CE11181 - BLE HTTP Proxy Code Examples with PSoC® 4 BLE

Objective

This code example demonstrate the HTTP Proxy Client and Server operation of the BLE PSoC® Creator Component.

Overview

HTTP Proxy Server and HTTP Proxy Client projects are used in a pair to demonstrate the BLE HTTP Proxy Service (HPS) operation. HTTP Proxy Server utilizes one instance of HTTP Proxy Service to simulate HTTP Server on the BLE device. HTTP Proxy Server can also operate with other devices that implement the HTTP Proxy Client Role. To conserve power the device switches to the DeepSleep mode between the BLE connection intervals.

HTTP Proxy Client is designed to operate with HTTP Proxy Server that can process GET and POST HTTP requests.

Requirements

Tool: PSoC Creator 4.0 or later

Programming Language: C (GCC 4.9 or later)

Associated Parts: PSoC® 4 BLE parts

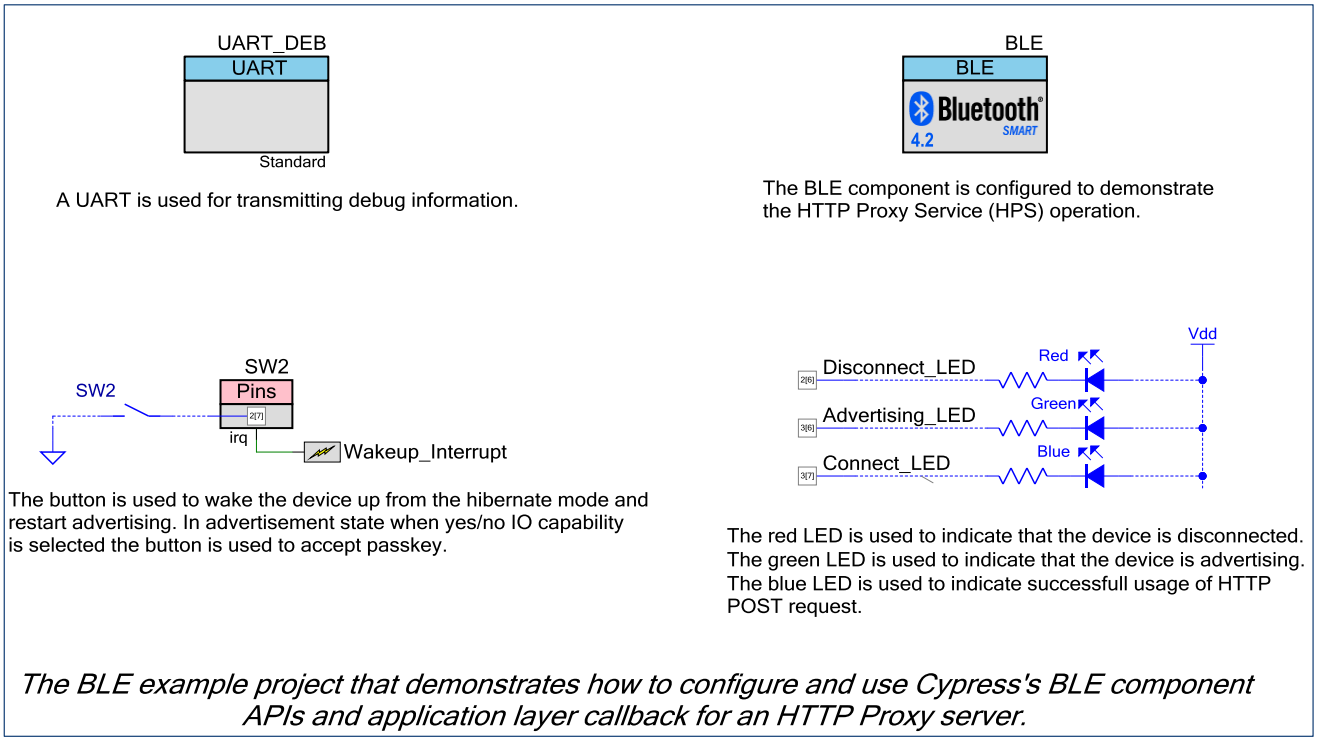
Related Hardware: Two CY8CKIT-042 PSoC® 4 Pioneer Kits with PSoC® 4 BLE devices

HTTP Proxy Server Code Example

Design

Figure 1 depicts the schematic of BLE HTTP Proxy Service Server Project.

Figure 1. BLE HTTP Proxy Service Server Project Schematic



The project demonstrates the functionality of the BLE component configured as HTTP Proxy Server. The project is designed to work with HTTP Proxy Client project provided with this document but it also can be used with CySmart.

After a startup, the device initializes the BLE component. To operate, the component requires two callback functions in order to receive events from the BLE Stack. The AppCallBack() is used to receive general BLE events. Another callback (HpsCallBack()) is used to receive events specific to the service's attribute operations.

The CYBLE_EVT_STACK_ON event indicates a successful initialization of the BLE Stack. After receiving this event the component starts fast advertising with the packet structure as configured in the BLE component customizer (see Figure 6). Once the 30-second advertising period expires, the component switches to the slow advertisement parameters. On an advertisement event timeout, if HTTP Proxy Server is not connected to any Client, the device goes to the Low Power mode (Hibernate mode) and waits for a SW2 button press to wake up the device again and start advertising.

You can connect to the HTTP Proxy Server device with HTTP Proxy Client or any BLE 4.1 or BLE 4.2 compatible device configured in the GAP Central role capable of discovering HTTP Proxy Service. To connect to an HTTP Proxy Server device, send a connection request to the device when the device is advertising. The blinking green LED indicates that the device is advertising. If the Client is connected to HTTP Proxy Server, the green LED will stop blinking.

Design Considerations

Using UART for debugging

A HyperTerminal program is required in a/your PC to receive debugging information. If you don't have a HyperTerminal program installed, download and install any serial port communication program. Freeware such as HyperTerminal, Bray's Terminal, Putty etc. is available on the web.

