Getting started with PSoC™ 4 CY8C4xxx-BL MCU with AIROC™ Bluetooth® LE

About this document

Scope and purpose

This application note introduces you to PSoC™ 4 with AIROC™ Bluetooth® LE (PSoC™ 4 CY8C4xxx-BL MCU with AIROC™ Bluetooth® LE), an Arm® Cortex®-M0 based Programmable System-on-Chip (PSoC™) that integrates a Bluetooth® Low Energy radio system. This application note helps you explore the PSoC™ 4 with AIROC™ Bluetooth® LE architecture and development tools and shows how easily you can create a Bluetooth® LE design using PSoC™ Creator, the development tool for PSoC™ 4 with AIROC™ Bluetooth® LE. It also guides you to more resources to accelerate your design development and in-depth learning about PSoC™ 4 with AIROC™ Bluetooth® LE.

Intended audience

This is intended for new users of PSoC™ 4 with AIROC™ Bluetooth® LE.

Associated part family

CY8C41x7-BL, CY8C42x7-BL, CY8C41x8-BL, CY8C42x8-BL

Software version

PSoC™ Creator 4.4

More code examples? We heard you.

To access an ever-growing list of hundreds of PSoC™ code examples, please visit our **code examples web page**. You can also explore the video training library **here**.

Table of contents

Table of contents **1**
1 Introduction **4**
1.1 Prerequisites 4
1.1.1 Hardware 4
1.1.2 Software 4
2 PSoC™ 4 with AIROC™ Bluetooth® LE resources **5**
2.1 PSoC™ Creator 5
2.2 PSoC™ Creator help 6
2.3 Code examples 6
3 PSoC™ 4 with AIROC™ Bluetooth® LE features **9**
4 PSoC™ is more than a Bluetooth® LE MCU **11**
5 Bluetooth® LE overview **13**
5.1 Bluetooth® LE link establishment 14
5.2 GATT data format 14
5.3 Bluetooth® LE profile 17

Table of contents

5.4 Bluetooth® Low Energy Component ... 19
5.4.1 Features ... 19
5.4.2 Configuration ... 19
6 PSoC™ 4 with AIROC™ Bluetooth® LE development setup ... 22
7 My first PSoC™ 4 with AIROC™ Bluetooth® LE design ... 24
7.1 About the design ... 24
7.2 Prerequisites ... 25
7.3 Stage 1: Create the design ... 25
7.4 Stage 2: Write the firmware ... 36
7.4.1 System initialization ... 36
7.4.2 Bluetooth® LE stack event handler ... 38
7.4.3 Bluetooth® LE service-specific event handler ... 41
7.4.4 Main loop and low-power implementation ... 42
7.5 Stage 3: Program the device ... 45
7.6 Stage 4: Test your design ... 47
7.6.1 With CySmart mobile app ... 47
7.6.2 With CySmart host emulation tool ... 48
8 Summary ... 52
9 Appendix A: Bluetooth® LE device family comparison ... 53
10 Appendix B: Terms of art ... 56
11 Appendix C: Bluetooth® LE development tools ... 57
11.1 PSoC™ 4 with AIROC™ Bluetooth® LE pioneer kit ... 57
11.2 CySmart host emulation tool ... 57
11.3 CySmart Mobile app ... 59
12 Appendix D: PSoC™ 4 with AIROC™ Bluetooth® LE device ... 61
12.1 Bluetooth® Low Energy Subsystem (BLESS) ... 61
12.1.1 Deep-sleep mode ... 61
12.1.2 Sleep mode ... 61
12.1.3 Idle mode ... 61
12.1.4 Transmit mode ... 61
12.1.5 Receive mode ... 62
12.2 Arm® Cortex®-M0, memory, and DMA ... 62
12.3 Programmable digital peripherals ... 62
12.3.1 Programmable SCBs ... 62
12.3.2 Programmable TCPWMs ... 63
12.3.3 Universal digital blocks ... 63
12.3.4 Applications ... 64
12.4 Programmable analog ... 64
12.4.1 Continuous Time Block mini (CTBm) ... 64
12.4.2 SAR ADC with hardware sequencer ... 64
12.4.3 Low-power comparators ... 64
12.4.4 Capacitive touch sensing (CapSense™) ... 65
12.4.5 Segment LCD direct driver ... 65
12.4.6 Applications ... 65
12.5 System-wide resources ... 66
12.5.1 Low-leakage power modes ... 66
12.5.2 Power supply and monitoring ... 67
12.5.3 Clocking system ... 67
12.5.4 Device security ... 68

Table of contents

12.6 Programmable GPIOs ... 68
13 Appendix E: Bluetooth® LE protocol ... 69
13.1 Overview ... 69
13.2 Physical Layer (PHY) ... 69
13.3 Link Layer (LL) ... 70
13.4 Host Control Interface (HCI) ... 71
13.5 Logical Link Control and Adaptation Protocol (L2CAP) ... 71
13.6 Security Manager (SM) ... 71
13.7 Attribute Protocol (ATT) ... 72
13.7.1 Attribute hierarchy ... 72
13.7.2 Attribute operations ... 75
13.8 Generic Attribute Profile (GATT) ... 75
13.9 Generic Access Profile (GAP) ... 76
Related application notes ... 79
Revision history ... 80

1 Introduction

The PSoC™ 4 with AIROC™ Bluetooth® LE device is a programmable embedded system-on-chip that integrates a Bluetooth® Low Energy radio, programmable analog and digital peripherals, memory, and an Arm® Cortex®-M0 microcontroller on a single chip. Bluetooth® LE is an ultra-low-power wireless standard defined by the Bluetooth Special Interest Group (SIG) for short-range communication. It features a physical layer, protocol stack, and application use cases, all designed and optimized for low power consumption.

PSoC™ 4 with AIROC™ Bluetooth® LE provides a cost-effective and small footprint alternative to the combination of an MCU and a Bluetooth® LE radio. The programmable analog and digital subsystems allow flexibility and dynamic fine-tuning of the design using PSoC™ Creator, the schematic-based design tool for developing PSoC™ 4 with AIROC™ Bluetooth® LE applications. To develop a Bluetooth® LE application, you do not need a working knowledge of Bluetooth® LE complex protocol stack. Infineon provides an easy-to-configure, free-of-cost graphical user interface (GUI) based Bluetooth® Low Energy Component in PSoC™ Creator that abstracts the protocol complexity. Bluetooth® Low Energy Component can be configured in minutes using a GUI, enabling you to jump-start your Bluetooth® LE design. PSoC™ 4 with AIROC™ Bluetooth® LE offers a current consumption of 150 nA while retaining the SRAM contents, programmable logic, and the ability to wake up from an interrupt. It consumes 1.3 µA in Deep-Sleep mode while maintaining an active Bluetooth® LE link. A combination of these low-power modes provides a best-in-class system power consumption for battery-operated Bluetooth® LE designs such as wearable fitness monitors and wireless sensor interfaces.

PSoC™ 4 with AIROC™ Bluetooth® LE simplifies the RF board design with its integrated balun circuit, which reduces the number of external components required for antenna matching.

Infineon capacitive touch-sensing feature in PSoC™ 4 with AIROC™ Bluetooth® LE, known as CapSense™, offers unprecedented signal-to-noise ratio, best-in-class waterproofing, and a wide variety of sensor types such as buttons, sliders, track pads, and proximity sensors. CapSense™ user interfaces are gaining popularity in wearable electronic devices such as activity monitors and health and fitness equipment.

This application note presents the basics of Bluetooth® LE that includes capabilities of PSoC™ 4 with AIROC™ Bluetooth® LE device and overview of development tools. For advanced application development, refer to **Designing Bluetooth® Low Energy applications** and **Creating a Bluetooth® Low Energy custom profile** application note.

1.1 Prerequisites

Before you get started, make sure you have the development kit and installed the required software.

1.1.1 Hardware

- **PSoC™ 4 Bluetooth® Low Energy Compliant Pioneer Kit**
- PC/Laptop with Windows 7 or later (If using the **CySmart Host Emulation Tool** PC application)
- Mobile phone with Android 5 or later or iOS 8 or later (If using the CySmart **iOS**/**Android** app)

1.1.2 Software

- **PSoC™ Creator 4.4** with **PSoC™ Programmer 3.29**
- **CySmart Host Emulation Tool** or CySmart **iOS**/**Android** app

2 PSoC™ 4 with AIROC™ Bluetooth® LE resources

Infineon provides a wealth of data at **www.cypress.com** to help you select the right PSoC™ device and quickly and effectively integrate it into your design. If you are new to PSoC™, it is recommended that you read **Appendix B: Terms of art** for a list of commonly used terms. For a comprehensive list of resources, see **KBA86521, How to Design with PSoC™ 3, PSoC™ 4, and PSoC™ 5LP**.

The following is an abbreviated list for PSoC™ 4 with AIROC™ Bluetooth® LE:

- **Overview: PSoC™ Portfolio, PSoC™ RF 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™ 4XX7-BL** and **PSoC™ 4XX8-BL** device families.
- **Application Notes and Code Examples** cover a broad range of topics, from basic to advanced level. Many of the application notes include code examples. PSoC™ Creator provides additional code examples—see **Code examples**.
- **Technical Reference Manuals (TRMs)** provide detailed descriptions of the architecture and registers in each PSoC™ 4 with AIROC™ Bluetooth® LE device family.
- **CapSense™ design guide:** Learn how to design capacitive touch-sensing applications with the PSoC™ 4 with AIROC™ Bluetooth® LE family of devices.
- **Development tools**
  - **CY8CKIT-042-BLE Bluetooth® Low Energy Pioneer Kit** is an easy-to-use and inexpensive development platform for Bluetooth® LE. This kit includes connectors for Arduino™ compatible shields and Digilent® Pmod™ daughter cards.
  - CySmart Bluetooth® LE Host Emulation Tool for **Windows**, **iOS**, and **Android** is an easy-to-use app that enables you to test and debug your Bluetooth® LE Peripheral applications.

See **Appendix C: Bluetooth® LE development tools** for an overview.

- **Training videos:** A video is worth a million words—here's a suggested list of PSoC™ 4 with AIROC™ Bluetooth® LE videos to get you started with your first Bluetooth® LE design:
  - **Introduction to PSoC™ 4-BLE**
  - **Getting to know PSoC™ Creator**
  - **PSoC™ 4 Bluetooth® Low Energy Compliant Pioneer Kit**
  - **Introduction to Bluetooth® Low Energy**
  - **Configuring your first Bluetooth® Low Energy design**
  - **Writing firmware for your first Bluetooth® Low Energy design**
- Visit our **Technical support** page questions or issues related to designs, compatibility, functionality, timing, datasheets, and general applications support.

2.1 PSoC™ Creator

PSoC™ Creator is a free Windows-based Integrated Design Environment (IDE). It enables you to design hardware and firmware systems concurrently, based on PSoC™ 4 with AIROC™ Bluetooth® LE and PRoC-BLE. As **Figure 1** shows, with PSoC™ Creator, you can:

1. Drag and drop Components to build your hardware system design in the main design workspace.
2. Co-design your application firmware with the PSoC™ hardware.

3. Configure the Components using configuration tools.
4. Explore the library of more than 100 Components.
5. Review the Component datasheets.

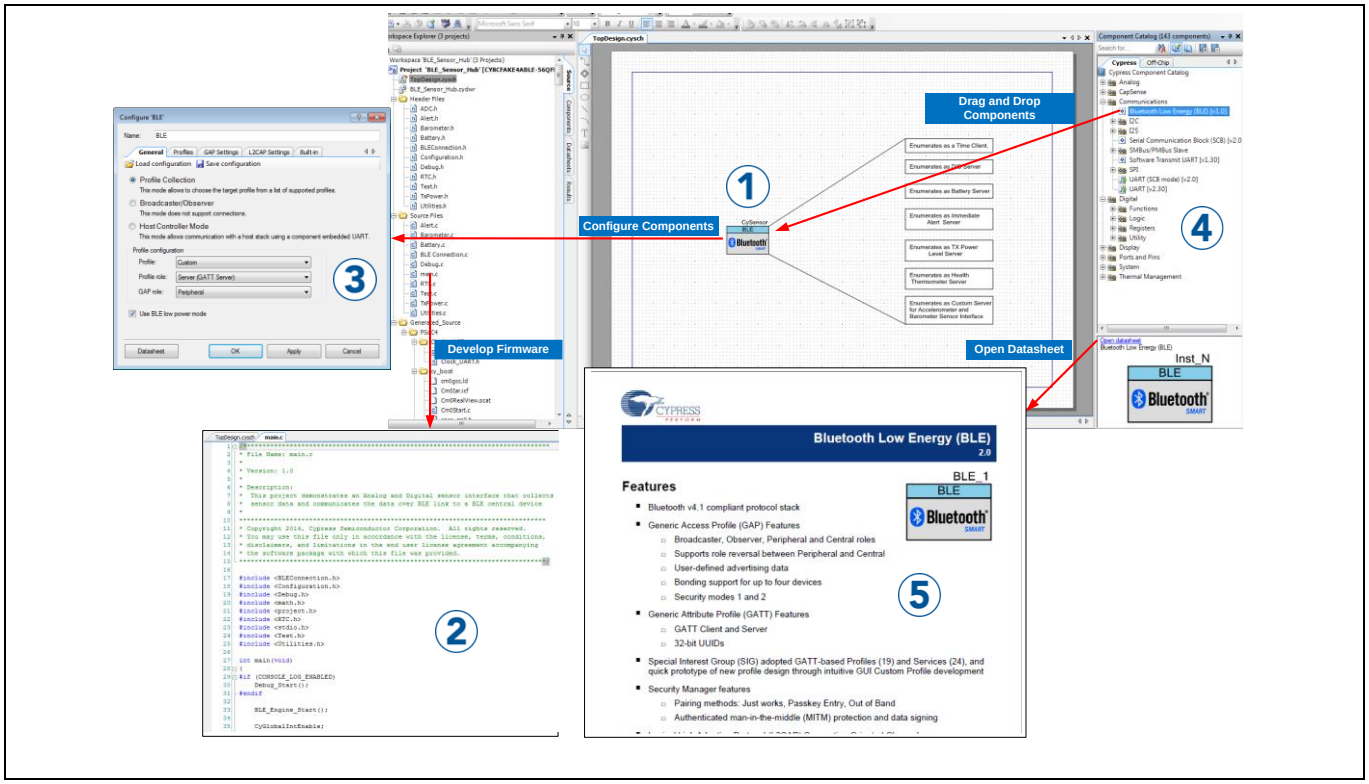


Figure 1 PSoC™ Creator schematic entry and Components

2.2 PSoC™ Creator help

Visit the **PSoC™ Creator** home page to download and install the latest version of PSoC™ Creator. Then launch PSoC™ Creator and navigate to the following items:

- **Quick start guide**: Choose **Help** > **Documentation** > **Quick Start Guide**. This guide gives you the basics for developing PSoC™ Creator projects.
- **Simple Component code examples**: Choose **File** > **Code Example**. These code examples demonstrate how to configure and use PSoC™ Creator Components.
- **System reference guide**: Choose **Help** > **System Reference Guides**. This guide lists and describes the system functions provided by PSoC™ Creator.
- **Component datasheets**: Right-click a Component and select “Open Datasheet.” Visit the **PSoC™ 4-BLE Component Datasheets** page for a list of all PSoC™ 4 with AIROC™ Bluetooth® LE Component datasheets.

2.3 Code examples

PSoC™ Creator includes a large number of code examples. These projects are available from the PSoC™ Creator Start Page, as **Figure 2** shows.

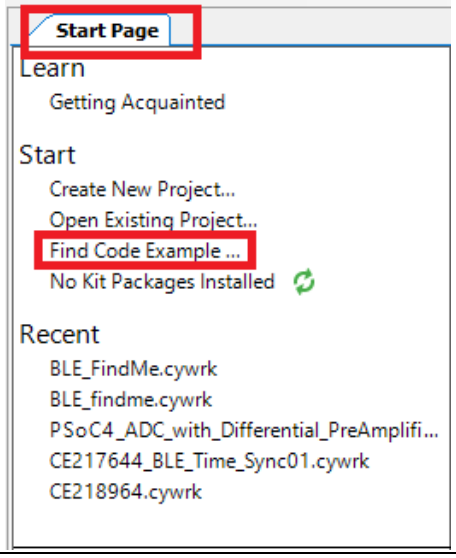


Figure 2 Code examples in PSoC™ Creator

Code examples can speed up your design process by starting you off with a complete design, instead of a blank page. The code examples also show how PSoC™ Creator Components can be used for various applications.

In the **Find Code Example** dialog shown in **Figure 3**, you have several options:

- Filter for examples based on architecture or device family, that is, PSoC™ 4, PSoC™ 4-BLE, PRoC-BLE, and so on; category; or keyword.
- Select from the menu of examples offered based on the **Filter Options**. There are more than 30 Bluetooth® LE code examples for your reference, as shown in **Figure 3**.
- View the documentation for the selection (in the **Documentation** tab)
- Review the code example for the selection. You can copy and paste code from this window to your project, which can help speed up code development.
- Or create a new project (and a new workspace if needed) based on the selection. This can speed up your design process by starting you off with a complete basic design. You can then adapt that design to your application.
- Apart from PSoC™ Creator code examples, you can also find more Bluetooth® LE reference examples on **this** GitHub repository.

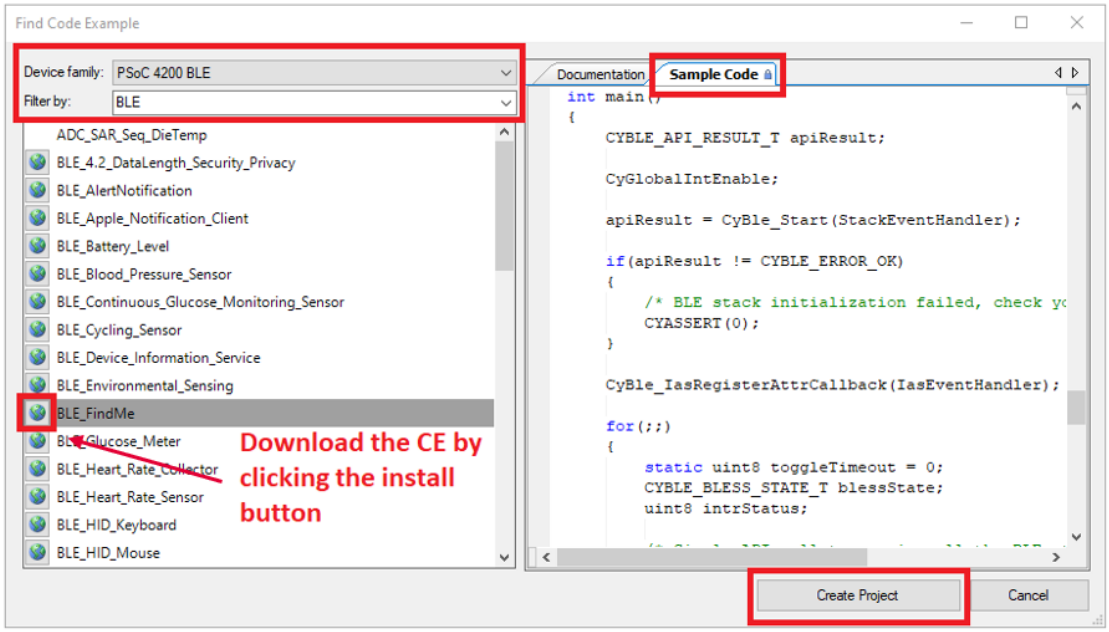


Figure 3 Code examples with Sample Code

3 PSoC™ 4 with AIROC™ Bluetooth® LE features

As shown in **Figure 4**, PSoC™ 4 with AIROC™ Bluetooth® LE device features:

- **32-bit MCU subsystem**
  - 48-MHz Arm® Cortex®-M0 CPU
  - Up to 256 KB flash and 32 KB SRAM
- **CapSense™ with SmartSense™ Auto-tuning**
  - One Capacitive Sigma-Delta™ (CSD) controller with touchpad capability
- **Programmable analog front end (AFE)**
  - Four opamps, configurable as PGAs, comparators, filters, etc.
  - One 12-bit, 1-Msps SAR2 ADC
- **Programmable digital logic**
  - Four Universal Digital Blocks (UDBs): custom digital peripherals
  - Four configurable TCPWM blocks: 16-bit timer, counter or PWM
  - Two configurable serial communication blocks (SCBs): I2C master or slave, SPI master or slave, or UART
- **Bluetooth Smart connectivity with Bluetooth 4.2**
  - 2.4-GHz Bluetooth® LE radio with integrated Balun
  - Bluetooth® 4.2 specification-compliant controller and host implementation

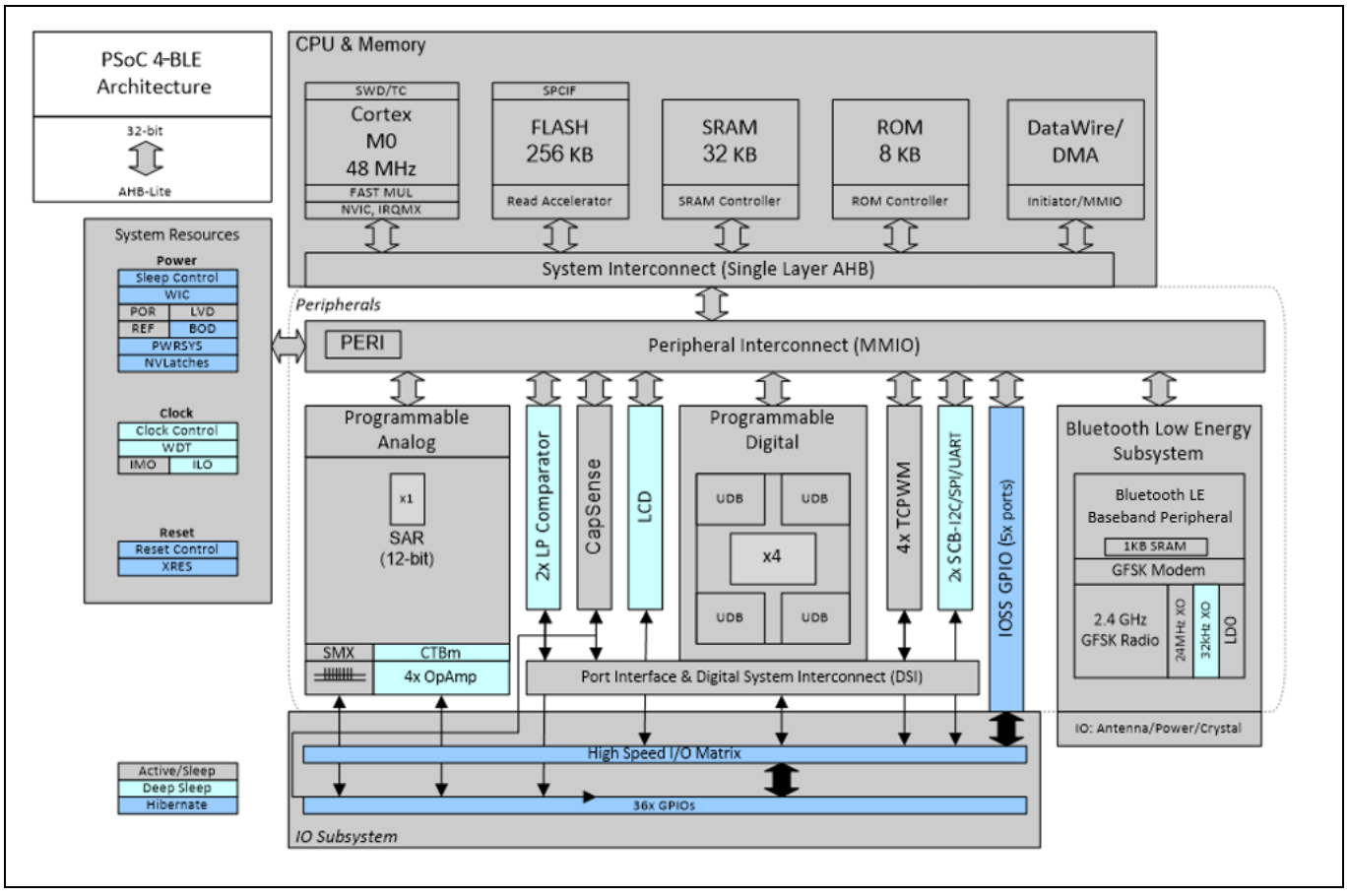


Figure 4 PSoC™ 4 with AIROC™ Bluetooth® LE architecture (CY8C4248-BL)

PSoC™ 4 with AIROC™ Bluetooth® LE features

See **Appendix D: PSoC™ 4 with AIROC™ Bluetooth® LE device** for a brief description of the features. For in-depth information, see the PSoC™ 4 with AIROC™ Bluetooth® LE family **datasheet**, **TRM**, and **application notes**. **Figure 4** shows the features available in the CY8C4248-BL family of devices. Subsets of these features are available in other device families; see **Table 2**.

4 PSoC™ is more than a Bluetooth® LE MCU

Figure 5 shows that a typical MCU contains a CPU (such as 8051 or an Arm® Cortex®) with a set of peripheral functions such as ADCs, DACs, UARTs, SPIs, Bluetooth® LE, and general I/O, all linked to the CPU's register interface. Within the MCU, the CPU is the "heart" of the device – the CPU manages everything from setup to data movement to timing. Without the CPU, the MCU cannot function.

Figure 6 shows that PSoC™ is quite different. With PSoC™, the CPU, analog, digital, and I/O are equally important resources in a programmable system. *It is the system's interconnect and programmability that is the heart of PSoC™ – not the CPU.* The peripheral analog and digital are interconnected with a highly configurable matrix of signal and data bus meshing that allows you to create custom designs that meet your application requirements. *You can program PSoC™ to emulate an MCU, but you cannot program an MCU to emulate PSoC™.*

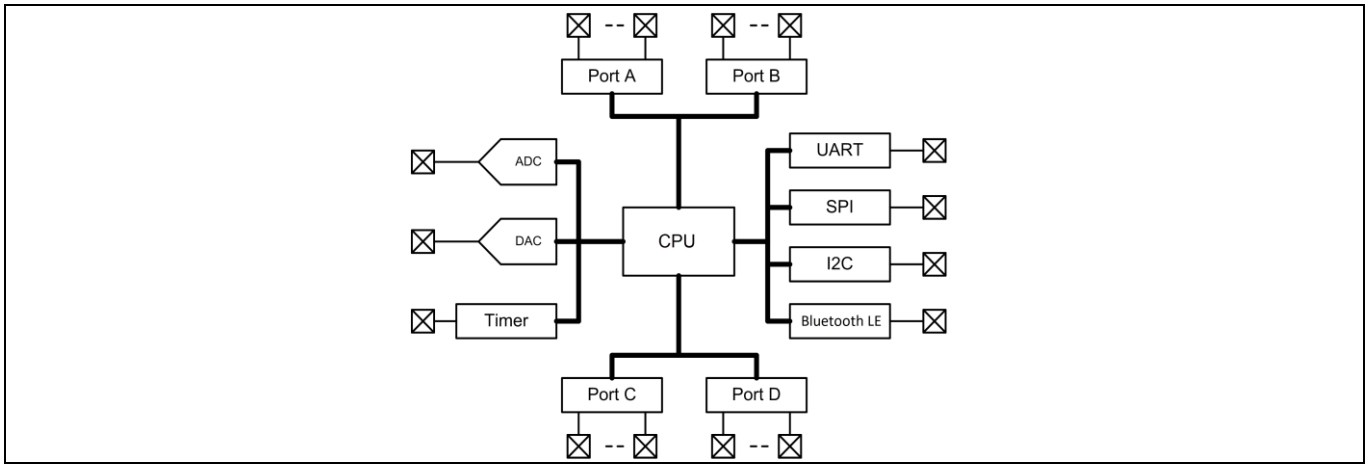


Figure 5 Block diagram of a typical Bluetooth® LE MCU

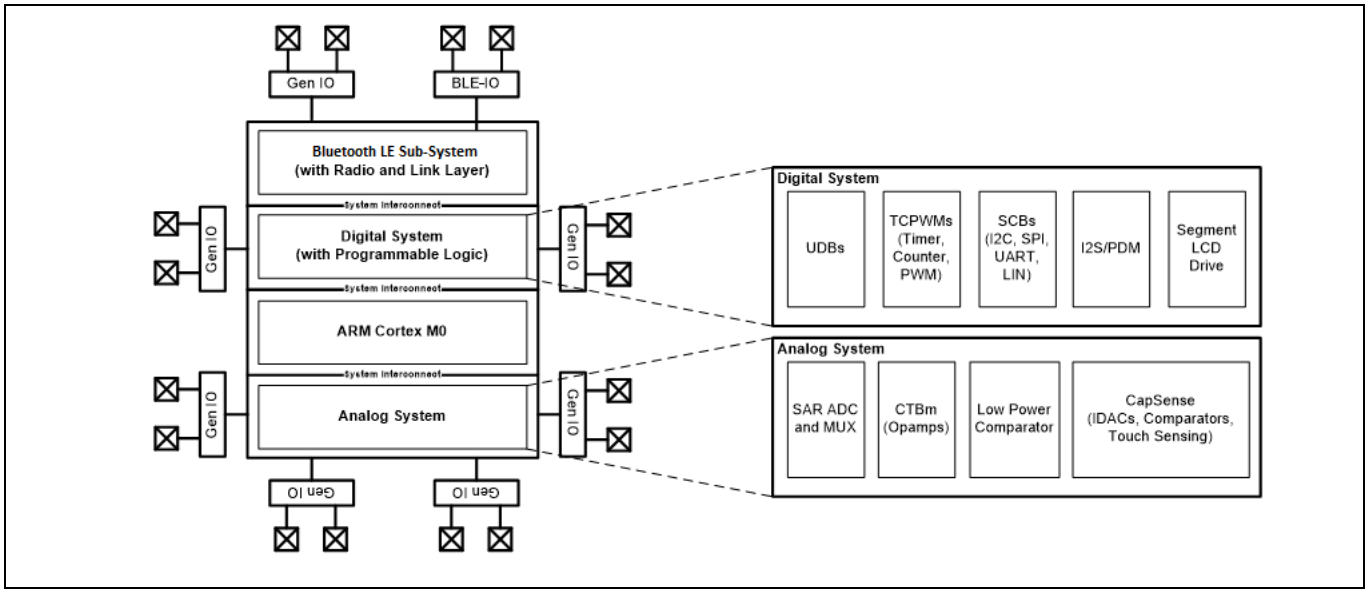


Figure 6 PSoC™ 4 with AIROC™ Bluetooth® LE block diagram

A typical MCU requires CPU firmware to process state machines, use a timer for timing, and drive an output pin. Thus, the functional path is almost always through the CPU. However, with PSoC™, asynchronous parallel processing is possible. You can configure a PSoC™ to have elements that operate independently from the CPU. PSoC™ Creator makes the interface between the analog, digital and programmable interconnect blocks

seamless and all the analog and digital peripherals are provided in the form of virtual ICs called Components in PSoC™ Creator. The example project in this application note will describe how to use PSoC™ Creator and its Components.

The PSoC™ also has programmable digital blocks known as Universal Digital Blocks (UDBs). PSoC™ Creator also provides several Components made out of UDBs, such as UART, SPI, I2S, Timer, PWM, Counter, Digital Gates (AND, OR, NOT, XOR, and so on), and many more. You can create custom state machines and digital logic using the UDBs in PSoC™ Creator. The method to create custom PSoC™ Creator Components is provided in the **PSoC™ Creator Component Author Guide**.

5 Bluetooth® LE overview

Bluetooth® LE or Bluetooth Smart™ is a low-power, short-range, low-data-rate wireless communication protocol that is defined by the Bluetooth® SIG. As shown in **Figure 7**, Bluetooth® LE has a layered protocol stack that is designed to efficiently transfer a small amount of data with low power consumption, making it the preferred wireless protocol for battery-operated devices.

The Bluetooth® LE stack consists of the following:

- 2.4-GHz RF physical layer (PHY) with a 1-Mbps data rate
- Link Layer (LL) that defines the timing and packet format for PHY
- Host Control Interface (HCI) that links the hardware controller (PHY + LL) layer with the firmware host layer of the stack
- Logical Link Control and Adaptation Protocol (L2CAP) that acts as a packet assembly/disassembly and protocol multiplexer layer
- Attribute Protocol (ATT) that defines how the application data is organized and accessed
- Security Manager (SM) that provides a toolbox for secure data exchange over the Bluetooth® LE link
- Generic Attribute Profile (GATT) that defines methods to access data defined by the ATT layer
- Generic Access Profile (GAP) that provides an application-oriented interface that determines whether the device acts as a Bluetooth® LE link master or slave, and configures the underlying layers accordingly
- See **Appendix E: Bluetooth® LE protocol** for a detailed description of the Bluetooth® LE protocol.

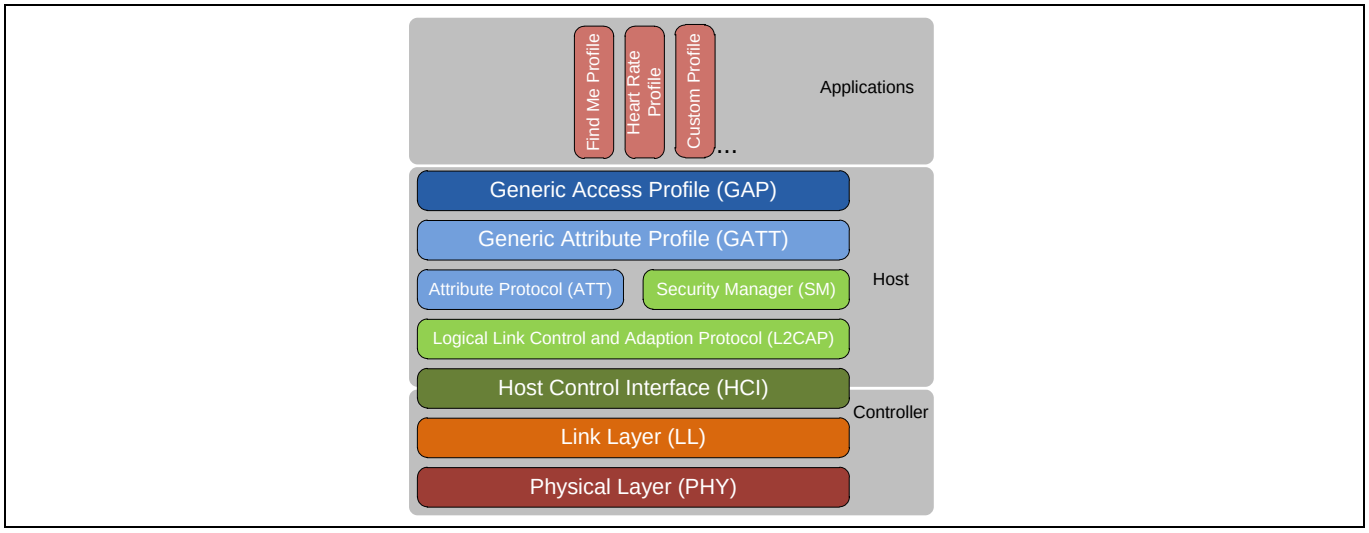


Figure 7 Bluetooth® LE protocol stack

To develop a Bluetooth® LE application, you do not need a working knowledge of this complex protocol stack. Infineon provides an easy-to-configure, GUI-based Bluetooth® Low Energy Component in PSoC™ Creator that abstracts the protocol complexity. To get started with Bluetooth® LE, it is sufficient to understand the following:

- Bluetooth® LE link establishment procedure
- Application data representation and abstraction
- Mapping of application requirements to Bluetooth® LE GAP and GATT layer configurations.

