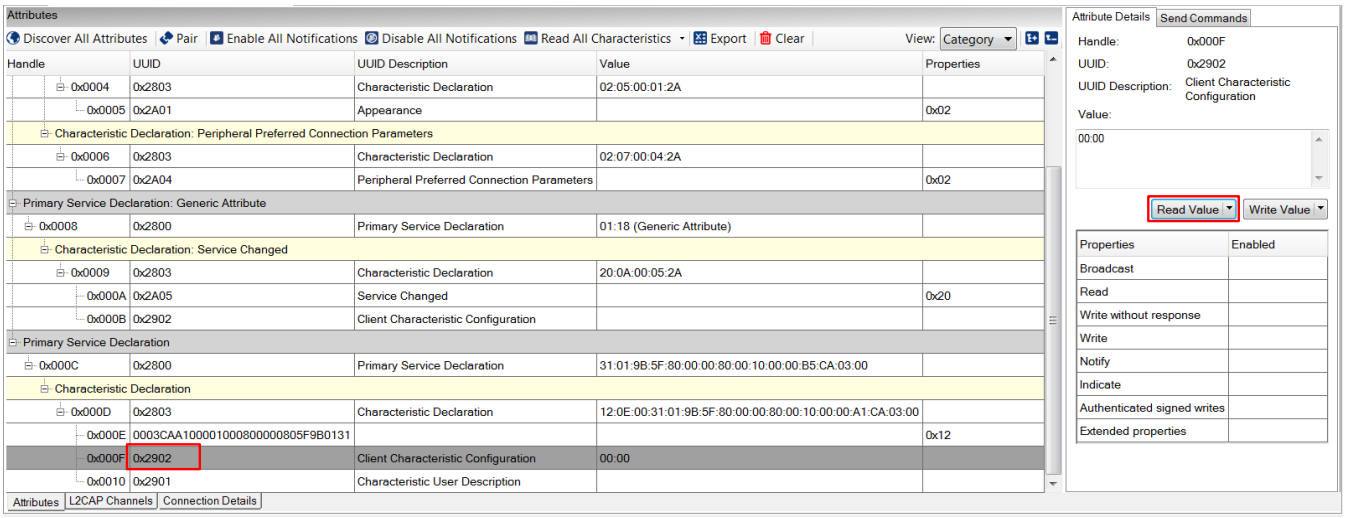
7. Click **Discover All Attributes** to find all attributes supported.

Figure 5. Discover All Attributes



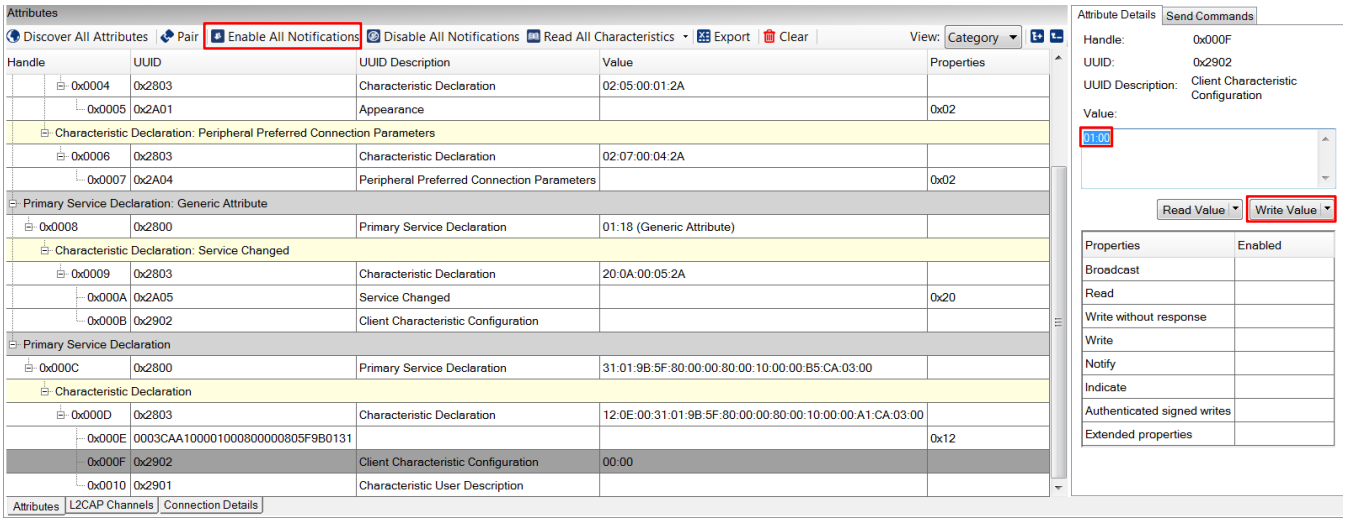
8. Locate the attribute **Client Characteristic Configuration** descriptor (UUID 0x2902) under the CapSense Proximity characteristic (**UUID 0x0003CAA2000010008000000805F9B0131**). Click **Read Value** to read the existing Client Characteristic Configuration Descriptor (CCCD) value as shown in Figure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

