PSoC 4 BLE Lab #3: CapSense Design with BLE Connectivity

Description

This lab teaches you how to create a Custom Profile by implementing an RGB LED controller through BLE. It also demonstrates how to combine CapSense and BLE in a system, by designing a slider application.

Pre-Reading

Custom Profile

The BLE standard provides you with an option to create your own Profile for a customized application. A Custom Profile can contain standard BLE Services as well as custom Services you define.

A Custom Service is user-defined, i.e., you can define your own Characteristics and their Descriptors. Each Service, Characteristic, and Descriptor will have its own UUID, which you define.

Recall, the UUID stands for Universally Unique Identifier, it's used to uniquely identify a Service, Characteristic, or Descriptor. All UUIDs are 128-bit in length but in order to reduce the data transfer over BLE, the UUIDs can be converted to a 16-bit or 32-bit value.

In an end-design it's recommended to use a 128-bit value for the UUID of custom services, characteristics, and descriptors. For simplicity, in this lab we'll use 16-bit values.

Custom Service for RGB LED

Cypress has defined a Custom Service for transferring RGB LED data over BLE. The Service has one Characteristic. The Characteristic details are given in Table 1. This Characteristic contains 4 bytes as described in Table 2. One byte each for the Red, Green and Blue LEDs and another byte for the overall LED intensity.

Custom Service UUID (16-bit): 0xCBBB

Custom Characteristic UUID (16-bit): 0xCBB1

Table 1: Custom Service for RGB LED

Characteristic	Details	Properties	Descriptors
Custom	Carries the RGB hue and brightness level information.	Read, Write	Characteristic User Description

The Read property signifies that the Characteristic can be read by the GATT Client. The Write property signifies that the Characteristic value can be changed by the GATT Client.

The Characteristic User Description Descriptor is a string to identify what the Custom Characteristic is. The GATT Client treats this as the name of the Characteristic.

Table 2: Custom Characteristic Fields for RGB LED

Field Name	Field Requirement	Size in Bytes	Additional Information
Red LED	Mandatory	1	Range: 0 to 255
Green LED	Mandatory	1	Range: 0 to 255
Blue LED	Mandatory	1	Range: 0 to 255
Intensity	Mandatory	1	Range: 0 to 255

Custom Service for CapSense Slider

Cypress has defined another Custom Service for CapSense functionality. The Service has one Characteristic for implementing sliders. See Table 3 for details.

Custom Service UUID (16-bit): 0xCAB5

Custom Characteristic UUID (16-bit): 0xCAA2

Table 3: Custom Service for CapSense Slider

Characteristic	Details	Properties	Descriptors
Custom	Carries the CapSense slider information. This is 1 byte data and the valid range is 0 to 255.	Notify	Client Characteristic Configuration Characteristic User Description

The Notify property means that the GATT Server (the BLE Pioneer Kit in this case) can send unsolicited notifications to the GATT Client, once notifications enabled by the GATT Client.

The Client Characteristic Configuration Descriptor (or CCCD) is used by the GATT Server to identify whether notifications are enabled or not. If the CCCD has a value of ‘1’, the notifications are enabled. A value of ‘0’ indicates disabled notifications.

Note: CySmart iOS/Android mobile apps can recognize the above defined custom UUIDs and display data in graphical format. A 3rd party app which is not programmed for these custom UUIDs will only be able to show data in raw format, similar to the GATT DB view of the CySmart iOS/Android mobile apps.

PSoC 4 BLE Lab #3: CapSense Design with BLE Connectivity

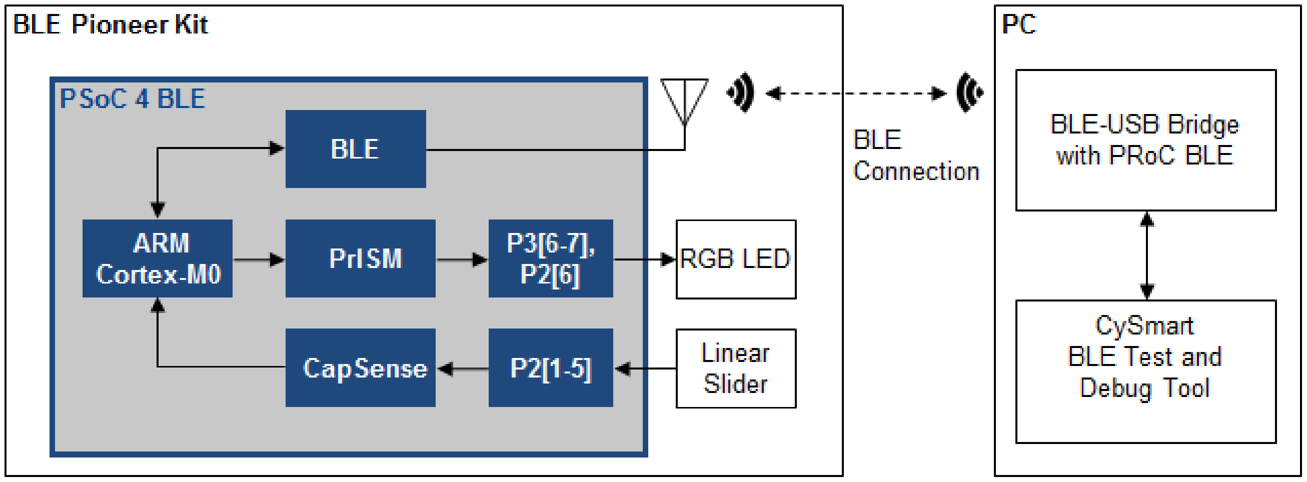
Objectives

1. Adjust RGB LED color and intensity using the PRiSM Component
2. Implement a custom BLE Profile with a custom Service to send RGB LED color and intensity over BLE
3. Implement a Custom Service to send CapSense slider data over BLE
4. Use the CySmart tool or mobile app to validate the operation

Requirements	**Details**
Hardware	BLE Pioneer Kit (CY8CKIT-042-BLE)
Software	PSoC Creator 3.1 (or newer)
	CySmart 1.0
	CySmart iOS or CySmart Android app

Block Diagram

Figure 1: Lab 3 Block Diagram



Background Check

This lab requires a basic working knowledge of PSoC Creator and the BLE Component. Ensure that you have covered Lab 1 and Lab 2 before proceeding.

Theory

This lab implements two Custom Services – one to control the RGB LED on the kit over BLE, and the other to send CapSense slider information over BLE.

To control the LED, the PrISM (Precision Illumination Signal Modulation) Component is used, implemented using Universal Digital Blocks (UDBs). The PrISM Component uses a Linear Feedback Shift Register (LFSR) to generate the user-adjustable pseudo random pulse densities, ranging from 0 to 100%. One PrISM block can generate two outputs, thus we need two PrISM blocks to drive the three LEDs on the kit. Together, these PrISM blocks control the hue and brightness of the RGB LEDs. This Component provides better EMI performance than using a standard PWM to control intensity. See the Component datasheet for additional details.

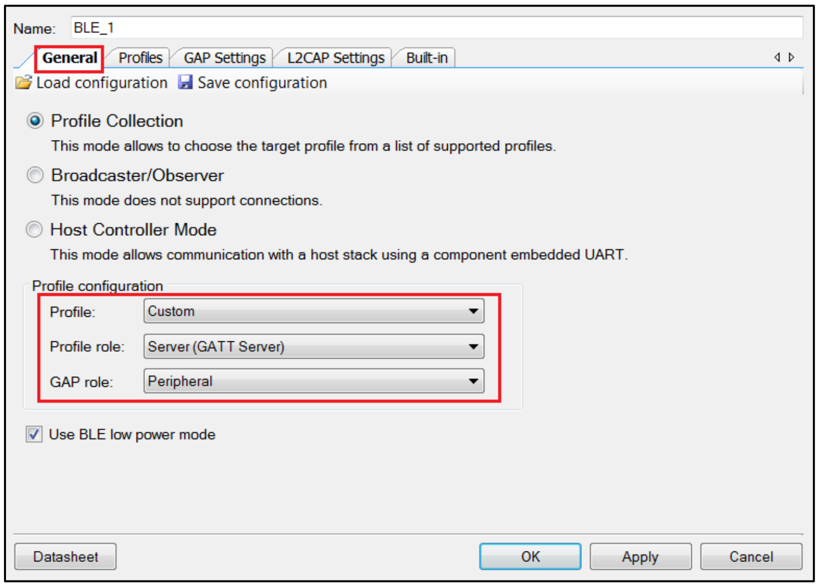
Procedure

We start this project with a template lab. Some of the details are already present, and you need to fill in the blanks as instructed. To get started, open the project **BLE Lab 3** and follow these instructions.