5.1 Bluetooth® LE link establishment

To establish a Bluetooth® LE link between two devices—the Bluetooth® Low Energy Pioneer Kit and a smartphone, for example—you need to understand the two Generic Access Profile (GAP) device roles as **Figure 8** shows:

- **GAP Peripheral:** A device that advertises its presence and accepts connection from a GAP Central device. A Bluetooth® Low Energy Pioneer Kit that implements a heart-rate measurement function is an example of a GAP Peripheral device.
- **GAP Central:** A device that scans for advertisements from GAP Peripherals and establishes a connection with them. A smartphone that connects to a heart-rate measurement device is an example of a GAP Central device.

After the Central device establishes a connection with the Peripheral, both devices are said to be connected over a Bluetooth® LE link. On a connected Bluetooth® LE link, independent of the GAP role, the Generic Attribute Profile (GATT) defines two profile roles based on the source and destination of data as **Figure 8** shows:

- **GATT server:** A GATT server is a device that contains data or state. When configured by a GATT client, it sends data to the GATT client or modifies its local state. For example, a heart-rate measurement device is a GATT server that sends heart-rate data to a smartphone GATT client. Similarly, a smart bulb is a GATT server that contains the state of the bulb (on/off) that can be configured by a smart switch GATT client.
- **GATT client:** A GATT client is a device that configures the state of a GATT server or receives data from a GATT server. For example, a smartphone that receives heart-rate information from the heart-rate device is a GATT client.

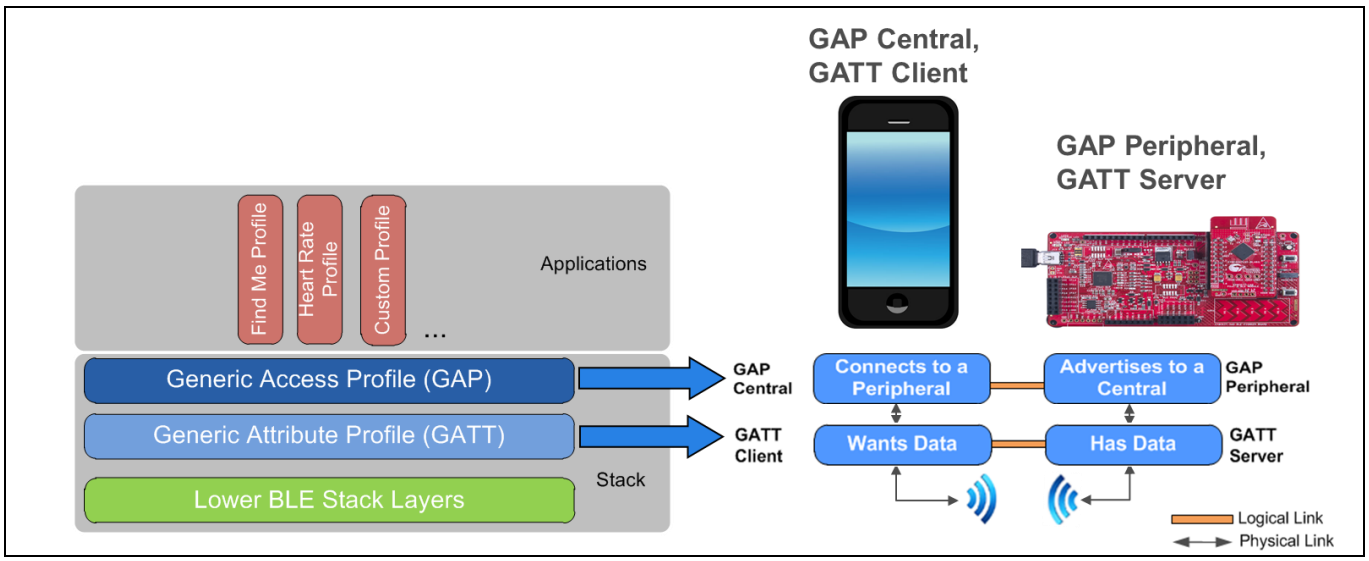


Figure 8 Bluetooth® LE application overview

After establishing a Bluetooth® LE link, the GATT client discovers all the data and state present on the GATT server. Once discovered, the GATT client can configure and/or read/write data or state of the GATT server.

5.2 GATT data format

After understanding the Bluetooth® LE GAP/GATT roles, next step is to understand how the data is stored on a GATT server and how it is retrieved by a GATT client. A GATT server uses Attributes, Characteristics, and Services

to represent and abstract data in a Bluetooth® LE device. As you will see in this section, a Service contains one or more Characteristics and each Characteristic is composed of multiple Attributes that contain the actual data.

- **Attribute:** An Attribute is the fundamental data container of the GATT layer that represents a discrete piece of information. The structure of an Attribute consists of the following, as shown in **Figure 9**.
  - Attribute Handle: Used to address the Attribute
  - Attribute Type: A 16-bit Universally Unique Identifier (UUID) assigned by the Bluetooth® SIG that specifies the data contained in the Attribute
  - Attribute Value: Contains the actual data
  - Attribute Permission: Specifies read/write permissions and security requirements for the Attribute

Heart-rate measurement, battery level, battery level units, and device name are a few examples of an Attribute.

A GATT server consists of a number of Attributes that are stored in a database called “Attribute database.” The GATT client performs read/write operations on one or more Attributes in the GATT server’s Attribute database using the Attribute handle. As a user, you only need to know the Attribute handle on which you want to perform a read/write operation; other details are abstracted by Bluetooth® Low Energy Component API functions.

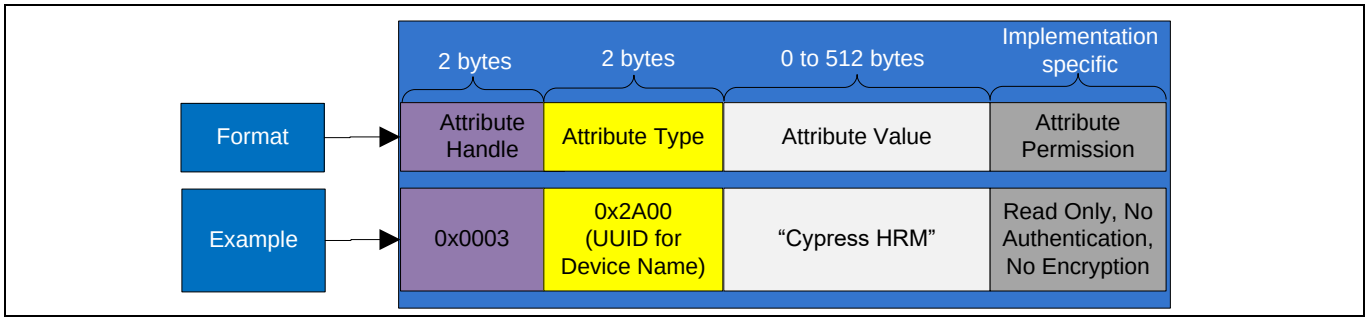


Figure 9 GATT attribute example

- **Characteristic:** A Characteristic is composed of multiple discrete Attributes that when combined, define the system information or meaningful data. A Characteristic consists of a Characteristic Declaration Attribute, a Characteristic Value Attribute, and, optionally one or more Characteristic Descriptor Attributes, as shown in **Figure 10**. For example, combining the Battery Level Attribute of “90” and Battery Level Descriptor Attribute of “%” provides the battery level information of a system as 90%.

  The Bluetooth® SIG offers a set of predefined **standard Characteristics** that you can use in your application. In addition, you can define your own custom Characteristics. On/off state of a smart bulb is an example for a custom Characteristic.

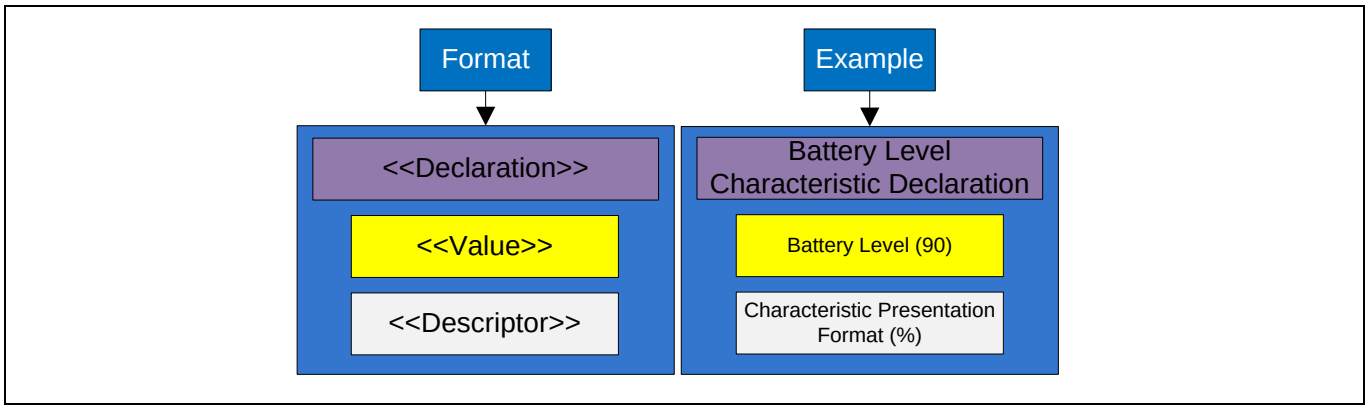


Figure 10 GATT characteristic example

- **Service:** A Service is composed of one or more related Characteristics that define a particular function or feature of a device. **Figure 11** shows an example (Heart Rate Service) that has three Characteristics describing the information related to measuring the heart-rate.

The Bluetooth® SIG offers a set of predefined **standard Services** for implementing commonly used Bluetooth® LE device functionalities. In addition, you can define your own custom Services that consist of standard or custom Characteristics. Smart bulb service is an example for a custom Service that contains the bulb on/off state Characteristic.

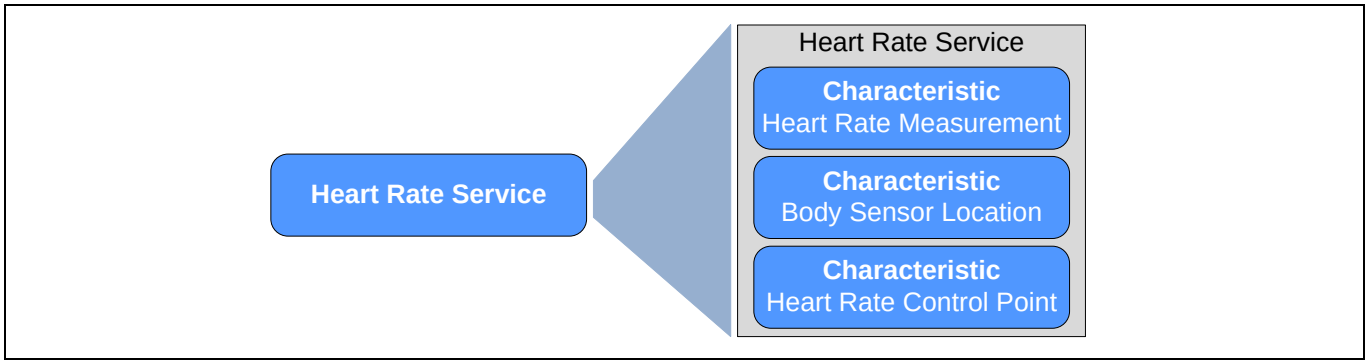


Figure 11 GATT service example

In addition to defining the data format, GATT layer also defines a set of procedures or methods to discover and access data in the GATT layer. After establishing a Bluetooth® LE connection, a GATT client uses these GATT procedures to discover the complete attribute database of the connected GATT server. After the discovery is complete, the GATT client uses Characteristic read/write GATT procedures to perform read/write operations on the Attributes using the Attribute handle.

Figure 12 shows how the heart-rate data is modeled on a heart-rate sensor and the GATT procedures used by a smartphone to get the heart-rate data. A typical heart-rate sensor implements a GAP Peripheral role where it advertises and connects to a GAP Central device such as a smartphone. After establishing the connection, the heart-rate sensor device exposes a GATT server that encapsulates the heart-rate data. The GATT server supports a Heart Rate Service that contains the Heart Rate Measurement Characteristic, which stores the actual measured heart-rate value in the form of a Heart Rate Value Attribute.

On the smartphone side, after establishing the connection, a GATT client on the smartphone initially discovers all the Services, Characteristics, and Attributes supported by the connected GATT server using GATT discovery procedures. After completing the discovery, the smartphone GATT client uses GATT Characteristic procedures to enable push notifications for heart-rate data from the GATT server.

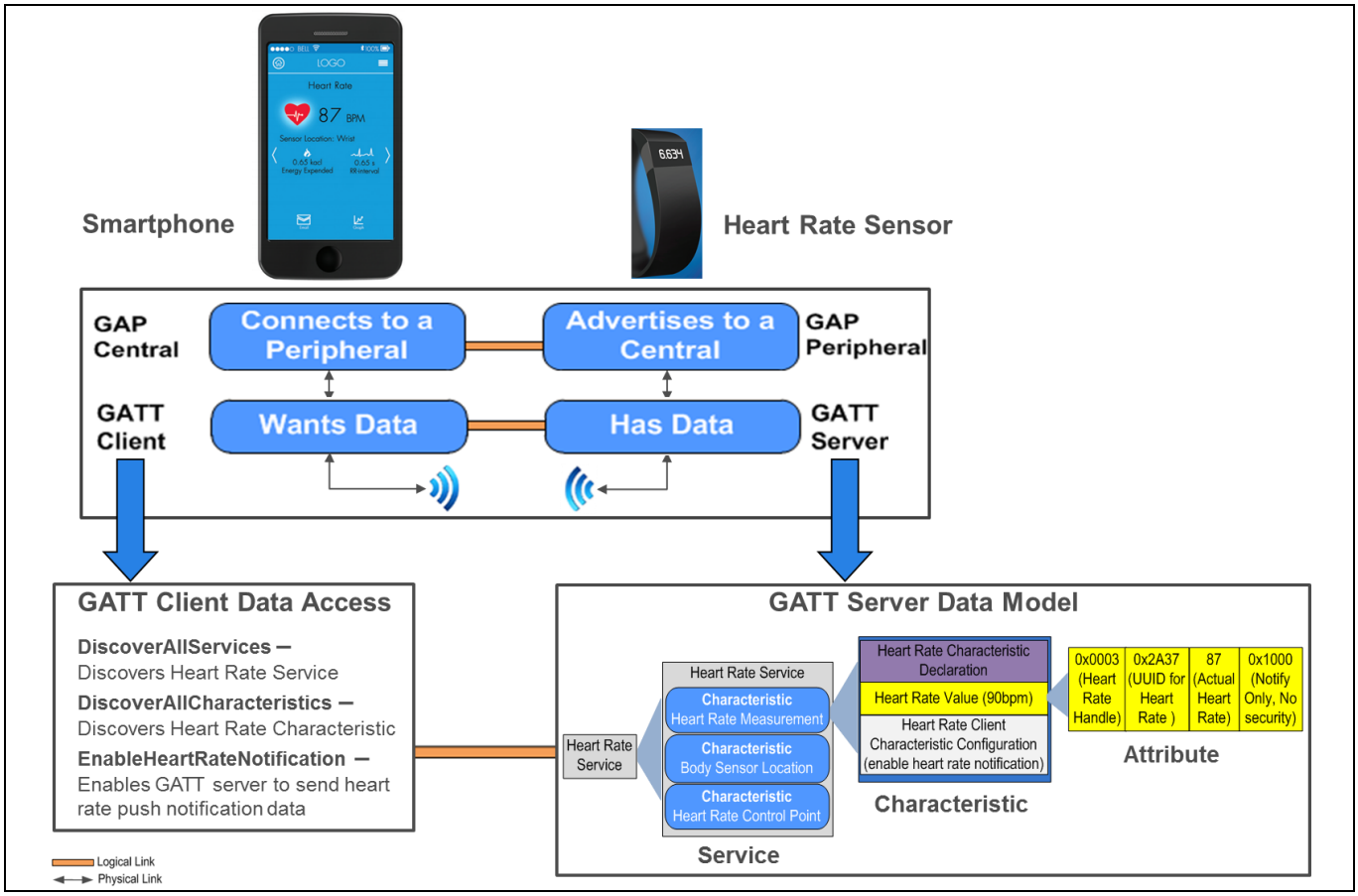


Figure 12 GATT data format

5.3 Bluetooth® LE profile

A Profile in Bluetooth® LE is a specification that guarantees application-level interoperability between Profile-compliant devices. It defines the role and configuration of different Bluetooth® LE layers and GATT Service(s) to be supported to create a specific end application or use case. For example, in the case of a heart-rate monitoring device, the Bluetooth® LE Heart Rate Profile defines the required GAP, GATT roles, and the GATT Services to be supported by the heart-rate monitoring device to create an interoperable heart-rate monitoring device. The Bluetooth® SIG offers a set of predefined **standard Profiles** for commonly used Bluetooth® LE end applications. In addition, you can create your own custom Profile that consists of standard or custom Services.

As shown in **Figure 13**, similar to the GATT layer, the Profile defines two application roles:

- **Sensor or server:** The Sensor Profile role is supported by the application that has data. The Sensor Profile specification defines the required roles (for example, GATT/GAP roles) and behavior (for example, advertisement interval, GATT Services to be supported) of the Bluetooth® LE device to support a Sensor application use case. Following the Sensor Profile specification guarantees interoperability of the Sensor application with any other device that implements the corresponding Collector Profile. For example, a heart-rate monitoring device that implements the Heart Rate Sensor Profile will be interoperable with all the smartphones that implements Heart Rate Collector Profile.
- **Collector or client:** The Collector Profile role is supported by the application that wants data. The Collector Profile specification defines the required Bluetooth® LE device roles and behavior to interoperate and collect information from any device that implements the corresponding Sensor Profile specification.

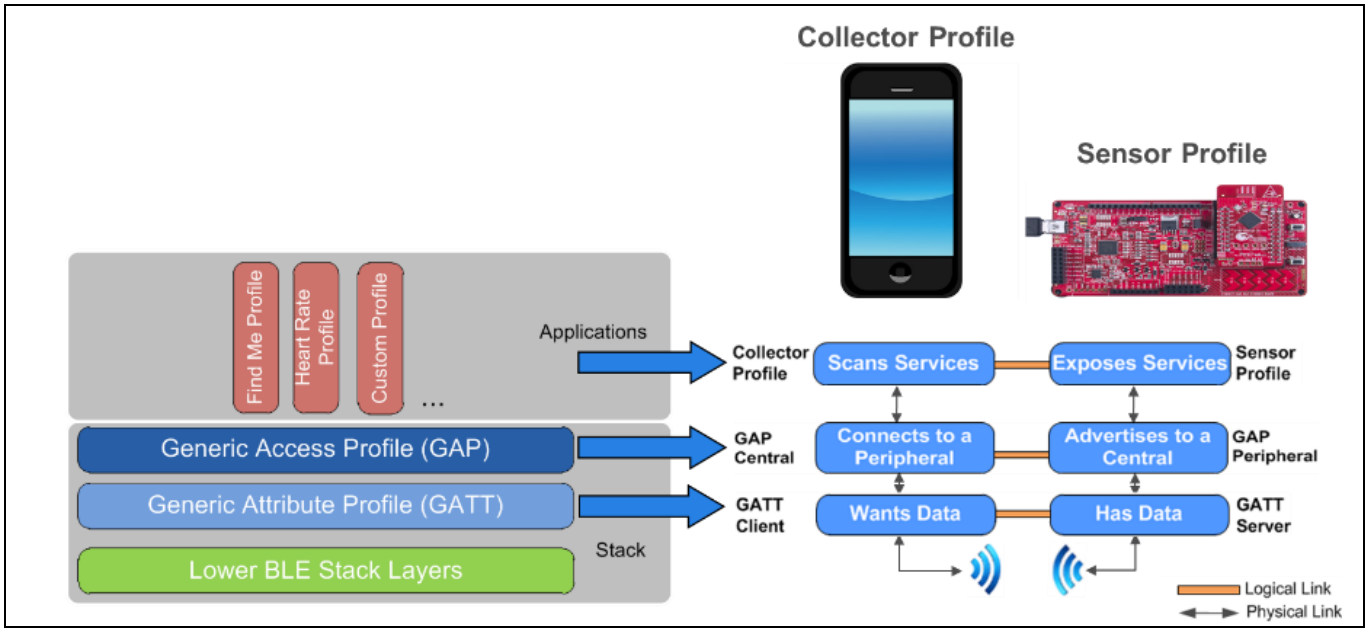


Figure 13 Bluetooth® LE profile example

A summary of the data abstraction and hierarchy in a Bluetooth® LE device is shown in **Figure 14**.

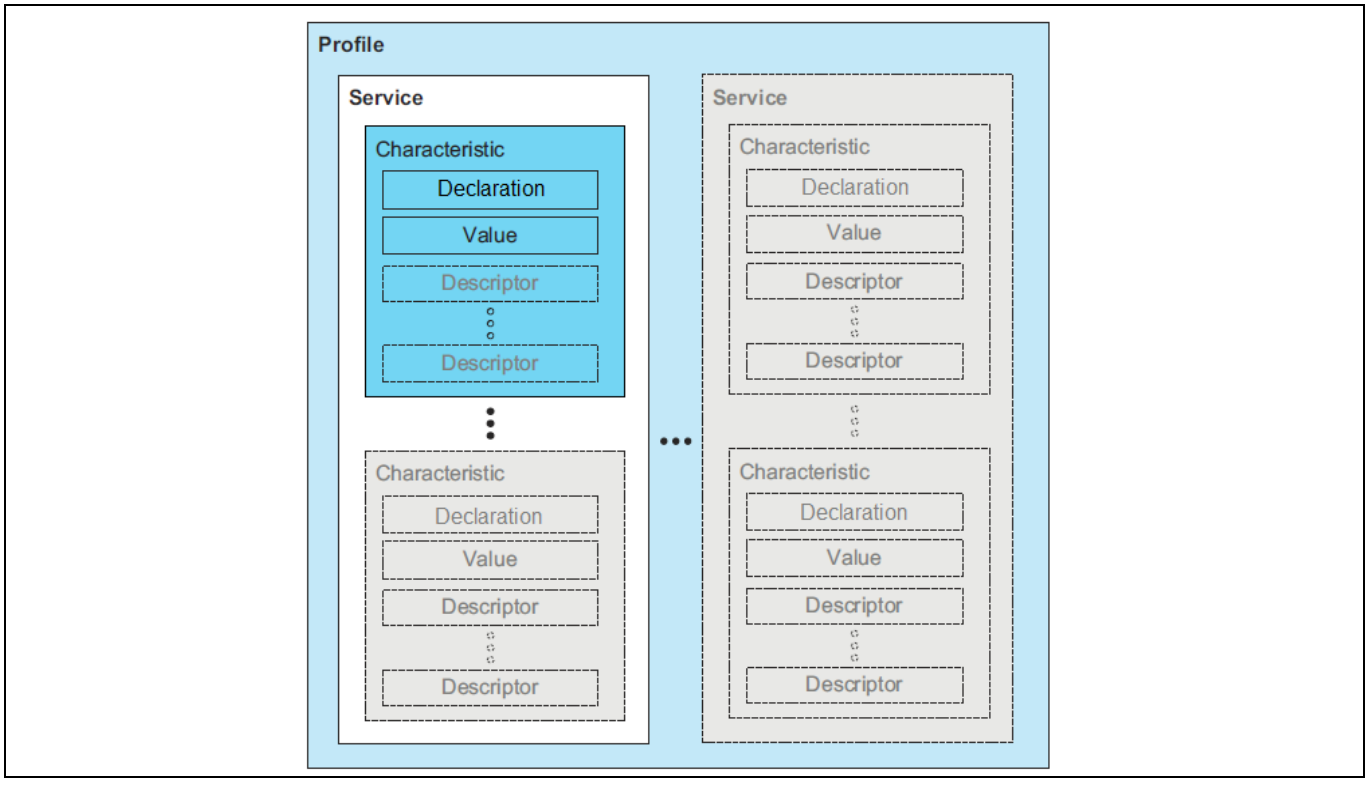


Figure 14 Bluetooth® LE data hierarchy*

* Image courtesy of Bluetooth® SIG.

5.4 Bluetooth® Low Energy Component

5.4.1 Features

The Bluetooth® Low Energy Component in PSoC™ Creator abstracts the Bluetooth® LE protocol into a simple and easy-to-use GUI and an API with just a few functions. Bluetooth® Low Energy Component features:

- Bluetooth 4.2-compliant protocol stack
- Supports all the GAP roles – Central, Peripheral, Broadcaster, and Observer. A limited simultaneous combination of GAP roles such as Central and Observer, Peripheral and Broadcaster are also supported.
- Supports all the Bluetooth® SIG adopted GATT based Profiles and Services with at least one example code per supported Profile
- Custom Profile creation and usage made easy by the Bluetooth® Low Energy Component GUI
- Supports L2CAP connection oriented channels, Link Layer low duty cycle advertising and LE ping features
- Supports up to 4 bonded devices and 8 device whitelist filter
- Supports all Bluetooth 4.2 security modes
- Supports link layer data length extension, link layer privacy, and LE secure connection features of Bluetooth 4.2 specification

See the Bluetooth® Low Energy Component **datasheet** for details.

5.4.2 Configuration

It takes five steps to create a Bluetooth® SIG defined Profile (standard Profile) based application using the Bluetooth® Low Energy Component. For example, to create a Find Me Target example described in **My First PSoC™ 4-BLE Design** section, follow these steps:

Step 1: Select a desired Profile and Profile role for your design. The Bluetooth® Low Energy Component automatically selects the required GAP and GATT roles for the selected Profile role. For the Bluetooth® SIG defined Find Me Profile in Target role shown in **Figure 15**, the GAP role is set to Peripheral and the GATT role is set to server as per the **Find Me Profile** specification.

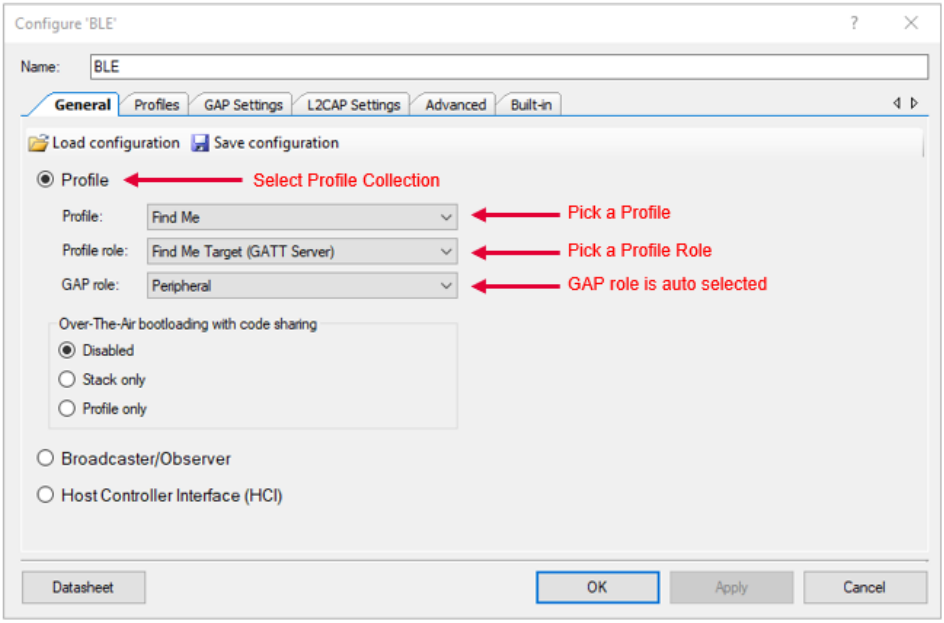


Figure 15 Bluetooth® Low Energy component profile configuration

Step 2: For the selected Profile role, all the supported GATT Services and their corresponding Characteristics are auto-generated per the Profile specification. Verify and/or edit the Service and Characteristic values if required based on your design. The settings that you make in the **Profiles** tab form the Attribute database for the selected design.

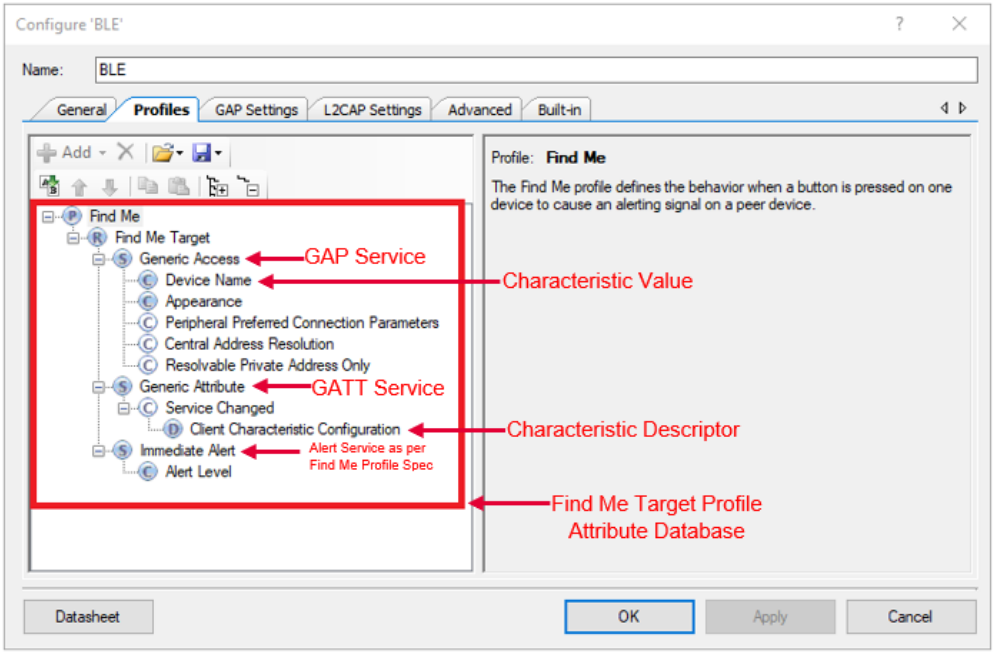


Figure 16 Bluetooth® LE attribute database configuration

Step 3: Configure the GAP general and advertisement data settings for your design as explained at the top of page **14**, as shown in **Figure 17** and **Figure 18**.

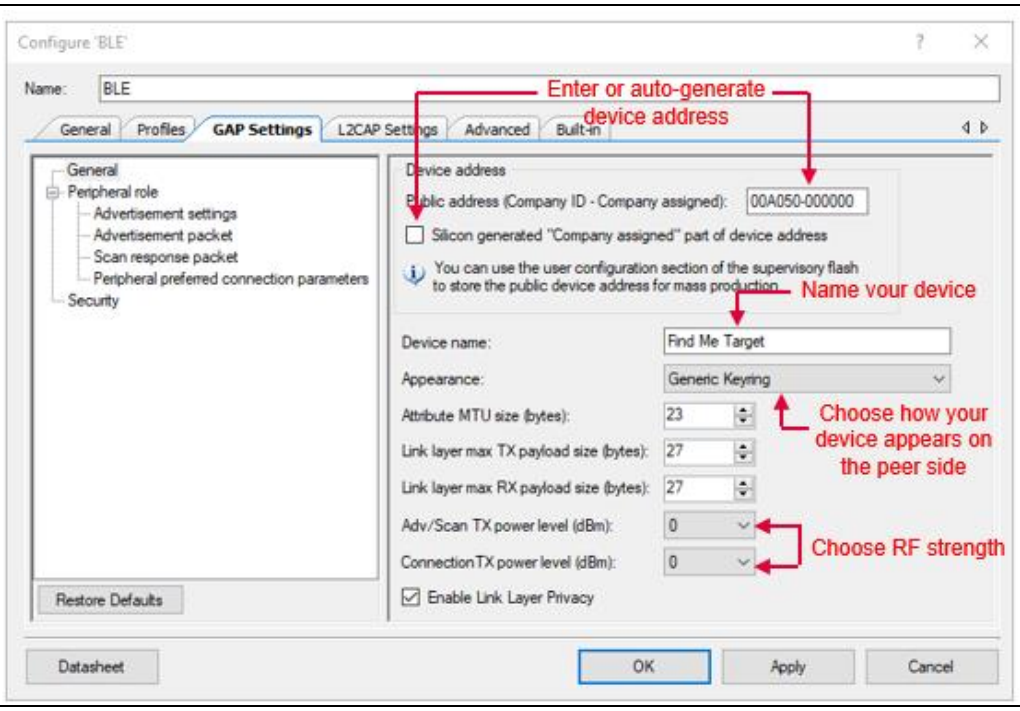


Figure 17 Bluetooth® LE GAP configuration

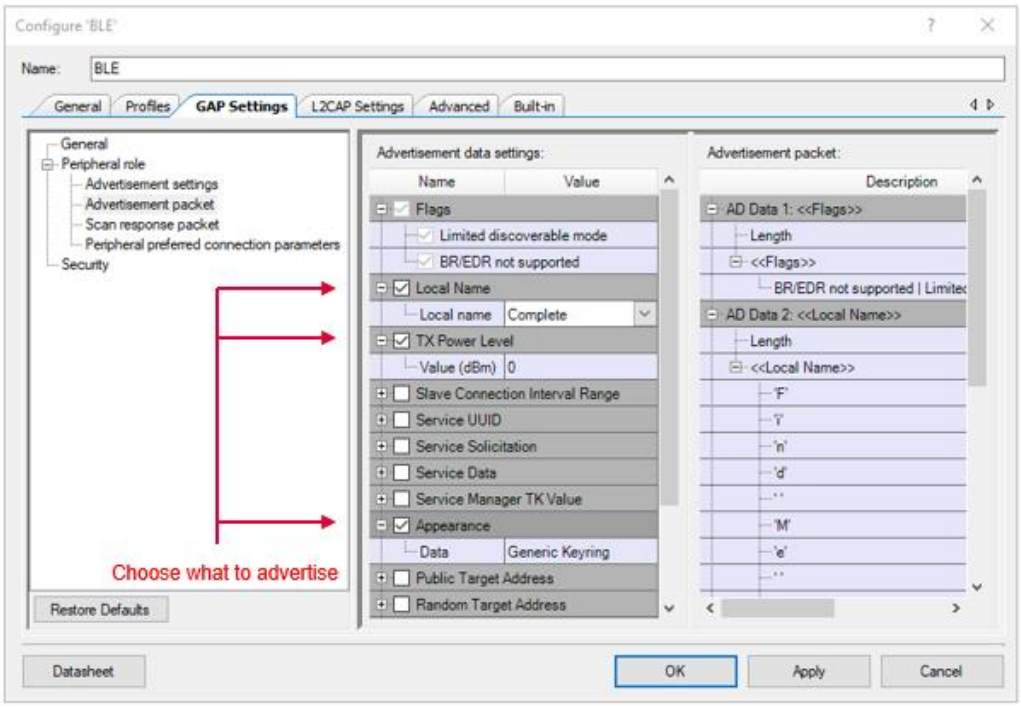


Figure 18 Bluetooth® LE advertisement data configuration

Step 4: Write firmware to initialize the design you just configured. Register event handlers with the Bluetooth® Low Energy Component to receive data and events. Event handler details are described **later** in this document.