1. Connect the PC and kit with a USB cable.
2. Open the device manager program in your PC, find the COM port in which the kit is connected, and note the port number.
3. Open the HyperTerminal program and select the COM port into which the kit is connected.
4. Configure the Baud rate, Parity, Stop bits, and Flow control information in the HyperTerminal configuration window. By default, the settings are following: Baud rate – 115200, Parity – None, Stop bits – 1 and Flow control – XON/XOFF. These settings have to match the configuration of the PSoC Creator UART component in the project
5. Start communicating with the device as explained in the Operation section.

The UART debugging can be disabled by setting the `DEBUG_UART_ENABLED` to `DEBUG_UART_DISABLED` in common.h.

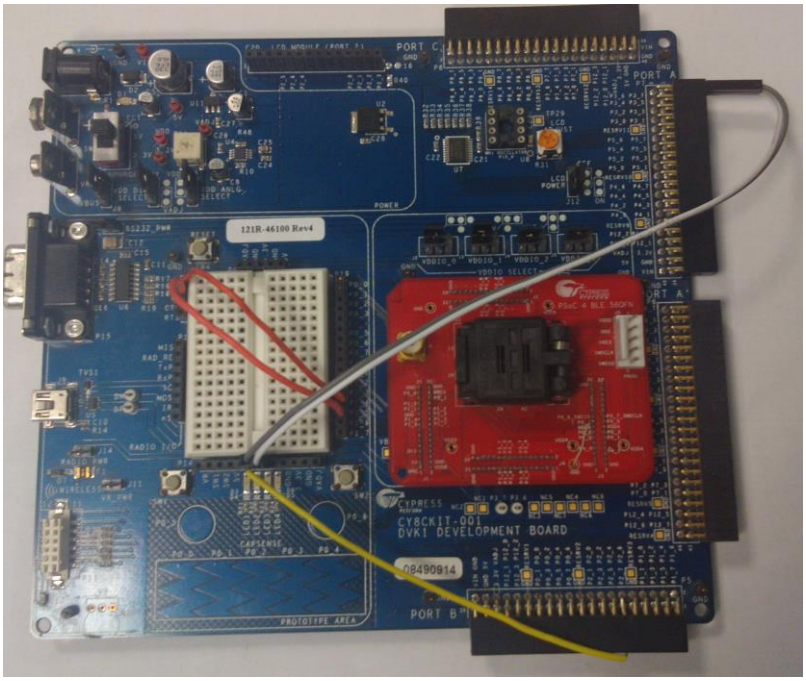
Hardware Setup

The code example was created for CY8CKIT-042 PSoC® 4 Pioneer Kit. The project utilizes the internal connections and default kit configuration so nothing needs to be done to configure the kit.

The project can also be run on CY8CKIT-001 PSoC® Development Kit. For this the following configuration should be done:

1. Connect pin 1[4] and pin 1[5] on header P19 to RX and TX pins on P16 header respectively.
2. Connect pin 2[7] on header P19 to pin SW1 or SW2 on header P14.
3. Connect pin 2[6] on header P5 and pins 3[6], 3[7] on header P7 to pins LED1, LED2 and LED3 on header P14. Keep in mind that RGB LED on CY8CKIT-042 PSoC® 4 Pioneer Kit is driven in reverse way then the LEDs on CY8CKIT-001 PSoC® Development Kit. In other words the high signal for any of RGB LEDs are disabling the LED and low signal enables the LED. So on CY8CKIT-001 PSoC® Development Kit the turned on LED will actually mean "LED is off" and vice versa.

Figure 2. CY8CKIT-001 connections for BLE HTTP Proxy Code Examples



Components

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

Table 1. List of PSoC Creator Components

Component	Hardware Resources
UART_DEB	1 SCB
BLE	1 BLE, 1 Interrupt
SW2	1 pin
Wakeup_Interrupt	1 interrupt
Disconnect_LED, Advertising_LED, Connect_LED	3 pins

Parameter Settings

The BLE component is configured as HPS Server in the GAP Peripheral role with the settings shown in the figures below. A custom profile is used as there's no standard profile defined for HPS. The BLE component that represents HTTP Proxy Server is configured as follows:

Figure 3. GATT Settings

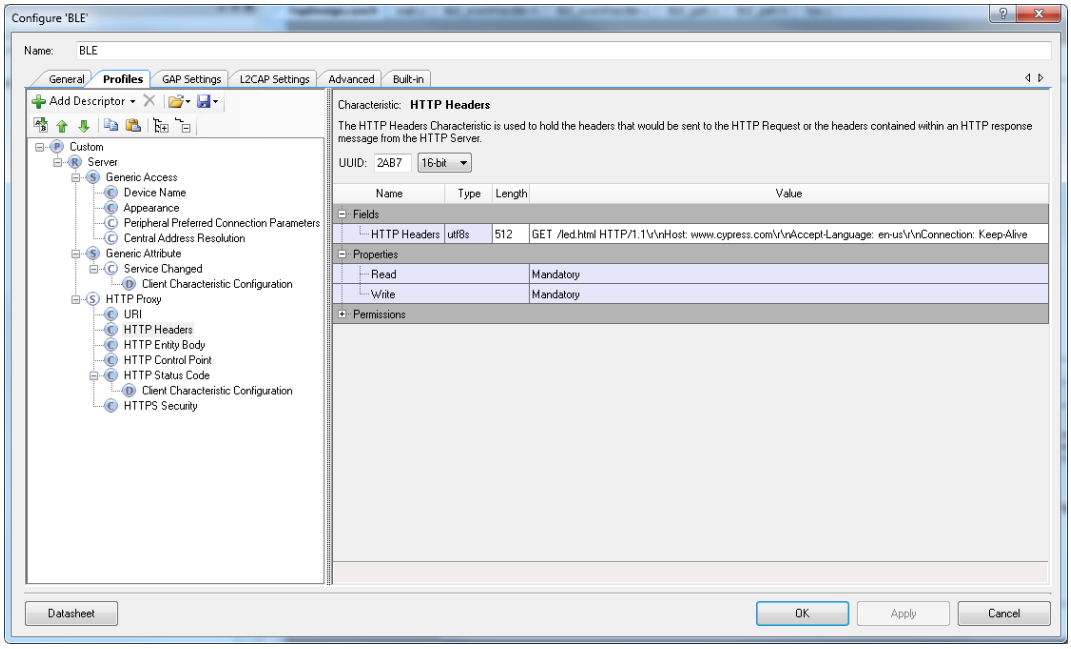


Figure 4. GAP Settings

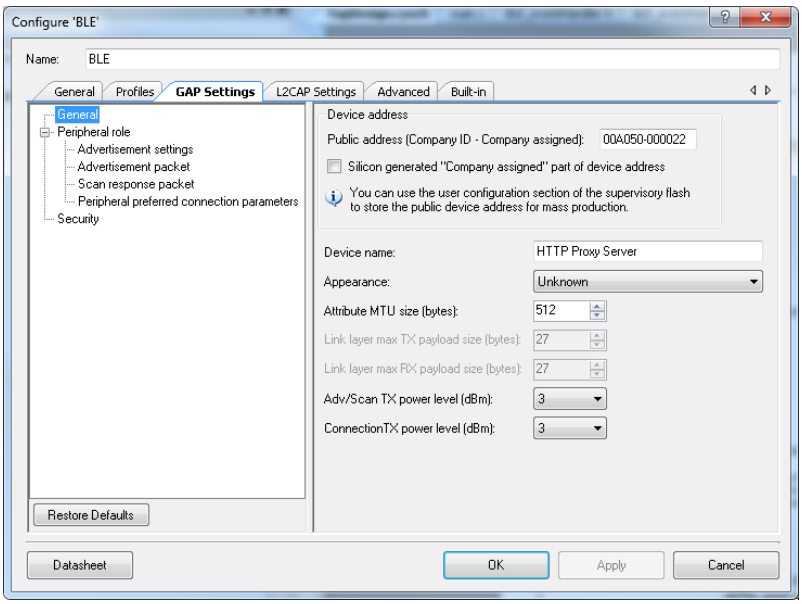


Figure 5. GAP Settings -> Advertisement Settings

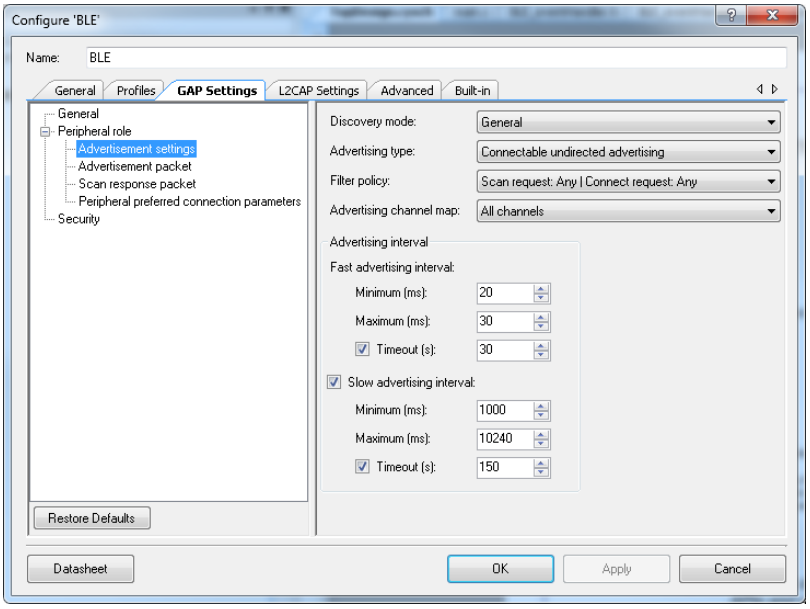


Figure 6. GAP Settings -> Advertisement Packet

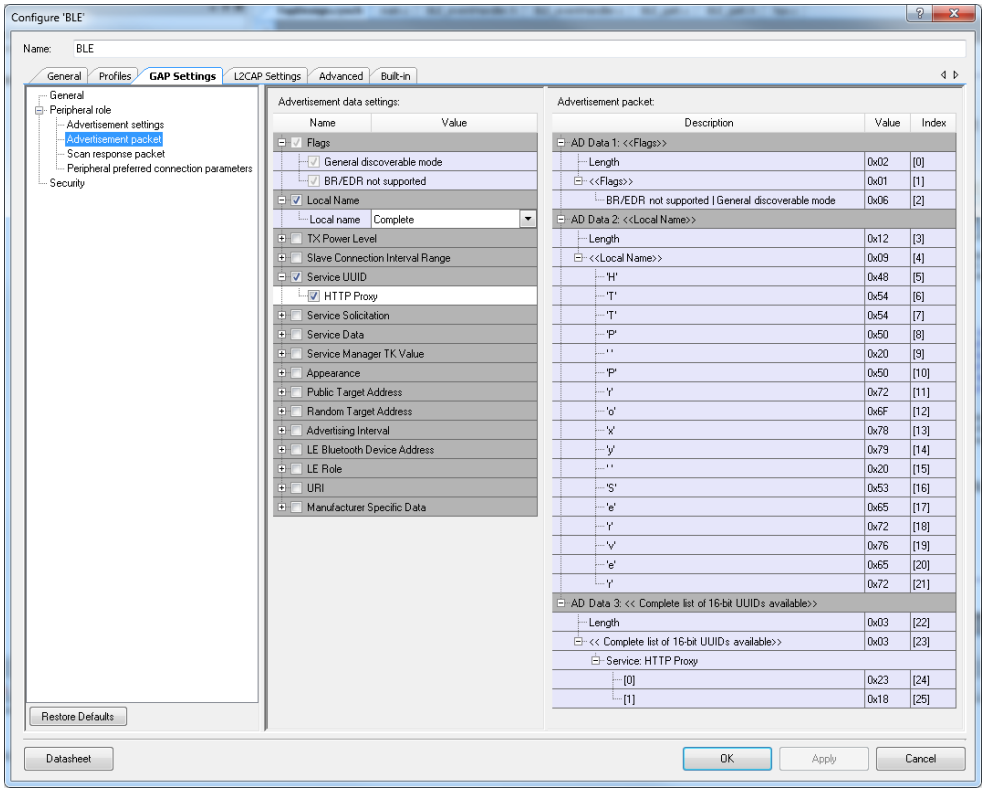
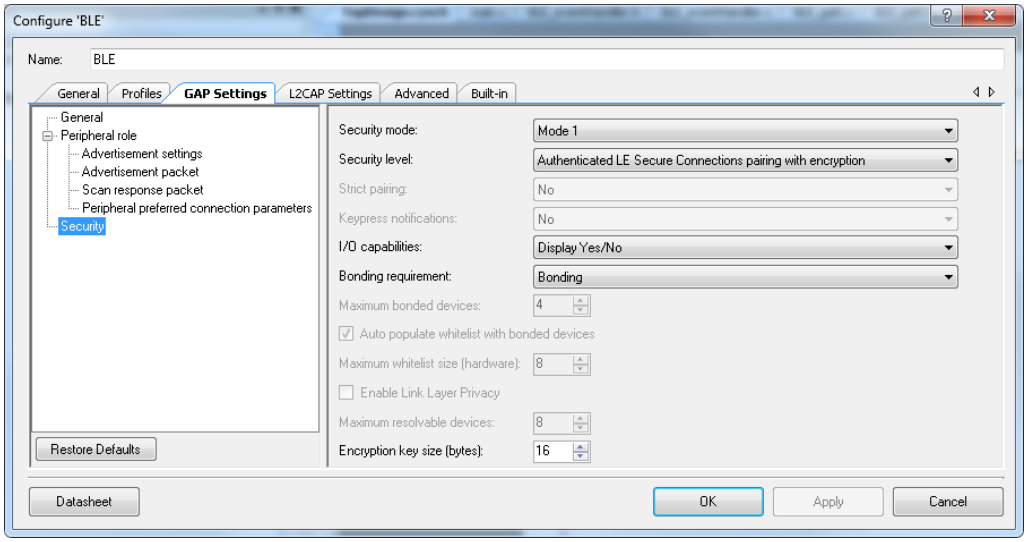