Configure Schematic

1. Open the schematic and place the **BLE Component**. Configure the Component as shown in the following steps. For more information, please refer to the BLE Component datasheet.
2. **General Tab** - Set the **Profile** to **Custom** and the **Profile role** to **Server (GATT Server)**. Set the **GAP role** to **Peripheral**. See Figure 2.

Figure 2: BLE Component - General Tab



Name: BLE_1

Profile Collection
This mode allows to choose the target profile from a list of supported profiles.
Broadcaster/Observer
This mode does not support connections.
Host Controller Mode
This mode allows communication with a host stack using a component embedded UART.

Profile configuration
Profile: Custom
Profile role: Server (GATT Server)
GAP role: Peripheral

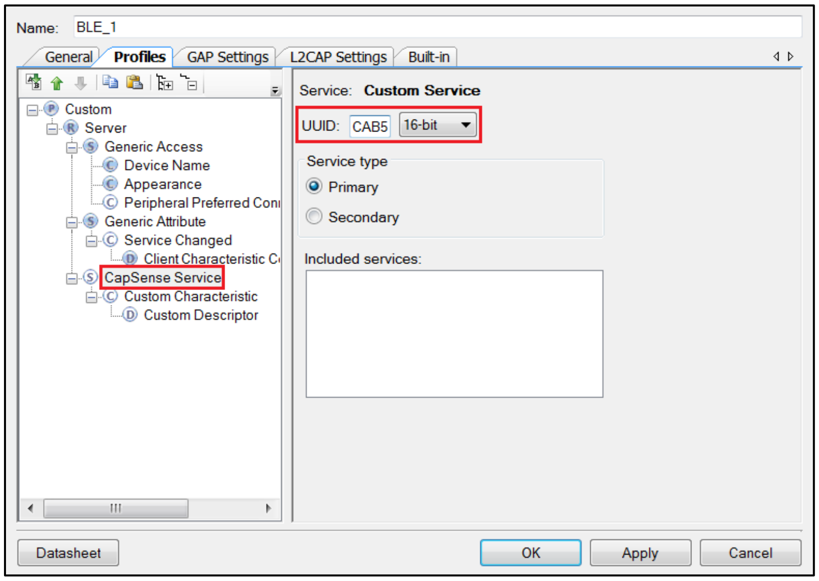
Use BLE low power mode

3. **Profiles Tab** - This tab has a custom service already. Configure that for CapSense Slider, and add another Custom Service for RGB LED. Follow these steps:

   3.1. **Custom Service for CapSense Slider**

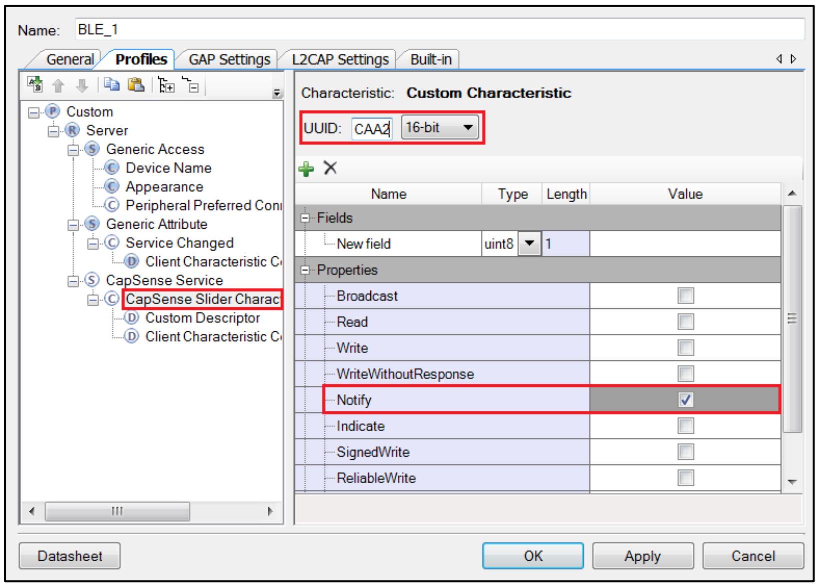
      a. Configure the Custom Service for CapSense Slider. Specifically:
      b. **Custom Service** - Rename this service to **CapSense Service** by right-clicking on **Custom Service** and choosing **Rename**.
      c. Set the **UUID length** as **16-bit** and **value** as **0xCAB5** (note: do NOT enter “0x” in the box, just the four hex values). This is the UUID defined by Cypress for this Service. See Figure 3.

Figure 3: Configuring Custom Service UUID for CapSense Slider



d. **Custom Characteristic** – Rename this to **CapSense Slider Characteristic**. Set the **UUID (16-bit)** to **0xCAA2**. Enable **Notify** in the **Properties** for the Characteristic. When you enable the Notify property, the **Client Characteristic Configuration Descriptor** is automatically added. See Figure 4.

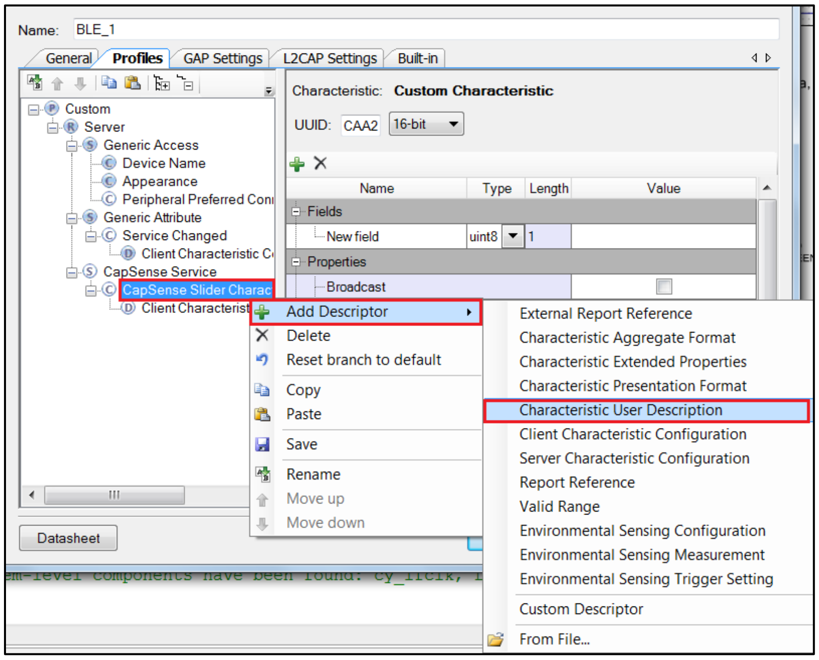
Figure 4: Configuring Custom Characteristic for CapSense Slider



e. Delete the **Custom Descriptor** by right-clicking on it and selecting **Delete**.

f. Click on the **CapSense Slider Characteristic**, and using the **Add Descriptor** drop down on the top, add the **Characteristic User Description** Descriptor. See Figure 5.