Step 5: Wait in the program main loop for an event from the Bluetooth® Low Energy Component and take the necessary action or send data to the Central device using Bluetooth® Low Energy Component API functions.

The **My First PSoC™ 4-BLE Design** section will walk you through a step-by-step configuration of the Bluetooth® Low Energy Component for creating a simple Peripheral application. See application notes **AN91184** and **AN91162** for a step-by-step description of how to use the Bluetooth® Low Energy Component to develop applications using Bluetooth® LE standard and custom Profiles.

6 PSoC™ 4 with AIROC™ Bluetooth® LE development setup

Figure 19 shows the hardware and software tools required for evaluating Bluetooth® LE Peripheral designs using the PSoC™ 4 with AIROC™ Bluetooth® LE device. In a typical use case, the Bluetooth® Low Energy Pioneer Kit (red board in **Figure 19**) is configured as a Peripheral that can communicate with a Central device such as CySmart iOS/Android app or CySmart Host Emulation Tool. The CySmart Host Emulation Tool also requires a Bluetooth® Low Energy Dongle (black board in **Figure 19**) for its operation.

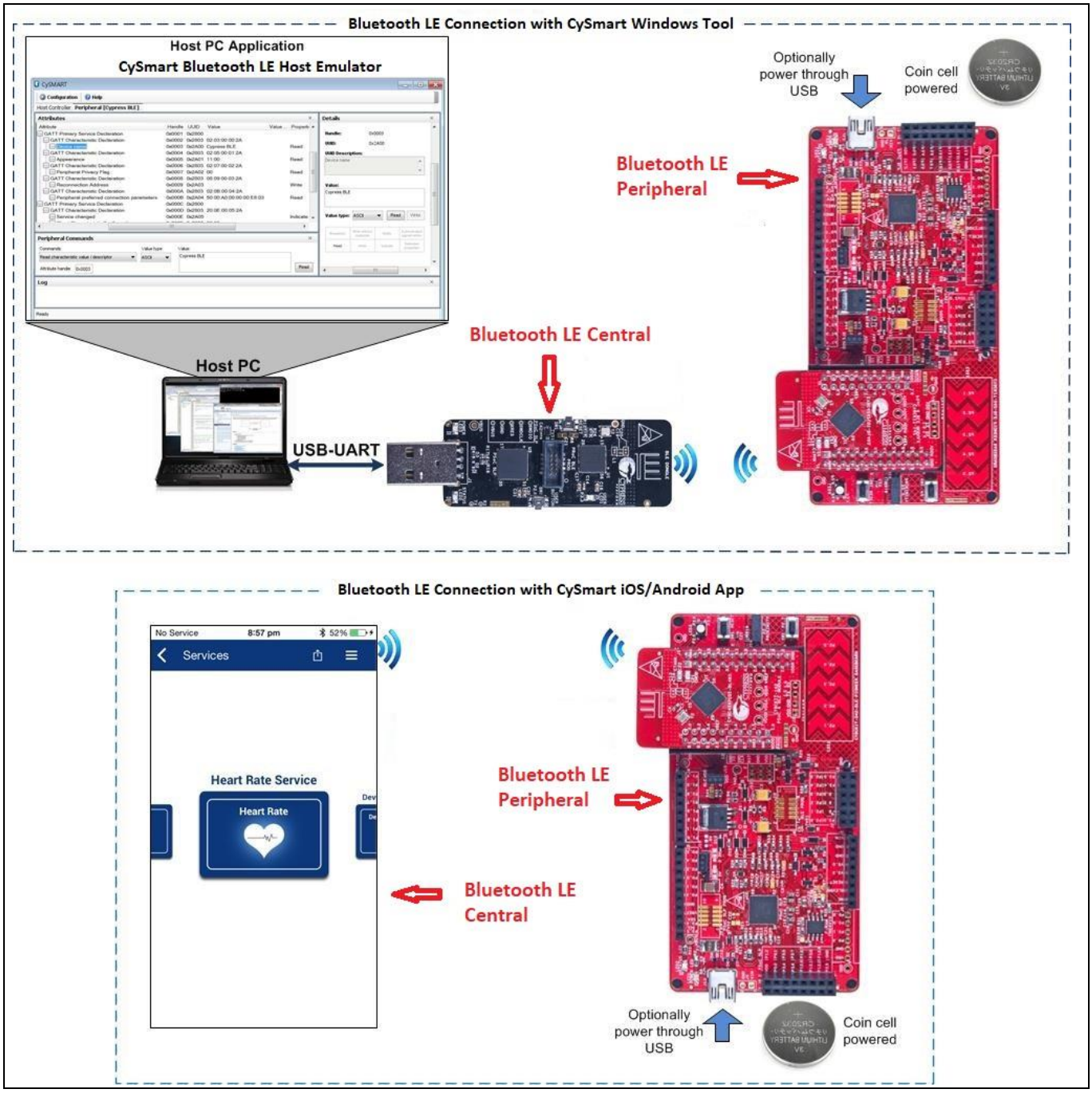


Figure 19 Bluetooth® LE functional setup

As shown in **Figure 20**, the Bluetooth® Low Energy Dongle is pre-programmed to work with Windows CySmart Host Emulation Tool. The Bluetooth® Low Energy Pioneer Kit has an on-board USB programmer that works with PSoC™ Creator for programming or debugging your Bluetooth® LE design. Bluetooth® Low Energy Pioneer

PSoC™ 4 with AIROC™ Bluetooth® LE development setup

Kit can either be powered over the USB interface or by a coin-cell battery. Both the Bluetooth® Low Energy Dongle and the Bluetooth® Low Energy Pioneer Kit can simultaneously be connected to a common host PC for development and testing.

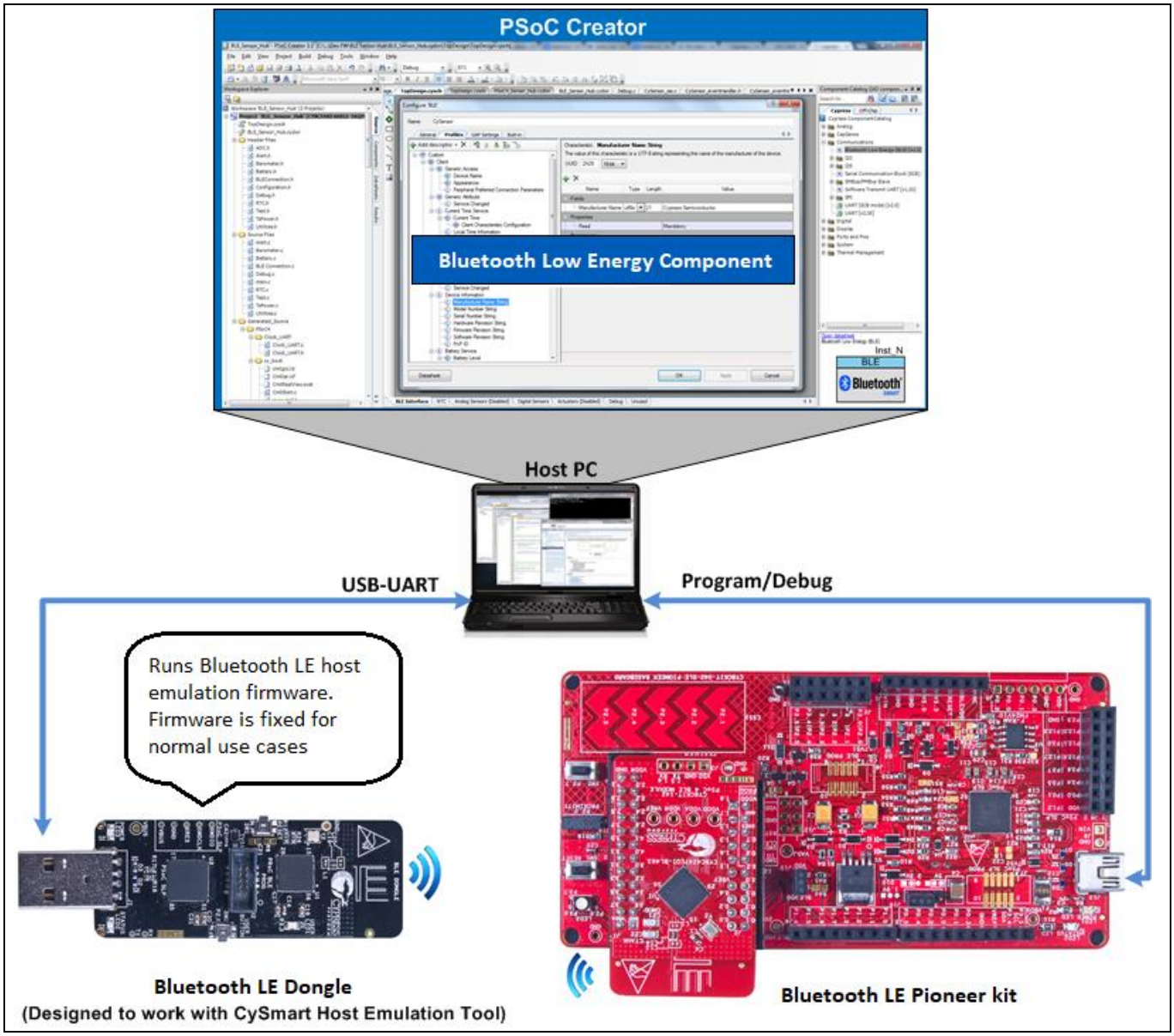


Figure 20 Bluetooth® LE development setup

7 My first PSoC™ 4 with AIROC™ Bluetooth® LE design

This section gives you a step-by-step process for building a simple Bluetooth® Low Energy Pioneer Kit–based design with the PSoC™ 4 with AIROC™ Bluetooth® LE device using PSoC™ Creator. A simple Bluetooth® SIG defined standard Profile design creation is described in this section. For creating advanced standard or custom Profile designs, refer to **Designing Bluetooth® Low Energy Applications** and **Creating a Bluetooth® Low Energy Custom Profile** application note.

7.1 About the design

This design implements a Bluetooth® LE **Find Me Profile (FMP)** in the Target role that consists of an Immediate Alert Service (**IAS**). FMP and IAS are Bluetooth® LE standard Profile and Service respectively defined by the Bluetooth® SIG. Alert levels triggered by the Find Me Locator are indicated by varying the state of an LED on the Bluetooth® Low Energy Pioneer Kit, as **Figure 21** shows. Two status LEDs indicate the state of the Bluetooth® LE interface.

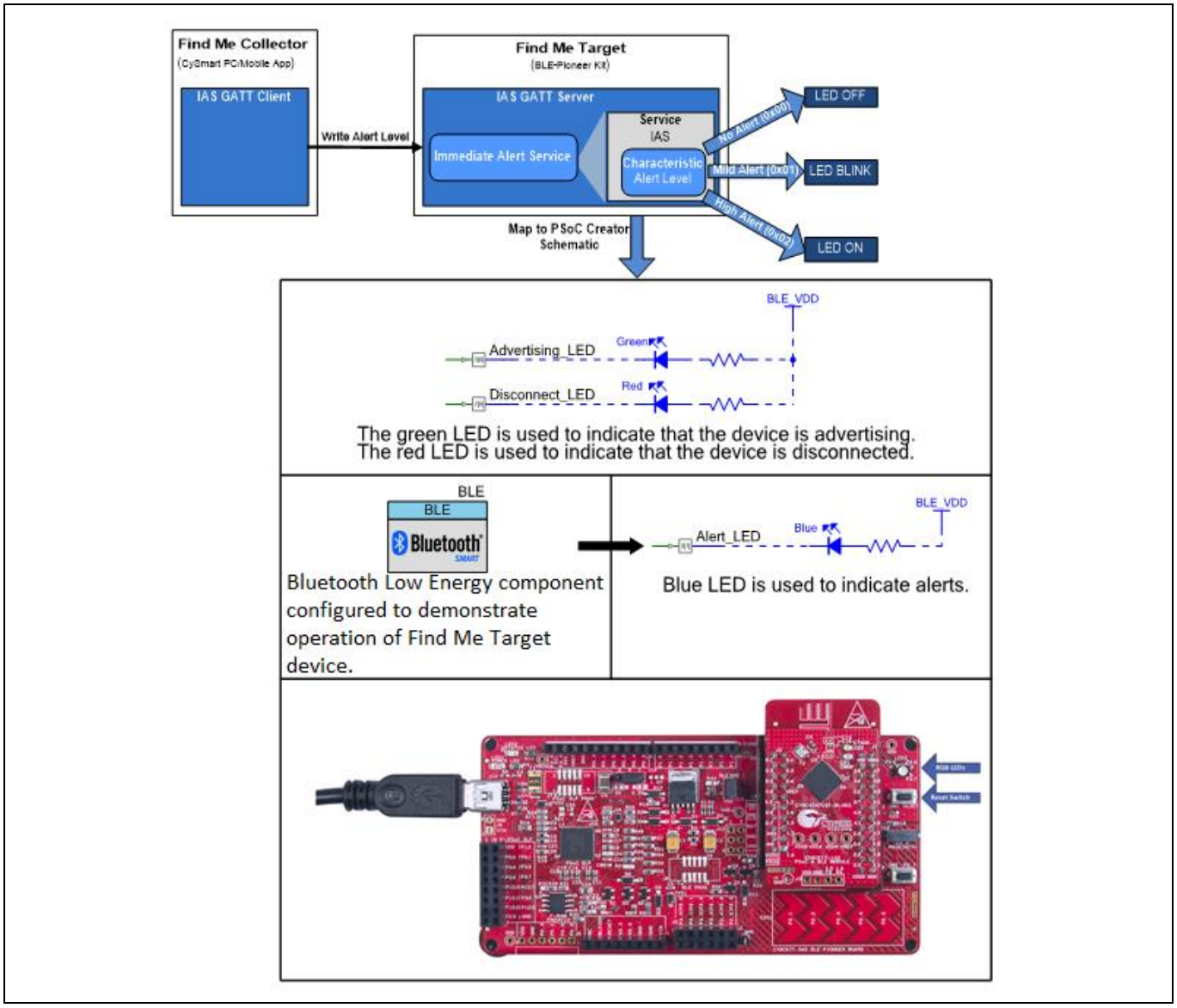


Figure 21 My First PSoC™ 4-Bluetooth® LE design

Create your first PSoC™ 4 with AIROC™ Bluetooth® LE design in four stages:

1. Create the design in the PSoC™ Creator schematic page.
2. Write the firmware to initialize and handle Bluetooth® LE events.
3. Program the PSoC™ 4 Bluetooth® LE device on the Bluetooth® Low Energy Pioneer Kit.
4. Test your design using the CySmart Host Emulation Tool or mobile app.

Note: *The functional PSoC™ Creator project for the Bluetooth® LE example design described in this application note is distributed as part of PSoC™ Creator code examples. You can choose to start with the code example, by selecting **File > Code Example**, and then selecting **Filter by** as **Find Me > BLE_FindMe.** If you use the code example, **skip** schematic configuration (stage 1) and firmware development (stage 2) mentioned above and go to programming (**stage 3**) and testing the design (**stage 4**).*

7.2 Prerequisites

Before you get started with the implementation, make sure you have a **Bluetooth® Low Energy Pioneer Kit** and have installed the following software:

- **PSoC™ Creator 4.4** with **PSoC™ Programmer 3.29**
- **CySmart Host Emulation Tool** or CySmart **iOS**/**Android** app

7.3 Stage 1: Create the design

This section takes you on a step-by-step guided tour of the design process. It starts with creating an empty project and guides you through schematic design entry.

1. Install PSoC™ Creator 4.4 on your PC.
2. Start PSoC™ Creator, and from the **File** menu, choose **New** > **Project**, as **Figure 22** shows.

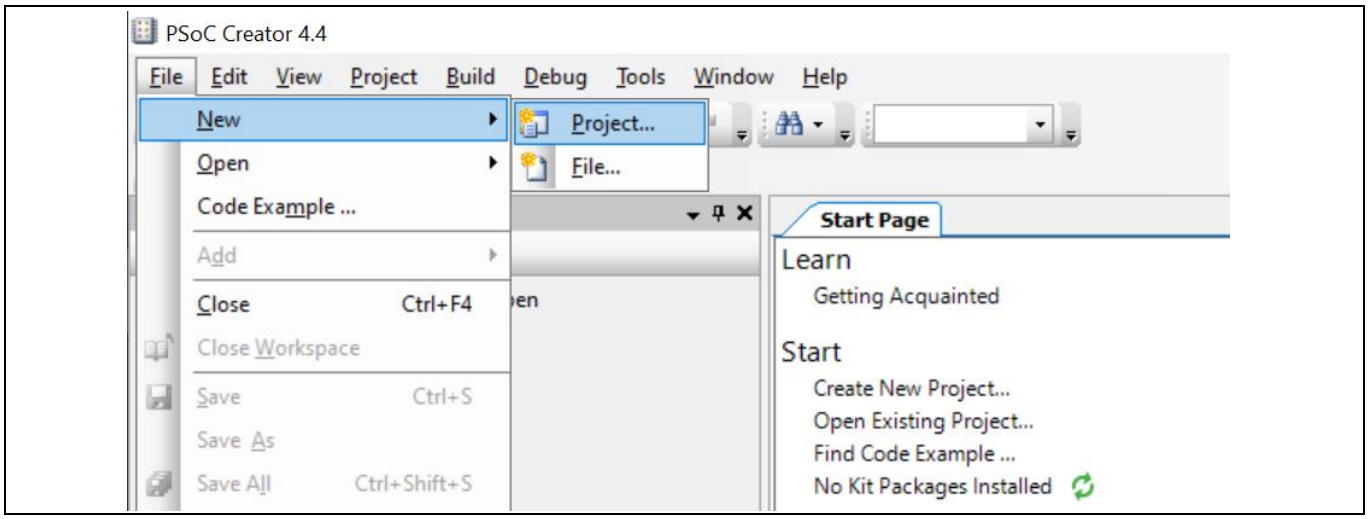


Figure 22 Creating a New project

3. Select the target device as “PSoC™ 4” as shown in **Figure 23** to select CY8C4248LQI-BL583 device used on **Bluetooth® Low Energy Pioneer Kit** and click **Next**. To select CY8C4248LQI-BL583 device, click on the dropdown and select “Launch Device Selector” and select the mentioned device. If you are using a custom PSoC™ 4 with AIROC™ Bluetooth® LE hardware or a different PSoC™ 4 with AIROC™ Bluetooth® LE part number, choose the “Launch Device Selector” option in Target device and select the appropriate part number.

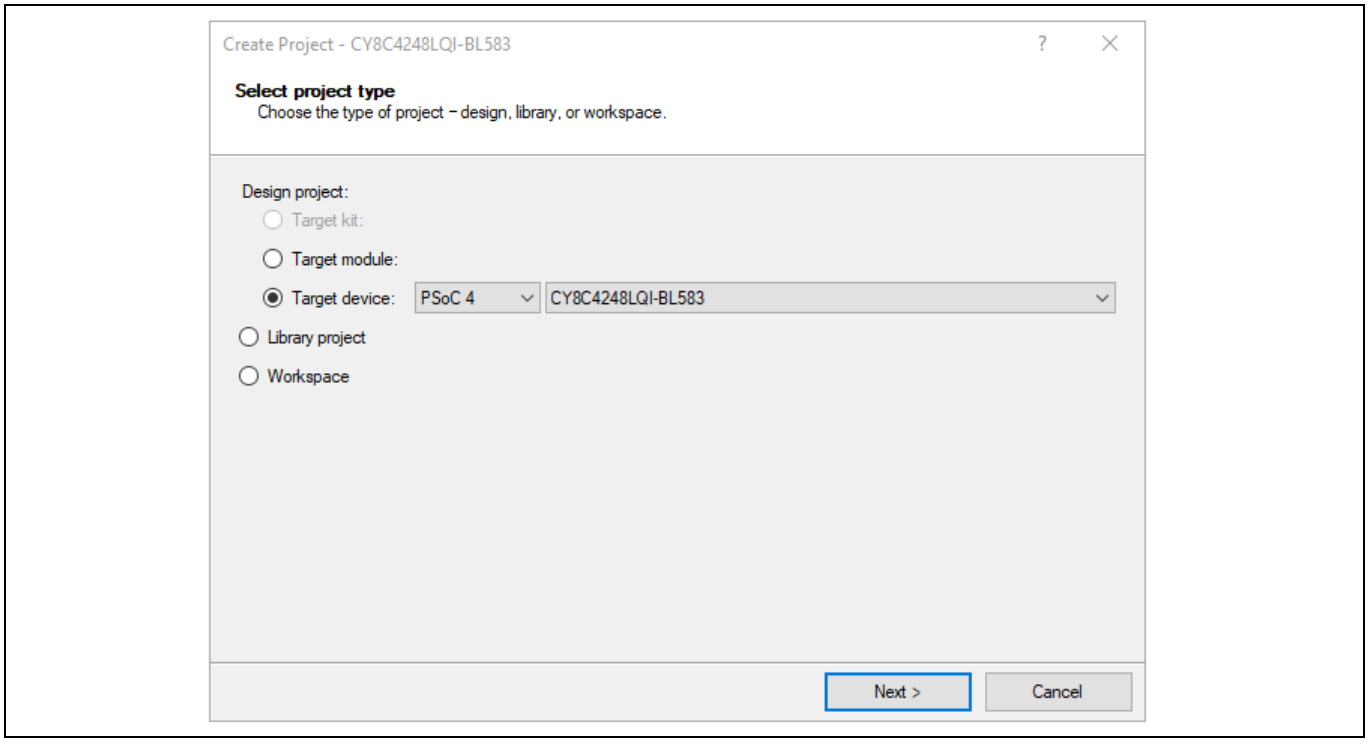


Figure 23 Selecting the Target device

4. Select an “Empty Schematic” project template as shown in **Figure 24** and click **Next**.

*Note: If you like to use the pre-built example for this design, select Code example in **Figure 24** and click **Next**. In the code example selection window, select **Filter by** as **Find Me > BLE_FindMe** and **skip** to programming section directly.*

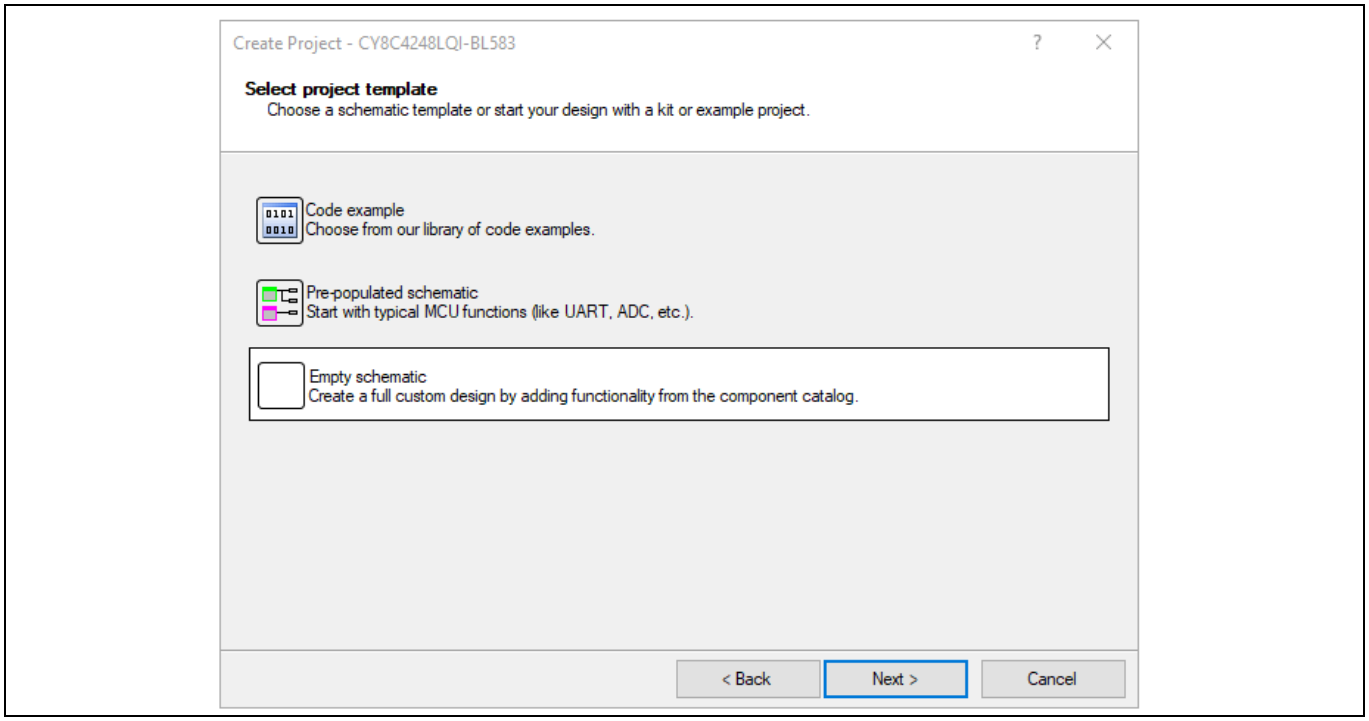


Figure 24 Selecting the project template

5. Give the workspace and the project a name such as “BLE_FindMe,” as **Figure 25** shows. Choose an appropriate location for your new project, and then click **Finish**.

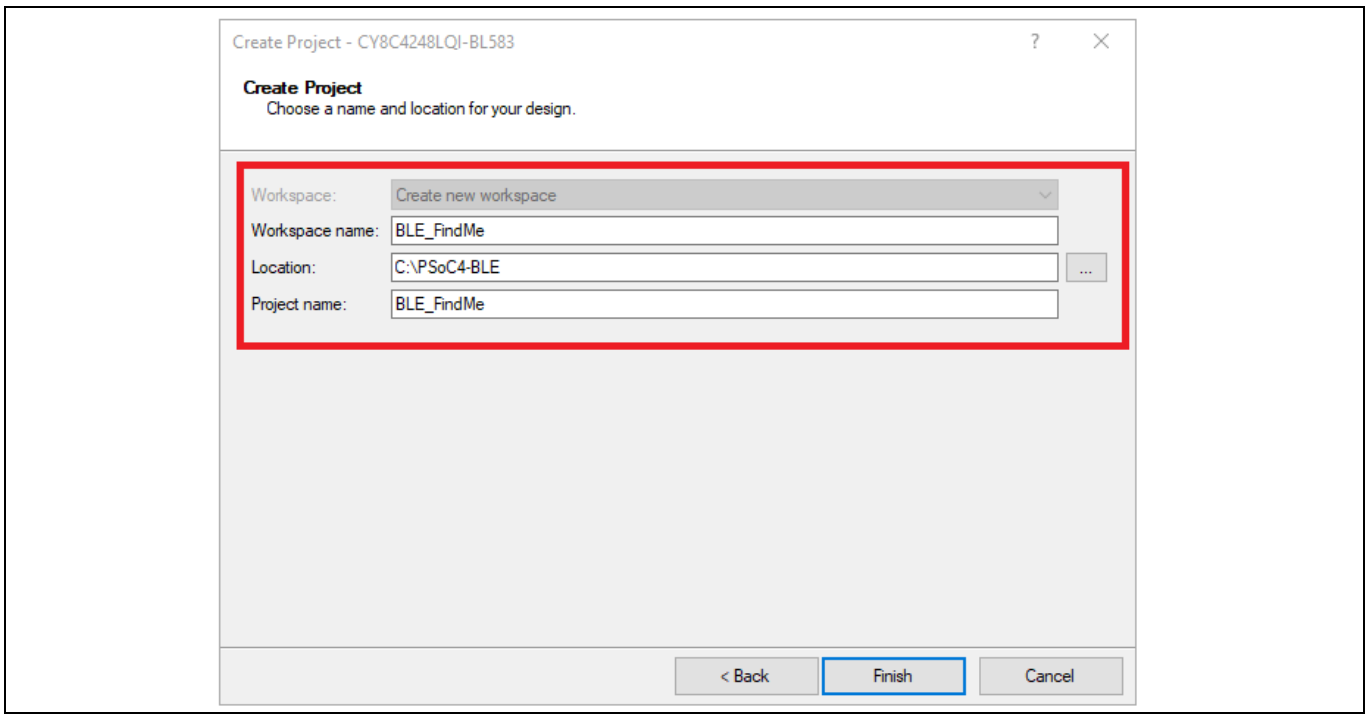


Figure 25 Project naming

Creating a new project generates a project folder with a baseline set of files. You can view these files in the **Workspace Explorer** window, as **Figure 26** shows.

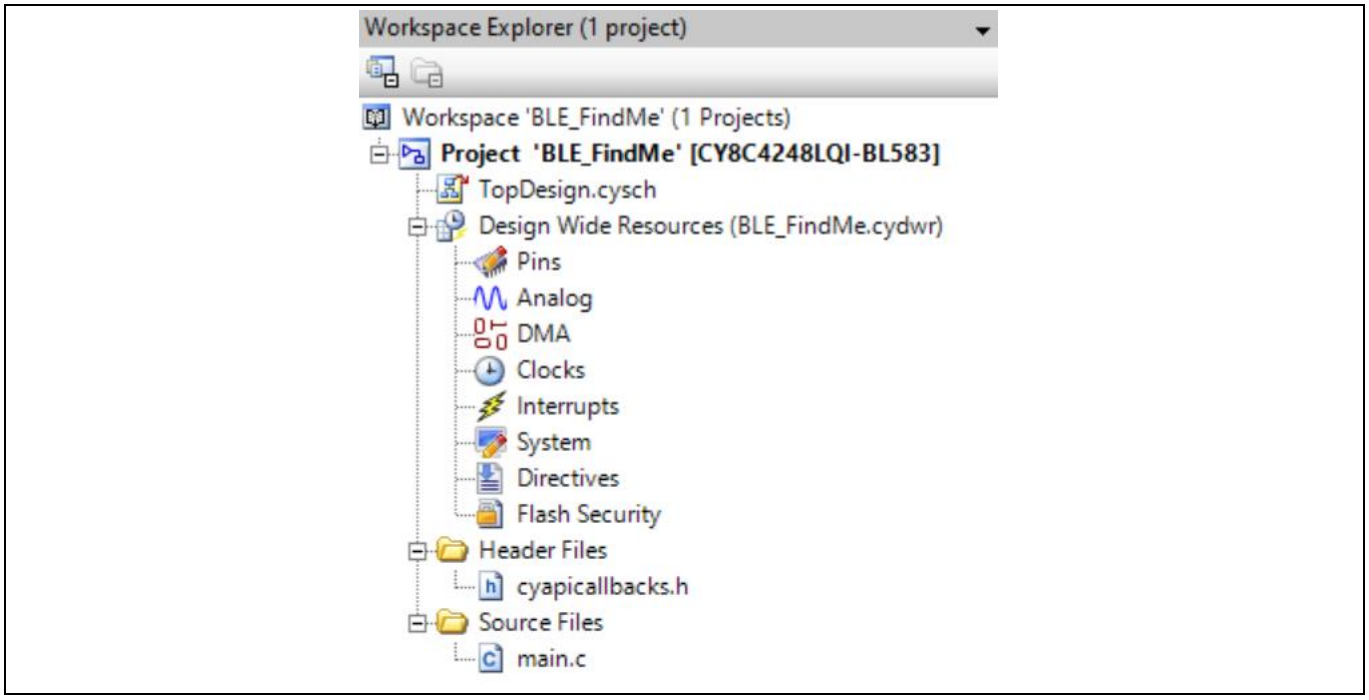


Figure 26 Opening TopDesign schematic

6. Open the project schematic file *TopDesign.cysch* by double-clicking it.

7. In the project schematic, you can create analog, digital, and communication designs by dragging and dropping Components and wiring them. For the current design, you need two Components—Bluetooth® Low Energy and Pin—as **Figure 21** shows.
8. Drag one Bluetooth® Low Energy Component from the **Component Catalog** onto the schematic, as **Figure 27** shows.