Figure 7. Security Settings

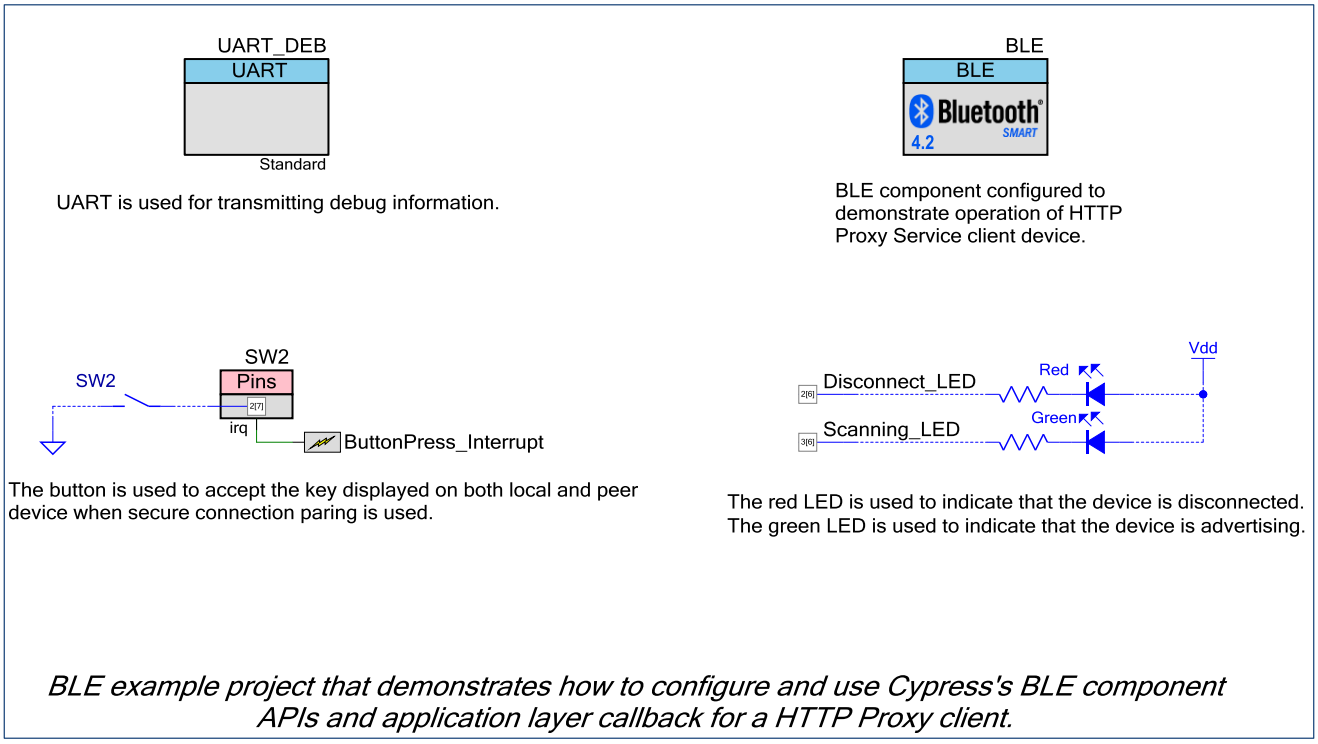


HTTP Proxy Client Code Example

Design

Figure 8 depicts the schematic of BLE HTTP Proxy Service Client Project.

Figure 8. BLE HTTP Proxy Service Client Project Schematic



The project demonstrates the functionality of BLE HTTP Proxy Client. This project is designed to work with BLE HTTP Proxy Server provided with this document.

The project is configured to operate using secure connection. Button “SW2” on the CY8CKIT-042 is used to accept the password displayed on HyperTerminal. This can also be done by pressing ‘y’ on HyperTerminal. Optionally, the project can use legacy Security Mode 1 Level 3 (Authenticated pairing with encryption).