Figure 6. Read CCCD for CapSense Proximity Characteristic



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

Figure 7. Write CCCD to Enable Notifications



10. Bring your hand close to the proximity sensor, as shown in Figure 8 and see the notification values in the CapSense Proximity value field, as shown in Figure 9.

Figure 8. CapSense Proximity Testing

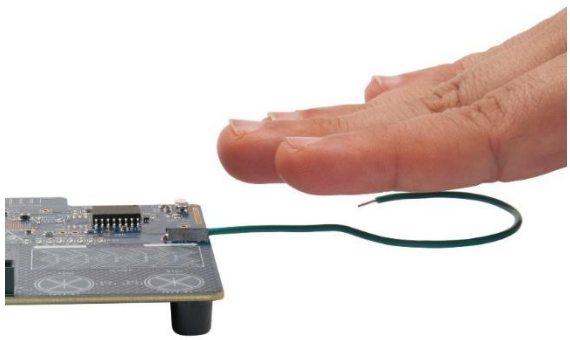
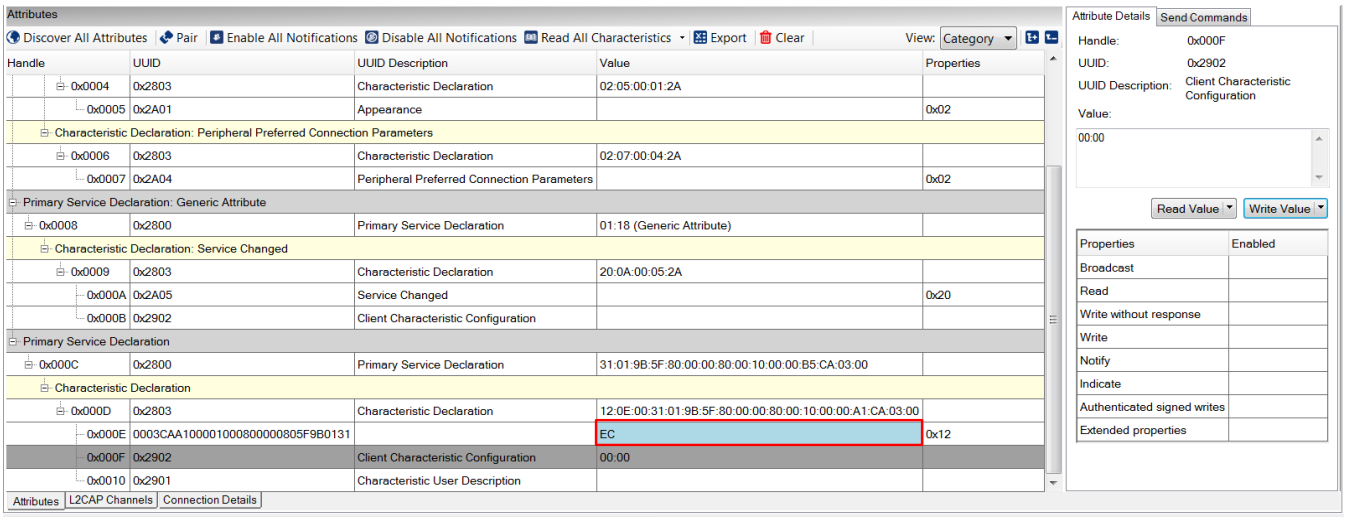
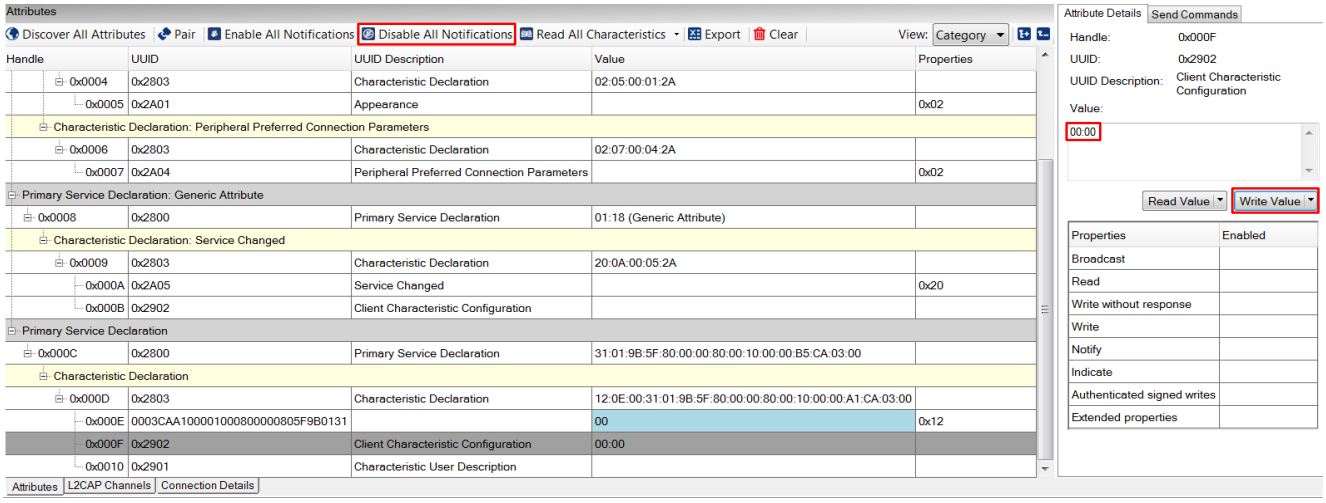