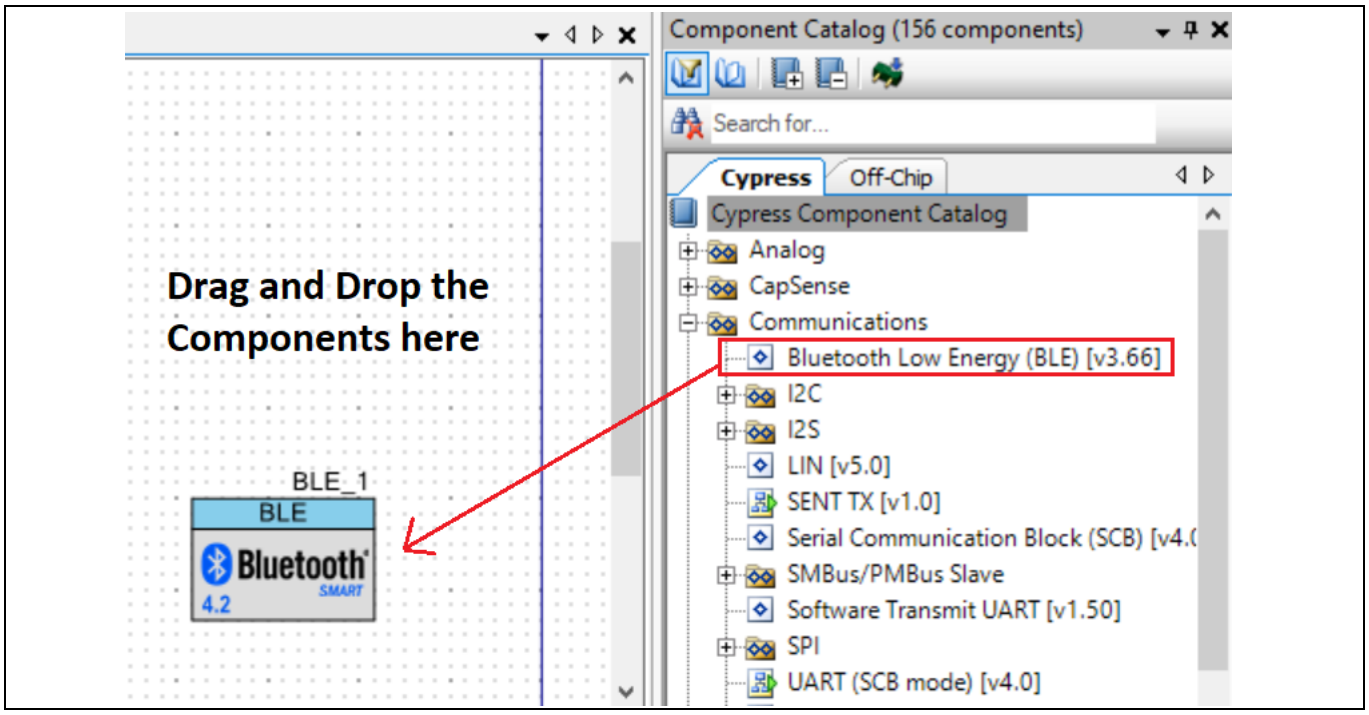


Figure 27 Location of the Bluetooth® Low Energy component

9. Double-click the Bluetooth® Low Energy Component on the schematic to configure it as a "BLE Find Me Target" with the following properties:

- GAP Peripheral role with Find Me Target (GATT Server) configuration as shown in **Figure 28**.
- GAP Device Name set to "Find Me Target", Appearance set to "Generic Keyring", and check the Silicon generated "Company assigned" part of device address as shown in **Figure 29**. This configures the device name and type that appears when another device attempts to discover your device and assigns a unique Bluetooth® LE device address to your device.
- Limited advertisement mode with an advertising timeout of 30 seconds and a fast advertisement interval of 20 to 30 ms as shown in **Figure 30**. Fast advertising allows quick discovery and connection but consumes more power due to increased RF advertisement packets. De-select the "Slow advertising interval" checkbox.
- Advertisement Packet with Immediate Alert Service enabled as shown in **Figure 31** and Scan Response Packet with Local Name, Tx Power Level, and Appearance fields enabled as shown in **Figure 32**.
- GAP security set to the lowest possible configuration that does not require authentication, encryption, or authorization for data exchange (Mode 1, No Security). Set the I/O capabilities to "No Input No Output" and Bonding requirement to "No Bonding".
- **Figure 28** to **Figure 34** show the Bluetooth® Low Energy Component screenshots for this configuration.

*Note: You do not need to change the default configuration of the Bluetooth® Low Energy Component in the **GAP Settings > Peripheral preferred connection parameters**, **Profiles, L2CAP Settings,** and **Advanced** tab for this design.*

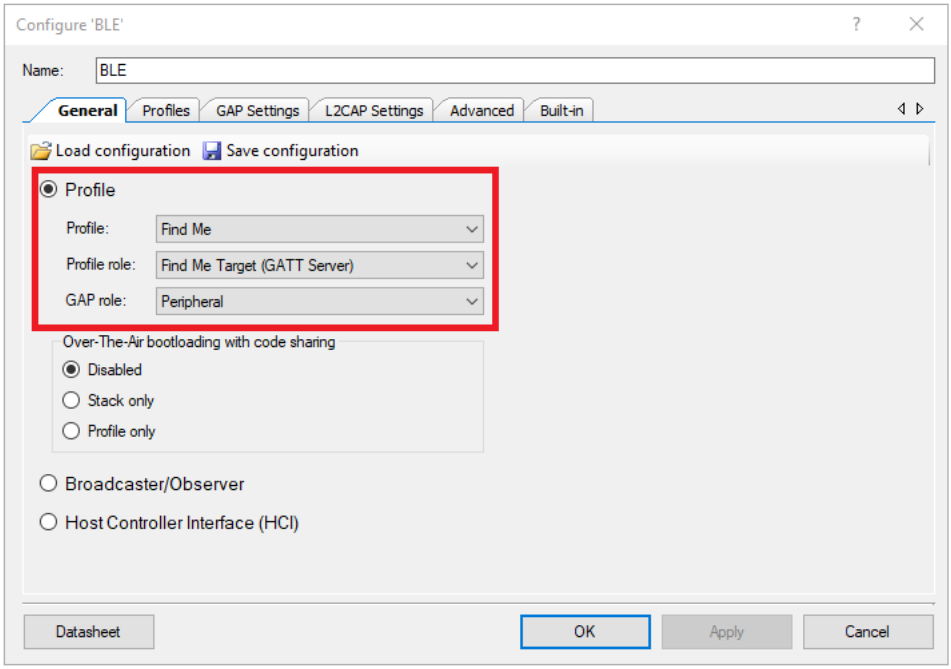


Figure 28 Bluetooth® Low Energy component General configuration

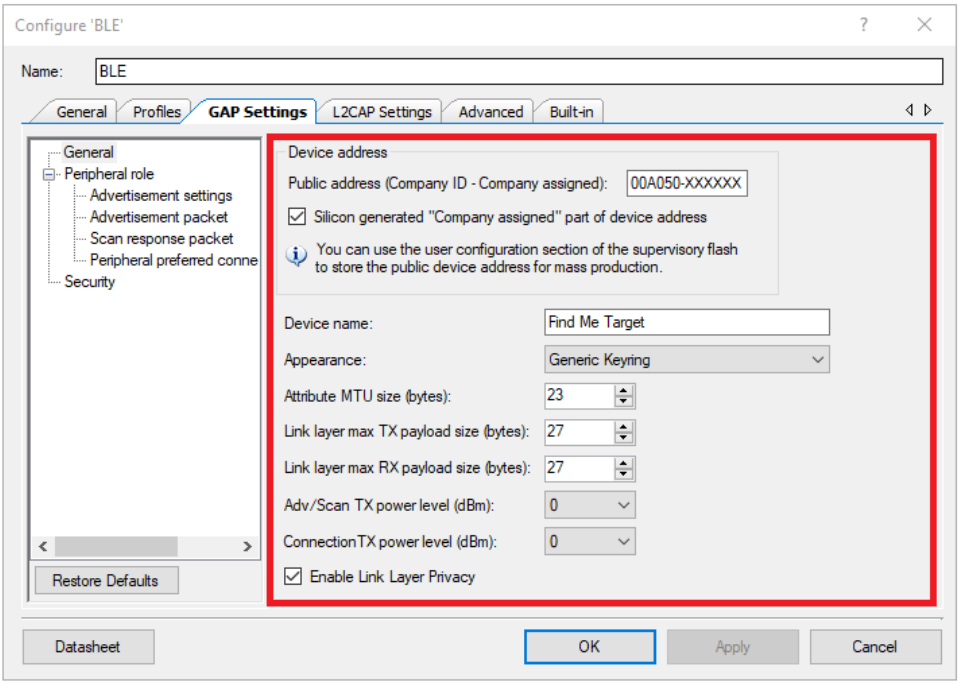


Figure 29 Bluetooth® Low Energy component GAP general settings

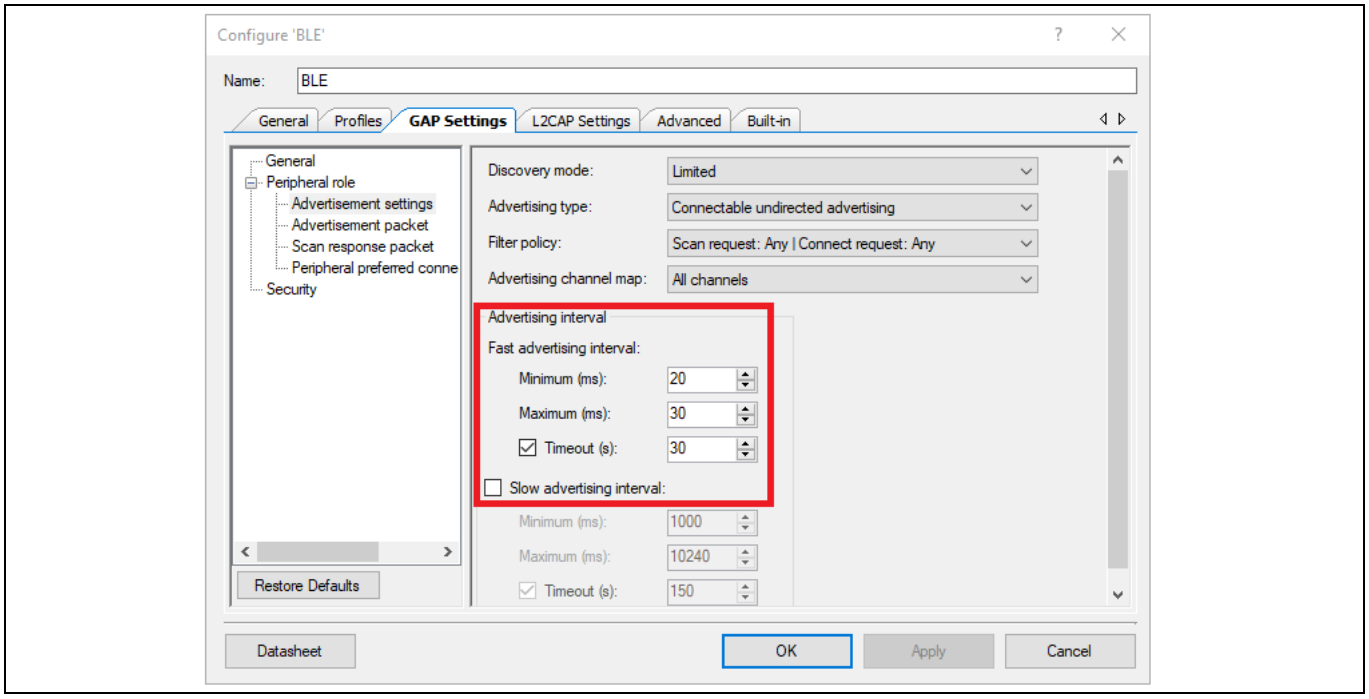


Figure 30 Bluetooth® Low Energy component GAP advertisement settings

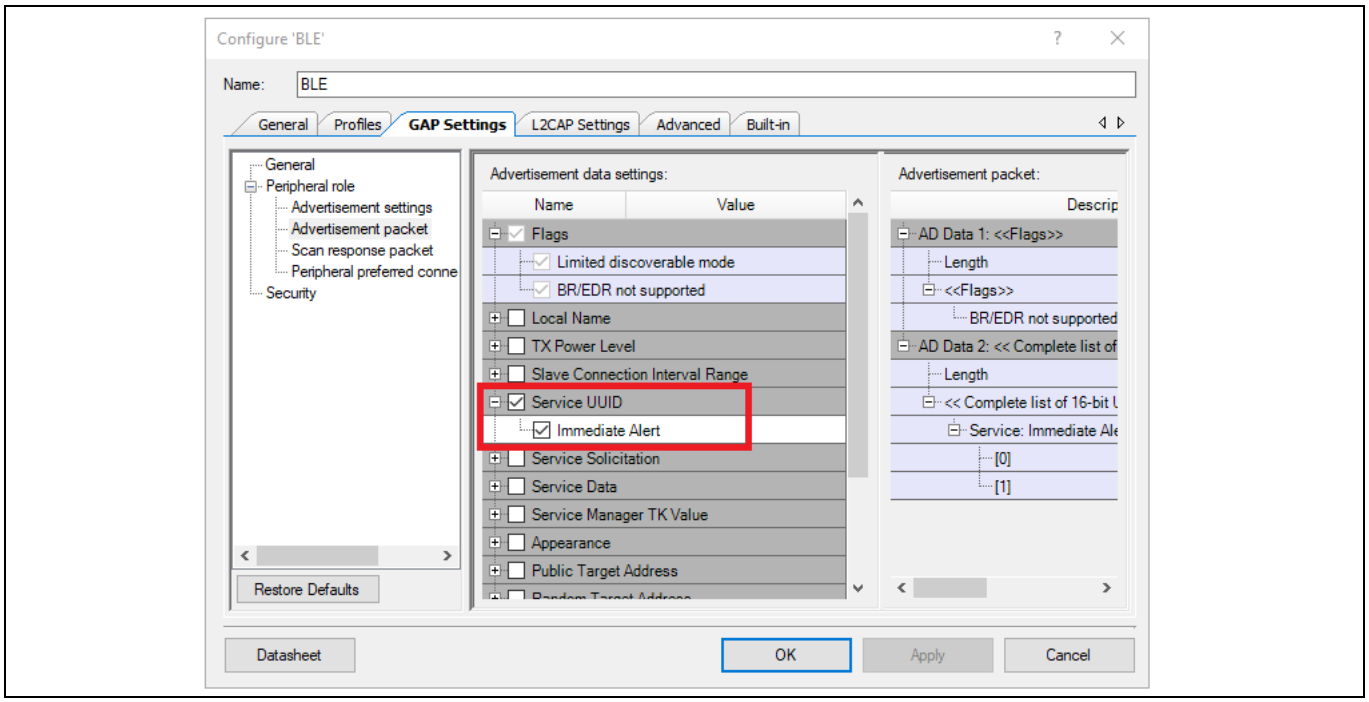


Figure 31 Bluetooth® Low Energy component GAP advertisement packet

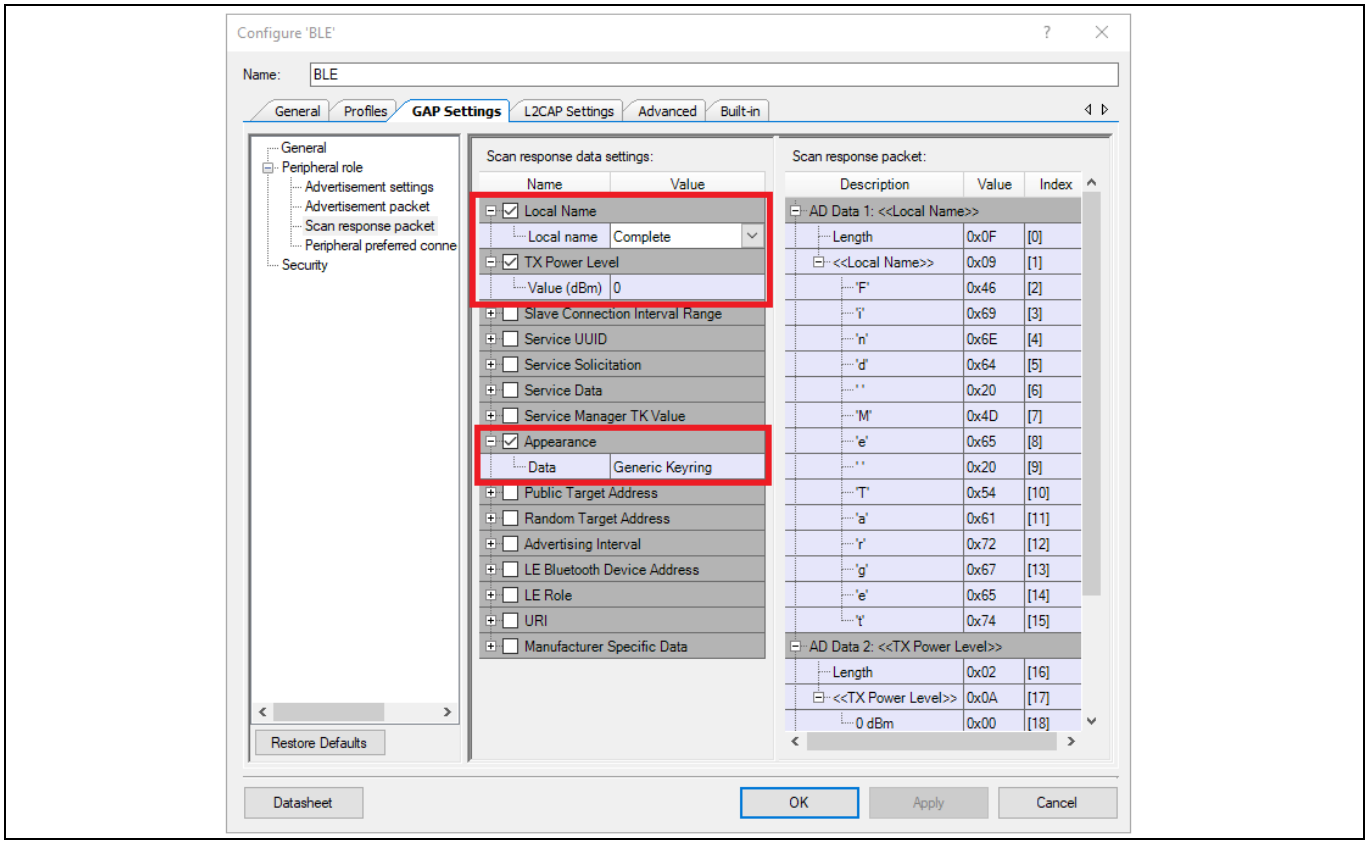


Figure 32 Bluetooth® Low Energy component GAP Scan response packet

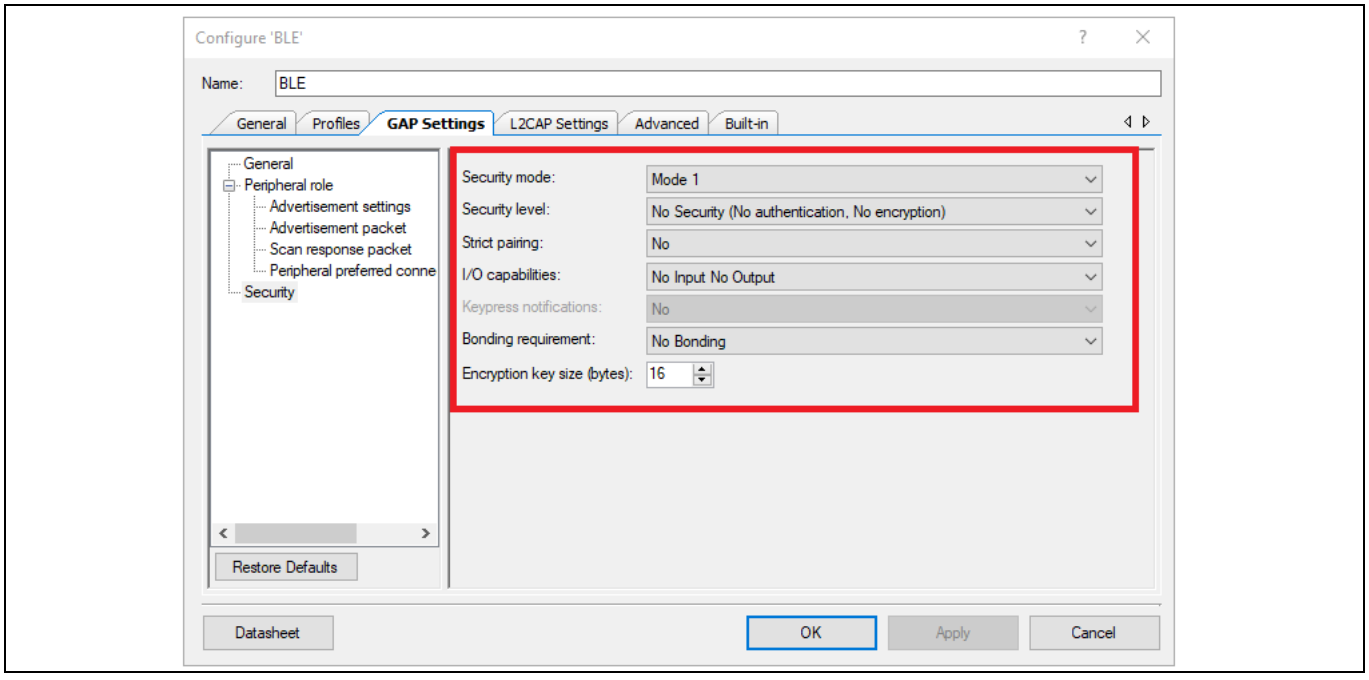


Figure 33 Bluetooth® Low Energy component GAP security settings

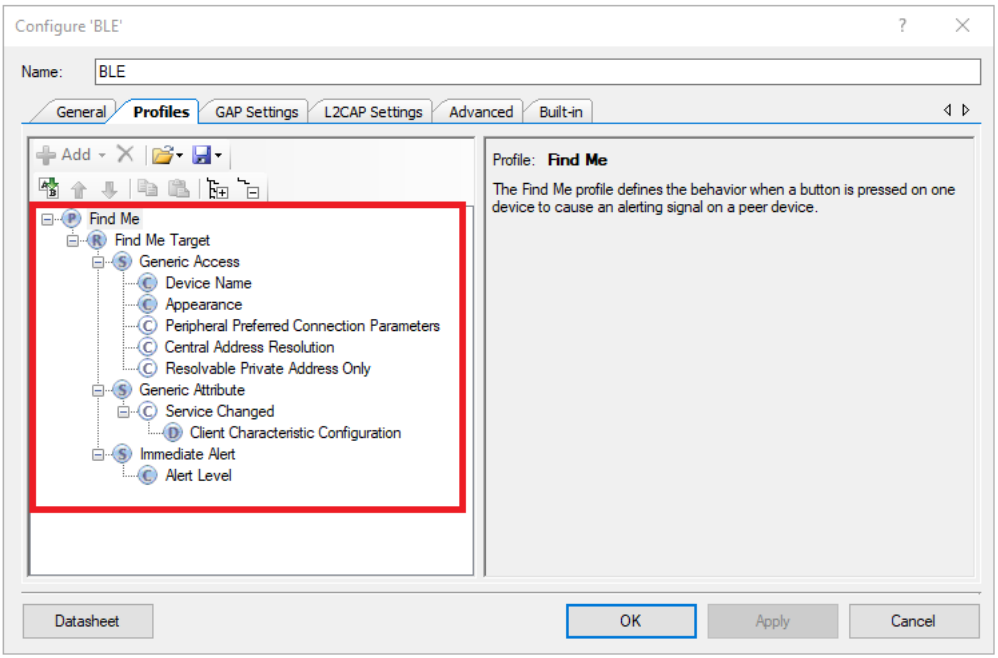


Figure 34 Bluetooth® Low Energy component Profiles configuration

10. Drag and drop a Digital Output Pin Component as shown in **Figure 35**.

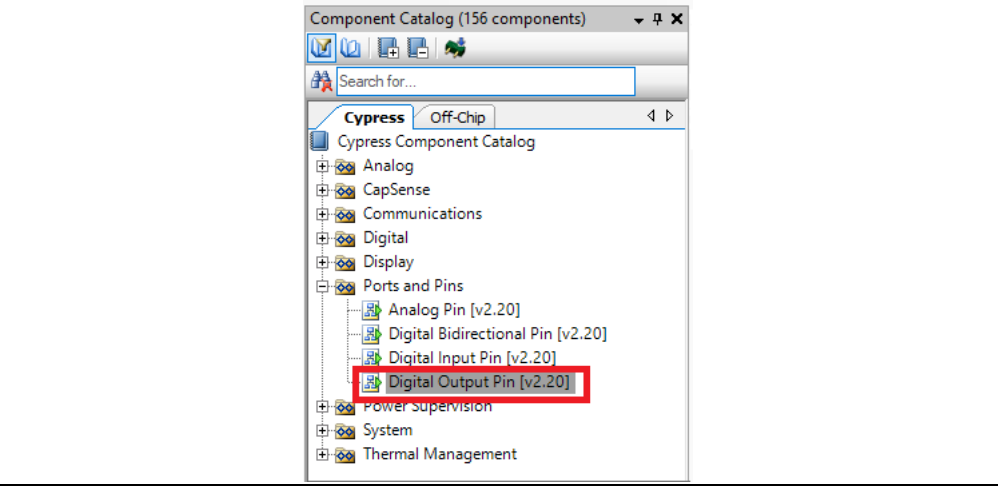


Figure 35 Location of the Digital Output Pin component

11. Double-click the Component and make the following changes as shown in **Figure 36**:

- Deselect **HW Connection**.
- Change the name to **Alert_LED**.
- Set **Initial drive state** to “High (1)”.

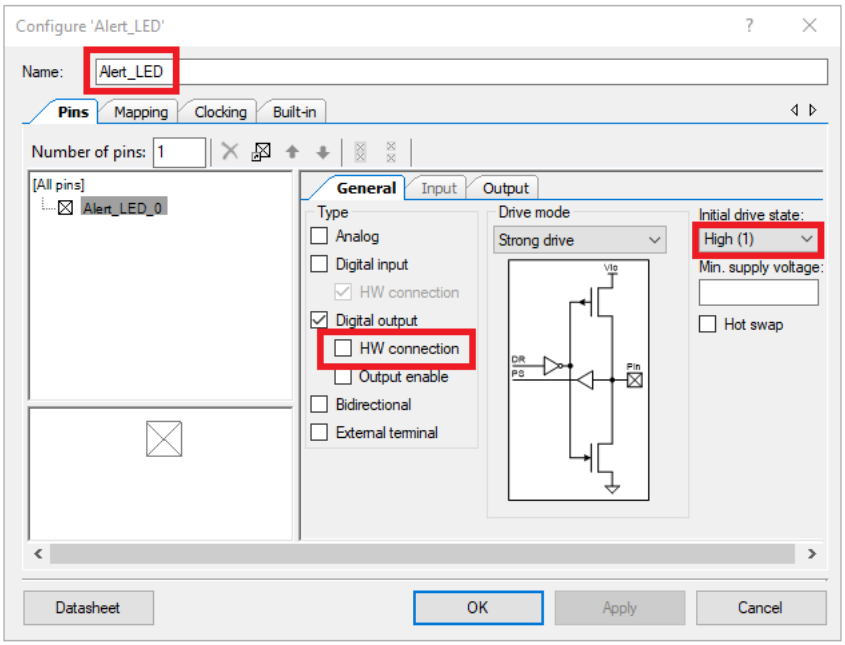


Figure 36 Renaming a pin component

12. Add two more Digital Output Pin Components. Double-click the Components and make the following changes as **Figure 37** and **Figure 38** show:

- Rename the Components as Advertising_LED and Disconnect_LED.
- Deselect **HW Connection.**
- Set **Initial drive state** to “High (1)”.

These pins will be used to drive the Bluetooth® LE advertising and disconnection state indicator LEDs. The LEDs on the Bluetooth® Low Energy Pioneer Kit are active LOW; that is, the high pin-drive state turns off the LEDs and the low pin-drive state turns them on.

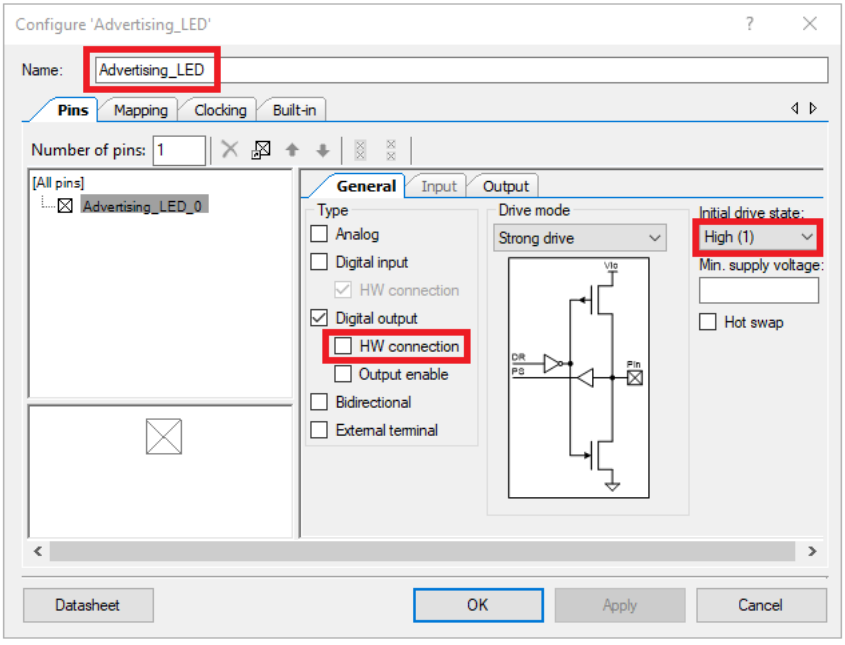


Figure 37 Advertising LED configuration

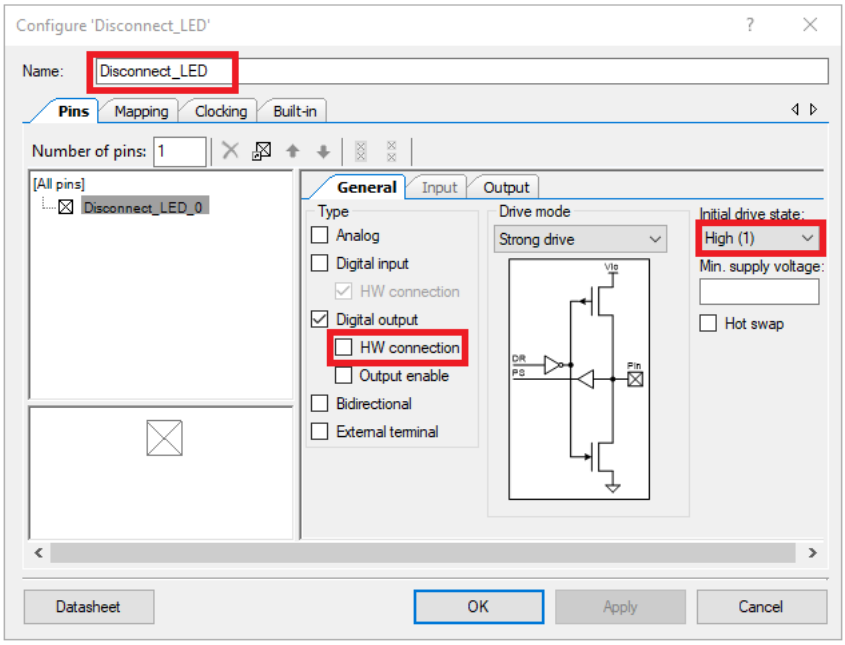


Figure 38 Disconnection LED configuration

13. After completing the schematic configuration, your design should look similar to **Figure 39**.

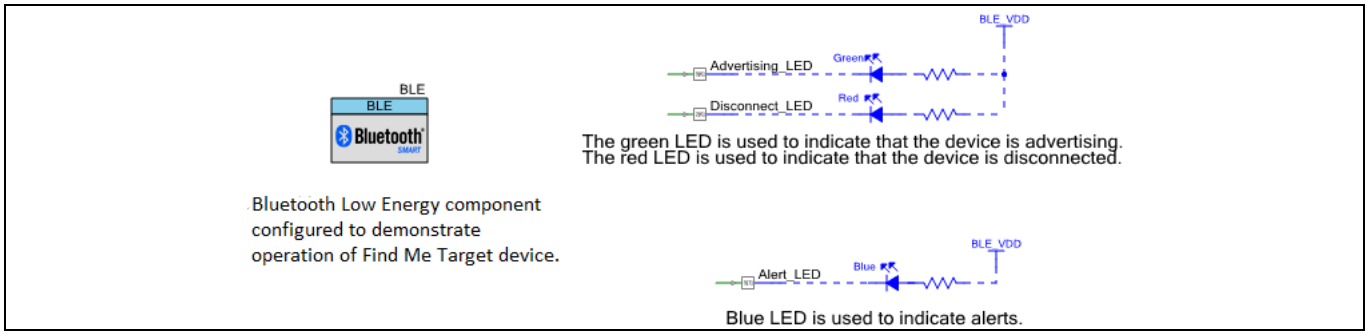


Figure 39 Schematic configuration

*Note: The blue dotted lines, the LED symbols, and resistor symbols shown in **Figure 39** are off-chip PSoC™ Creator Components that are present only for descriptive purposes and are not required for the functioning of your design. You can add off-chip Components to your design by dragging and dropping the required off-chip Components on to your project schematic page from PSoC™ Creator's off-chip Component Catalog.*

14. Under Design-Wide Resources (*BLE_FindMe.cydwr)* file from **Workspace Explorer** double click on the **Pins** section. You can use this section to select the device pins for the outputs (Advertising_LED, Disconnect_LED, and Alert_LED). **Figure 40** shows the pin configuration to connect the Advertising_LED, Disconnect LED, and Alert_LED pins to the green, red, and blue LEDs on the Bluetooth® Low Energy Pioneer Kit respectively.