The project supports only one URI - "http://www.cypress.com/led" and two HTTP requests – HTTP GET and HTTP POST. The HTTP GET request is used to read the state of the blue LED on a device programmed with BLE HTTP Proxy Server project. The HTTP POST request is used to set the state of the blue LED on BLE HTTP Proxy Server device.

To start the project operation, build it and program it onto PSoC 4100-BL or PSoC 4200-BL device. Right after the startup, the device initializes the BLE component as well as UART and ISR components. In this project, two callback functions are required for the BLE operation. One callback function (AppCallBack()) is required for receiving general BLE events from BLE Stack and the HpsAppEventHandler() required for receiving events related to the HTTP Proxy Service.

The CYBLE_EVT_STACK_ON event indicates a successful initialization of the BLE Stack. After this event is received, the project will prompt a message asking for start scanning. The green LED will start blinking to indicate that the device has started scanning. During the scanning, the project should prompt scan reports from the BLE devices that are advertising and are available for connection. After selecting the proper device from a prompted list, the project will connect to the selected device and initiate secure connection paring. The green LED will stop blinking after the device stops scanning. When secure pairing is confirmed, the project will discover the connected Server device and configure the HPS service for notifications. Then it will prompt a message asking to send an HTTP GET request to HTTP Proxy Server. Next, upon successful execution of the HTTP GET request, the project will ask to send an HTTP POST request to an HTTP Proxy Server.

In a connected state, the device is configured in Deepsleep low power mode between consecutive connection intervals. When the device is not in a connected state and scanning is inactive, the red LED will glow to indicate that the device is in disconnected state.

Design Considerations

Using UART for operation

A HyperTerminal program is required in a/your PC to receive debugging information. If you don't have a HyperTerminal program installed, download and install any serial port communication program. Freeware such as HyperTerminal, Bray's Terminal, Putty etc. is available on the web.

1. Connect the PC and kit with a USB cable.
2. Open the device manager program in your PC, find the COM port in which the kit is connected, and note the port number.
3. Open the HyperTerminal program and select the COM port into which the kit is connected.
4. Configure the Baud rate, Parity, Stop bits, and Flow control information in the HyperTerminal configuration window. By default, the settings are following: Baud rate – 115200, Parity – None, Stop bits – 1 and Flow control – XON/XOFF. These settings have to match the configuration of the PSoC Creator UART component in the project
5. Start communicating with the device as explained in the Operation section.

Hardware Setup

The code example was created for CY8CKIT-042 PSoC® 4 Pioneer Kit. The project utilizes the internal connections and default kit configuration so nothing needs to be done to configure the kit.

The project can also be run on CY8CKIT-001 PSoC® Development Kit. The configuration steps for the project are the same as for HTTP Proxy Server Code Example shown above in this document.

Components

Table 2 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each.

Table 2. List of PSoC Creator Components

Component	Hardware Resources
UART_DEB	1 SCB
BLE	1 BLE, 1 Interrupt
SW2	1 pin
Wakeup_Interrupt	1 interrupt
Disconnect_LED, Scanning_LED	2 pins

Parameter Settings

The BLE component is configured as HPS Client in the GAP Central role with the settings shown in the figures below. A custom profile is used as there's no standard profile defined for HPS.

HTTP Proxy Client is configured as follows:

Figure 9. GATT Settings

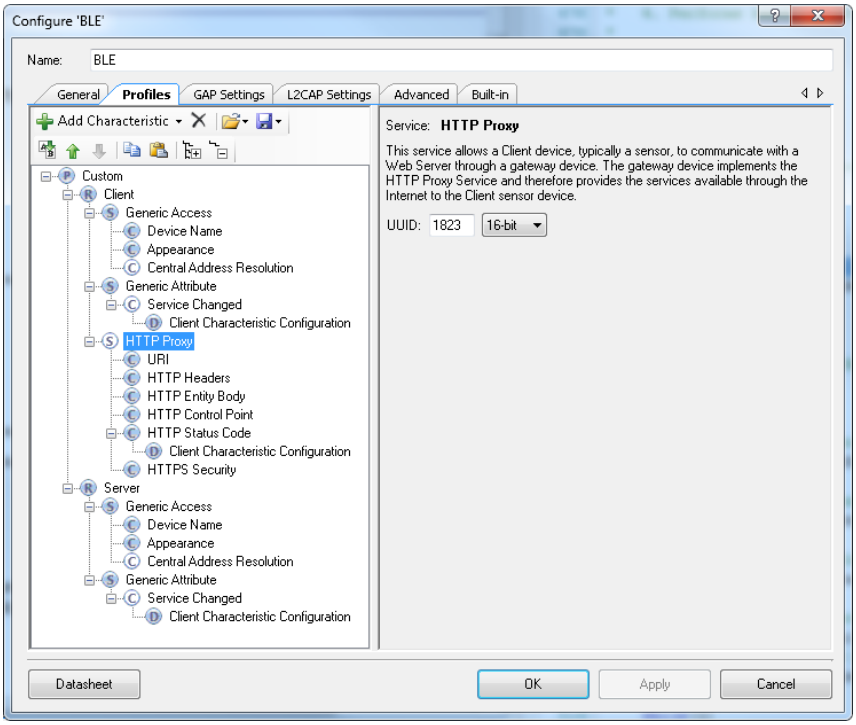


Figure 10. GAP Settings

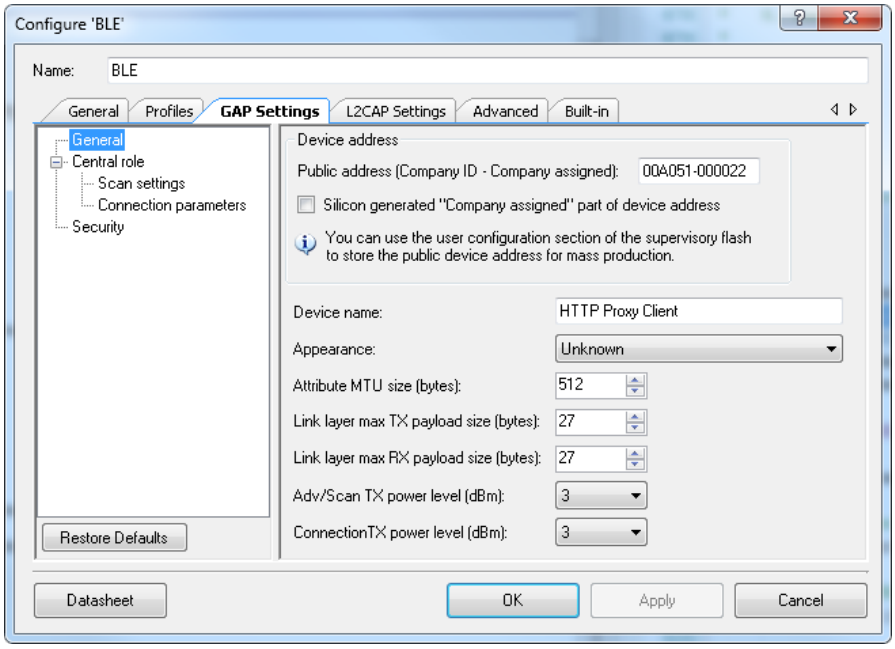


Figure 11. GAP Settings -> Advertisement Settings

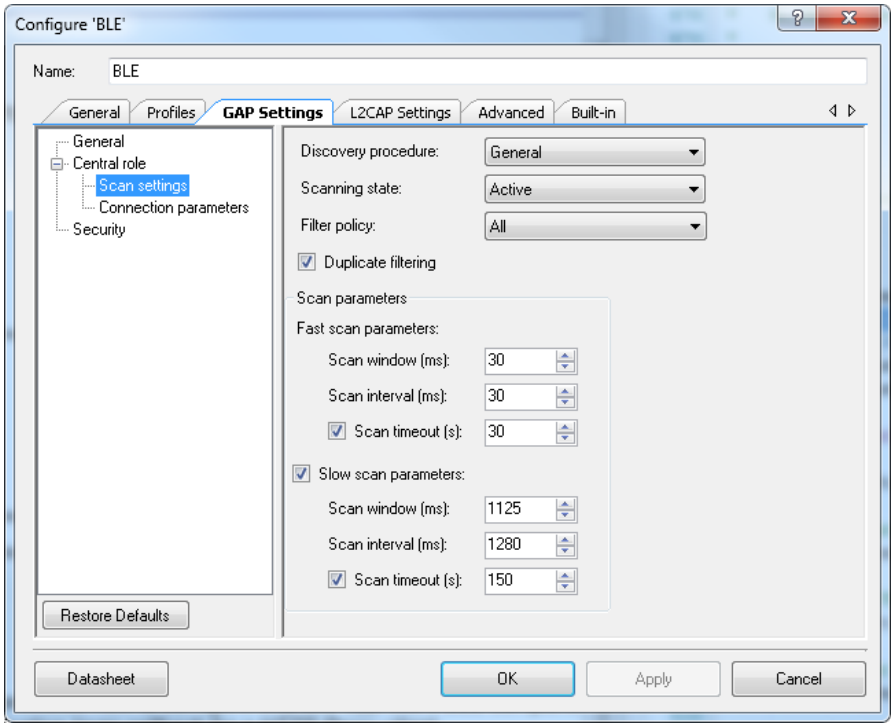
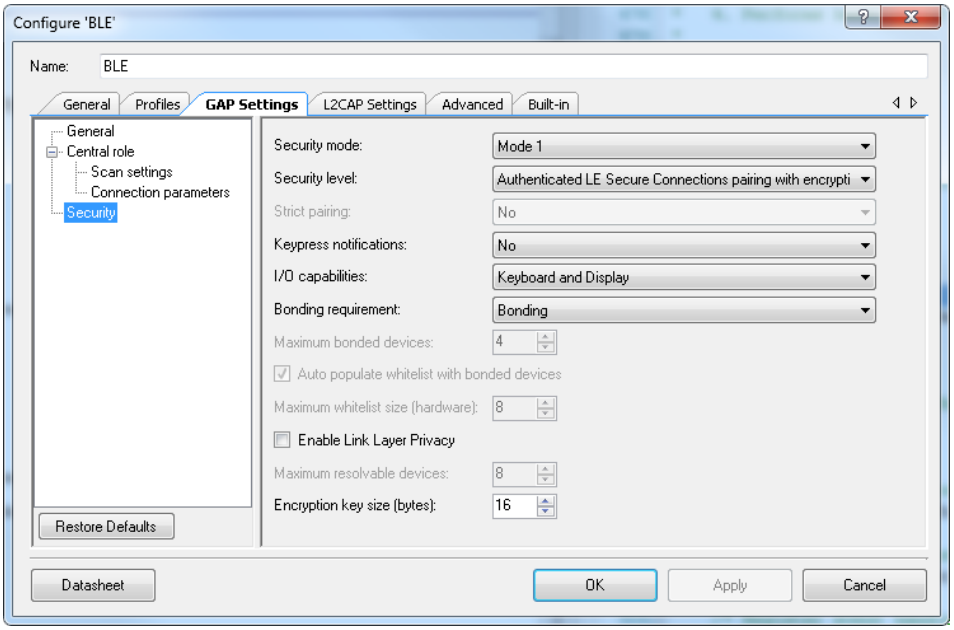