Figure 5: Add Descriptor for the CapSense Service



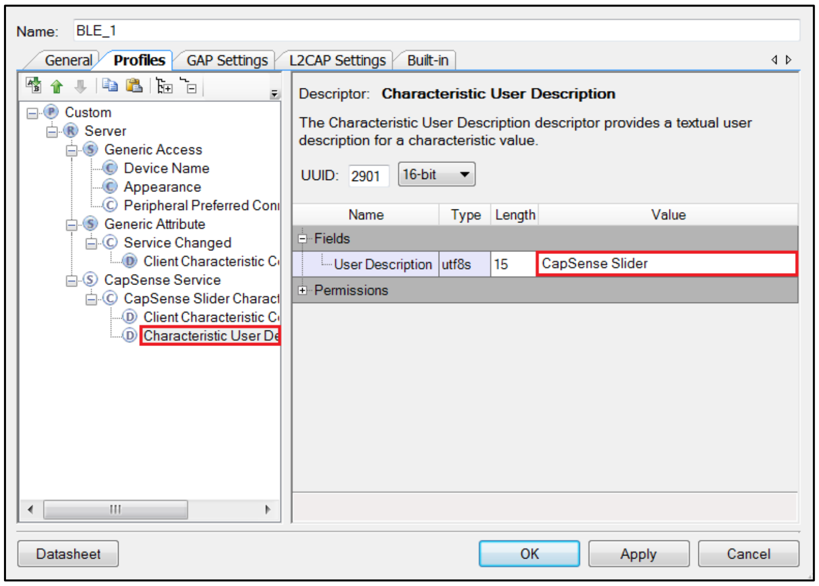
Characteristic: **Custom Characteristic**

UUID: CAA2 16-bit

Name	Type	Length	Value
New field	uint8	1	

g. Set the value of **Characteristic User Description** to **CapSense Slider**. See Figure 6.

Figure 6: Characteristic User Description for CapSense Slider Characteristic



Descriptor: **Characteristic User Description**

The Characteristic User Description descriptor provides a textual user description for a characteristic value.

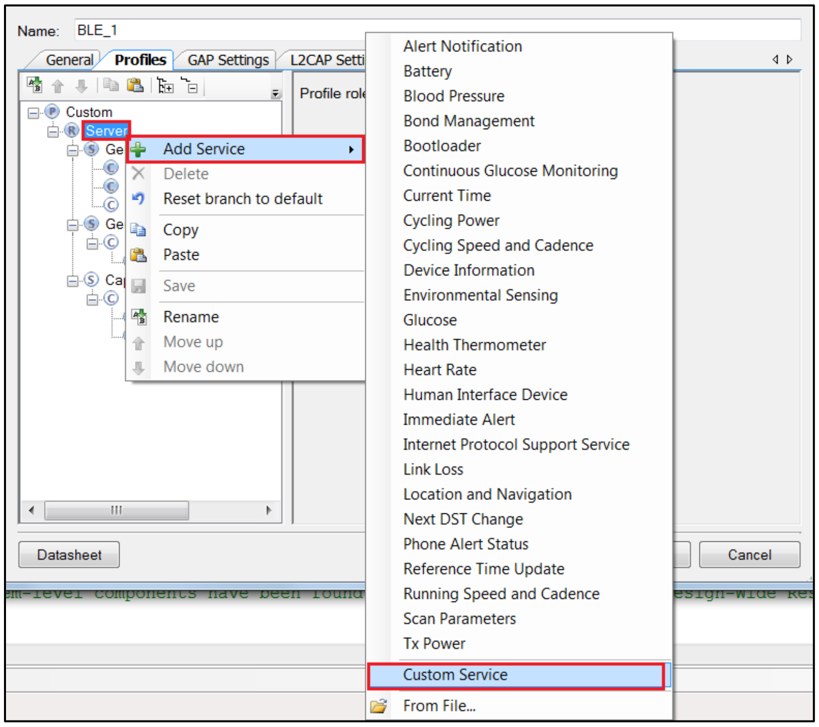
UUID: 2901 16-bit

Name	Type	Length	Value
User Description	utf8s	15	CapSense Slider

3.2. Custom Service for RGB LED

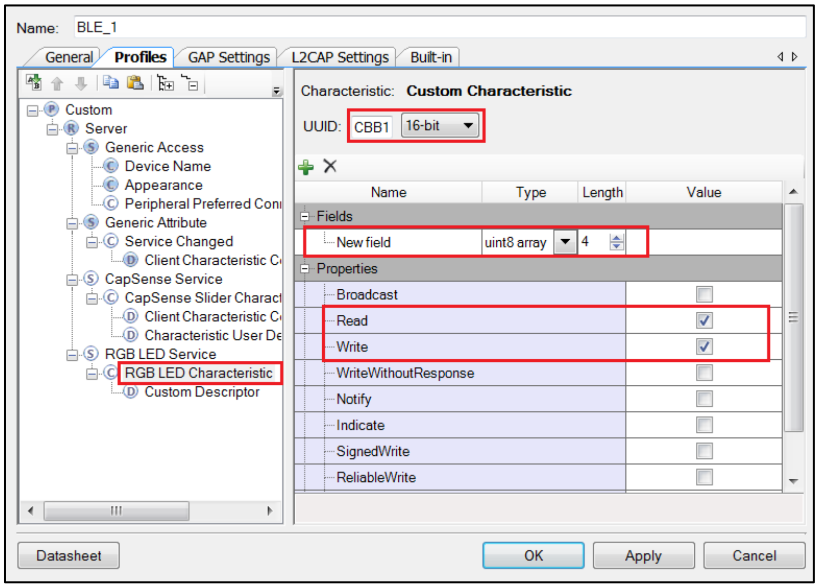
a. Add a new Custom Service to the **Profiles** page by right-clicking on **Server** and then selecting **Add Service -> Custom Service**. See Figure 7.

Figure 7: Add Custom Service for RGB LED



b. Rename this **Custom Service** to **RGB LED Service**. Set its **UUID (16-bit)** to **0xCBBB**.
c. Rename the **Custom Characteristic** in this service to **RGB LED Characteristic**. Set its **UUID (16-bit)** to **0xCBB1**. See Figure 8.
d. For the **RGB LED Characteristic**, change the Type of **New Field** under the **Fields** column to **uint8 array** and the **Length** to **4**, see Figure 8. This is because the Characteristic contains four bytes – for Red LED, Green LED, Blue LED, and overall intensity. See Table 2 on Page 1 for details on the characteristic.
e. Enable **Read** and **Write** in the Characteristic properties. See Figure 8.

Figure 8: Configuring Characteristic for RGB LED Service



Name: BLE_1

General | Profiles | GAP Settings | L2CAP Settings | Built-in

- Custom
  - Server
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Con
    - Generic Attribute
      - Service Changed
        - Client Characteristic C
    - CapSense Service
      - CapSense Slider Charact
        - Client Characteristic C
        - Characteristic User De
    - RGB LED Service
      - RGB LED Characteristic
        - Custom Descriptor

Characteristic: **Custom Characteristic**

UUID: CBB1 16-bit

Name	Type	Length	Value
Fields			
New field	uint8 array	4	
Properties			
Broadcast			☐
Read			✓
Write			✓
WriteWithoutResponse			☐
Notify			☐
Indicate			☐
SignedWrite			☐
ReliableWrite			☐

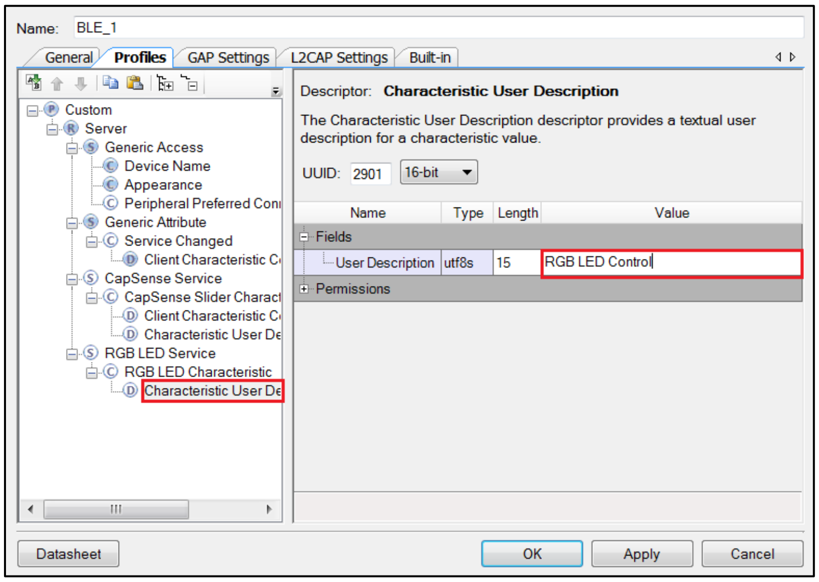
Datasheet | OK | Apply | Cancel

f. **Delete** the **Custom Descriptor**.

g. **Add** the **Characteristic User Description Descriptor**. Give it a value of **RGB LED Control**.

h. Your **Profiles** tab should now look like Figure 9.