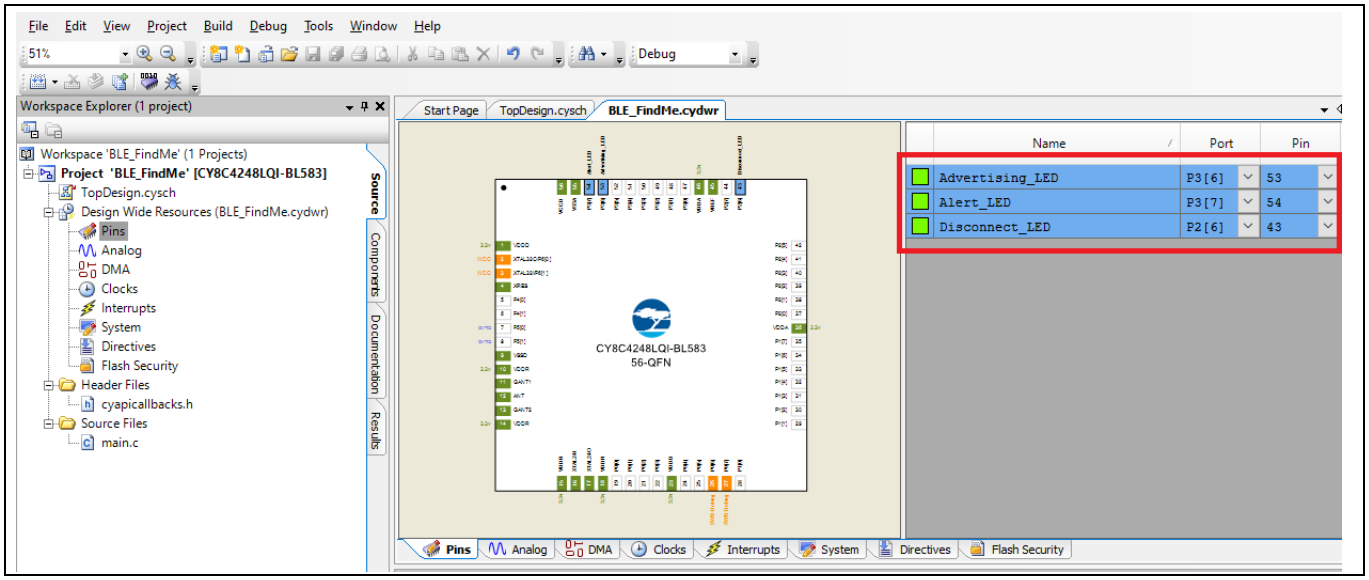


Figure 40 Pin selection

If you are using your own board or a development kit with no LEDs, select the appropriate pins. You can connect external LEDs to the selected pins, as **Figure 21** shows.

15. Similarly, in the **Clocks** section under the Design Wide Resources (*BLE_FindMe.cydwr)* file, click on the **Edit Clock** button. A window to configure the clocks opens up. Go to **High Frequency Clocks** tab and modify IMO to configure the internal main oscillator (IMO) to 12 MHz, as **Figure 41** shows. Click **OK**.

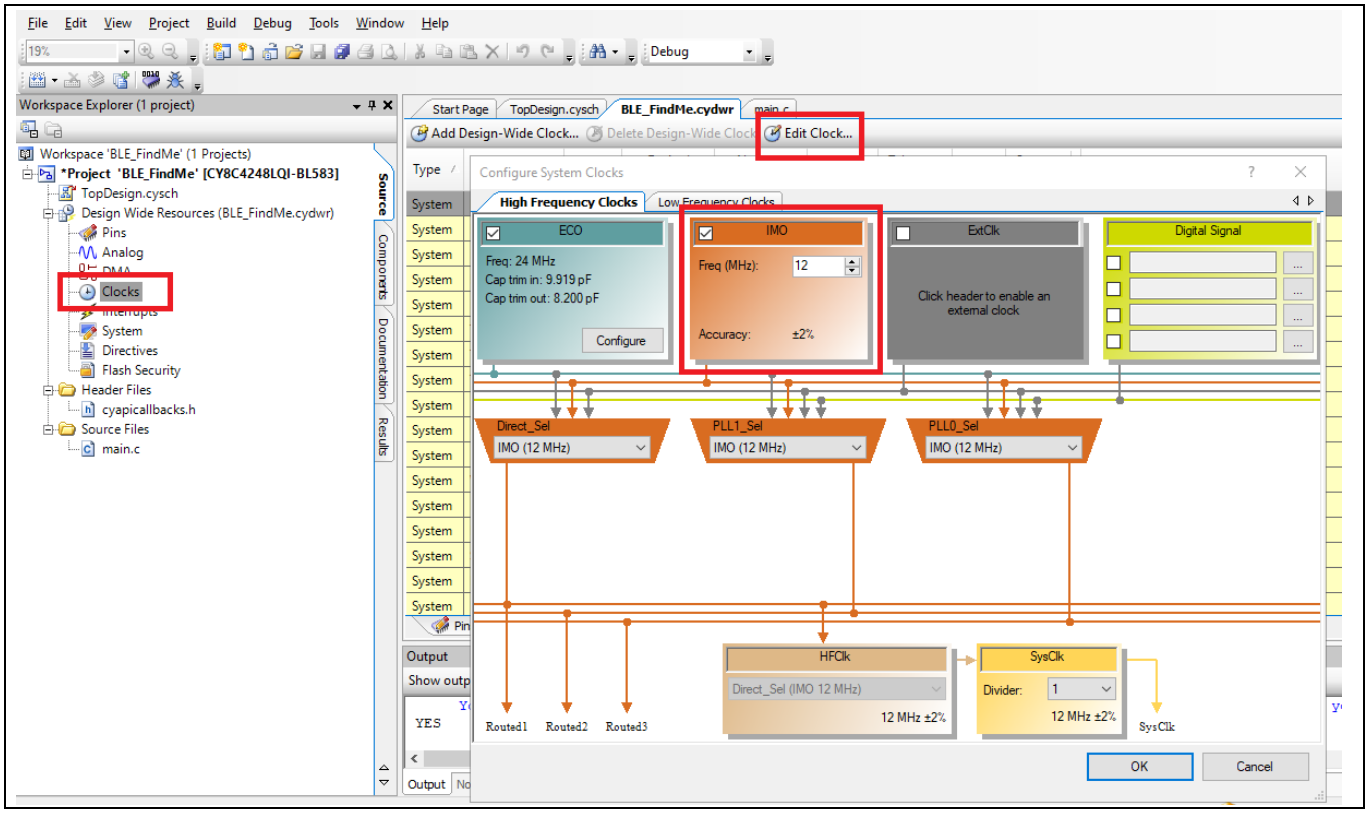


Figure 41 Clock configuration

16. Select **Generate Application** from the **Build** menu. Notice in the **Workspace Explorer** window that PSoC™ Creator automatically generates source code files for the Bluetooth® Low Energy and Digital Output Pin Components, as **Figure 42** shows. Clear the contents of generated main.c file before proceeding to next section.

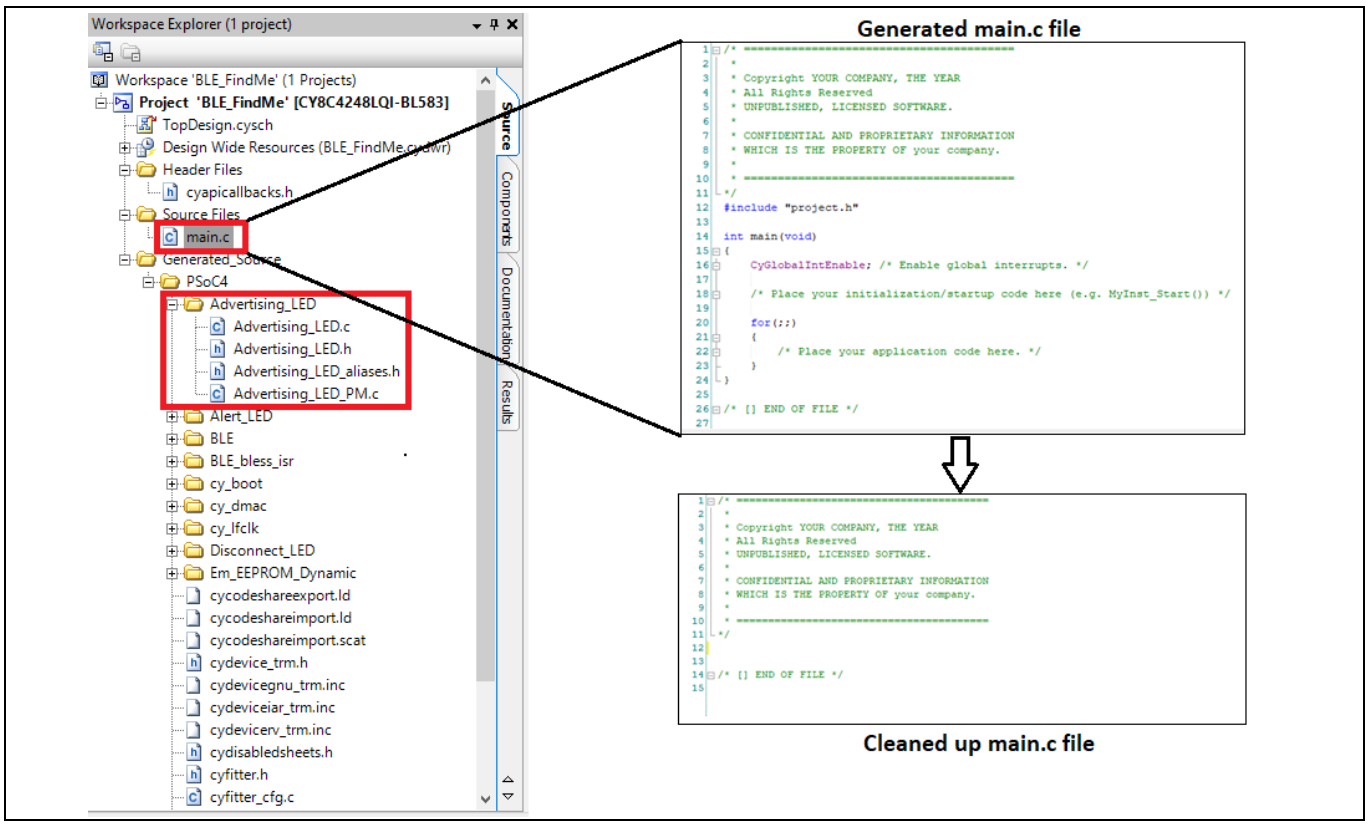


Figure 42 Generated source files

7.4 Stage 2: Write the firmware

Four main firmware blocks are required for designing Bluetooth® LE standard Profile applications using PSoC™ Creator:

- System initialization
- Bluetooth® LE stack event handler
- Bluetooth® LE service-specific event handler
- Main loop and low-power implementation

The following sections discuss these blocks with respect to the design that you configured in **Stage 1: Create the design**. The code snippets provided in this section are to be placed inside the main.c file shown in **Figure 42**.

7.4.1 System initialization

When the PSoC™ 4 with AIROC™ Bluetooth® LE device is reset, the firmware first performs the system initialization, which includes enabling global interrupts and enabling other Components used in the design. After the system is initialized, the firmware initializes the Bluetooth® Low Energy Component, which internally initializes the complete Bluetooth® LE subsystem.

As a part of the Bluetooth® Low Energy Component initialization, you must pass the event-handler function, which will be called by the Bluetooth® LE stack to notify pending events. The Bluetooth® LE stack event handler shown in **Code Listing 2** is registered as a part of the Bluetooth® LE initialization. If the Bluetooth® Low Energy Component initializes successfully, the firmware registers another event handler for the events specific to the Immediate Alert Service (IAS) and switches control to the main loop. **Figure 43** and **Code Listing 1** show the flow chart and the firmware source code for system initialization.

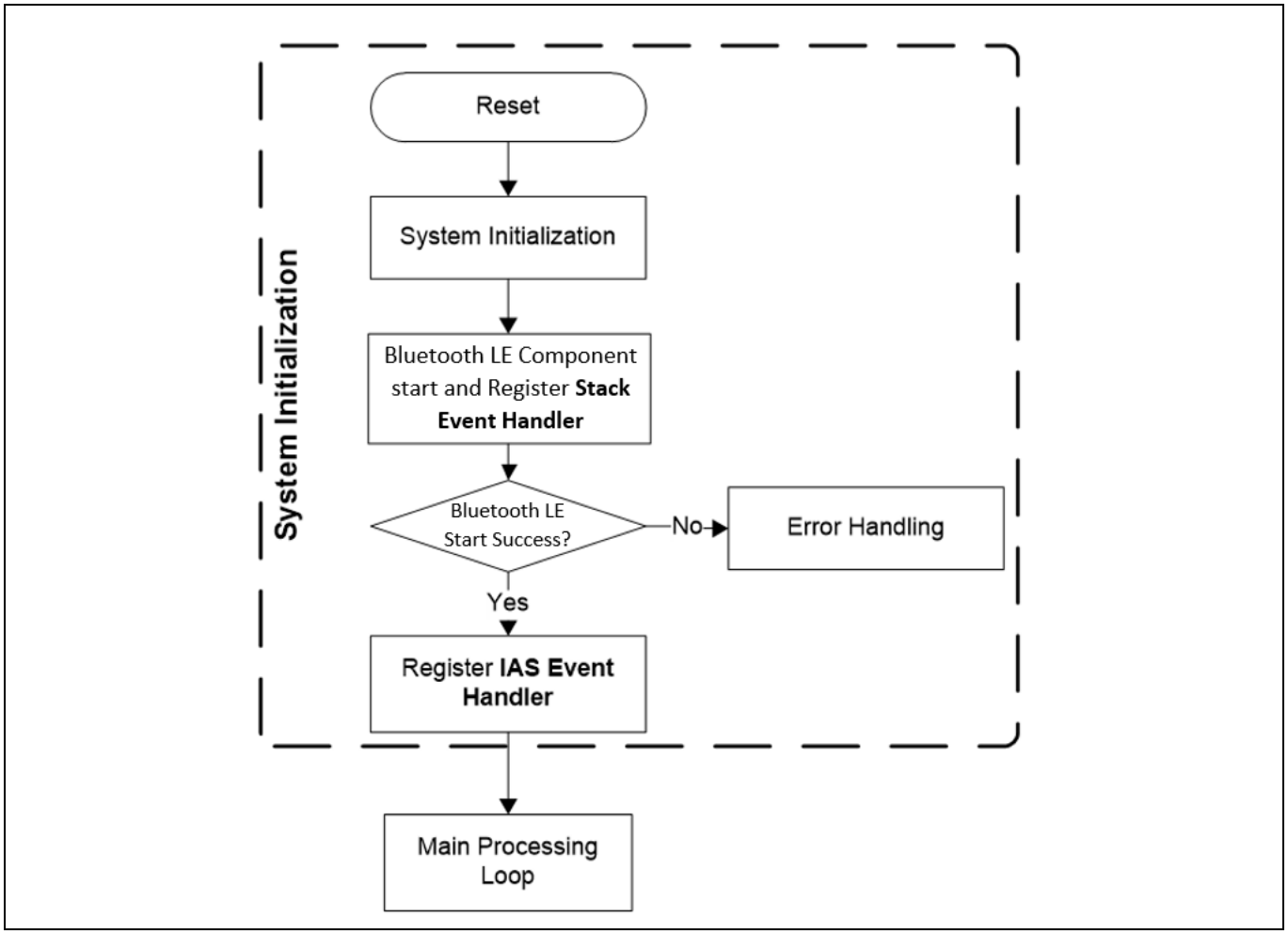


Figure 43 System initialization flow chart

Code Listing 1 System initialization firmware

```
#include <project.h>

#define LED_ON                  (0u)
#define LED_OFF                 (1u)

#define NO_ALERT                (0u)
#define MILD_ALERT              (1u)
#define HIGH_ALERT              (2u)
```

```
#define LED_TOGGLE_TIMEOUT      (100u)

void StackEventHandler(uint32 event, void *eventParam);
void IasEventHandler(uint32 event, void *eventParam);

uint8 alertLevel;

int main()
{
  CYBLE_API_RESULT_T apiResult;

  CyGlobalIntEnable;

  apiResult = CyBle_Start(StackEventHandler);

  if(apiResult != CYBLE_ERROR_OK)
  {
    /* BLE stack initialization failed, check your configuration */
    CYASSERT(0);
  }

  CyBle_IasRegisterAttrCallback(IasEventHandler);

  /* Place the main application loop here */
}
```

7.4.2 Bluetooth® LE stack event handler

The Bluetooth® LE stack within the Bluetooth® Low Energy Component generates events to provide the Bluetooth® LE interface status and data to the application firmware through the Bluetooth® LE stack event handler registered by you. The event handler must handle a few basic events from the stack, such as device connection and stack on, and configure the stack accordingly to establish and maintain the Bluetooth® LE link. For the Find Me Target application that you are creating, the Bluetooth® LE stack event handler must process all the events described in **Table 1**. The flow chart and the firmware for handling Bluetooth® LE stack events are shown in **Figure 44** and **Code Listing 2**.

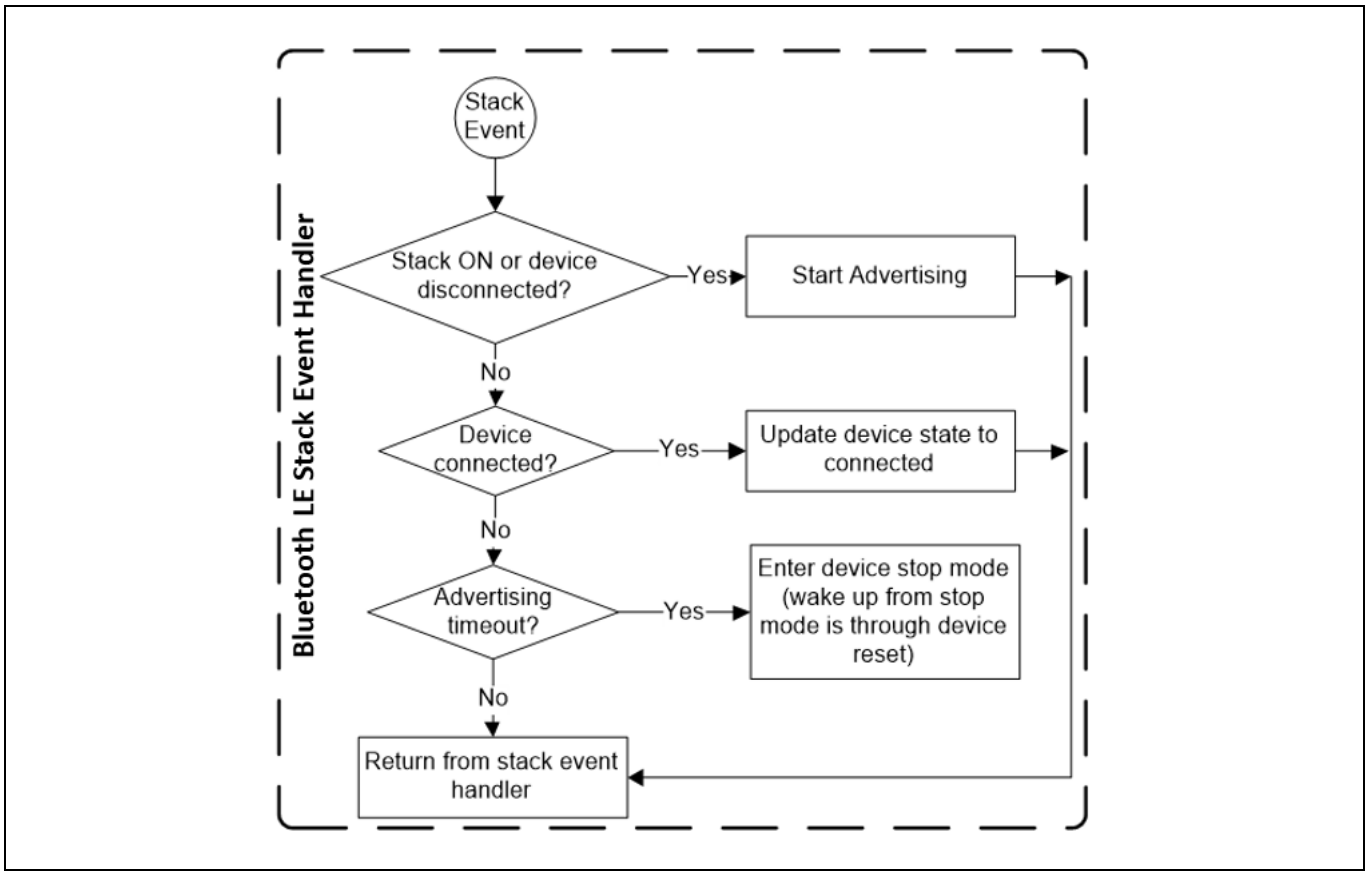


Table 1 Bluetooth® LE stack events

Bluetooth® LE stack event name	Event description	Event handler action
CYBLE_EVT_STACK_ON	Bluetooth® LE stack initialization completed successfully.	Start advertisement and reflect the advertisement state on the LED.
CYBLE_EVT_GAP_DEVICE_DISCONNECTED	Bluetooth® LE link with the peer device is disconnected.	Restart advertisement and reflect the advertisement state on the LED.
CYBLE_EVT_GAP_DEVICE_CONNECTED	Bluetooth® LE link with the peer device is established.	Update the Bluetooth® LE link state on the LED.
CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP	Bluetooth® LE stack advertisement start/stop event.	Configure the device in Stop mode if the advertisement has timed out.

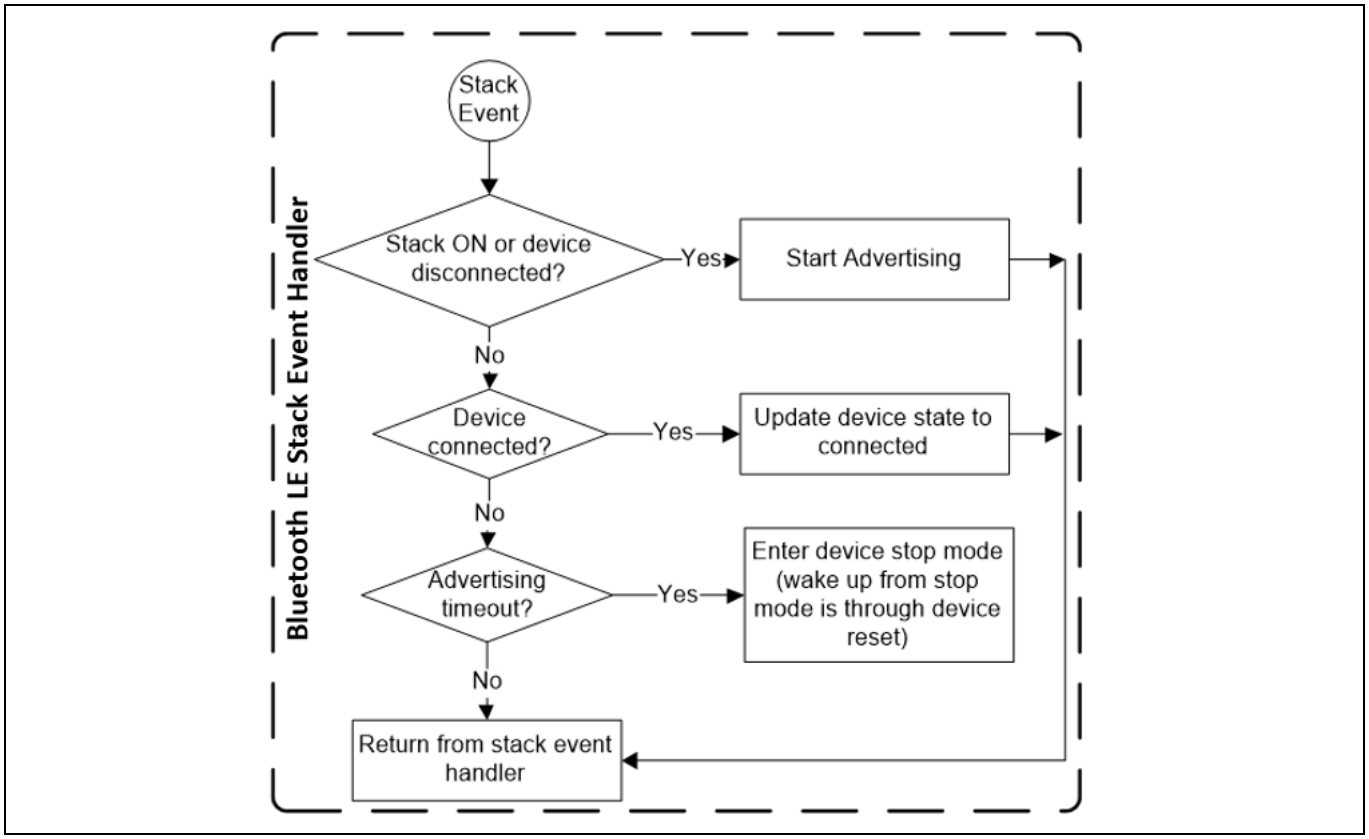


Figure 44 Bluetooth® LE Stack event handler flow chart

Code Listing 2 Bluetooth® LE Stack event handler firmware

```
void StackEventHandler(uint32 event, void *eventParam)
{
    switch(event)
    {
        /* Mandatory events to be handled by Find Me Target design */
        case CYBLE_EVT_STACK_ON:
        case CYBLE_EVT_GAP_DEVICE_DISCONNECTED:
            /* Start BLE advertisement for 30 seconds and update link
             * status on LEDs */
            CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
            Advertising_LED_Write(LED_ON);
            alertLevel = NO_ALERT;
        break;

        case CYBLE_EVT_GAP_DEVICE_CONNECTED:
            /* BLE link is established */
```

```
            Advertising_LED_Write(LED_OFF);
            Disconnect_LED_Write(LED_OFF);
        break;

        case CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP:
            if(CyBle_GetState() == CYBLE_STATE_DISCONNECTED)
            {
                /* Advertisement event timed out, go to low power
                 * mode (Stop mode) and wait for device reset
                 * event to wake up the device again */
                Advertising_LED_Write(LED_OFF);
                Disconnect_LED_Write(LED_ON);
                CySysPmSetWakeupPolarity(CY_PM_STOP_WAKEUP_ACTIVE_HIGH);
                CySysPmStop();

                /* Code execution will not reach here */
            }
        break;

        default:
        break;
    }
}
```

7.4.3 Bluetooth® LE service-specific event handler

The Bluetooth® Low Energy Component also generates events corresponding to each of the Services supported by your design. For the Find Me Target application that you are creating, the Bluetooth® Low Energy Component will generate IAS events that will let the application know if the Alert Level Characteristic is updated with a new value. The flow chart and the firmware for handling Bluetooth® LE IAS events are shown in **Figure 43** and **Code Listing 3** respectively.

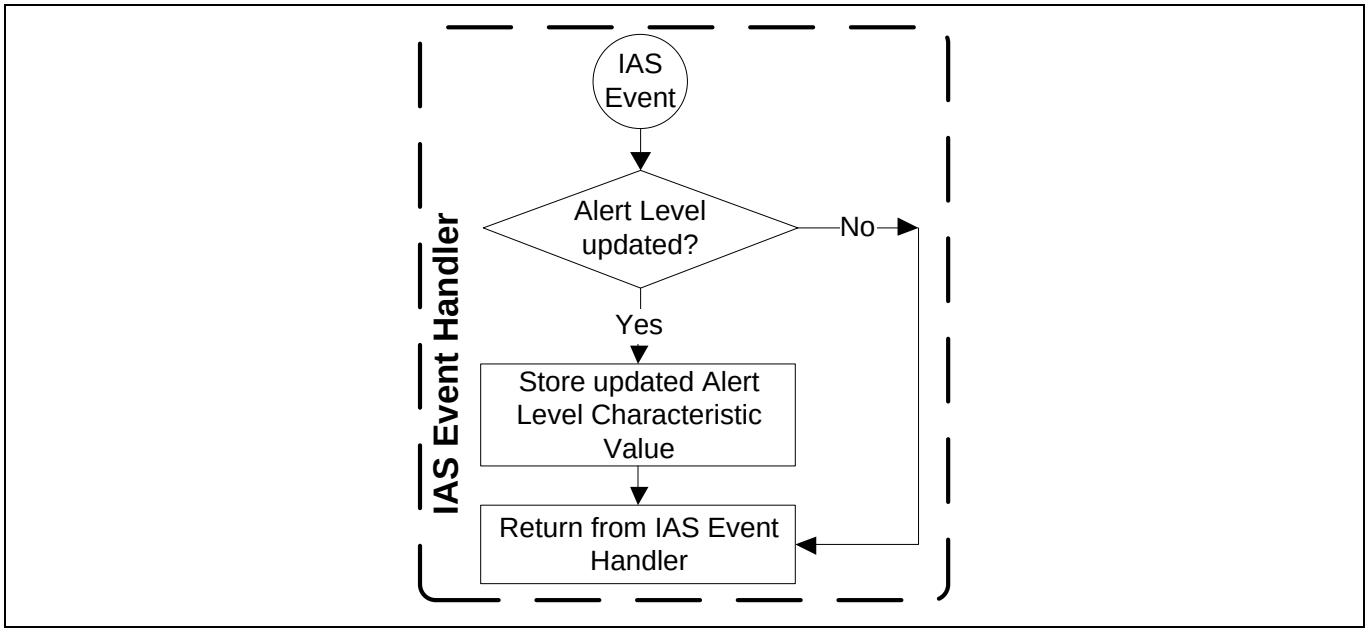


Figure 45 Bluetooth® LE IAS event handler flow chart

Code Listing 3 Bluetooth® LE IAS event handler firmware

```
void IasEventHandler(uint32 event, void *eventParam)
{
    /* Alert Level Characteristic write event */
    if(event == CYBLE_EVT_IASS_WRITE_CHAR_CMD)
    {
        /* Read the updated Alert Level value from the GATT database */
        CyBle_IassGetCharacteristicValue(CYBLE_IAS_ALERT_LEVEL,
            sizeof(alertLevel), &alertLevel);
    }
}
```

7.4.4 Main loop and low-power implementation

The main loop firmware in your design must periodically service the Bluetooth® LE stack-processing event, update the blue alert LED state per the IAS Alert Level Characteristic value and configure the **BLE subsystem (BLESS)** block and the PSoC™ 4 with AIROC™ Bluetooth® LE device into the low-power mode. The main loop flow chart and the firmware are shown in **Figure 46** and **Code Listing 4**. You should place the block of code in **Code Listing 4** inside the main function shown in **Code Listing 1** after the IasEventHandler function callback is registered.

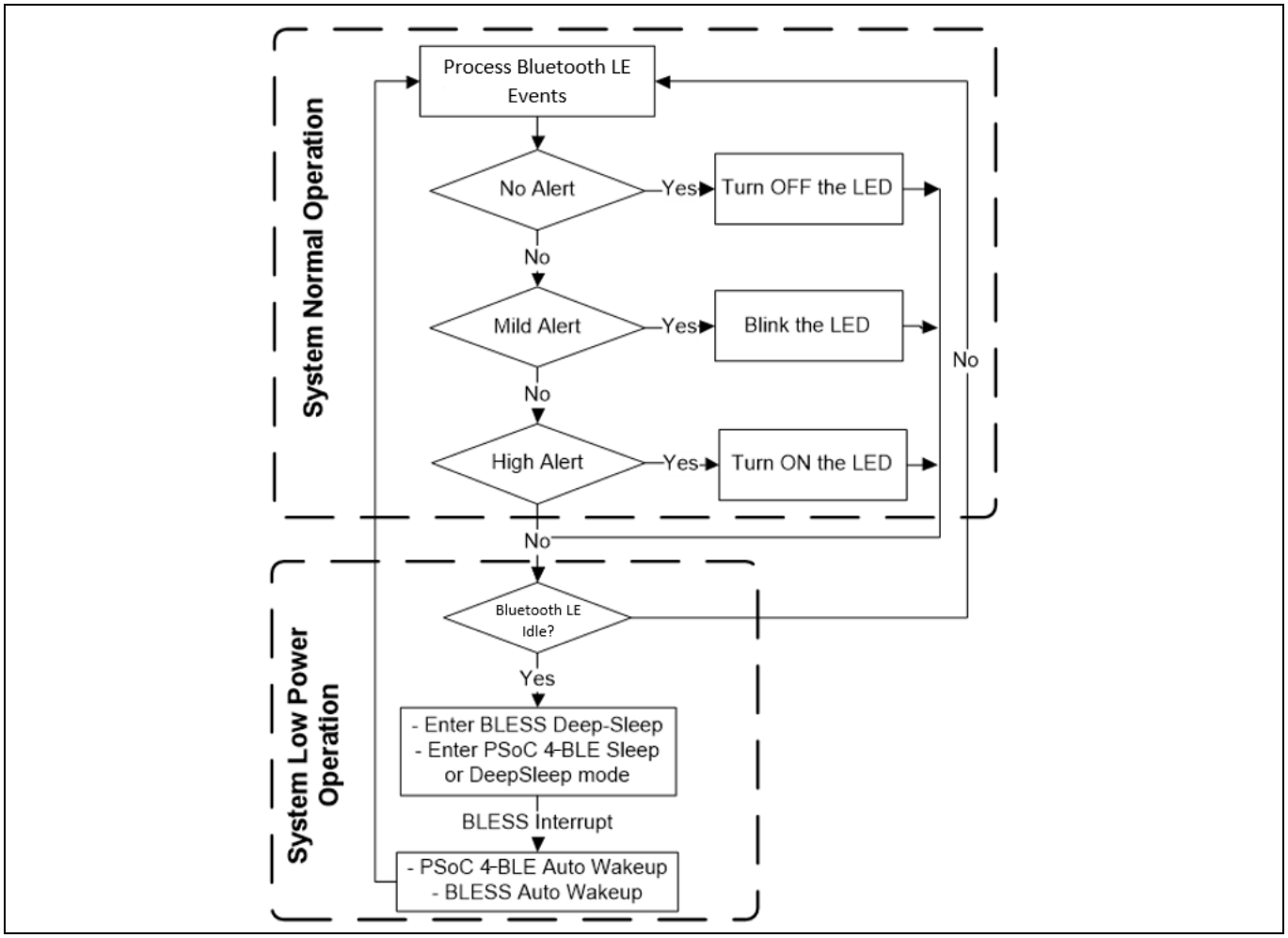


Figure 46 Firmware main loop flow chart

Code Listing 4 Main loop firmware

```
/* Place the main application loop here */


for(;;)
{
    static uint8 toggleTimeout = 0;
    CYBLE_BLESS_STATE_T blessState;
    uint8 intrStatus;


    /* Single API call to service all the BLE stack events. Must be
     * called at least once in a BLE connection interval */
                                        CyBle_ProcessEvents();
    /* Update Alert Level value on the blue LED */
    switch(alertLevel)
```

```
    {
        case NO_ALERT:
        Alert_LED_Write(LED_OFF);
        break;

        case MILD_ALERT:
        toggleTimeout++;
        if(toggleTimeout == LED_TOGGLE_TIMEOUT)
        {
            /* Toggle alert LED after timeout */
            Alert_LED_Write(Alert_LED_Read() ^ 0x01);
            toggleTimeout = 0;
        }
        break;

        case HIGH_ALERT:
        Alert_LED_Write(LED_ON);
        break;
    }

    /* Configure BLESS in Deep-Sleep mode */
    CyBle_EnterLPM(CYBLE_BLESS_DEEPSLEEP);

    /* Prevent interrupts while entering system low power modes */
    intrStatus = CyEnterCriticalSection();

    /* Get the current state of BLESS block */
    blessState = CyBle_GetBleSsState();

    /* If BLESS is in Deep-Sleep mode or the XTAL oscillator is turning on,
     * then PSoC™ 4 BLE can enter Deep-Sleep mode (1.3uA current consumption) */
    if(blessState == CYBLE_BLESS_STATE_ECO_ON ||
        blessState == CYBLE_BLESS_STATE_DEEPSLEEP)
    {
        CySysPmDeepSleep();
    }
```

```
    else if(blessState != CYBLE_BLESS_STATE_EVENT_CLOSE)
    {
        /* If BLESS is active, then configure PSoC™ 4 BLE system in
         * Sleep mode (~1.6mA current consumption) */
        CySysPmSleep();
    }
    else
    {
        /* Keep trying to enter either Sleep or Deep-Sleep mode */
    }
    CyExitCriticalSection(intrStatus);

    /* BLE link layer timing interrupt will wake up the system from Sleep
     * and Deep-Sleep modes */
}
```

7.5 Stage 3: Program the device

This section shows how to program the PSoC™ 4 with AIROC™ Bluetooth® LE device. If you are using a development kit with a built-in programmer (the Bluetooth® Low Energy Pioneer Kit, for example), connect the kit board to your computer using the USB cable. For other kits, refer to the kit guide. If you are developing on your own hardware, you need a hardware debugger, for example, a **CY8CKIT-002 MiniProg3**.