Figure 12. Security Settings



Operation

1. Build and program BLE HTTP Proxy Server and BLE HTTP Proxy Client into CY8CKIT-042 PSoC® 4 Pioneer Kits with PSoC® 4 BLE devices.
2. Run two HyperTerminal (Bray's Terminal, Putty etc.) instances. One - for BLE HTTP Proxy Server and one for BLE HTTP Proxy Client.
3. In the BLE HTTP Proxy Client HyperTerminal window, press 's' to start scanning for advertising devices. When the scan report from the device with address **0x00A050000022** is received – press 'z'. Then select the number corresponding to the device with address **0x00A050000022** and press 'c'. An approximate example of an output on Client's HyperTerminal should look like following:

```
Bluetooth On. Device address is: 00a051000022
Press 's' to start scanning for BLE devices.
When you see a scan report from the device you wish to connect to - press 'z'.
CyBle_GapcStartScan API Success
GAPC_START_SCANNING
CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT:
   eventType: 0
   peerAddrType: 0
   peerBdAddr: 0a0500022
   dataLen: 1a
   data: 02 01 06 12 09 48 54 54 50 20 50 72 6f 78 79 20 53 65 72 76 65 72 03 03 23 18
   rssi: 0
CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT:
   eventType: 4
   peerAddrType: 0
   peerBdAddr: 0a0500022
   dataLen: 4
   data: 03 19 00 00
   rssi: 4
CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT:
   eventType: 3
   peerAddrType: 0
   peerBdAddr: 321321321321
```

```
    dataLen: 1c
    data: 02 01 06 04 08 49 50 53 03 03 21 18 0f 25 7d e5 06 00 80 00 00 00 80 aa ff ff ff 00
    rssi: 3
Device 0: 00a050000022
Device 1: 321321321321
Select the device from the list above using numbers from 0 to 1 and then press 'c'.0
Stop Scanning ...
Scan complete!
Connect to the device: 0
Connect to the device with address - 00a050000022
CYBLE_EVT_GATT_CONNECT_IND
CyBle_GapcStartScan API Success
CYBLE_EVT_DEVICE_CONNECTED: 0, 17( ms), 0, 3e8
CYBLE_EVT_GAP_SMP_NEGOTIATED_AUTH_INFO: security:3, bonding:1, ekeySize:10, authErr 0
Compare this passkey with the one displayed in your peer device and press 'y' or 'n': 677338
```

4. When message "**Compare this passkey with displayed in your peer device and press 'y' or 'n':**" is prompted on HyperTerminal, and if the passkey that follows that message matches on both HyperTerminals, press button "SW2" on CY8CKIT-042 PSoC® 4 Pioneer Kits or optionally press 'y' on both HyperTerminals. Wait until the Client completes pairing and the peer device discovery.

5. When message "**Press 'g' to send GET request to get the state of the LED on the peer device**" is prompted on HyperTerminal, press 'g'. Wait until the Client handles sending an HTTP GET request and receiving a response from BLE HTTP Proxy Server. A Client's HyperTerminal's output should look like following:

```
An HTTP 'GET' request was sent
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE    Characteristic: CYBLE_HPS_URI
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE    Characteristic: CYBLE_HPS_HTTP_HEADERS
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE    Characteristic: CYBLE_HPS_HTTP_ENTITY_BODY
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE    Characteristic: CYBLE_HPS_HTTP_CP
CYBLE_EVT_HPSC_NOTIFICATION           Characteristic: CYBLE_HPS_HTTP_STATUS_CODE
                    Status code: 200  "Succes"
CYBLE_EVT_HPSC_READ_CHAR_RESPONSE     Characteristic: CYBLE_HPS_HTTP_HEADERS
Char data:Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=iso-8859-1
Connection: Closed
CYBLE_EVT_HPSC_READ_CHAR_RESPONSE     Characteristic: CYBLE_HPS_HTTP_ENTITY_BODY
Char data:<html>
<head>
<title>LED State page</title>
</head>
<body>
<p>LED is off.</p>
</body>
</html>
```

6. When message "**Press '1' to send POST request to turn on the LED on the peer device or press '0' to send POST request to turn off the LED on the peer device.**" is prompted on HyperTerminal, press '1'. Wait until the Client handles sending an HTTP POST request (to set the state of the blue LED) and receiving a response from BLE HTTP Proxy Server. Observe the blue LED is turned on BLE HTTP Proxy Server device. Press '0' and wait until the Client handles sending the HTTP POST request (to clear the state of the blue LED) and receiving a response from BLE HTTP Proxy Server. Observe the blue LED is turned off on the BLE HTTP Proxy Server device. A Client's HyperTerminal's output for LED on request should look like following:

```
An HTTP 'POST' request with parameter 'LED ON' was sent
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE    Characteristic: CYBLE_HPS_URI
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE    Characteristic: CYBLE_HPS_HTTP_HEADERS
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE    Characteristic: CYBLE_HPS_HTTP_ENTITY_BODY
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE    Characteristic: CYBLE_HPS_HTTP_CP
CYBLE_EVT_HPSC_NOTIFICATION           Characteristic: CYBLE_HPS_HTTP_STATUS_CODE
                    Status code: 200  "Succes"
CYBLE_EVT_HPSC_READ_CHAR_RESPONSE     Characteristic: CYBLE_HPS_HTTP_HEADERS
```

```
Char data:Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=iso-8859-1
Connection: Closed
CYBLE_EVT_HPSC_READ_CHAR_RESPONSE      Characteristic: CYBLE_HPS_HTTP_ENTITY_BODY
Char data:<html>
<head>
<title>LED State page</title>
</head>
<body>
<p>LED is on.</p>
</body>
</html>
```

A Client's HyperTerminal's output for LED off request should look like following:

```
An HTTP 'POST' request with parameter 'LED OFF' was sent
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE     Characteristic: CYBLE_HPS_URI
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE     Characteristic: CYBLE_HPS_HTTP_HEADERS
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE     Characteristic: CYBLE_HPS_HTTP_ENTITY_BODY
CYBLE_EVT_HPSC_WRITE_CHAR_RESPONSE     Characteristic: CYBLE_HPS_HTTP_CP
CYBLE_EVT_HPSC_NOTIFICATION            Characteristic: CYBLE_HPS_HTTP_STATUS_CODE
                    Status code: 200  "Succes"
CYBLE_EVT_HPSC_READ_CHAR_RESPONSE      Characteristic: CYBLE_HPS_HTTP_HEADERS
Char data:Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=iso-8859-1
Connection: Closed
CYBLE_EVT_HPSC_READ_CHAR_RESPONSE      Characteristic: CYBLE_HPS_HTTP_ENTITY_BODY
Char data:<html>
<head>
<title>LED State page</title>
</head>
<body>
<p>LED is off.</p>
</body>
</html>
```

7. Press 'd' on any of HyperTerminal to disconnect the devices.

Note When the bonding info is removed from BLE HTTP Proxy Server (this may happen when the devices were previously bonded and the Server device is reprogrammed but the Client isn't), a pairing attempt will fail with INSUFFICIENT_ENCRYPTION_KEY_SIZE error. To address this, reprogram the Client device or delete the bonding information by pressing 'r' on BLE HTTP Proxy Client's HyperTerminal.