Figure 9. CapSense Proximity Notification Received



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. To disconnect from the device, click **Disconnect**, as shown in Figure 11. The red LED (**LED9**) will turn ON for three seconds to indicate a disconnect event. Press **SW2** to restart the advertisement, if required.

Figure 11. Disconnect from the Device

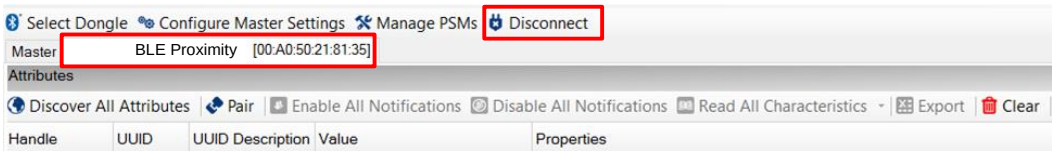
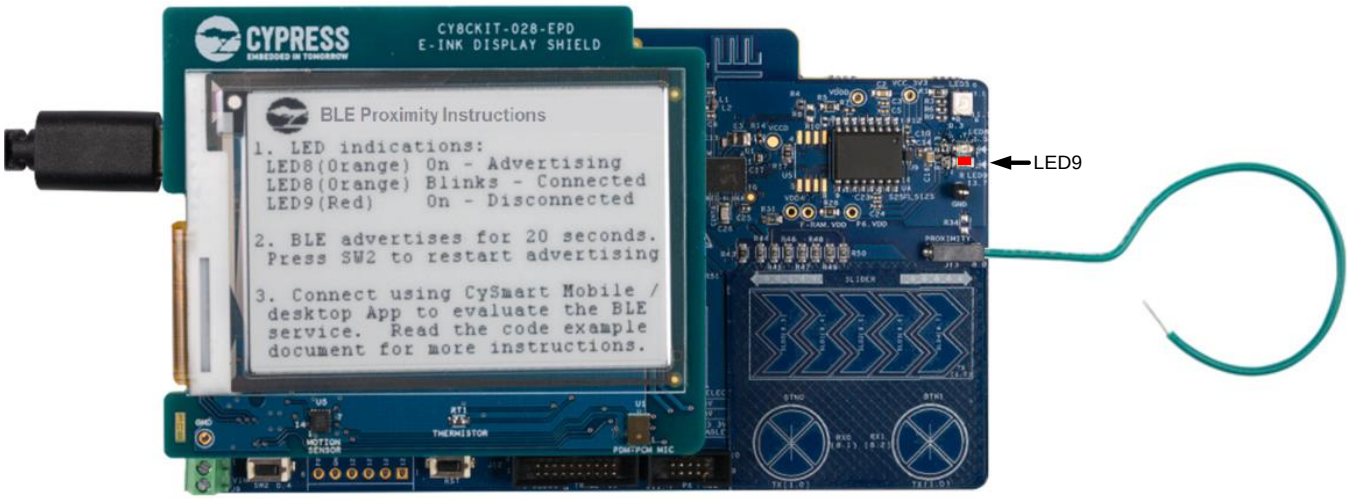