Note: *The source project for this design is in PSoC™ Creator 4.4 or later under **File > Code Example**, select **Filter by** as **Find Me > BLE_FindMe .***

1. In PSoC™ Creator, choose Debug > Select Debug Target, as **Figure 47** shows.

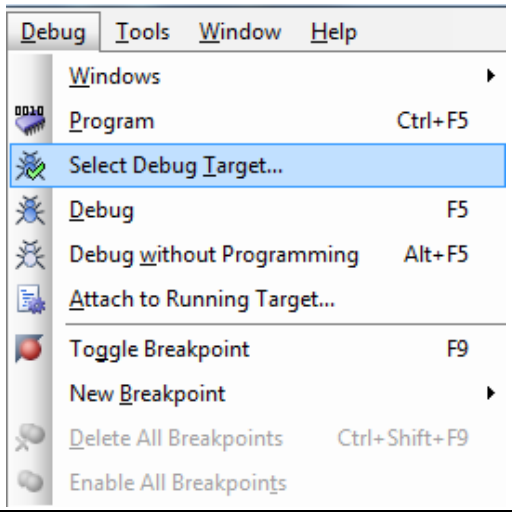


Figure 47 Selecting Debug Target

2. In the **Select Debug Target** dialog box, click **Port Acquire**, and then click **Connect**, as **Figure 48** shows. Click **OK** to close the dialog box.

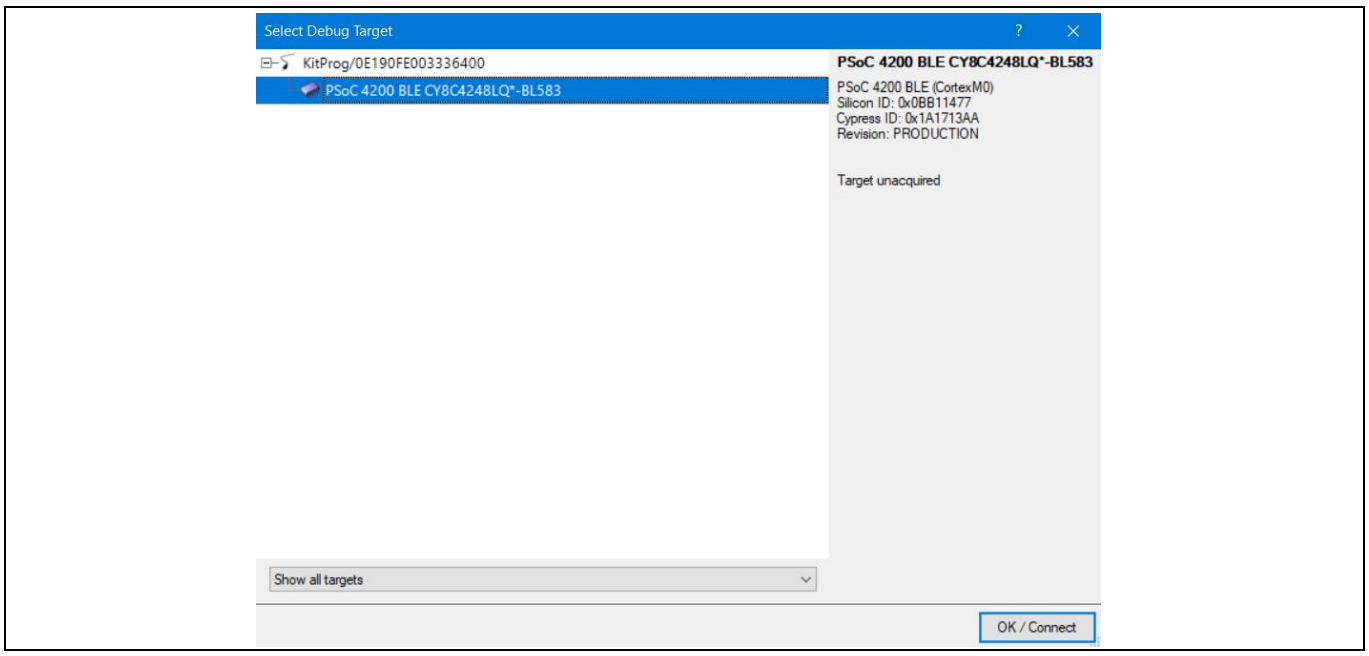


Figure 48 Connecting to a device

If you are using your own hardware, make sure the **Port Setting** configuration in the **Select Debug Target** window for your programming hardware is configured per your setup.

3. In PSoC™ Creator, choose **Debug** > **Program** to program the device with the project, as **Figure 49** shows.

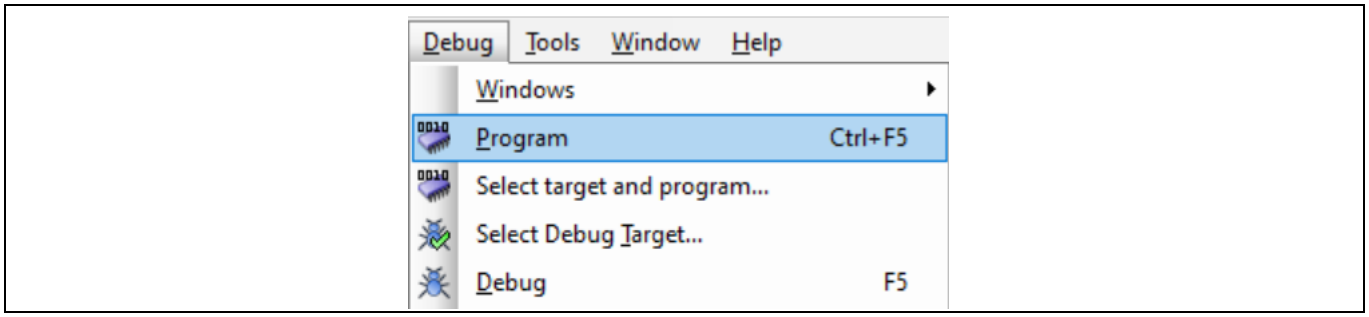


Figure 49 Programming the device

4. You can view the programming status on the PSoC™ Creator status bar (lower left corner of the window), as **Figure 50** shows.

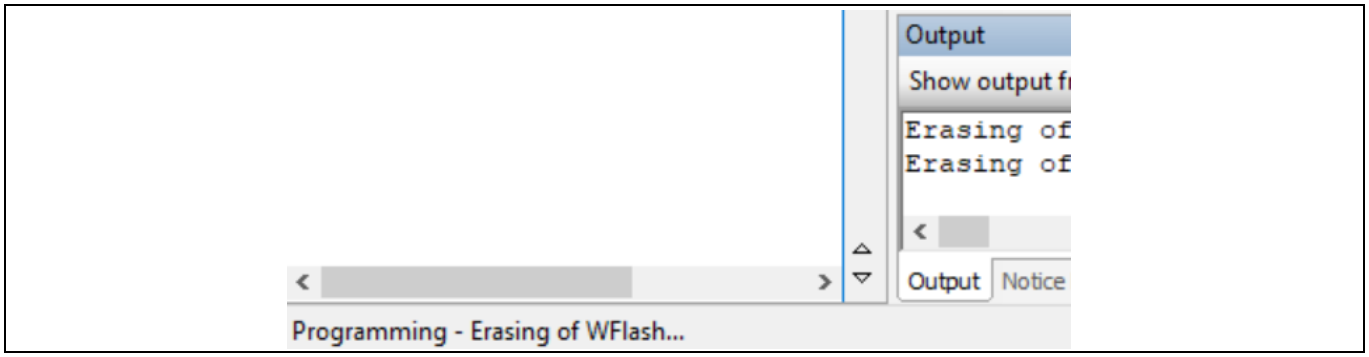


Figure 50 Programming status

7.6 Stage 4: Test your design

This section describes how to test your Bluetooth® LE design using the CySmart mobile apps and PC tool. The setup for testing your design using the Bluetooth® Low Energy Pioneer Kit is shown in **Figure 19**.

See the following appendixes for more information on these tools:

- **CySmart host emulation tool**
- **CySmart Mobile app**

7.6.1 With CySmart mobile app

1. Turn on Bluetooth® on your iOS or Android device.
2. Launch the CySmart app.
3. Press the reset switch on the Bluetooth® Low Energy Pioneer Kit to start Bluetooth® LE advertisements from your design.
4. Pull down the CySmart app home screen to start scanning for Bluetooth® LE Peripherals, your device will now appear in the CySmart app home screen. Select your device to establish a Bluetooth® LE connection.
5. Select the “Find Me” Profile from the carousel view.
6. Select one of the Alert Level values on the Find Me **Profile** screen and observe the state of the LED on your device change per your selection.

A step-by-step configuration screenshot of the CySmart mobile app is shown in **Figure 51** and **Figure 52**.

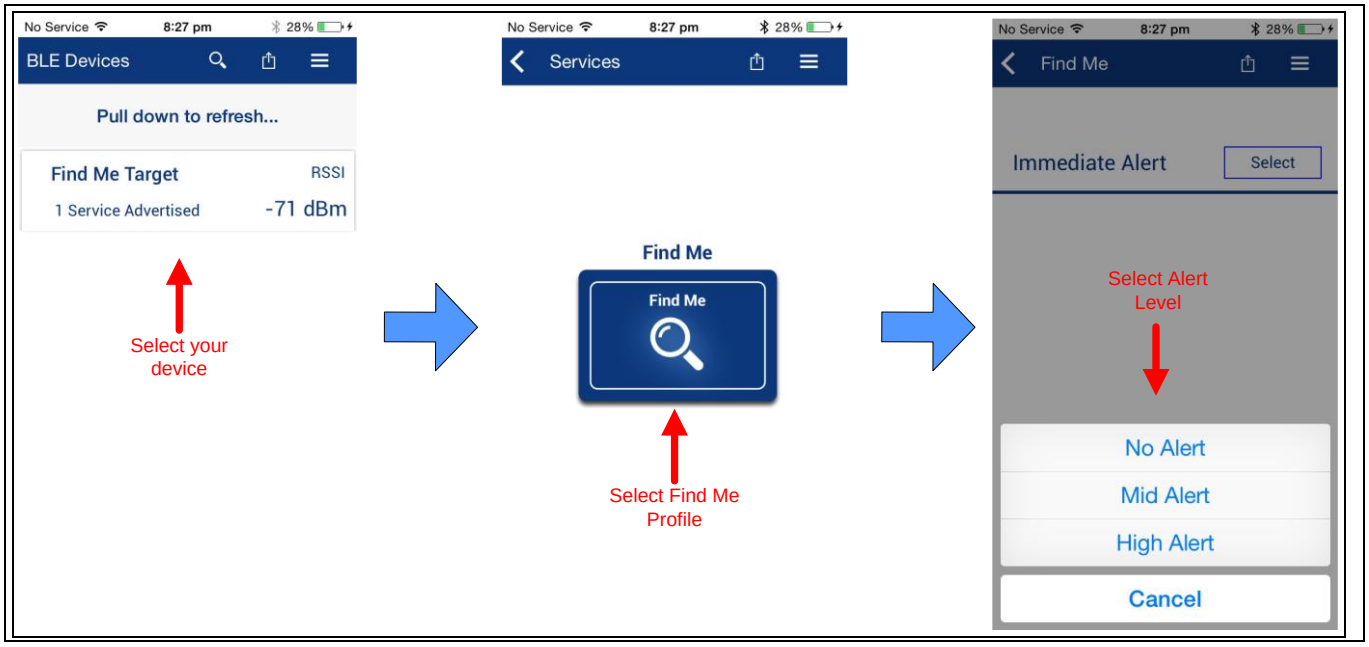


Figure 51 Testing with CySmart iOS app

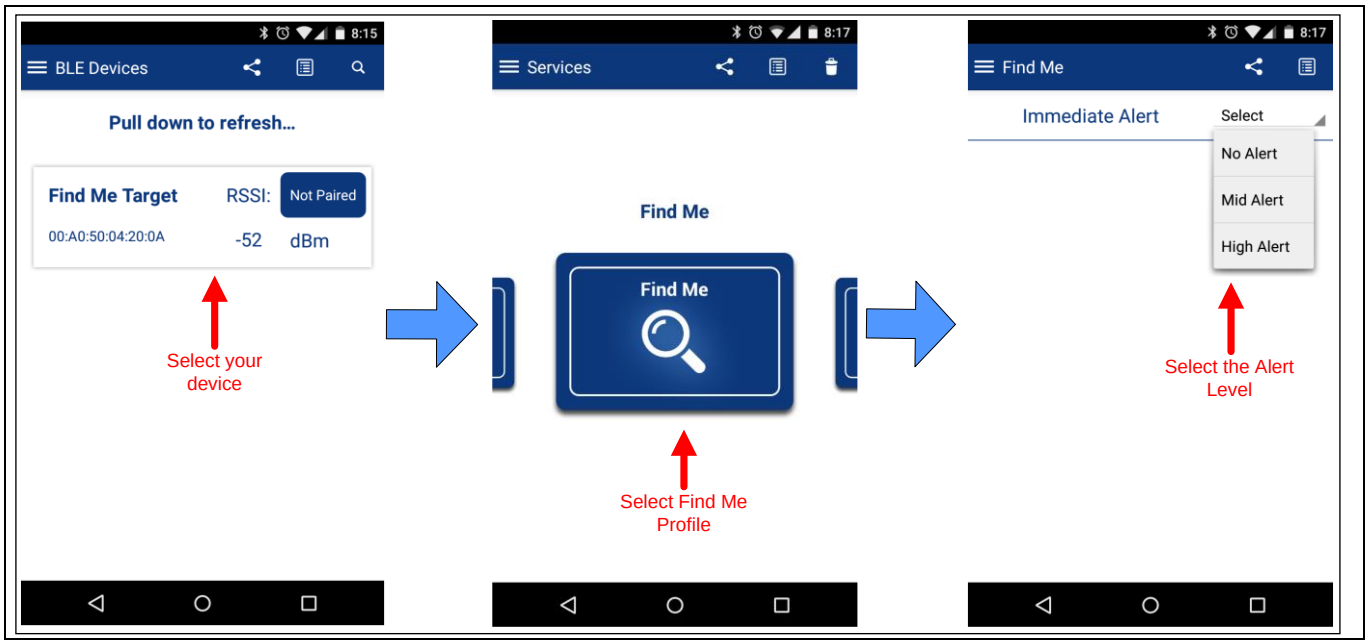


Figure 52 Testing with CySmart android app

7.6.2 With CySmart host emulation tool

Similar to the CySmart mobile app, you can also use the CySmart Host Emulation Tool to establish a Bluetooth® LE connection with your design and perform read or write operations on Bluetooth® LE Characteristics.

1. Connect the Bluetooth® Low Energy Dongle to your Windows machine. Wait for the driver installation to be completed.
2. Launch the CySmart Host Emulation Tool; it automatically detects the Bluetooth® Low Energy Dongle. Click **Refresh** if the Bluetooth® Low Energy Dongle does not appear in the **Select BLE Dongle Target** pop-up window. Click **Connect,** as shown in **Figure 53**.

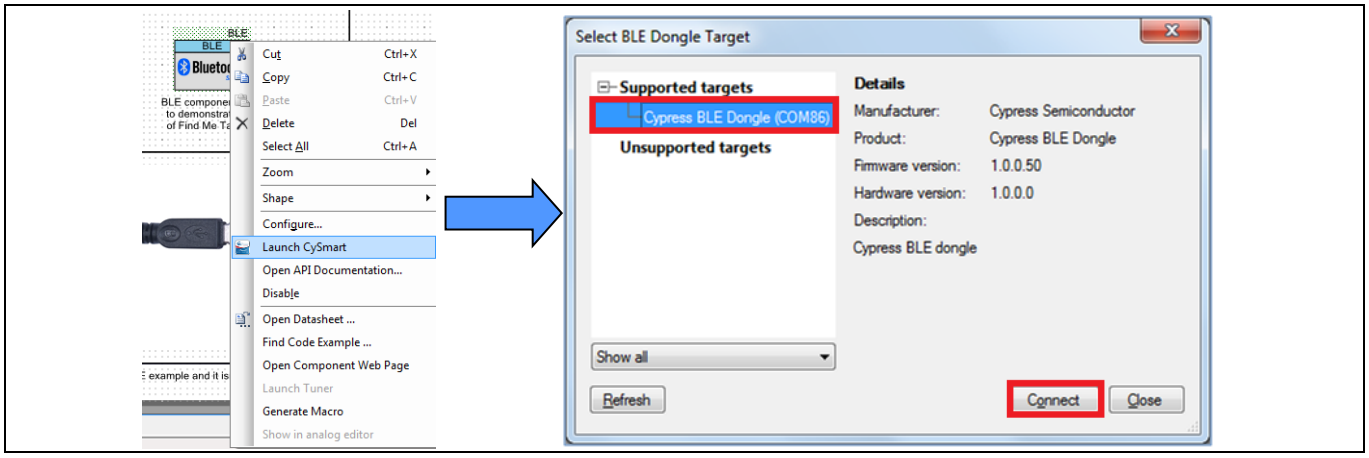


Figure 53 CySmart Bluetooth® Low Energy dongle selection

3. Select **Configure Master Settings** and restore the values to the default settings, as shown in **Figure 54**.

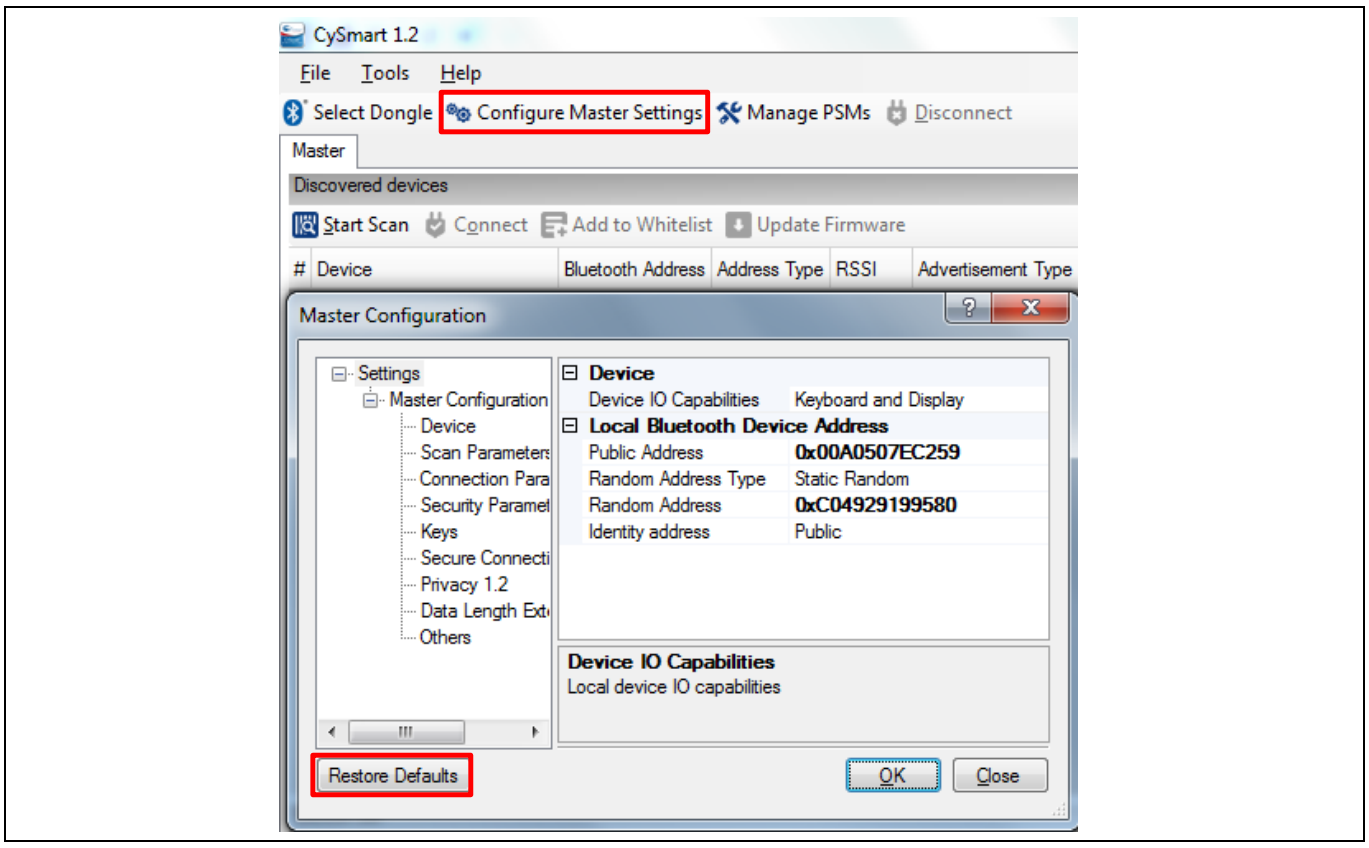


Figure 54 CySmart master settings configuration

4. Press the reset switch on the Bluetooth® Low Energy Pioneer Kit to start Bluetooth® LE advertisements from your design.
5. On the CySmart Host Emulation Tool, click **Start Scan**. Your device name should appear in the **Discovered devices** list, as shown in **Figure 55**.

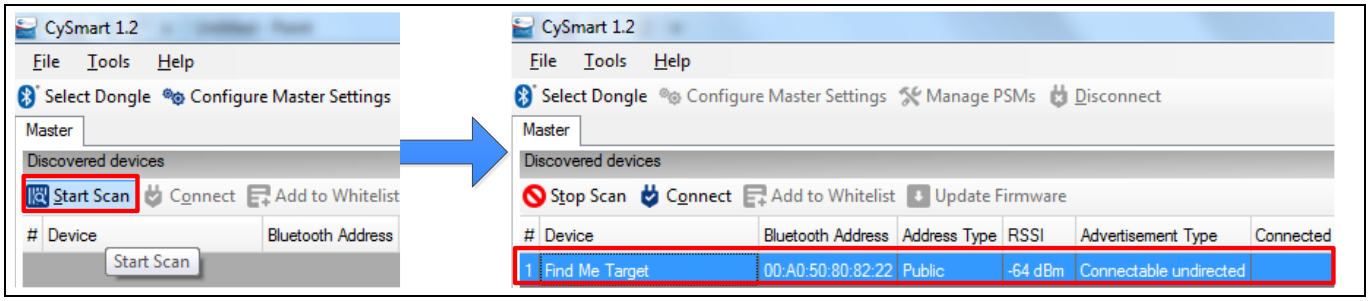


Figure 55 CySmart device discovery

6. Select your device and click **Connect** to establish a Bluetooth® LE connection between the CySmart Host Emulation Tool and your device, as shown in **Figure 56**.

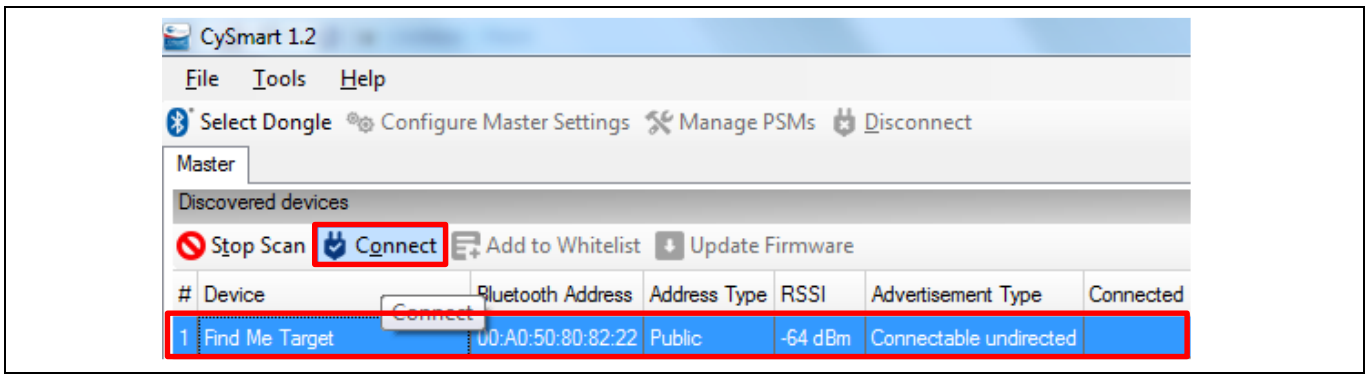


Figure 56 CySmart device connection

7. Once connected, discover all the Attributes on your design from the CySmart Host Emulation Tool, as shown in **Figure 57**.

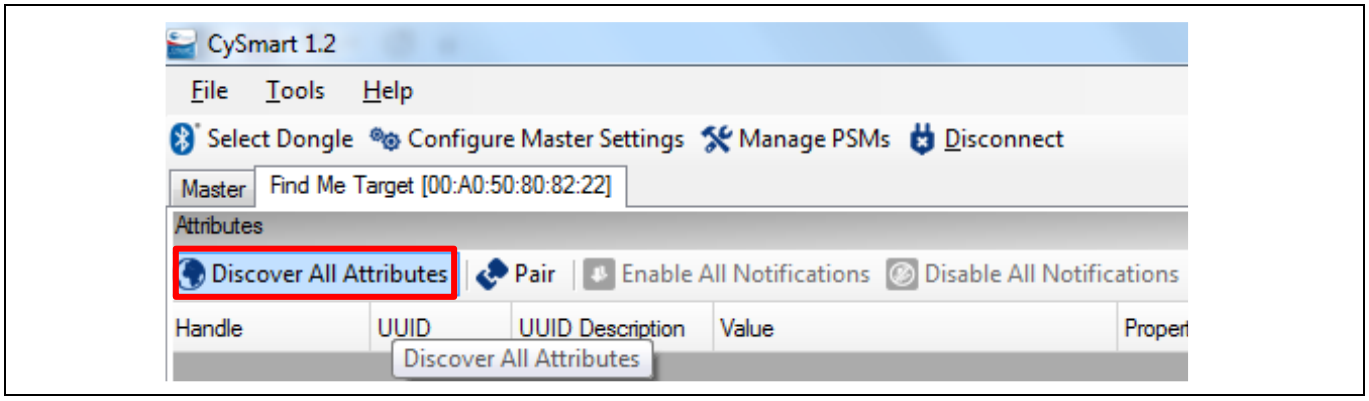


Figure 57 CySmart attribute discovery

8. Write a value of 0, 1, or 2 to the Alert Level Characteristic under the Immediate Alert Service, as **Figure 58** shows. Observe the state of the LED on your device change per your Alert Level Characteristic configuration.

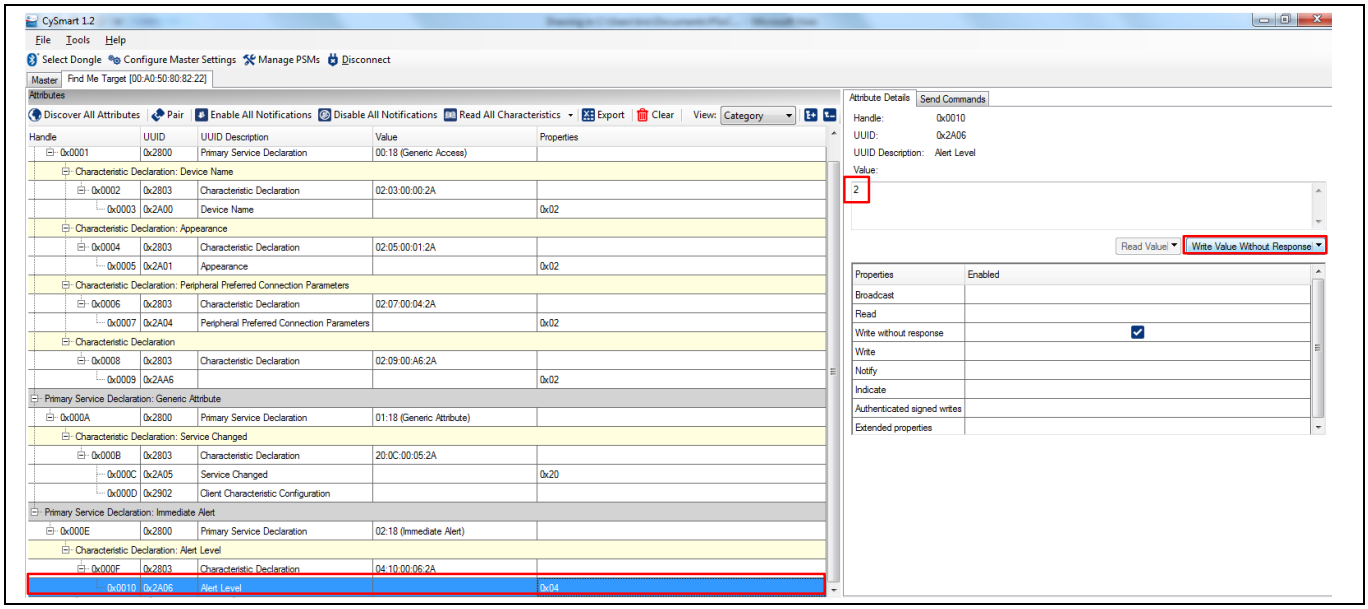


Figure 58 Testing with CySmart host emulation tool

8 Summary

This application note explored the basics of the Bluetooth® LE protocol and PSoC™ 4 with AIROC™ Bluetooth® LE device architecture and development tools. PSoC™ 4 with AIROC™ Bluetooth® LE is a truly programmable embedded system-on-chip, integrating Bluetooth® LE radio, configurable analog and digital peripheral functions, memory, and an Arm® Cortex®-M0 microcontroller on a single chip. Because of the integrated features and low-power modes, PSoC™ 4 with AIROC™ Bluetooth® LE is an ideal choice for battery-operated wearable, health, and fitness Bluetooth® LE applications.

This application note also guided you to a comprehensive collection of resources to accelerate in-depth learning about PSoC™ 4 with AIROC™ Bluetooth® LE.

9 Appendix A: Bluetooth® LE device family comparison

Table 2 summarizes the features and capabilities of the Bluetooth® LE device family from Infineon.