Related Documents

The following lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component datasheets.

Table 3. Related Documents

Application Notes		
AN94020	Getting Started with PRoC™ BLE	Introduces you to PRoC™ BLE, an ARM® Cortex™-M0 based programmable radio-on-chip with Bluetooth Low Energy.
AN91267	AN91267 - Getting Started with PSoC® 4 BLE	Introduces you to PSoC® 4 BLE, an ARM® Cortex™-M0 based Programmable System-on-Chip (PSoC) that integrates a Bluetooth Low Energy (BLE) radio system.

Application Notes		
AN91184	PSoC 4 BLE - Designing BLE Applications	Shows how to design the Bluetooth® Low Energy (BLE) application based on PSoC 4 BLE, using standard profiles defined by the Bluetooth SIG included in the BLE Component in PSoC Creator. Demonstrates how to build an application with the BLE Health Thermometer Profile on the CY8CKIT-042-BLE kit.
AN91162	Creating a BLE Custom Profile	Describes the methodology for developing a Bluetooth® Low Energy (BLE) application with PSoC 4 BLE or PRoC BLE devices using a custom BLE profile.
Video		
PSoC 4 BLE 101: Intro to Bluetooth Low Energy		This is the first installment of a series of getting-started videos on Cypress Bluetooth Low Energy solutions.
PSoC 4 BLE 101: 2 Configuring a Find Me Profile with BLE		Using Cypress Pioneer kit with a PSoC 4 Radio module. Alan Hawse walks through a simple example for a find-me tag.
PSoC 4 BLE 101: 3 Finishing the Find Me Application with Firmware		In this lesson, we take the Find Me profile you configured in the previous video and add the firmware required to make it work on the PSoC 4 BLE.
PSoC 4 BLE 101: 4 Adding Battery Level Service and Testing with CySmart		This lesson takes the Find Me profile built in the first two lessons and adds a Battery Level service.
PSoC 4 BLE 101: 5 Using CapSense with Bluetooth Low Energy		In this BLE lesson, we show how to use PSoC Creator's Custom Service to quickly and easily add a CapSense slider to a BLE (Bluetooth Low Energy) design.
PSoC 4 BLE 101: 6 Extending Battery Life with PSoC Low Energy Modes		Adds power savings into your BLE designs easily using PSoC and PSoC Creator. In the last lessons, we created Find Me peripheral with the Battery Level service.
Software and Drivers		
CySmart – Bluetooth® LE Test and Debug Tool		CySmart is a Bluetooth® LE host emulation tool for Windows PCs. The tool provides an easy-to-use Graphical User Interface (GUI) to enable customers to test and debug their Bluetooth LE peripheral applications.
PSoC Creator Component Datasheets		
Bluetooth Low Energy (BLE) Component		The Bluetooth Low Energy (BLE) Component provides a comprehensive GUI-based configuration window to facilitate designing applications requiring BLE connectivity.
PSoC 4 Serial Communication Block (SCB) Component		Supports a PSoC 4 multifunction hardware block that implements I^2C, SPI, UART, and EZI2C communications
Device Documentation		
PSoC® 4: PSoC 4XX7_BLE Family Datasheet Programmable System-on-Chip (PSoC®)		
PSoC® 4: PSoC 4XX8_BLE Family Datasheet - Programmable System-on-Chip (PSoC®)		
PSoC® 4: PSoC 4XX8_BLE 4.2 Family Datasheet Programmable System-on-Chip (PSoC®)		
Development Kit (DVK) Documentation		
Bluetooth® Low Energy Pioneer Kit (CY8CKIT-042-BLE)		

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 x:

- **Overview:** PSoC Portfolio, PSoC Roadmap
- **Product Selectors:** PSoC 1, PSoC 3, PSoC 4, or PSoC 5LP. In addition, PSoC Creator includes a device selection tool.
- **Datasheets:** Describe and provide electrical specifications for the PSoC 3, PSoC 4, and PSoC 5LP device families.
- **CapSense Design Guides:** Learn how to design capacitive touch-sensing applications with the PSoC 3, PSoC 4, and PSoC 5LP families of devices.
- **Application Notes** and **Code Examples:** Cover a broad range of topics, from basic to advanced level. Many of the application notes include code examples.
- **Technical Reference Manuals (TRM):** Provide detailed descriptions of the architecture and registers in each of the PSoC 3, PSoC 4, and PSoC 5LP device families.
- **PSoC Training Videos**: These videos provide step-by-step instructions on getting started building complex designs with PSoC.
- **Development Kits:**
  - CY8CKIT-042 and CY8CKIT-040, PSoC 4 Pioneer kits, are easy-to-use and inexpensive development platforms. These kits include connectors for Arduino™ compatible shields and Digilent® Pmod™ daughter cards.
  - CY8CKIT-049 is a series of very low-cost prototyping platform for sampling PSoC 4 devices.
  - CY8CKIT-030 and CY8CKIT-050 are designed for analog performance. They enable you to evaluate, develop, and prototype high-precision analog, low-power, and low-voltage applications powered by PSoC 3 and PSoC 5LP, respectively.
  - CY8CKIT-001 is a common development platform for all PSoC family devices.
- The MiniProg3 device provides an interface for flash programming and debug.

Document History

Document Title: BLE HTTP Proxy Code Examples with PSoC® 4 BLE - CE11181

Document Number: 002-11181

Revision	Date	Description of Change
**	08/18/16	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
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

cypress.com/psoc

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

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

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

© Cypress Semiconductor Corporation, 2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you under its copyright rights in the Software, a personal, non-exclusive, nontransferable license (without the right to sublicense) (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. Cypress also grants you a personal, non-exclusive, nontransferable, license (without the right to sublicense) 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 to the minimum extent that is necessary for you to exercise your rights under the copyright license granted in the previous sentence. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. 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 Company shall and hereby does release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. Company 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, 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.