Figure 12. Disconnect Indication



CySmart Mobile Application

To verify the *CE218137_BLE_Proximity* code example using the CySmart mobile application (See the CySmart Mobile App webpage), follow these steps:

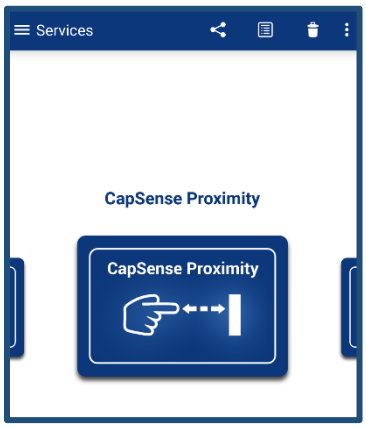
1. Install the CySmart app.
2. Power the Pioneer Board through the USB connector **J10**.
3. Program the Pioneer Board with the *CE218137_BLE_Proximity* project. See the Pioneer Kit guide for details on how to program firmware into the device.

   After programming successfully, the E-INK display will refresh and show the instructions to use this code example and the 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.
4. If the BLE advertisement has timed out (**LED8** is OFF), press **SW2** to restart advertisement. See the figures in the prior 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 ask to enable it.
6. After Bluetooth is enabled, the CySmart mobile application will automatically search for available devices and will list them. Select the **BLE Proximity** peripheral as shown in Figure 13. A successful connection is indicated by **LED8** continuously blinking at half-second intervals.

Figure 13. BLE Proximity Peripheral

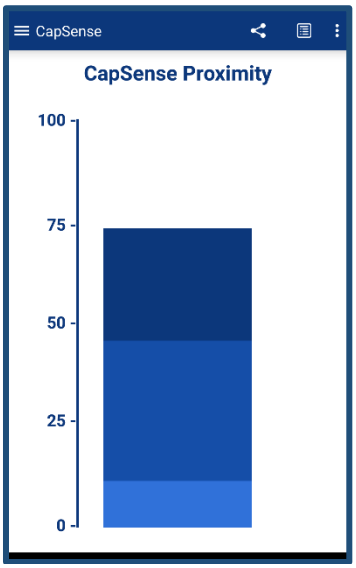
7. When connected, the CySmart mobile application will list the services supported by the device. Scroll and select the CapSense Proximity icon, as shown in Figure 14.

Figure 14. CapSense Proximity Service Page



8. Bring your hand close to the proximity sensor, as shown in Figure 8, and see a similar response on the CapSense Proximity bar graph in the CySmart application (see Figure 15).

Figure 15. CapSense Proximity Response



9. 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 as well as enable and disable notifications.

10. 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.

Design and Implementation

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).

The BLE profile in this code example consists of a BLE custom service called CapSense Proximity. The CapSense Proximity service consists of a custom characteristic that is used to send data as notifications to the GATT Client device. The notification data consists of the proximity signal read by the CapSense Component from a proximity wire attached to header **J13** on the Pioneer Board. This characteristic supports notification, which allows the GATT Server to send data to the connected Client device whenever new data is available. The properties for the custom service/characteristics are configured in the BLE Component under the **GATT Settings** tab, as shown in Figure 16.