Figure 9: Profiles Tab for Lab 3



Name: BLE_1

General | Profiles | GAP Settings | L2CAP Settings | Built-in

- Custom
  - Server
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Con
    - Generic Attribute
      - Service Changed
        - Client Characteristic C
    - CapSense Service
      - CapSense Slider Charact
        - Client Characteristic C
        - Characteristic User De
    - RGB LED Service
      - RGB LED Characteristic
        - Characteristic User De

Descriptor: **Characteristic User Description**

The Characteristic User Description descriptor provides a textual user description for a characteristic value.

UUID: 2901 16-bit

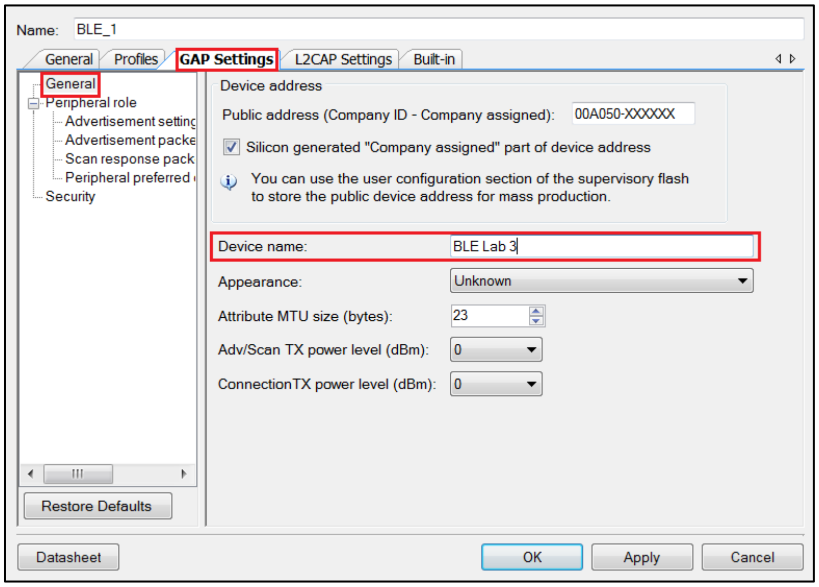
Name	Type	Length	Value
Fields			
User Description	utf8s	15	RGB LED Control
Permissions			

Datasheet | OK | Apply | Cancel

4. **GAP Settings Tab.** Refer to the BLE Component datasheet for more information on the configuration parameters.

   4.1. **General** - Set the **Device Address**, **Device name**, and **Appearance** as shown in Figure 10.

Figure 10: GAP Settings - General


Name: BLE_1

General | Profiles | GAP Settings | L2CAP Settings | Built-in

General
Peripheral role
Advertisement setting
Advertisement packe
Scan response pack
Peripheral preferred
Security

Device address

Public address (Company ID - Company assigned): 00A050-XXXXXX

✓ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name: BLE Lab 3

Appearance: Unknown

Attribute MTU size (bytes): 23

Adv/Scan TX power level (dBm): 0

ConnectionTX power level (dBm): 0

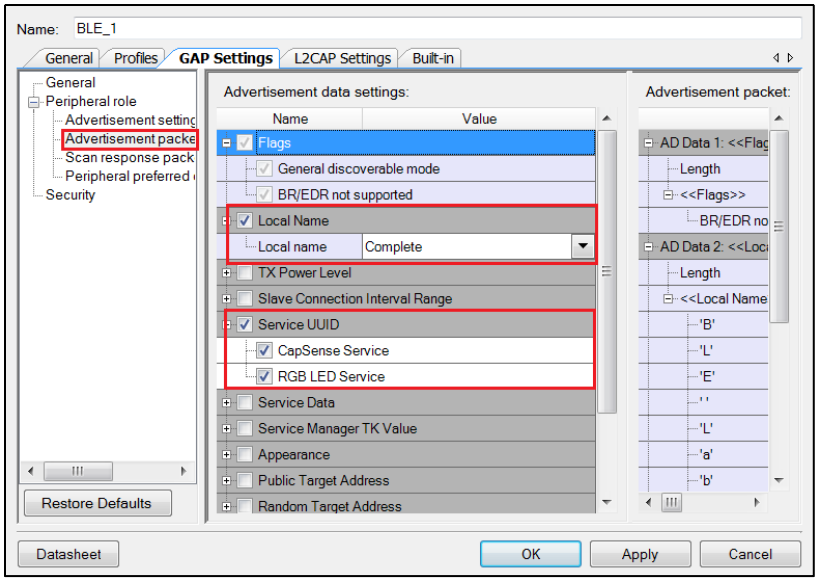
Restore Defaults

Datasheet | OK | Apply | Cancel

4.2. **Advertising Settings** - Leave these default settings.

4.3. **Advertisement Packet** – Enable the **Local Name** and configure other settings as per your choice. See Figure 11.

Figure 11: GAP Settings – Advertisement Packets


Name: BLE_1

General | Profiles | GAP Settings | L2CAP Settings | Built-in

General
Peripheral role
Advertisement setting
Advertisement packe
Scan response pack
Peripheral preferred
Security

Advertisement data settings:

Name	Value
✓ Flags	
✓ General discoverable mode	
✓ BR/EDR not supported	
✓ Local Name	
Local name	Complete
TX Power Level	
Slave Connection Interval Range	
✓ Service UUID	
✓ CapSense Service	
✓ RGB LED Service	
Service Data	
Service Manager TK Value	
Appearance	
Public Target Address	
Random Target Address	

Advertisement packet:

AD Data 1: <<Flag
Length
<<Flags>>
BR/EDR no
AD Data 2: <<Loca
Length
<<Local Name
'B'
'L'
'E'
' '
'L'
'a'
'b'

Restore Defaults

Datasheet | OK | Apply | Cancel

4.4. **Scan Response Packet** – Leave these default settings.

4.5. **Peripheral preferred connection parameters** - Leave these default settings.