Table 2 Bluetooth® LE device families

Features	Device Family				
	CY8C41x7-BLxxx	CY8C42x7-BLxxx	CYBL10xxx***	CY8C41x8-BL	CY8C42x8-BL
Bluetooth® Low Energy Subsystem	Bluetooth® LE radio and link-layer hardware blocks with Bluetooth 4.1-compatible protocol stack	Bluetooth® LE radio and link-layer hardware blocks with Bluetooth 4.1-compatible protocol stack	Bluetooth® LE radio and link-layer hardware blocks with Bluetooth 4.2-compatible protocol stack	Bluetooth® LE radio and link-layer hardware blocks with Bluetooth 4.2-compatible protocol stack**	Bluetooth® LE radio and link-layer hardware blocks with Bluetooth 4.2-compatible protocol stack**
Bluetooth® 4.2 features	LE secure connection	LE secure connection	LE secure connection, link layer privacy, and link layer data length extension	LE secure connection, link layer privacy, and link layer data length extension**	LE secure connection, link layer privacy, and link layer data length extension**
CPU	24-MHz Arm® Cortex®-M0 CPU with single-cycle multiply	48-MHz Arm® Cortex®-M0 CPU with single-cycle multiply	48-MHz Arm® Cortex®-M0 CPU with single-cycle multiply	24-MHz Arm® Cortex®-M0 CPU with single-cycle multiply	48-MHz Arm® Cortex®-M0 CPU with single-cycle multiply
Flash Memory	128 KB	128 KB	256 KB	256 KB	256 KB
SRAM	16 KB	16 KB	32 KB	32 KB	32 KB
GPIOs	Up to 36	Up to 36	Up to 36	Up to 36	Up to 36
CapSense™	Up to 35 sensors	Up to 35 sensors	Up to 35 sensors	Up to 35 sensors	Up to 35 sensors
CapSense™ Gestures	On selected devices	On selected devices	On selected devices	On selected devices	On selected devices
ADC	12-bit, 806-ksps SAR ADC with sequencer	12-bit, 1-Msps SAR ADC with sequencer	12-bit, 1-Msps SAR ADC with sequencer	12-bit, 806-ksps SAR ADC with sequencer	12-bit, 1-Msps SAR ADC with sequencer
Opamps	2 programmable opamps that are active in Deep-Sleep mode	4 programmable opamps that are active in Deep-Sleep mode	None	2 programmable opamps that are active in Deep-Sleep mode	4 programmable opamps that are active in Deep-Sleep mode
Comparators	2 low-power comparators	2 low-power comparators	None	2 low-power comparators	2 low-power comparators

Features	Device Family				
	CY8C41x7-BLxxx	**CY8C42x7-BLxxx**	**CYBL10xxx*** **	**CY8C41x8-BL**	**CY8C42x8-BL**
	with the wakeup feature	with the wakeup feature		with the wakeup feature	with the wakeup feature
Current DACs	One 7-bit, and one 8-bit	One 7-bit, and one 8-bit	None	One 7-bit, and one 8-bit	One 7-bit, and one 8-bit
Power Supply Range	1.9 V to 5.5 V	1.9 V to 5.5 V	1.9 V to 5.5 V	1.9 V to 5.5 V	1.9 V to 5.5 V
Low-Power Modes	Deep-Sleep mode at 1.3 µA Hibernate mode at 150 nA Stop mode at 60 nA	Deep-Sleep mode at 1.3 µA Hibernate mode at 150 nA Stop mode at 60 nA	Deep-Sleep mode at 1.3 µA Hibernate mode at 150 nA Stop mode at 60 nA	Deep-Sleep mode at 1.3 µA Hibernate mode at 150 nA Stop mode at 60 nA	Deep-Sleep mode at 1.3 µA Hibernate mode at 150 nA Stop mode at 60 nA
Segment LCD Drive	4-COM, 32-segment LCD drive on select devices	4-COM, 32-segment LCD drive on select devices	4-COM, 32-segment LCD drive on select devices	4-COM, 32-segment LCD drive on select devices	4-COM, 32-segment LCD drive on select devices
Serial Communication	2 independent serial communication blocks (SCBs) with programmable I^2C, SPI, or UART	2 independent SCBs with programmable I^2C, SPI, or UART	2 independent SCBs with programmable I^2C, SPI, or UART	2 independent serial communication blocks (SCBs) with programmable I^2C, SPI, or UART	2 independent SCBs with programmable I^2C, SPI, or UART
Timer Counter Pulse-Width Modulator (TCPWM)	4	4	4	4	4
Universal Digital Blocks (UDBs)	None	4, each with 8 macrocells and one data path. Can be used to synthesize additional digital peripherals (Timer, Counter, PWM) or communication interfaces (UART, SPI)	None	None	4, each with 8 macrocells and one data path. Can be used to synthesize additional digital peripherals (Timer, Counter, PWM) or communication interfaces (UART, SPI)
Additional Digital	None	Yes (UDB-based digital	Yes (fixed-function blocks	None	Yes (UDB-based digital

Appendix A: Bluetooth® LE device family comparison

Features	Device Family				
	CY8C41x7-BLxxx	**CY8C42x7-BLxxx**	**CYBL10xxx*** **	**CY8C41x8-BL**	**CY8C42x8-BL**
Peripherals (I^2S, PWM)		peripherals on select devices)	on select devices)		peripherals on select devices)
Clocks	3-MHz to 24-MHz IMO 32-kHz ILO 24-MHz ECO 32-kHz WCO	3-MHz to 48-MHz IMO 32-kHz ILO 24-MHz ECO 32-kHz WCO	3-MHz to 48-MHz IMO 32-kHz ILO 24-MHz ECO 32-kHz WCO	3-MHz to 24-MHz IMO 32-kHz ILO 24-MHz ECO 32-kHz WCO	3-MHz to 48-MHz IMO 32-kHz ILO 24-MHz ECO 32-kHz WCO
Power Supply Monitoring	Power-on reset (POR) Brown-out detection (BOD) Low-voltage detection (LVD)	POR BOD LVD	POR BOD LVD	Power-on reset (POR) Brown-out detection (BOD) Low-voltage detection (LVD)	POR BOD LVD
Package	56-QFN (7.0 × 7.0 × 0.6 mm) and 68-WLCSP (3.52 × 3.91 × 0.55 mm)	56-QFN (7.0 × 7.0 × 0.6 mm) and 68-WLCSP (3.52 × 3.91 × 0.55 mm)	56-QFN* (7.0 × 7.0 × 0.6 mm) and 76-WLCSP (4.04 × 3.87 × 0.55 mm)	56-QFN* (7.0 × 7.0 × 0.6 mm) and 76-WLCSP (4.04 × 3.87 × 0.55 mm)	56-QFN* (7.0 × 7.0 × 0.6 mm) and 76-WLCSP (4.04 × 3.87 × 0.55 mm)
DMA	None	None	Up to 8 channels	Up to 8 channels**	Up to 8 channels**

* = CY8C41x8-BL and CY8C42x8-BL (256K FLASH) QFN packages are pin-to-pin compatible with CY8C41x7-BL and CY8C42x7-BL (128K FLASH) QFN package, respectively.

** = CY8C41x8-BL and CY8C42x8-BL family has two sub-families –BL4xx and –BL5xx. The –BL4xx series does not support Bluetooth 4.2 link layer privacy, link layer data length extension, and DMA engine; the –BL5xx series does.

*** = CY8CBL10xxx PRoC family of devices have multiple sub-families and the features described here are for the super set device. See **AN94020 - Getting Started with PRoC-BLE** for details.

10 Appendix B: Terms of art

This section lists the most commonly used terms that you might encounter while working with PSoC™ family of devices.

Component Configuration Tool: Simple GUI in PSoC™ Creator that is embedded in each Component. It is used to customize the Component parameters and is accessed by right-clicking a Component.

Components: Free embedded ICs represented by an icon in PSoC™ Creator software. These are used to integrate multiple ICs and system interfaces into one PSoC™ Component that is inherently connected to the MCU via the main system bus. For example, the Bluetooth® Low Energy Component creates Bluetooth Smart products in minutes. Similarly, you can use the Programmable Analog Components for sensors.

MiniProg3: A programming hardware for development that is used to program PSoC™ devices on your custom board or PSoC™ development kits that do not support a built-in programmer.

PSoC™: A programmable, embedded design platform that includes a CPU, such as the 32-bit Arm® Cortex®-M0, with both analog and digital programmable blocks. It accelerates embedded system design with reliable, easy-to-use solutions, such as touch sensing, and enables low-power designs.

PSoC™ 4 Bluetooth® LE: A PSoC™ 4 IC with an integrated Bluetooth® LE radio that includes a royalty-free Bluetooth® LE protocol stack compatible with the Bluetooth 4.2 specification.

PSoC™ Creator: PSoC™ 3, PSoC™ 4, and PSoC™ 5LP Integrated Design Environment (IDE) software that installs on your PC and allows concurrent hardware and firmware design of PSoC™ systems, or hardware design followed by export to other popular IDEs.

PSoC™ Programmer: A flexible, integrated programming application for programming PSoC™ devices. PSoC™ Programmer is integrated with PSoC™ Creator to program PSoC™ 3, PSoC™ 4, PRoC, and PSoC™ 5LP designs.

11 Appendix C: Bluetooth® LE development tools

11.1 PSoC™ 4 with AIROC™ Bluetooth® LE pioneer kit

The Bluetooth® Low Energy Pioneer Kit shown in **Figure 59** is a Bluetooth® LE development kit from Infineon that supports PSoC™ 4 with AIROC™ Bluetooth® LE device. This kit consists of pluggable PSoC™ 4 with AIROC™ Bluetooth® LE module that connects to a Bluetooth® LE Pioneer Baseboard. The kit can be powered either with a coin-cell battery or through the USB interface.

The Bluetooth® LE Pioneer Baseboard and the Bluetooth® LE module combination enables you to develop battery-operated low-power Bluetooth® LE designs that work in conjunction with standard, Arduino Uno connector-compliant shields or the onboard PSoC™ 4 with AIROC™ Bluetooth® LE device capabilities such as the CapSense™ user interface.

The Bluetooth® LE Pioneer Baseboard has an onboard programming and debugging interface, making the Bluetooth® LE design debugging through the PSoC™ Creator IDE quick and easy. The Bluetooth® Low Energy Pioneer Kit has an additional header that supports interfacing with Pmod™ daughter cards from third-party vendors such as Digilent®. The kit also has a Bluetooth® Low Energy Dongle that acts as a Bluetooth® LE link master and works with CySmart Host Emulation Tool to provide a Bluetooth® LE host emulation platform on non-Bluetooth® LE Windows PCs.

The kit consists of a set of Bluetooth® LE example projects and documentation that help you get started on developing your own Bluetooth® LE applications. Visit **PSoC™ 4 Bluetooth® Low Energy Pioneer Kit** webpage to get the latest updates on the kit and download the kit design, example projects, and documentation files.

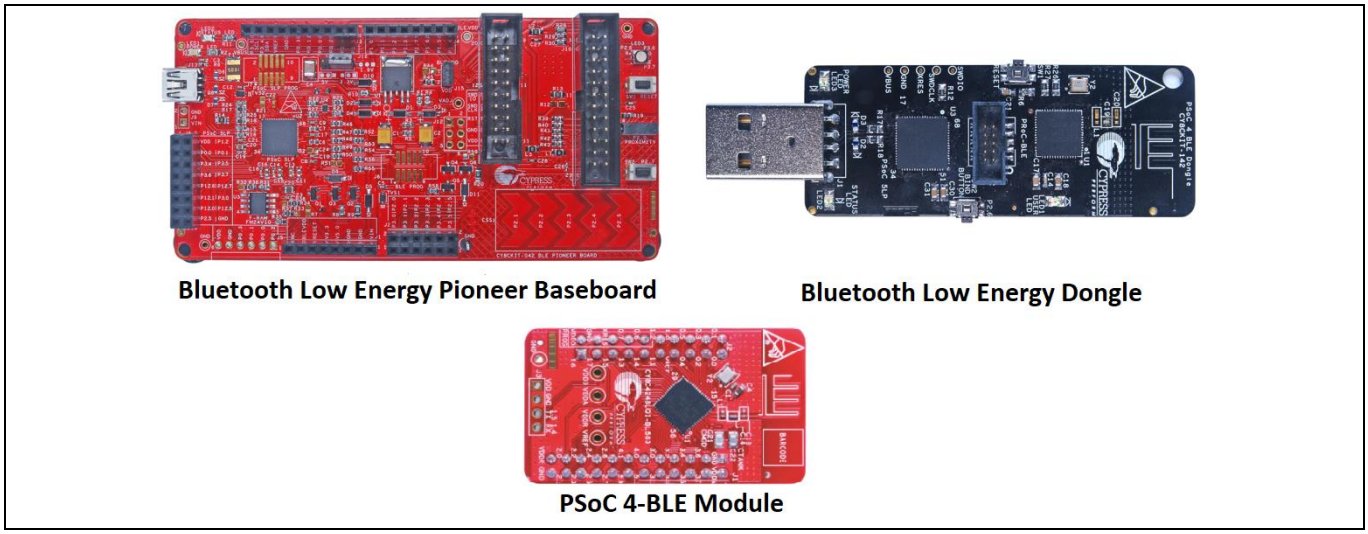


Figure 59 Bluetooth® Low Energy pioneer kit

11.2 CySmart host emulation tool

The CySmart Host Emulation Tool is a Windows application that emulates a Bluetooth® LE Central device using the Bluetooth® Low Energy Pioneer Kit's Bluetooth® Low Energy Dongle; see **Figure 19**. It is installed as part of the Bluetooth® Low Energy Pioneer Kit installation and can be launched from right-click options in the Bluetooth® Low Energy Component. It provides a platform for you to test your PSoC™ 4 with AIROC™ Bluetooth® LE Peripheral implementation over GATT or L2CAP connection-oriented channels by allowing you to discover and configure the Bluetooth® LE Services, Characteristics, and Attributes on your Peripheral.

Operations that you can perform with CySmart Host Emulation Tool include, but are not limited to:

- Scan Bluetooth® LE Peripherals to discover available devices to which you can connect.
- Discover available Bluetooth® LE Attributes including Services and Characteristics on the connected Peripheral device.
- Perform read and write operations on Characteristic values and descriptors.
- Receive Characteristic notifications and indications from the connected Peripheral device.
- Establish a bond with the connected Peripheral device using Bluetooth® LE Security Manager procedures.
- Establish a Bluetooth® LE L2CAP connection-oriented session with the Peripheral device and exchange data per the Bluetooth 4.2 specification.
- Over-the-air (OTA) firmware upgrade of Bluetooth® LE Peripheral devices.

Figure 60 and **Figure 61** show the user interface of CySmart Host Emulation Tool. For more information on how to set up and use this tool, see the CySmart user guide from the **Help** menu.

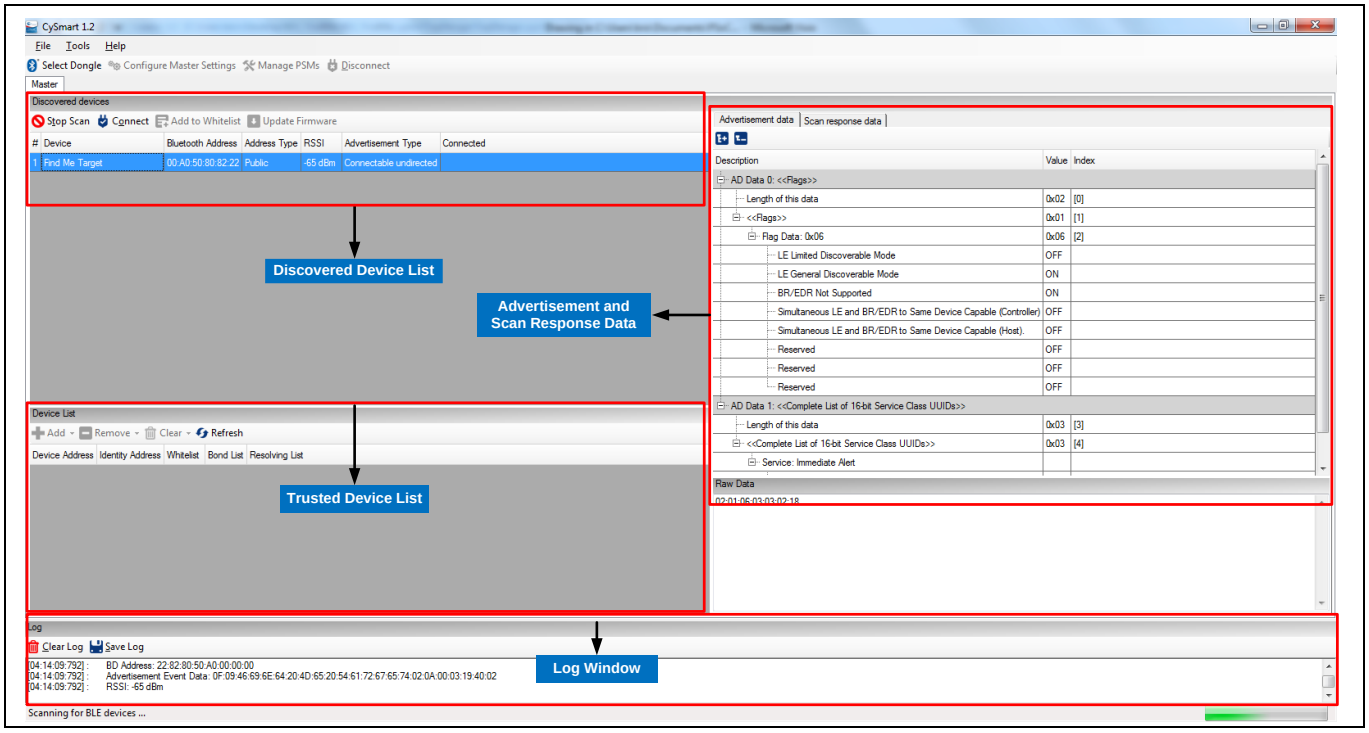


Figure 60 CySmart host emulation tool master device tab

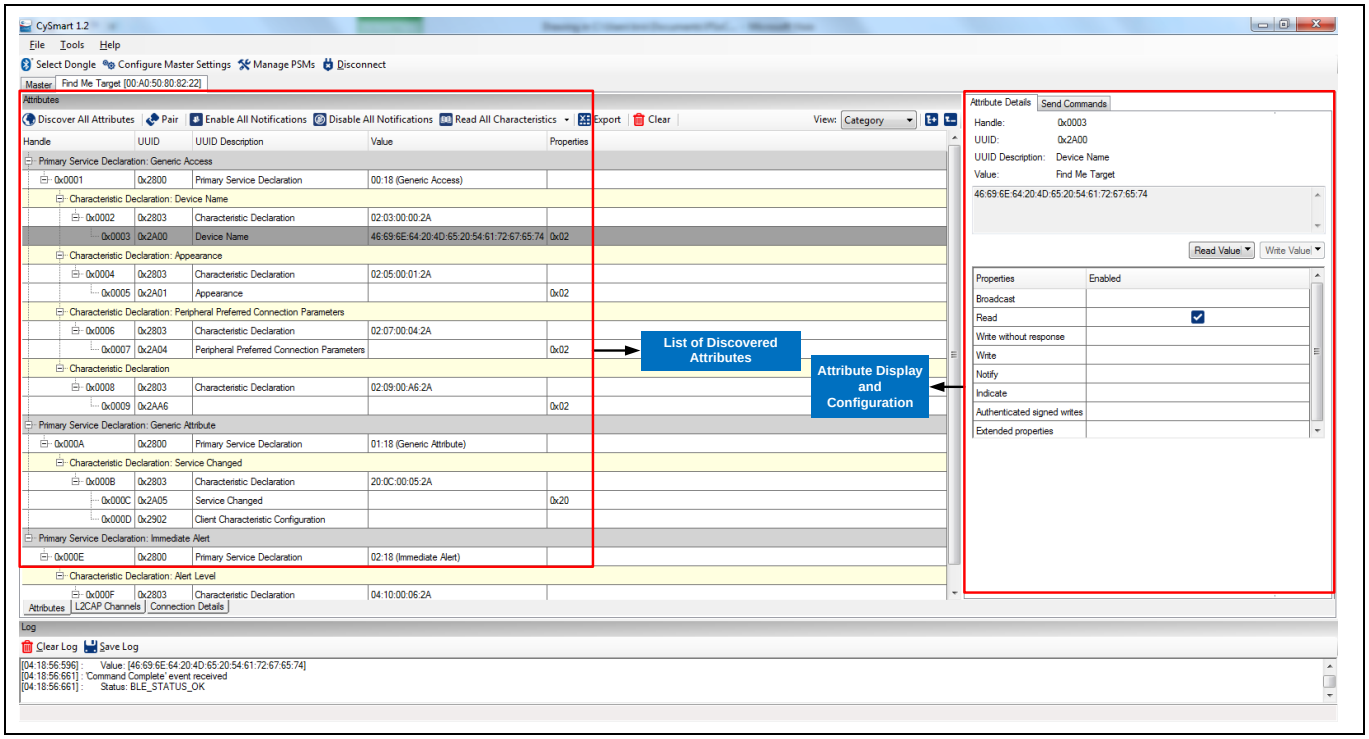


Figure 61 CySmart host emulation tool peripheral device attributes tab

11.3 CySmart Mobile app

In addition to the PC tool, you can download the CySmart mobile app for iOS or Android from the respective app stores. This app uses the iOS Core Bluetooth framework and the Android built-in platform framework for Bluetooth® LE respectively to configure your Bluetooth® LE-enabled smartphone as a Central device that can scan and connect to Peripheral devices.

The mobile app supports SIG-adopted Bluetooth® LE standard Profiles through an intuitive GUI and abstracts the underlying Bluetooth® LE Service and Characteristic details. In addition to the Bluetooth® LE standard Profiles, the app demonstrates a custom Profile implementation using LED and CapSense™ demo examples. **Figure 62** and **Figure 63** show a set of CySmart app screenshots for the Heart Rate Profile user interface. For a description of how to use the app with Bluetooth® Low Energy Pioneer Kit example projects, see the Bluetooth® Low Energy Pioneer Kit guide.

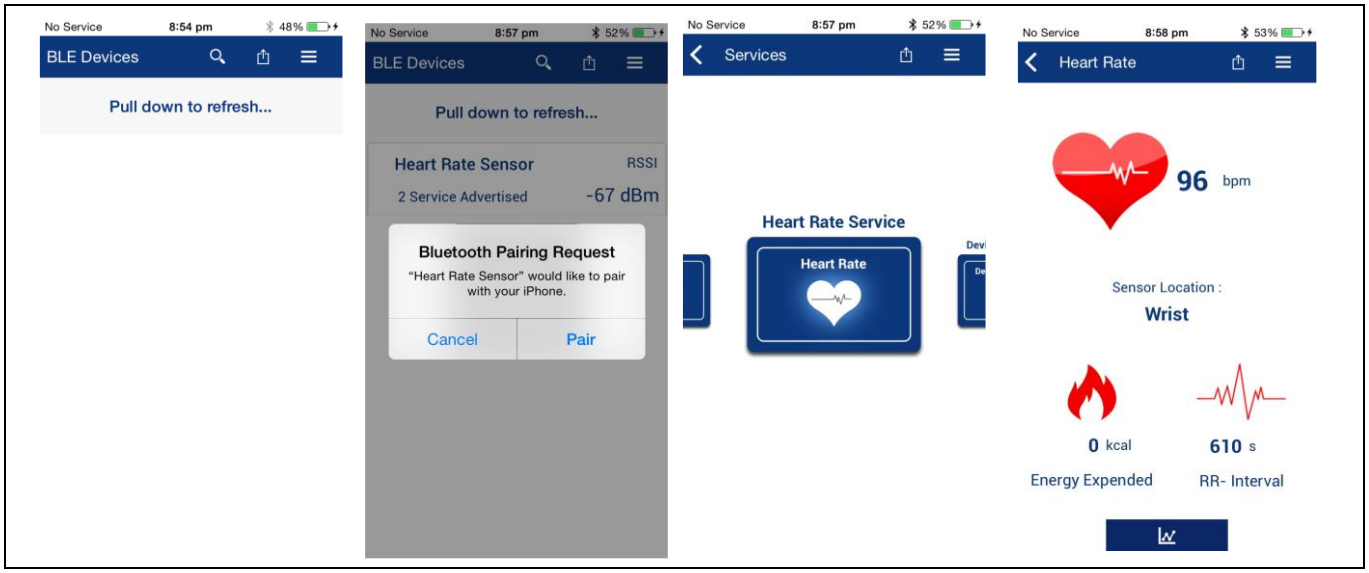


Figure 62 CySmart iOS app heart rate profile example

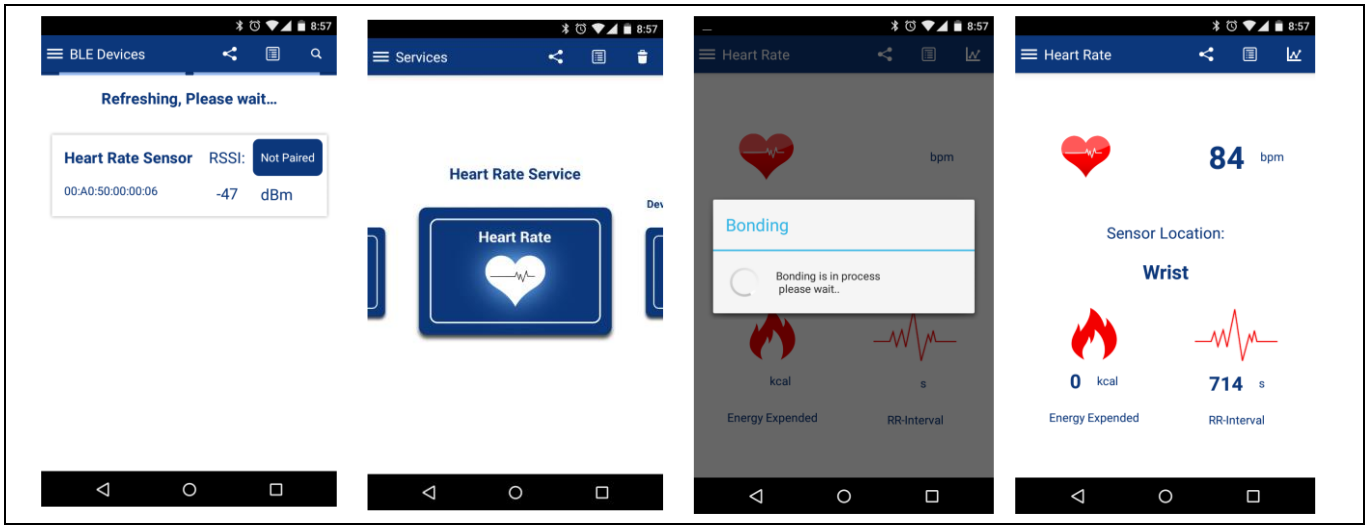


Figure 63 CySmart android app heart rate profile example

12 Appendix D: PSoC™ 4 with AIROC™ Bluetooth® LE device

12.1 Bluetooth® Low Energy Subsystem (BLESS)

The Bluetooth® Low Energy subsystem contains the physical layer (PHY) and link-layer engine with an embedded AES-128 security engine.

The physical layer consists of a digital PHY and RF transceiver compliant with the Bluetooth 4.2 specification. The transceiver transmits and receives GFSK packets at 1 Mbps over the 2.4-GHz ISM band. The baseband controller is a composite hardware/firmware implementation that supports both master and slave modes. The key protocol elements such as HCI and link control are implemented in firmware, while time-critical functions such as encryption, CRC, data whitening, and access code correlation are implemented in hardware.

The BLESS is Bluetooth 4.2 compliant with support for all the features of the Bluetooth 4.0 specification and some additional features of the Bluetooth 4.2 specification such as low-duty-cycle advertising, LE ping, L2CAP connection-oriented channels, link layer privacy, link layer data length extension, and LE secure connection. The BLESS block also contains an external crystal oscillator (ECO) and watch crystal oscillator (WCO) that are required for generating an accurate RF frequency and keeping the time between successive connection intervals on the Bluetooth® LE link respectively. The ECO and the WCO from the BLESS block can also be used as high-frequency and low-frequency clock sources for the PSoC™ 4 with AIROC™ Bluetooth® LE system. See **Clocking system**.

The BLESS supports five functional power modes: Deep-Sleep, Sleep, Idle, Transmit, and Receive.

Note: *The power modes discussed in this section are specific to the BLESS block. For PSoC™ 4 with AIROC™ Bluetooth® LE system power modes, see **Low-leakage power modes**.*

12.1.1 Deep-sleep mode

Deep-Sleep mode is the lowest power functional mode supported by the BLESS. In this mode, the radio is off. This mode is entered for maximum power saving during an advertising or connection interval after the packet transmission and reception is complete. The ECO can be turned off in this mode for power saving; the WCO, which is the low-frequency clock, is on for maintaining the Bluetooth® LE link-layer timing reference logic. The CPU controls the entry to and exit from this state.

12.1.2 Sleep mode

In Sleep mode, the radio is off. The block maintains all the configurations. The ECO and WCO are turned on, but the clock to the core BLESS logic is turned off. The CPU controls the entry to and exit from this state.

12.1.3 Idle mode

The Idle mode is the preparation state for the Transmit and Receive states. In this state, the radio is turned off but the link-layer clock is enabled for the link-layer logic so that the CPU starts the protocol state machines.

12.1.4 Transmit mode

Transmit mode is the active functional mode; all the blocks within BLESS are powered on. The link-layer clock is enabled to complete the logic within the link layer and RF-PHY. In this mode, RF-PHY gets 1-Mbps of serial data from the link layer and transmits the 2.4-GHz GFSK-modulated data to the antenna port. Bluetooth® LE enters Transmit mode from Idle mode.

12.1.5 Receive mode

This mode enables the BLESS to move into the receive state to perform Bluetooth® LE-specific receiver operations. RF-PHY translates the 1-Mbps data received from the RF analog block and forwards it to the link-layer controller after demodulation. A summary of the BLESS power modes and operational sub-blocks is shown in **Table 3**.

Table 3 BLESS power modes

BLESS Power Mode	ECO	WCO	RF Tx	RF Rx	BLESS Core
Deep-Sleep	Off	On	Off	Off	Off
Sleep	On	On	Off	Off	Off
Idle	On	On	Off	Off	On
Transmit	On	On	On	Off	On
Receive	On	On	Off	On	On

12.2 Arm® Cortex®-M0, memory, and DMA

PSoC™ 4 with AIROC™ Bluetooth® LE has a 32-bit Arm® Cortex®-M0 CPU, capable of operating at a maximum frequency of 48 MHz, providing a 43-DMIPS performance. The CPU supports single-cycle 32-bit multiplication.

PSoC™ 4 with AIROC™ Bluetooth® LE has 16 KB/32 KB of SRAM and 128 KB/256 of flash memory that can service most of the Bluetooth® LE application use; the flash includes a read accelerator. The device also provides 512 bytes of supervisory flash area for you to store user-specific data such as Bluetooth® LE device address and encryption keys.

A DMA engine, with eight channels, is provided on PSoC™ 4XX8-BL family of devices that can do 32-bit transfers among peripherals, among memory, or between peripherals and memory.

12.3 Programmable digital peripherals

PSoC™ 4 with AIROC™ Bluetooth® LE provides a rich set of digital peripherals including programmable serial communication blocks (SCBs), timer counter pulse width modulators (TCPWMs), and programmable logic arrays called universal digital blocks (UDBs).

12.3.1 Programmable SCBs

PSoC™ 4 with AIROC™ Bluetooth® LE has independent run-time programmable SCBs with I^2C, SPI, or UART. The SCB supports the following features:

- Standard SPI master and slave functionality with Motorola®, Texas Instruments®, and National Semiconductor® protocols
- Standard UART functionality with smart-card reader, Local Interconnect Network (LIN), and Infrared Data Association (IrDA) protocols
- Standard I^2C master and slave functionality
- SPI and EZI^2C mode, which allows operation without CPU intervention
- Low-power (Deep-Sleep) mode of operation for SPI and I^2C protocols (using an external clock)

For more information, refer to the **PSoC™ 4 SCB Component datasheet**.

12.3.2 Programmable TCPWMs

PSoC™ 4 with AIROC™ Bluetooth® LE has four programmable 16-bit TCPWM blocks. Each TCPWM can implement a 16-bit timer, counter, PWM, or quadrature decoder. TCPWMs provide complementary outputs and selectable start, reload, stop, count, and capture event signals. The PWM mode supports center-aligned, edge, and pseudo random operations.

For more information, refer to the **PSoC™ 4 TCPWM Component datasheet**.

12.3.3 Universal digital blocks

UDBs are programmable logic blocks that provide functionalities similar to CPLD and FPGA blocks, as **Figure 64** shows. UDBs allow you to create a variety of digital functions such as timer, counter, PWM, pseudo random sequence (PRS), CRC, shift register, SPI, UART, I2S, and custom combinational and sequential logic circuits.

Each UDB has two programmable logic devices (PLDs), each with 12 inputs and 8 product terms. PLDs can form registered or combinational sum-of-products logic. Additionally, an 8-bit single-cycle arithmetic logic unit (ALU), known as a “datapath,” is present in each UDB. The datapath helps with the efficient implementation of functions such as timer, counter, PWM, and CRC.

UDBs also provide a switched digital signal interconnect (DSI) fabric that allows signals from peripherals and ports to be routed to and through the UDBs for communication and control.

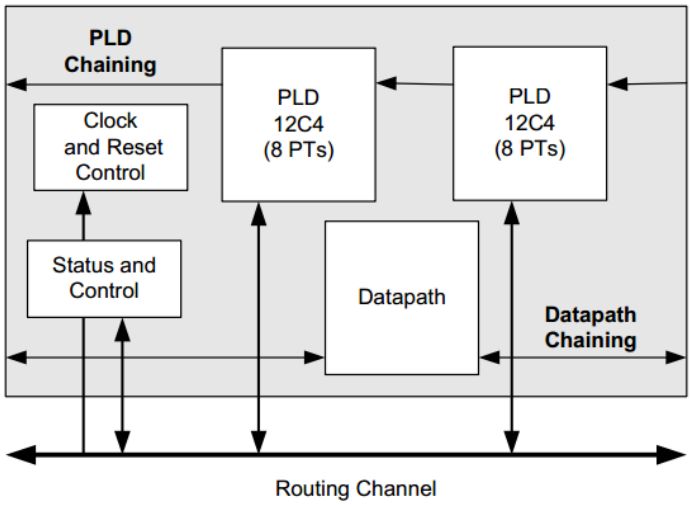


Figure 64 Universal digital block diagram

You do not necessarily need to know any hardware description language (HDL) to use UDBs. PSoC™ Creator, Infineon development tool for PSoC™ 4 with AIROC™ Bluetooth® LE, can generate the required function for you from a schematic. If required, advanced users can implement custom logic on UDBs using Verilog.

For more information, refer to the following application notes.

- **AN62510 – Implementing State Machines with PSoC™ 3, PSoC™ 4, and PSoC™ 5LP**
- **AN82156 – PSoC™ 3, PSoC™ 4, and PSoC™ 5LP – Designing PSoC™ Creator Components with UDB Datapaths**
- **AN82250 – PSoC™ 3, PSoC™ 4, and PSoC™ 5LP – Implementing Programmable Logic Designs with Verilog**

12.3.4 Applications

The use of programmable digital peripherals in Bluetooth® LE applications is shown in **Table 4**.

Table 4 Applications of programmable digital peripherals

Applications	Digital peripherals
Sensor Hub	I^2C (digital sensor interface), PWM (actuators), I2S (voice input)
Health and Fitness	PWM (user interface), Counter (waveform peak measurement), SPI (external memory interface)
Industrial	UART (modbus), Counter (event counting)
Home Automation	PWM (garage door control), PWM (lighting control)

12.4 Programmable analog

PSoC™ 4 with AIROC™ Bluetooth® LE provides the industry's best-in-class analog integration. The analog system includes Continuous Time Block mini (CTBm) blocks, a fast 12-bit SAR ADC, low-power comparators, capacitive touch-sensing (CapSense™), and a segment LCD direct drive.