Figure 16. BLE CapSense Proximity Service Configuration

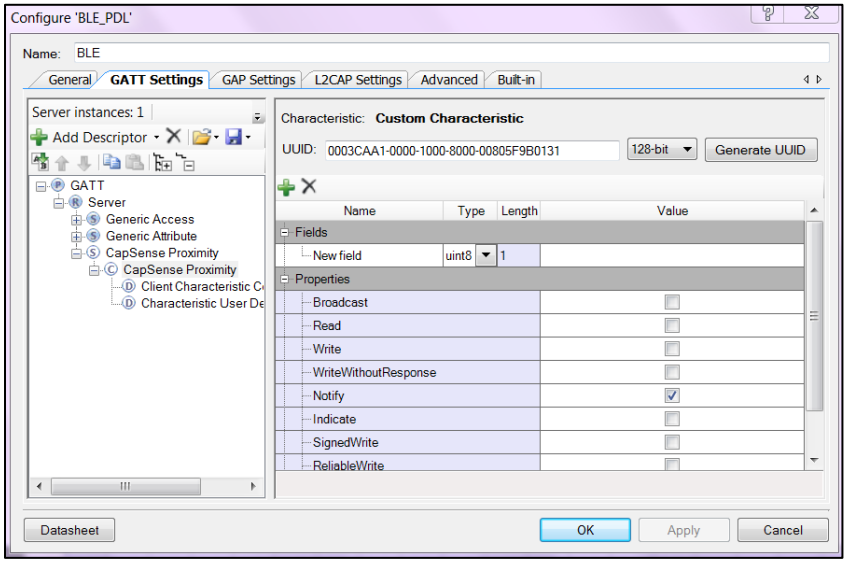


Figure 17 , Figure 18 and Figure 19 show the TopDesign schematic of this code example.