4.6. **Security –** Configure as below:

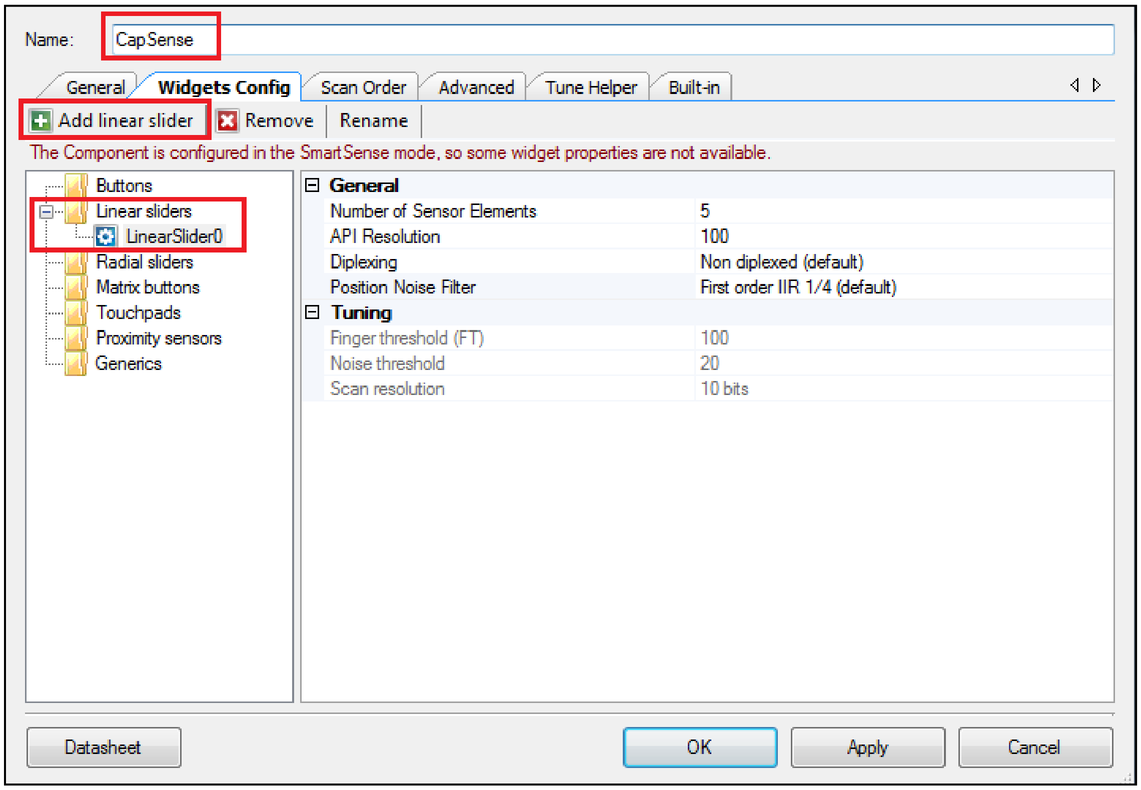
a. **Security mode**: Select **Mode 1** security

b. **Security level**: Select **No Security (No Authentication, No Encryption)**

c. **I/O Capabilities**: Set this to **No Input No Output**

   d. **Bonding requirement**: Set this to **No Bonding**
   e. **Encryption key size (bytes)**: Leave this parameter to the default value of **16**
5. Click **OK** to finish your BLE Component configuration.
6. Add the **CapSense CSD** Component to the schematic. This Component is used to sense your finger position on the slider on the BLE Pioneer Kit.
7. **Configure** the CapSense CSD Component's name to be **CapSense**. On the **General** tab, leave the default settings. The Component automatically tunes its parameters for the best performance, using SmartSense Auto-Tuning – an algorithm that sets, monitors and continuously maintains optimal capacitive sensor performance.
8. On the **Widgets Config** tab, add a linear slider by clicking **Linear sliders** and then the **Add linear slider button**, as shown in Figure 12.
9. Leave the slider's settings at default. Note that each element's sensitivity can be changed on the **Scan Order** tab. Values can be from 1 – 10 with lower values providing higher sensitivity. See the Component's datasheet for additional information on this and other CapSense settings.
10. Click **OK** to complete the configuration.

Figure 12: CapSense Component Configuration - Adding a Linear Slider

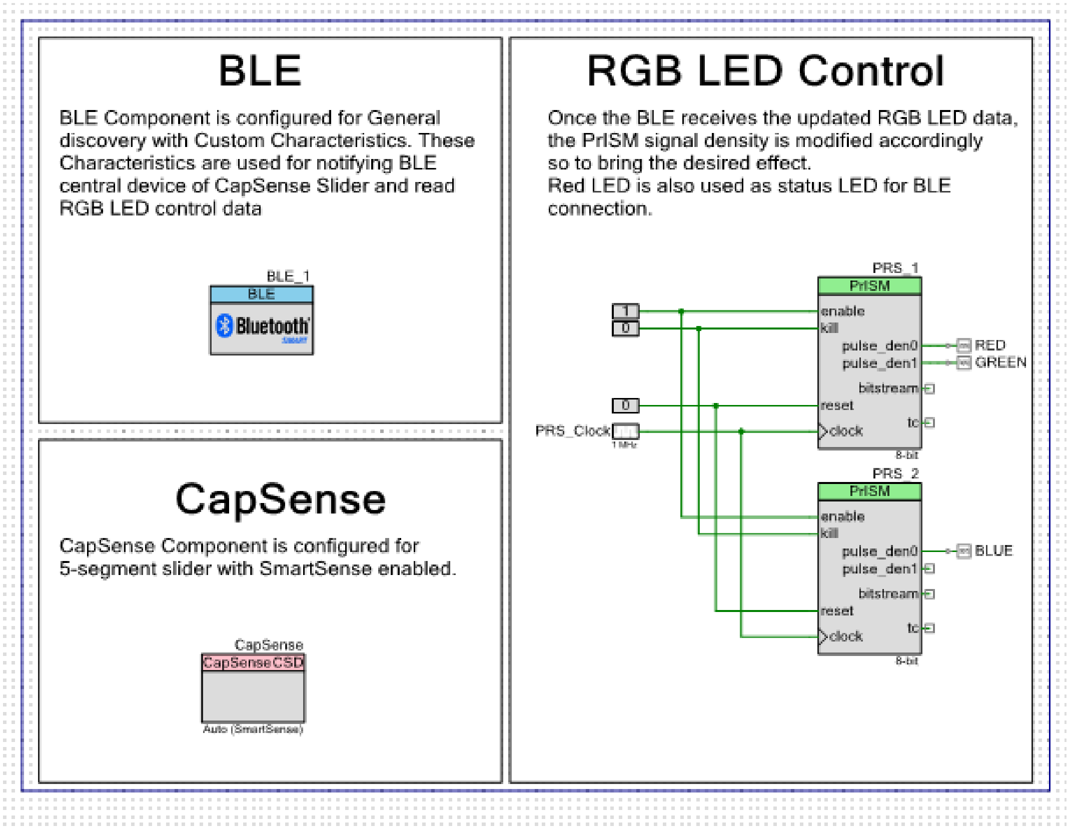


The Component is configured in the SmartSense mode, so some widget properties are not available.

General	
Number of Sensor Elements	5
API Resolution	100
Diplexing	Non diplexed (default)
Position Noise Filter	First order IIR 1/4 (default)
Tuning	
Finger threshold (FT)	100
Noise threshold	20
Scan resolution	10 bits

11. The other Components have already been placed on the schematic. Double-click one of the **PrISM** Components and examine the settings. Click the **Datasheet** button and review the datasheet for some additional information on this Component and how it works.
12. Your schematic should now look as Figure 13 shows.

Figure 13: Schematic for Lab 3



BLE

BLE Component is configured for General discovery with Custom Characteristics. These Characteristics are used for notifying BLE central device of CapSense Slider and read RGB LED control data

CapSense

CapSense Component is configured for 5-segment slider with SmartSense enabled.

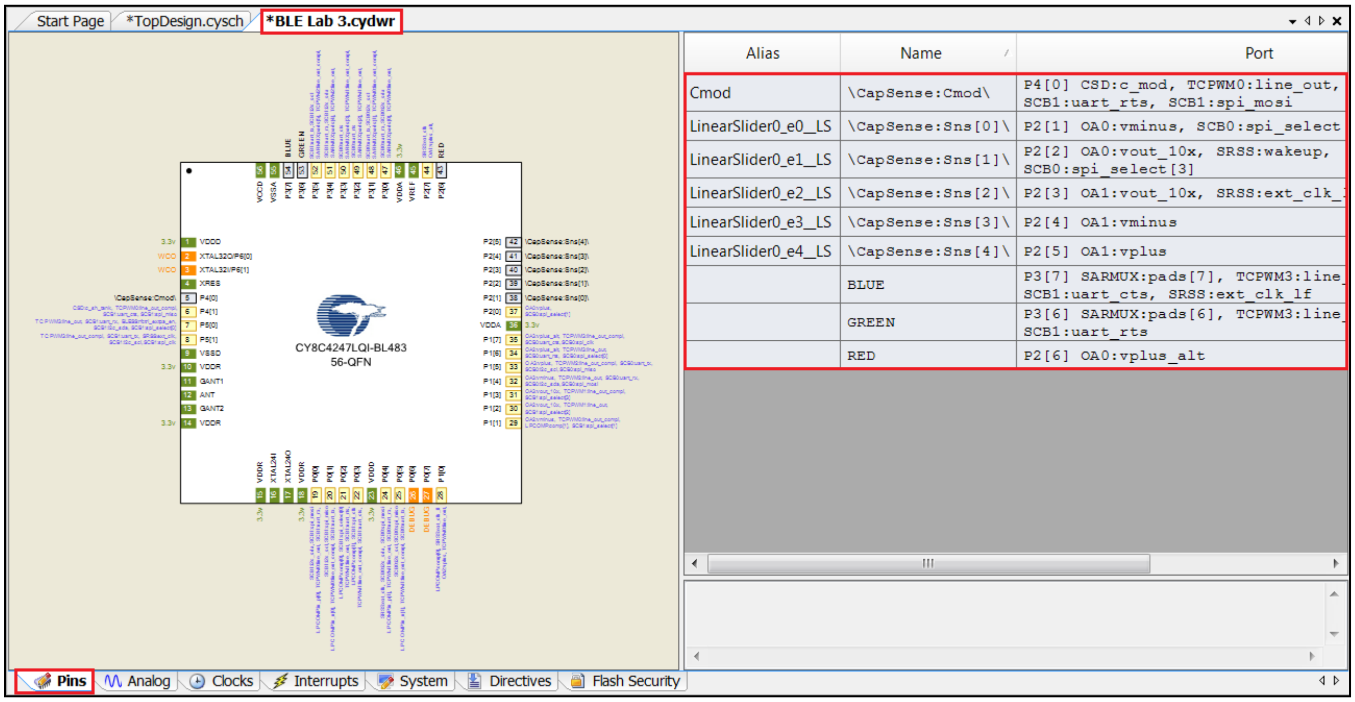
RGB LED Control

Once the BLE receives the updated RGB LED data, the PrISM signal density is modified accordingly so to bring the desired effect.
Red LED is also used as status LED for BLE connection.

Configure DWR

Open the DWR file. On the **Pins** tab, assign the segments of the linear slider to pins **P2[1], P2[2], P2[3], P2[4], and P2[5]**, in increasing order. The **CMOD** capacitor is on **P4[0]**. The three LEDs are already set to the correct device pins.

Figure 14: Design Wide Resources – Pins Assignment



Alias	Name	Port
Cmod	\CapSense:Cmod\	P4[0] CSD:c_mod, TCPWM0:line_out, SCB1:uart_rts, SCB1:spi_mosi
LinearSlider0_e0__LS	\CapSense:Sns[0]\	P2[1] OA0:vminus, SCB0:spi_select
LinearSlider0_e1__LS	\CapSense:Sns[1]\	P2[2] OA0:vout_10x, SRSS:wakeup, SCB0:spi_select[3]
LinearSlider0_e2__LS	\CapSense:Sns[2]\	P2[3] OA1:vout_10x, SRSS:ext_clk_
LinearSlider0_e3__LS	\CapSense:Sns[3]\	P2[4] OA1:vminus
LinearSlider0_e4__LS	\CapSense:Sns[4]\	P2[5] OA1:vplus
	BLUE	P3[7] SARMUX:pads[7], TCPWM3:line_ SCB1:uart_cts, SRSS:ext_clk_lf
	GREEN	P3[6] SARMUX:pads[6], TCPWM3:line_ SCB1:uart_rts
	RED	P2[6] OA0:vplus_alt

Build your project to generate the Component source files. This helps you while writing firmware because PSoC Creator can auto-complete the API names, variables, and macros for you. The build will produce some errors because the firmware is not yet completed. Don't worry about these errors yet.

Firmware

The firmware consists of these high-level blocks:

1. **CapSense slider**: The CapSense slider on the kit is scanned periodically. If the finger position is different from the previous scan, a notification packet is sent over BLE. The scan happens only when the notifications for this CapSense Slider Characteristic are enabled by the GATT Client. The functions used for the CapSense slider are in main.c.
2. **RGB LED**: The PrISM blocks used to drive the RGB LED are configured based on the Attribute values written by the GATT Client. The value can be in a range of 0 and 255 for each LED individually. This value is converted into a percentage of intensity for that LED and the corresponding PrISM block is configured. The overall LED brightness is a separate input (one of the four bytes in that characteristic) which controls the final intensity of all LEDs. The functions used for controlling the LEDs in BLEApplications.c

3. **BLE**: The events generated by the BLE Stack are handled to keep track of advertisement, connection, and disconnection states. The Attribute Write request event is handled by first identifying the Attribute which was written to by the GATT Client. These functions are contained in BLEApplications.c
   If the Attribute is the CapSense Slider Characteristic's Client Characteristic Configuration Descriptor (CCCD), then the slider notifications are correspondingly enabled (CCCD = 1) or disabled (CCCD = 0). On the other hand, if the Attribute is the RGB LED Characteristic, then the LED is controlled accordingly.
4. **Application Layer**: A top-level application layer is written for the firmware. The top-level application is contained in main.c.

The firmware flow is shown in Figure 15. Table 4 lists the files present in the firmware. This table describes the different functions defined in these files and their usage.

Note that for standard applications (e.g. Lab 1 and Lab 2) we had two BLE callback functions: one for general events, and one specific to the standard service. In the case of a custom service, we just have a single callback function which handles both general events and event related to our custom service(s). This single callback function is registered when the BLE Component is started.