Figure 17. TopDesign Schematic: BLE and Proximity

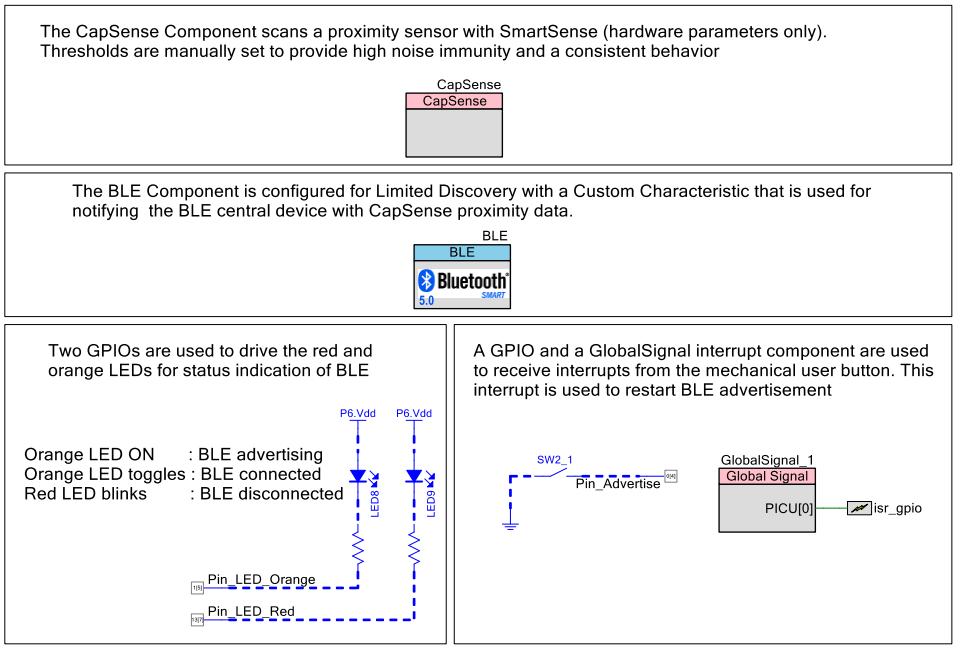


Figure 18. TopDesign Schematic: E-INK Display

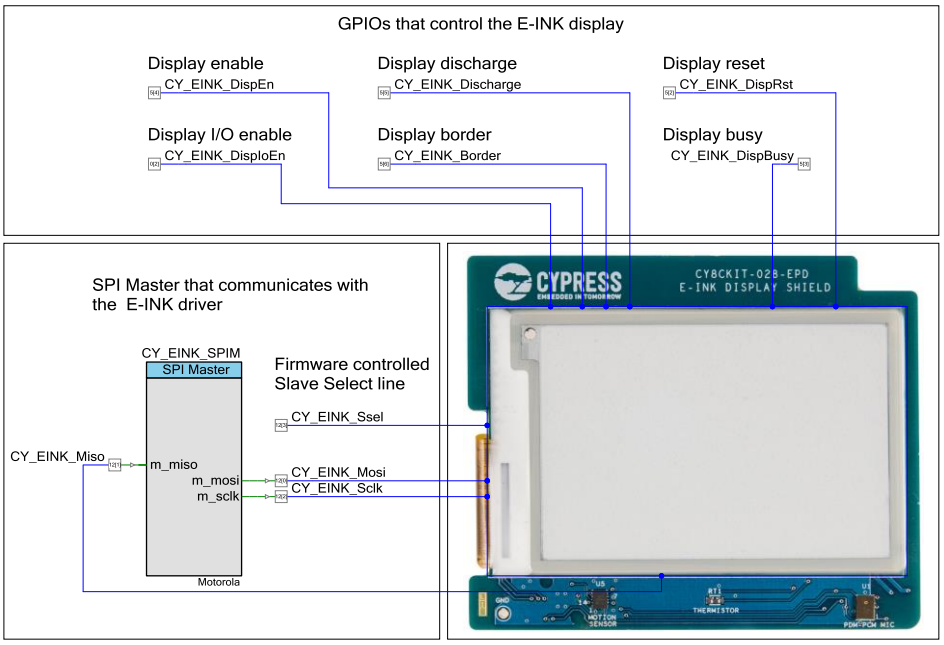
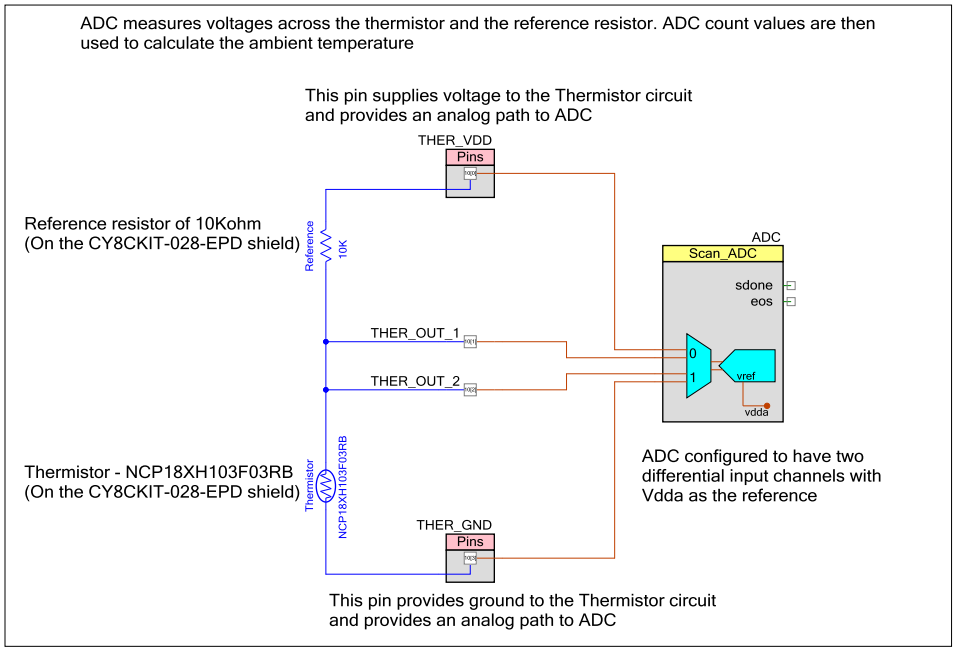