Figure 15: Firmware Flow

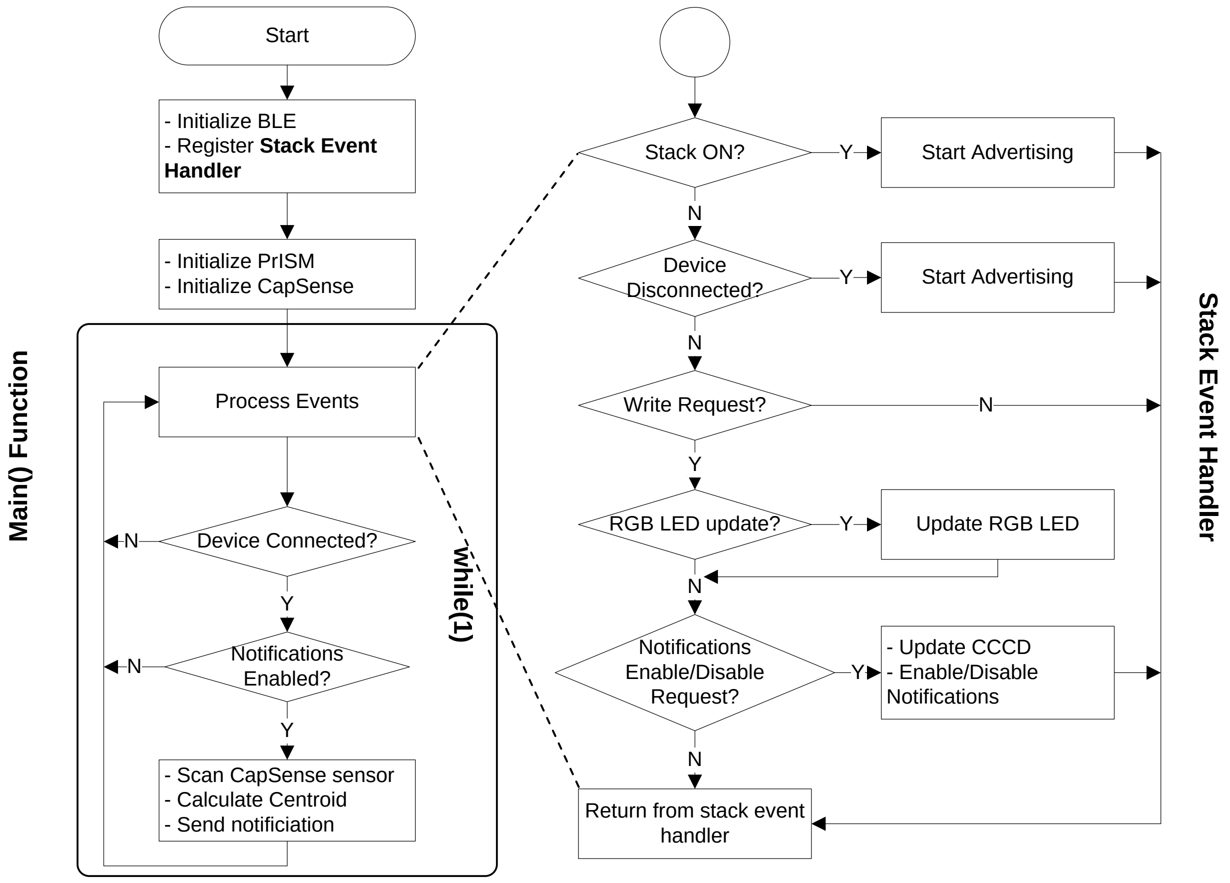


Table 4: Main Files Present in the Lab 3 Project

File name	Details
main.c	This is the top level application file. It initializes the system and runs the main loop. It also handles the CapSense slider functionality. This file has three functions: ***main()*** – The main function for the application ***InitializeSystem()*** – Initializes all the blocks of the system ***HandleCapSenseSlider()*** – Scans the CapSense slider and finds the finger position on the slider. When the finger position changes relative to the previous scan, it sends the new position as a notification over BLE. The notifications are sent by calling the SendCapSenseNotification() function.
BLEApplications.c	This file handles the BLE specific functionality of the project. It handles the BLE events and notifications as well as LED control. The file has these functions: ***CustomEventHandler()***: Handles the events for BLE advertisement, connection, and disconnection. Also services the write requests to the Custom Characteristics and Descriptors. This function is a callback from the BLE Stack for all events. ***SendCapSenseNotification():*** Creates a CapSense Slider Characteristic notification packet and sends it. This function is called by HandleCapSenseSlider() function in main.c. ***UpdateRGBled():*** Configures the PrISM blocks to drive the RGB LED as per the latest data. Also updates the RGB LED Characteristic in the database with the latest data for a future read by the GATT Client.

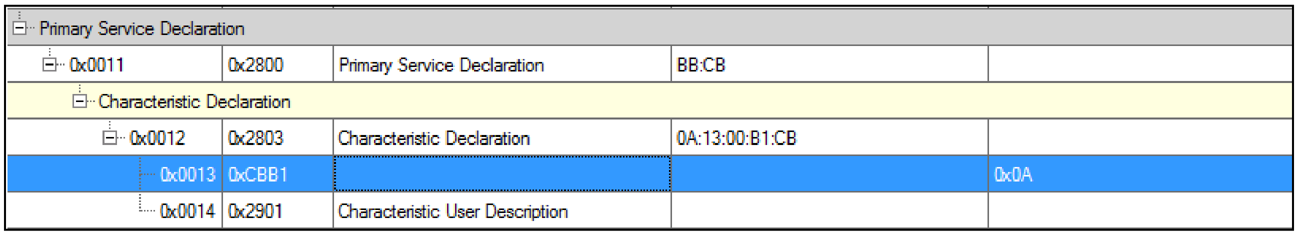
Build and Program

1. The firmware for this lab has been implemented as a part of the template project.
2. **Build** your project to generate the hex file, and **Program** it to your kit.

Testing with CySmart Central Emulation Tool

1. Open **CySmart 1.0** and **Connect** it to the **BLE-USB Bridge**.
2. **Start Scan** and **Connect** to your GATT Server device.
3. **Discover all Attributes** and then scroll down the Attribute list to the **RGB LED Characteristic** (it has the **UUID 0xCBB1**). See Figure 16.

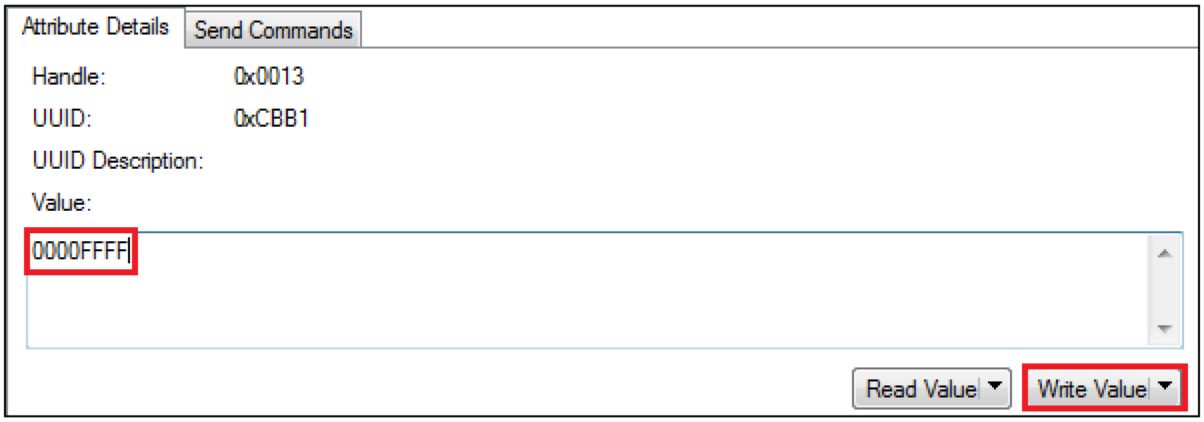
Figure 16. CySmart - RGB LED Characteristic



Primary Service Declaration				
0x0011	0x2800	Primary Service Declaration	BB:CB	
Characteristic Declaration				
0x0012	0x2803	Characteristic Declaration	0A:13:00:B1:CB	
0x0013	0xCBB1			0x0A
0x0014	0x2901	Characteristic User Description		

4. **Write** a 4 byte value to this Characteristic on the right and notice the corresponding color and intensity of the RGB LED on the kit. Byte 0 corresponds to the Red color, Byte 1 corresponds to the Green color, Byte 2 corresponds to the Blue color, and Byte 3 corresponds to the intensity. For example, writing 00:00:FF:FF to this Characteristic turns on the Blue LED with full intensity. See Figure 17.

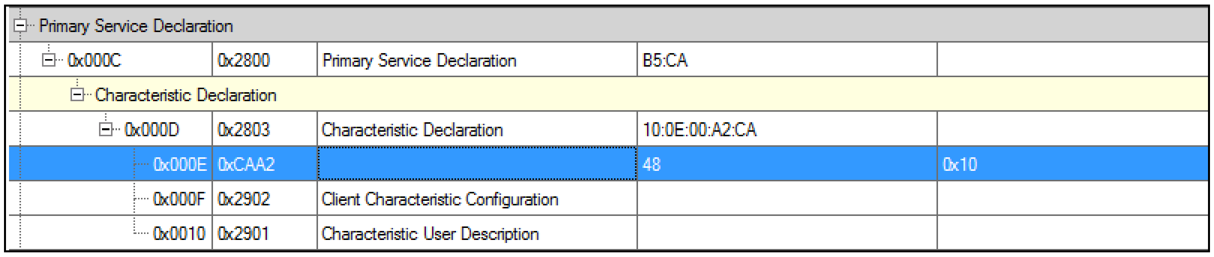
Figure 17: CySmart - Write Attribute



Handle: 0x0013
UUID: 0xCBB1
UUID Description:
Value:
0000FFFF

5. Now locate the **CapSense Slider Characteristic (UUID = 0xCAA2)** and enable notifications for it, either by clicking **Enable All Notifications** or by writing **1** to its CCCD descriptor (UUID=0x2902).
6. Move your finger over the slider on the kit and observe that the value of the Characteristic changes in the CySmart tool, while the tool's log shows notification packets being received. See Figure 18.