Figure 19. TopDesign Schematic: Temperature Compensation for E-INK Display



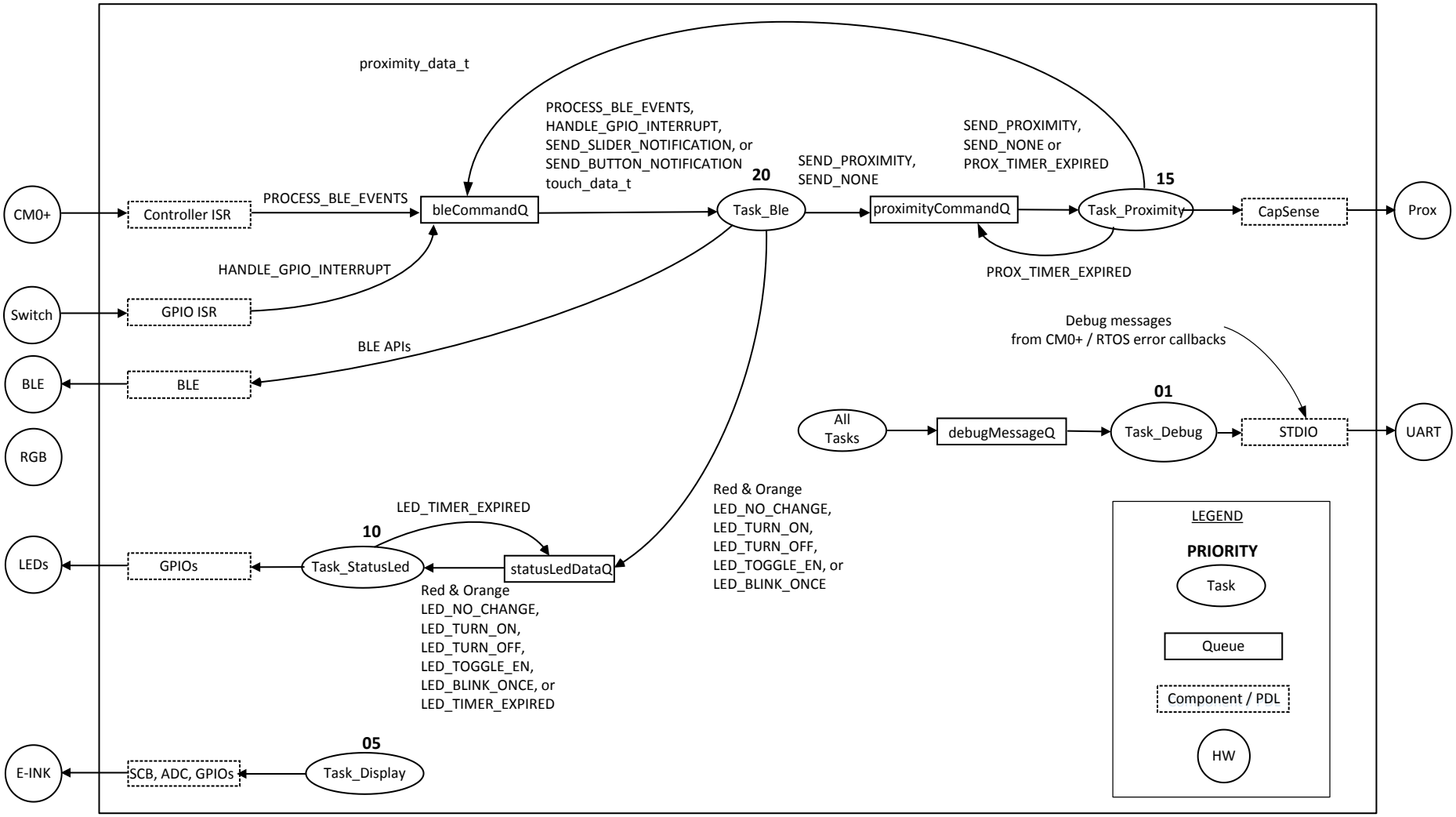
The code example consists of the following files:

- *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_proximity_service.h* contains the macros and datatypes used for the custom BLE proximity service.
- *touch_task.c/h* contain the task that scan CapSense sensors and process the data.
- *rgb_led.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.

Figure 20 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 20. RTOS Firmware Flow



Components

Table 2. List of PSoC Creator Components

Component	Instance Name	Function
BLE	BLE	The BLE Component is configured for Limited Discovery with a Custom Characteristic that is used for notifying the BLE Central device of CapSense Proximity data.
CapSense	CapSense	The CapSense Component scans a proximity sensor with SmartSense (hardware parameters only). Thresholds are manually set to provide high noise immunity and a consistent behavior.
MCWDT	MCWDT	The MCWDT Counter0 is configured to generate periodic interrupts at 0.5-second intervals. MCWDT interrupts are used to control status LEDs and turn them OFF when not required to save power.
Digital Output Pin	Pin_LED_Red Pin_LED_Orange	These GPIOs are configured as firmware-controlled digital output pins that control status LEDs.
Digital Input 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 **Advertise** pin.

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.

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: CE218137 – PSoC 6 MCU with BLE Connectivity: BLE with Proximity (RTOS)

Document Number: 002-18137

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6096840	NIDH	04/30/2018	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.