Figure 18: CySmart - Slider Notifications



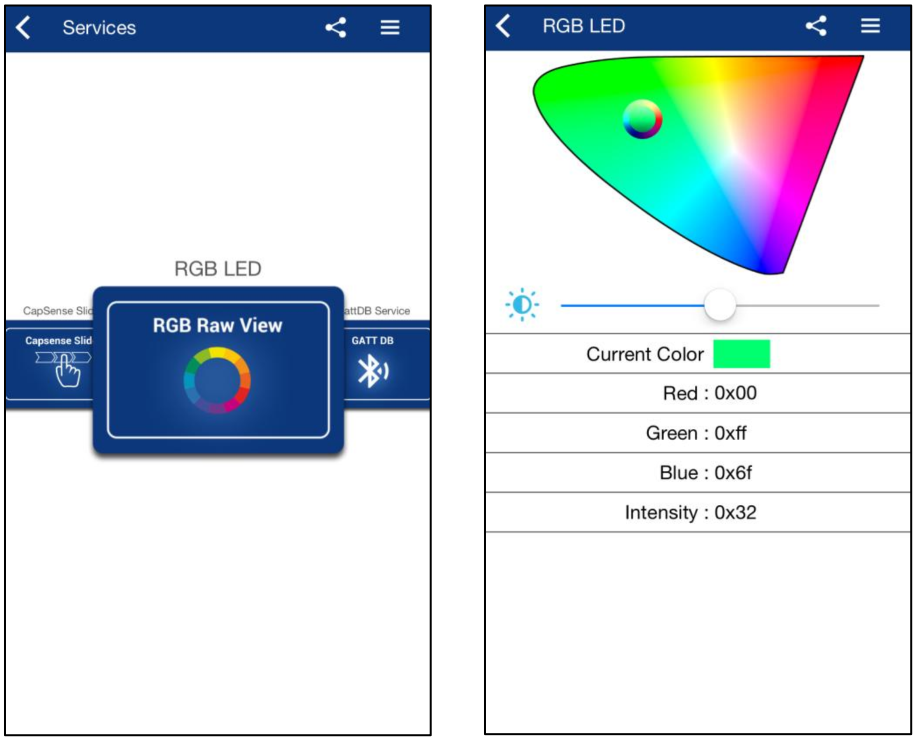
Primary Service Declaration				
0x000C	0x2800	Primary Service Declaration	B5:CA	
Characteristic Declaration				
0x000D	0x2803	Characteristic Declaration	10:0E:00:A2:CA	
0x000E	0xCAA2		48	0x10
0x000F	0x2902	Client Characteristic Configuration		
0x0010	0x2901	Characteristic User Description		

7. Disable notifications either by clicking **Disable All Notifications** or by writing **0** to its CCCD descriptor. Move your finger on the slider again. Notice that the slider position is no longer reported because notifications have been disabled.

Testing with CySmart Mobile App

1. Open the **CySmart Mobile App** on your phone. If you do not have Bluetooth switched on already, the app will ask you to do it.
2. Connect to your GATT Server device on the app. Once connected, the app shows you all the Services exposed by the GATT Server. It automatically detects the Custom Services for RGB LED and CapSense Slider and lists them respectively.
3. Select the RGB LED Service. You will see that a color gamut is available. Tap anywhere on the gamut to see the corresponding color on the RGB LED on the kit. See Figure 19.

Figure 19: CySmart iOS Mobile App – RGB LED Control

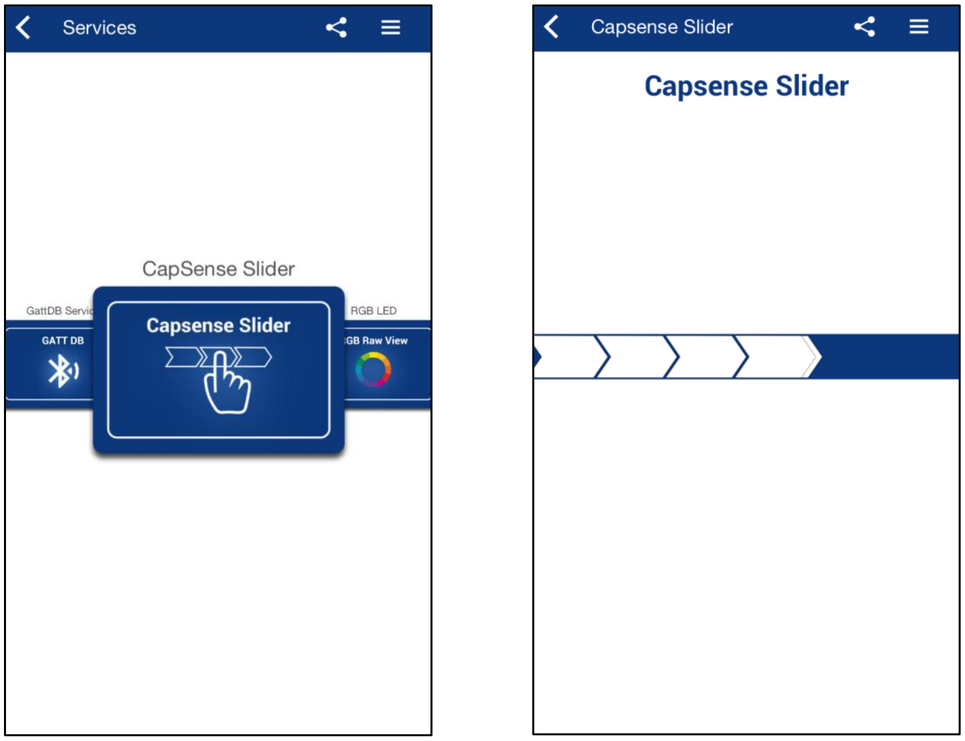


Current Color 🟩
Red : 0x00
Green : 0xff
Blue : 0x6f
Intensity : 0x32

4. Move the slider position on the app page to change the brightness level of the LED.
5. Now go back one page in the app and select the CapSense Slider Service.

6. Once on the CapSense Slider page, move your finger on the slider on the kit. You will see a corresponding slider update on the app page. See Figure 20.

Figure 20: CySmart iOS Mobile App – CapSense Slider



Congratulations! You have completed lab 3!

Additional Exercises

1. Replace the CapSense Slider with a CapSense Proximity Sensor.
   Hints:
   - Use CAA1 as the UUID for CapSense Proximity Service.
   - Remove the slider widget from the CapSense Component and add a Proximity widget
   - Assign the proximity sensor to the pin assigned for proximity header on the CY8CKIT-042-BLE
   - Enable the Proximity Sensor using the API function CapSense_EnableWidget(CapSense_PROXIMITYSENSOR0__PROX) before starting the CapSense Component.
   - Use the API function CapSense_GetDiffCountData(CapSense_PROXIMITYSENSOR0__PROX) instead of CapSense_GetCentroidPos(CapSense_LINEARSLIDER0__LS) to extract the proximity value
   - Modify the code to always send the CapSense Proximity sensor Notification data

2. Implement low-power operation for lab 3.
   Hints:
   - Follow the firmware implementation from PSoC 4 BLE Lab 2.
   - Add a switch to wake-up the device from Hibernate mode.
   - The PrISM Component is not active during the Deep-Sleep mode, so change the code for putting the device in the Deep-Sleep mode to use Sleep mode instead.

   Note: Refer to PSoC_4_CapSense_Slider_LED example project (installed with CY8CKIT-042-BLE) for Deep-Sleep operation with the PrISM Component.

Document Revision History (001-96274)

Revision	By	Description
**	PMAD	Initial Release
*A	GUL	Edits for BLE terminology

Document Revision History (001-98279)

Revision	By	Description
**	PMAD	Labs moved to new spec number Update to PSoC Creator 3.2 Added details for additional exercises