12.4.1 Continuous Time Block mini (CTBm)

PSoC™ 4 with AIROC™ Bluetooth® LE contains two CTBm blocks, each consisting of two programmable opamps and a switch matrix. You can configure each opamp individually as a comparator, voltage follower, or an opamp with external feedback. If required, the CTBm block can be configured to function even in device Deep-Sleep mode.

For more information, refer to the following Component datasheets:

- **PSoC™ 4 Opamp**
- **PSoC™ 4 Voltage Comparator**

12.4.2 SAR ADC with hardware sequencer

PSoC™ 4 with AIROC™ Bluetooth® LE has a 12-bit, 1-Msps Successive Approximation Register (SAR) ADC with input channels that support programmable resolution and single-ended or differential input options. The number of GPIOs limits the number of ADC input channels that can be implemented.

The SAR ADC has a hardware sequencer that can perform an automatic scan on as many as eight channels without CPU intervention. It also supports preprocessing operations such as accumulation and averaging of the output data on these eight channels.

You can trigger a scan with a variety of methods, such as firmware, timer, pin, or UDB, giving you additional design flexibility.

For more information, refer to the **PSoC™ 4 SAR ADC Component datasheet**.

12.4.3 Low-power comparators

PSoC™ 4 with AIROC™ Bluetooth® LE devices have low-power comparators capable of operating in all system power modes except the **Stop** mode. In a power-sensitive design, when the device goes into low-power modes, you can use the low-power comparator to monitor analog inputs and generate an interrupt that can wake up the system.

For more information, refer to the **PSoC™ 4 Low-Power Comparator Component datasheet**.

12.4.4 Capacitive touch sensing (CapSense™)

Capacitive touch sensors use human-body capacitance to detect the presence of a finger on or near a sensor. Capacitive sensors are aesthetically superior, easy to use, and have long lifetimes.

The CapSense™ feature in PSoC™ 4 with AIROC™ Bluetooth® LE offers unprecedented signal-to-noise ratio; best-in-class liquid tolerance; and a wide variety of sensor types such as buttons, sliders, track pads, and proximity sensors.

A Infineon-supplied software Component makes capacitive sensing design very easy; the Component supports an automatic hardware-tuning feature called SmartSense™ and provides a gesture-recognition library for trackpads and proximity sensors.

Two current DACs (IDACs), one 7-bit and one 8-bit, in the CapSense™ block are available for general-purpose use if capacitive sensing is not used. The comparator in the CapSense™ block is also available for general-purpose use.

For more information, see the **PSoC™ 4 CapSense™ Design Guide**.

12.4.5 Segment LCD direct driver

Most low-power, portable, handheld devices such as glucose meters, multimeters, and blood pressure monitors use a segment LCD to display information. Segment LCDs typically require an external driver to interface with a microcontroller. PSoC™ 4 with AIROC™ Bluetooth® LE includes an integrated low-power LCD driver that can directly drive segment LCD glass.

PSoC™ 4 with AIROC™ Bluetooth® LE can drive LCDs with as many as 4 common and 32 segment electrodes. The segment LCD driver can retain a static display in Deep-Sleep mode with a system current consumption as low as 7 µA.

For more information, see **AN87391 – PSoC™ 4 Segment LCD Direct Drive**.

12.4.6 Applications

The use of programmable analog peripherals in different Bluetooth® LE applications is listed in **Table 5**.

Table 5 Applications of programmable analog peripherals

Applications	Analog peripherals
Health and Fitness	Opamp: TIA (heart-rate measurement) Opamp: Follower (analog- reference buffer) ADC (sampling heart-rate signal) CapSense™ (user interface on a wrist band) Segment LCD (display on a wrist band)
Sensor Hub	ADC (analog sensor interface) Analog Mux (multiple-sensor input) Opamps (signal amplifier) Segment LCD (UI)
Industrial	ADC: Differential mode (temperature measurement) IDAC (temperature-sensor drive) Low-Power Comparator (wakeup on threshold detection)

Applications	Analog peripherals
	Opamp: PGA (4-mA to 20-mA current-loop system)
Home Automation	Opamp: PGA (motion sensor, light sensor) Comparator (door sensors) ADC and opamp filter (smoke detector)

12.5 System-wide resources

This section explains the system-wide resources available for all peripherals in PSoC™ 4 with AIROC™ Bluetooth® LE.

12.5.1 Low-leakage power modes

PSoC™ 4 with AIROC™ Bluetooth® LE offers the following power modes. Note that these are PSoC™ 4 with AIROC™ Bluetooth® LE device power modes, which are different from the power modes described in the **Bluetooth® Low Energy Subsystem (BLESS)** section.

- **Active mode:** This is the primary mode of operation. In this mode, all peripherals are available.
- **Sleep mode:** In this mode, the CPU is in Sleep mode, SRAM is in retention, and all peripherals are available. Any interrupt wakes up the CPU and returns the system to Active mode.
- **Deep-Sleep mode:** In this mode, the high-frequency clock (IMO) and all high-speed peripherals are off. Optionally, the low-frequency clocks (32-kHz ILO and WCO) and low-speed peripherals are available. Interrupts from low-speed, asynchronous, or low-power analog peripherals can cause a wakeup. The current consumption in this mode is 1.3 µA.
- **Hibernate mode:** This power mode provides a best-in-class current consumption of 150 nA while retaining SRAM and the ability to wake up from an interrupt generated by a low-power comparator or a GPIO.
- **Stop mode:** This power mode retains the GPIO states. Wakeup is possible from a fixed WAKEUP pin. The current consumption in this mode is only 60 nA.

You can use a combination of Sleep, Deep-Sleep, Hibernate, and Stop modes in a battery-operated Bluetooth® LE system to achieve best-in-class system power with longer battery life.

Table 6 shows the dependency between PSoC™ 4 with AIROC™ Bluetooth® LE system power modes and BLESS power modes. All these dependencies are handled by simple APIs; see the **Main loop and low-power** section for an example.

In a typical Bluetooth® LE application such as heart-rate monitoring, the PSoC™ 4 with AIROC™ Bluetooth® LE device will be in Active mode while measuring the heart-rate, in Sleep mode while the Bluetooth® LE radio is transmitting or receiving packets, in Deep-Sleep mode between consecutive Bluetooth® LE connection intervals, and in Hibernate or Stop mode on Bluetooth® LE advertisement timeout.

Table 6 PSoC™ 4 with AIROC™ Bluetooth® LE power modes

BLESS modes	PSoC™ 4 with AIROC™ Bluetooth® LE system power modes				
	Active	Sleep	Deep-Sleep	Hibernate	Stop
Transmit	✓	✓	✖	✖	✖
Receive	✓	✓	✖	✖	✖
Idle	✓	✓	✖	✖	✖
Sleep	✓	✓	✖	✖	✖
Deep-Sleep	✓	✓	✓	✖	✖

BLESS modes	PSoC™ 4 with AIROC™ Bluetooth® LE system power modes				
	Active	Sleep	Deep-Sleep	Hibernate	Stop
Powered	✘	✘	✘	✓	✓

12.5.2 Power supply and monitoring

PSoC™ 4 with AIROC™ Bluetooth® LE is capable of operating from a single 1.9-V to 5.5-V supply. There are multiple internal regulators to support the different device power modes. PSoC™ 4 with AIROC™ Bluetooth® LE has three types of voltage-monitoring capabilities: POR, BOD, and LVD.

12.5.3 Clocking system

PSoC™ 4 with AIROC™ Bluetooth® LE has the following clock sources:

- **Internal main oscillator (IMO)**: The IMO is the primary source of internal clocking in PSoC™ 4 with AIROC™ Bluetooth® LE. The CPU and all high-speed peripherals can operate from the IMO or an external crystal oscillator (ECO). PSoC™ 4 with AIROC™ Bluetooth® LE has multiple peripheral clock dividers operating from either the IMO or the ECO, which generate clocks for high-speed peripherals. The IMO can generate clocks in the range of 3 MHz to 48 MHz in 1-MHz increments with an accuracy of ±2 percent.
- **Internal low-speed oscillator (ILO):** The ILO is a very-low-power 32-kHz oscillator, which primarily generates clocks for low-speed peripherals operating in Deep-Sleep mode except the BLESS (see **WCO**).
- **External crystal oscillator (ECO):** The external crystal oscillator with a built-in tunable crystal load capacitance is used to generate a highly accurate 24-MHz clock. It is primarily used to clock the Bluetooth® Low Energy subsystem that generates the RF clocks. The high-accuracy ECO clock can also be used as a clock source for the PSoC™ 4 with AIROC™ Bluetooth® LE device's high-frequency clock (HFCLK).
- **Watch crystal oscillator (WCO):** The 32-kHz WCO is used as one of the sources for LFCLK (along with ILO). WCO is used to accurately maintain the time interval for Bluetooth® LE advertising and connection events. Similar to ILO, WCO is also available in all modes except the Hibernate and Stop modes.

Figure 65 shows the clocking architecture of a PSoC™ 4 with AIROC™ Bluetooth® LE device.

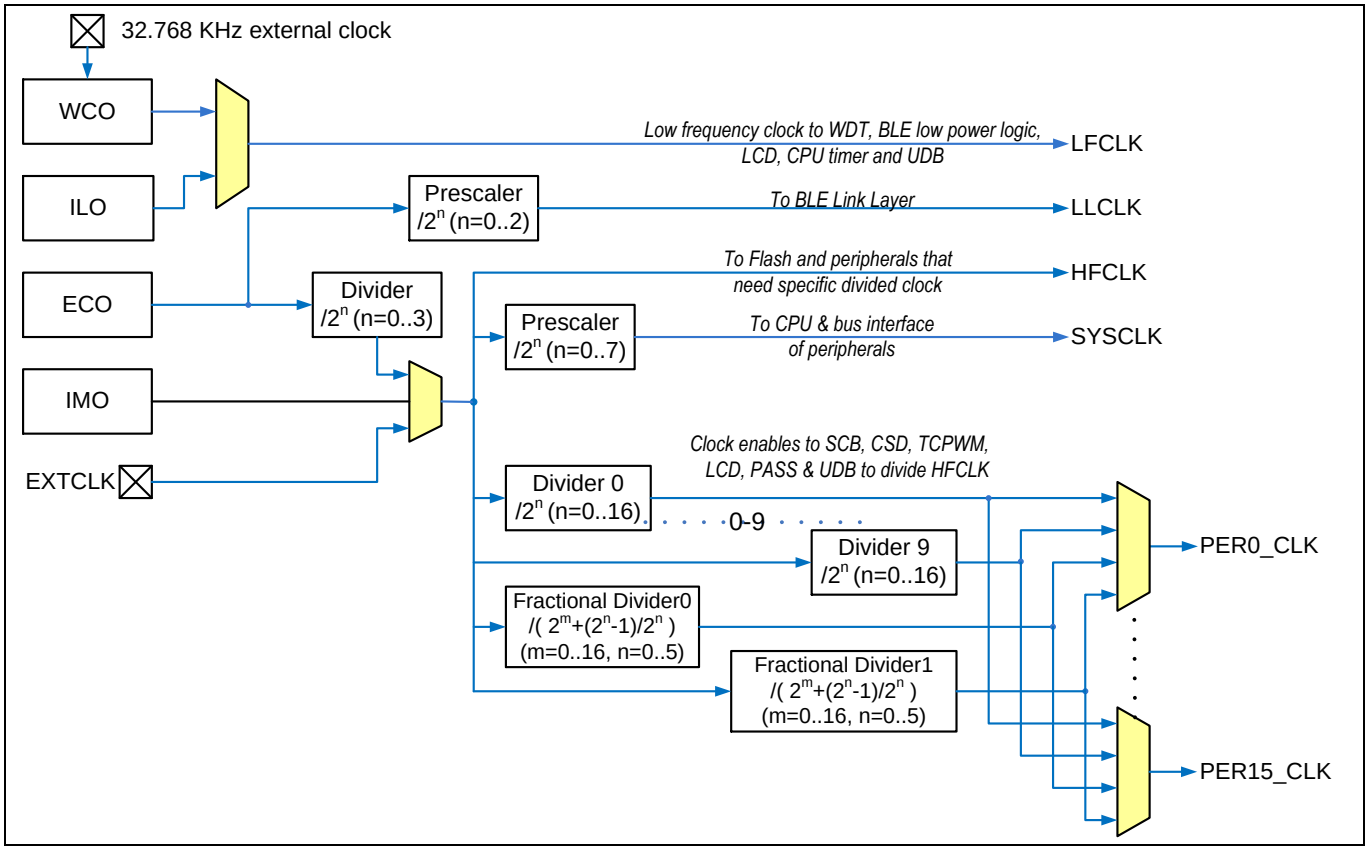


Figure 65 PSoC™ 4 with AIROC™ Bluetooth® LE clocking system

12.5.4 Device security

PSoC™ 4 with AIROC™ Bluetooth® LE provides a number of options to protect the flash memory from unauthorized access or copying. Each row of flash has a single protection bit; these bits are stored in a supervisory flash row.

12.6 Programmable GPIOs

The I/O system provides an interface between the CPU and the peripherals and the outside world. PSoC™ 4 with AIROC™ Bluetooth® LE has up to 36 programmable GPIO pins. You can configure the GPIOs for CapSense™, LCD, analog, or digital signals. PSoC™ 4 with AIROC™ Bluetooth® LE GPIOs support multiple drive modes, drive strengths, and slew rates.

PSoC™ 4 with AIROC™ Bluetooth® LE offers an intelligent routing system that gives multiple choices for connecting an internal signal to a GPIO. This flexible routing simplifies circuit design and board layout.

13 Appendix E: Bluetooth® LE protocol

13.1 Overview

Bluetooth® LE, also known as Bluetooth Smart, was introduced by the Bluetooth® SIG as a low-power wireless standard operating in the 2.4-GHz ISM band. **Figure 66** shows the Bluetooth® LE protocol stack.

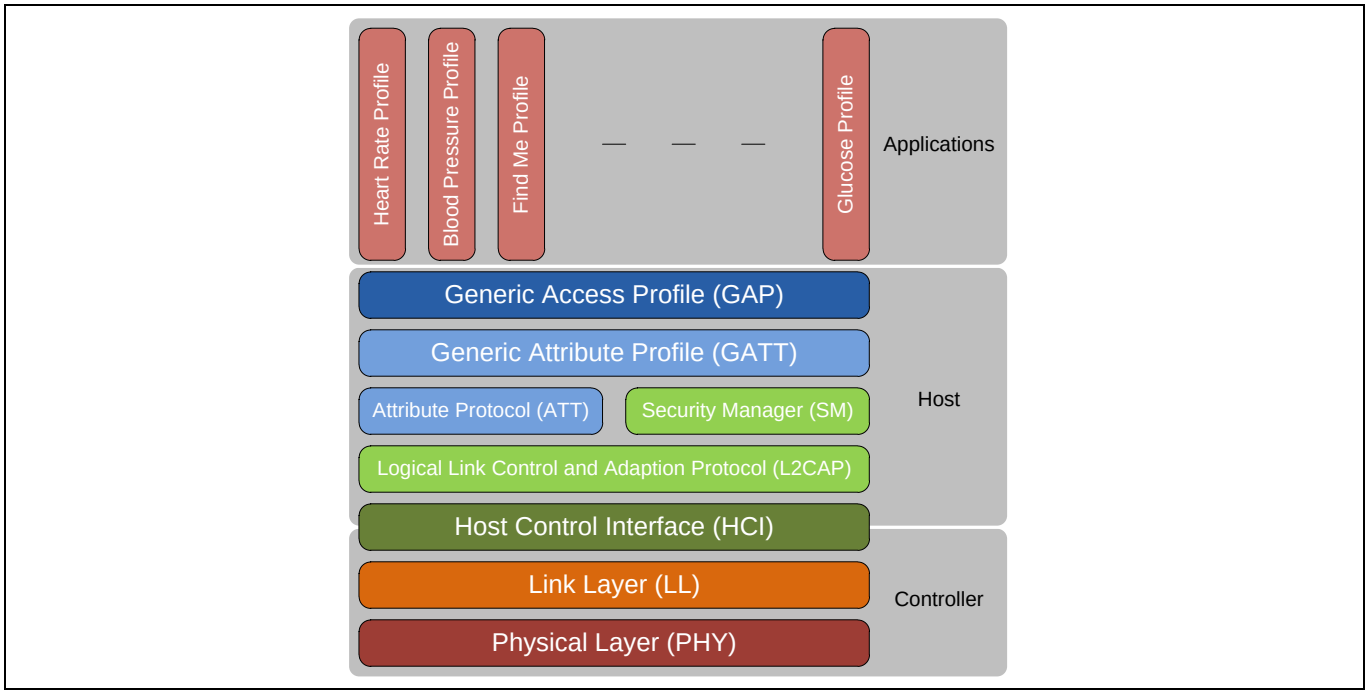


Figure 66 Bluetooth® LE architecture

The Bluetooth® LE stack can be subdivided into three groups:

- **Controller:** A physical device that encodes the packet and transmits it as radio signals. On reception, the controller decodes the radio signals and reconstructs the packet.
- **Host:** A software stack consisting of various protocols and Profiles (Security Manager, Attribute Protocol, and so on) that manages how two or more devices communicate with one another.
- **Application:** A use case that uses the software stack and the controller to implement a particular functionality.

The following sections provide an overview of the multiple layers of the Bluetooth® LE stack, using the standard Heart Rate and Battery Service as examples. For a detailed Bluetooth® LE architecture description, see the Bluetooth 4.2 specification or the training videos on the **Bluetooth Developer** website.

13.2 Physical Layer (PHY)

The physical layer transmits or receives digital data at 1 Mbps using Gaussian frequency-shift keying (GFSK) modulation in the 2.4-GHz ISM band. The Bluetooth® LE physical layer divides the ISM band into 40 RF channels with a channel spacing of 2 MHz, 37 of which are data channels and 3 are advertisement channels.

13.3 Link Layer (LL)

The link layer implements key procedures to establish a reliable physical link (using an acknowledgement and flow-control-based architecture) and features that help make the Bluetooth® LE protocol robust and low power. Some link layer functions include:

- Advertising, scanning, creating, and maintaining connections to establish a physical link
- 24-bit CRC and AES-128-bit encryption for robust and secure data exchange
- Establishing fast connections and low-duty-cycle advertising for low-power operation
- Adaptive Frequency Hopping (AFH), which changes the communication channel used for packet transmission so that the interference from other devices is reduced

At the link layer, two roles are defined:

- **Master:** A smartphone is an example that configures the link layer in the master configuration.
- **Slave:** A heart-rate monitor device is an example that configures the link layer in the slave configuration.

PSoC™/PRoC-BLE devices can operate in either configuration.

The link-layer slave is the one that advertises its presence to another link-layer master. A link-layer master receives the advertisement packets and can choose to connect to the slave based on the request from an application (see **Figure 67**). In this example implementation of a heart-rate monitor application, a heart-rate monitor device acts as the slave and sends the data to a smartphone, which acts as the master. A smartphone app then can display the reading on the smartphone.

PSoC™/PRoC-BLE devices implement the time-critical and processor-intensive parts of the link layer such as advertising, CRC, and AES encryption in hardware. Link-layer control operations such as entering the advertisement state and starting encryption are implemented in firmware.

Figure 67 shows the Bluetooth® LE link-layer packet structure and sizes of the individual fields in the link-layer packet. The link-layer packet carries all upper layer data in its payload field. It has a 4-byte access address that is used to uniquely identify communications on a physical link, and ignore packets from a nearby Bluetooth® LE device operating in the same RF channel. 24-bit CRC provides data robustness.

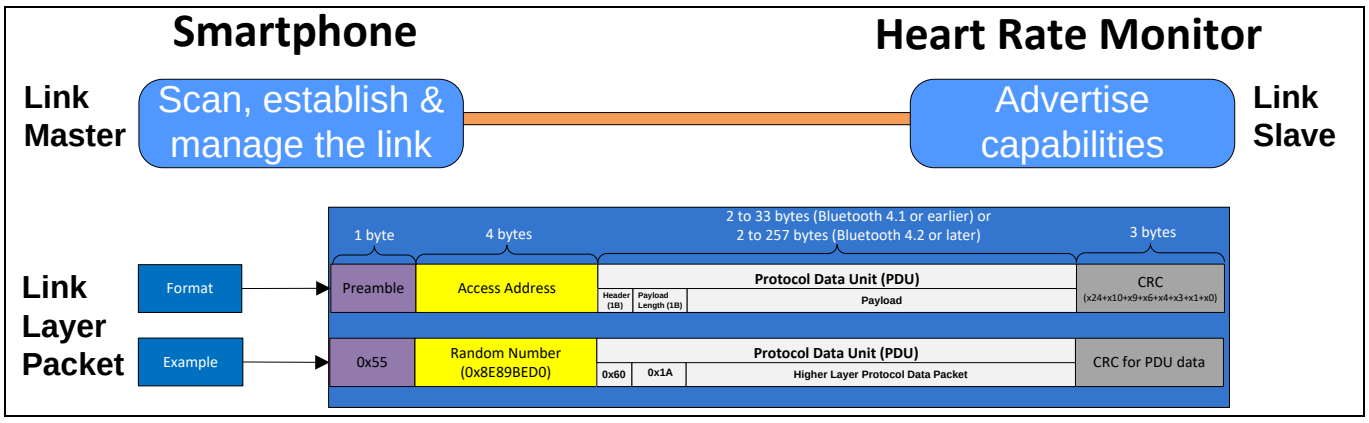


Figure 67 Bluetooth® LE Link Layer protocol

13.4 Host Control Interface (HCI)

The HCI is the standard-defined interface between the host and the controller. It allows the host and the controller to exchange information such as commands, data, and events over different physical transports such as USB or UART. The HCI requires a physical transport only when the controller and the host are different devices.

In PSoC™/PRoC-BLE devices, the HCI is just a firmware protocol layer that passes the messages and events between the controller and the host.

13.5 Logical Link Control and Adaptation Protocol (L2CAP)

L2CAP provides protocol multiplexing, segmentation, and reassembly services to upper-layer protocols. Segmentation breaks the packet received from the upper layer into smaller packets that the link layer can transmit, while reassembly combines the smaller packets received from the link layer into a meaningful packet. The L2CAP layer supports three protocol channel IDs for **Attribute Protocol (ATT)**, **Security Manager (SM)**, and L2CAP control, as shown in **Figure 68**. Bluetooth 4.2 allows direct data channels through the L2CAP connection-oriented channels on top of these protocol channels.

The L2CAP and the layers above it are implemented in firmware in PSoC™/PRoC-BLE.

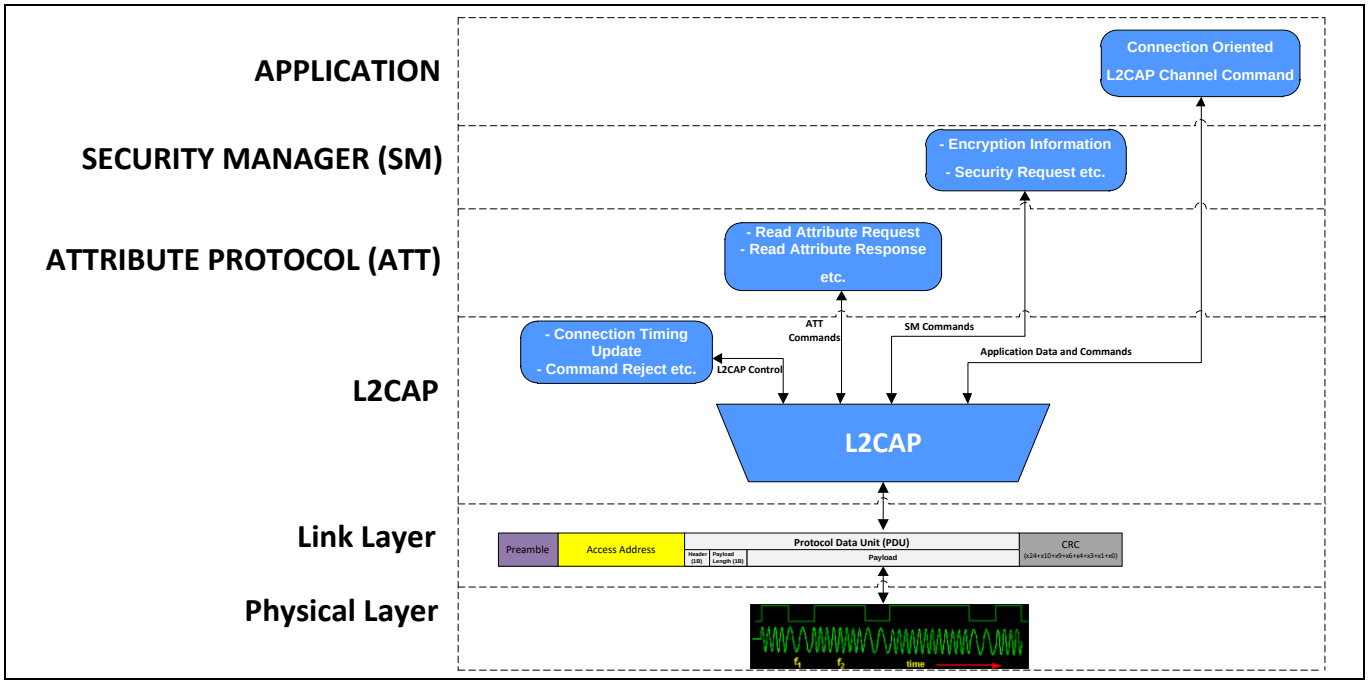


Figure 68 Bluetooth® LE L2CAP layer

13.6 Security Manager (SM)

The SM layer defines the methods used for pairing, encryption, and key distribution.

- **Pairing** is the process to enable security features. In this process, two devices are authenticated, the link is encrypted, and then the encryption keys are exchanged. This enables the secure exchange of data over the Bluetooth® LE interface without being snooped on by a silent listener on the RF channel.
- **Bonding** is the process in which the keys and the identity information exchanged during the pairing process are saved. After devices are bonded, they do not have to go through the pairing process again when reconnected.

Bluetooth® LE uses 128-bit **AES** for data encryption.

13.7 Attribute Protocol (ATT)

There are two GATT roles in Bluetooth® LE that you should know to understand the ATT and GATT layers:

- **GATT server:** A GATT server contains the data or information. It receives requests from a GATT client and responds with data. For example, a heart-rate monitor GATT server contains heart-rate information; a Bluetooth® LE HID keyboard GATT server contains user key press information.
- **GATT client:** A GATT client requests and/or receives data from a GATT server. For example, a smartphone is a GATT client that receives heart-rate information from the heart-rate GATT server; a laptop is a GATT client that receives key-press information from a Bluetooth® LE keyboard.

ATT forms the basis of Bluetooth® LE communication. This protocol enables the GATT client to find and access data or Attributes on the GATT server. For more details about the GATT client and server architecture, refer to **Generic Attribute Profile (GATT)**.

An Attribute is the fundamental data container in the ATT/GATT layer, which consists of the following:

- **Attribute Handle:** The 16-bit address used to address and access an Attribute.
- **Attribute Type:** This specifies the type of data stored in an Attribute. It is represented by a 16-bit UUID defined by the Bluetooth® SIG.

For example, the 16-bit UUID of the Heart-Rate Service is 0x180D; the UUID for the Device Name Attribute is 0x2A00. Visit the Bluetooth **webpage** for a list of 16-bit UUIDs assigned by the SIG.

- **Attribute Value:** This is the actual data stored in the Attribute.
- **Attribute Permission:** This specifies the Attribute access, authentication, and authorization requirements. Attribute permission is set by the higher layer specification and is not discoverable through the Attribute protocol.

Figure 69 shows the structure of a Device Name Attribute as an example.

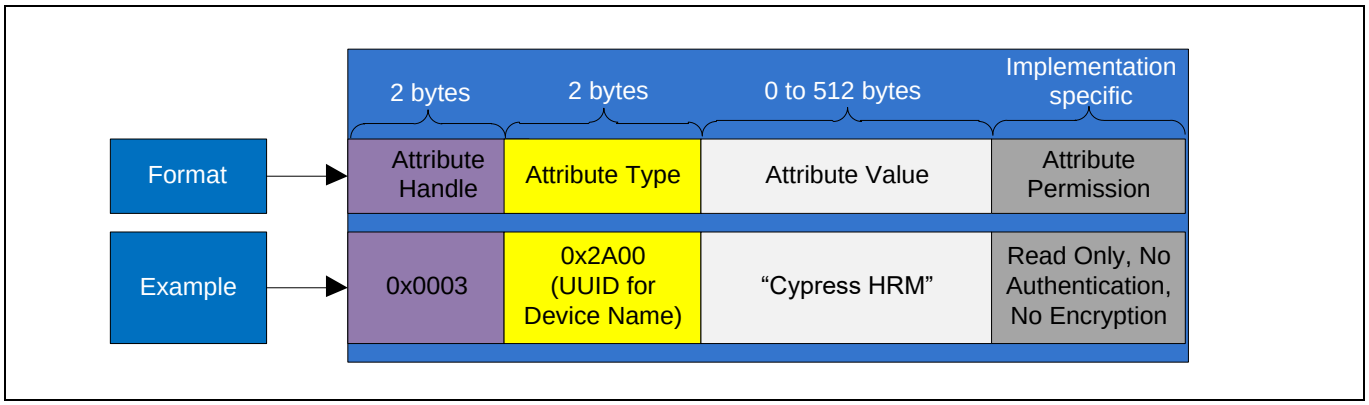


Figure 69 Attribute format example

13.7.1 Attribute hierarchy

Attributes are the building blocks for representing data in ATT/GATT. Attributes can be broadly classified into the following two groups to provide hierarchy and abstraction of data:

- **Characteristic:** A collection of Attributes that exposes the system information or meaningful data. A Characteristic consists of the following Attributes:

- Characteristic Declaration Attribute: This defines the beginning of a Characteristic.
- Characteristic Value Attribute: This holds the actual data.
- Characteristic Descriptor Attributes: These are optional Attributes, which provide additional information about the Characteristic value.

"Battery Level" is an example of a Characteristic in the Battery Service (BAS). Representing the battery level in percentage values is an example of a Characteristic descriptor.

Figure 70 shows the structure of a Characteristic with Battery Level as an example.

- The first part of a Characteristic is the declaration of the Characteristic (it marks the beginning of a Characteristic) indicated by the Battery Level Characteristic in **Figure 70**.
- Next is the actual Characteristic value or the real data, which in the case of the Battery Level Characteristic is the current battery level. The battery level is expressed as a percentage of full scale, for example "65," "90," and so on.
- Characteristic descriptors provide additional information that is required to make sense of the Characteristic value. For example, the Characteristic Presentation Format Descriptor for Battery Level indicates that the battery level is expressed as a percentage. Therefore, when "90" is read, the GATT client knows this is 90 percent and not 90 mV or 90 mAh. Similarly, the Valid Range Characteristic descriptor (not shown in **Figure 70**) indicates that the battery level range is between 0 and 100 percent.
- A Client Characteristic Configuration Descriptor (CCCD) is another commonly used Characteristic descriptor that allows a GATT client to configure the behavior of a Characteristic on the GATT server. When the GATT client writes a value of 0x01 to the CCCD of a Characteristic, it enables asynchronous notifications (described in the next section) to be sent from the GATT server. In the case of a Battery Level Characteristic, writing 0x01 to the Battery Level CCCD enables the Battery Service to notify its battery level periodically or on any change in battery-level value.

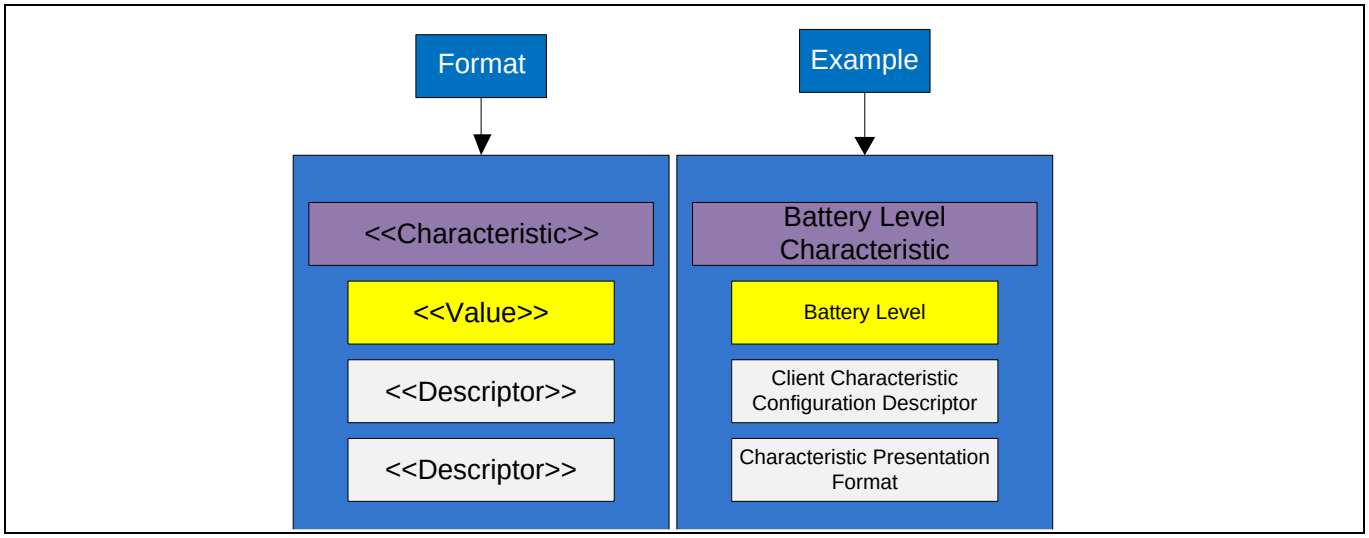


Figure 70 Characteristic format and example

- **Service:** The type of Attribute that defines a function performed by the GATT server. A Service is a collection of Characteristics and can include other Services. The concept of a Service is used to establish the grouping of relative data and provide a data hierarchy. See **Figure 71** for an example of a Heart Rate Service (**HRS**).

A Service can be of two types: A primary Service or a secondary Service. A primary Service exposes the main functionality of the device, while the secondary Service provides additional functionality. For example, in a heart-rate monitoring device, the HRS is a primary Service and BAS is a secondary Service.

A Service can also include other Services that are present on the GATT server. The entire included Services become part of the new Service.

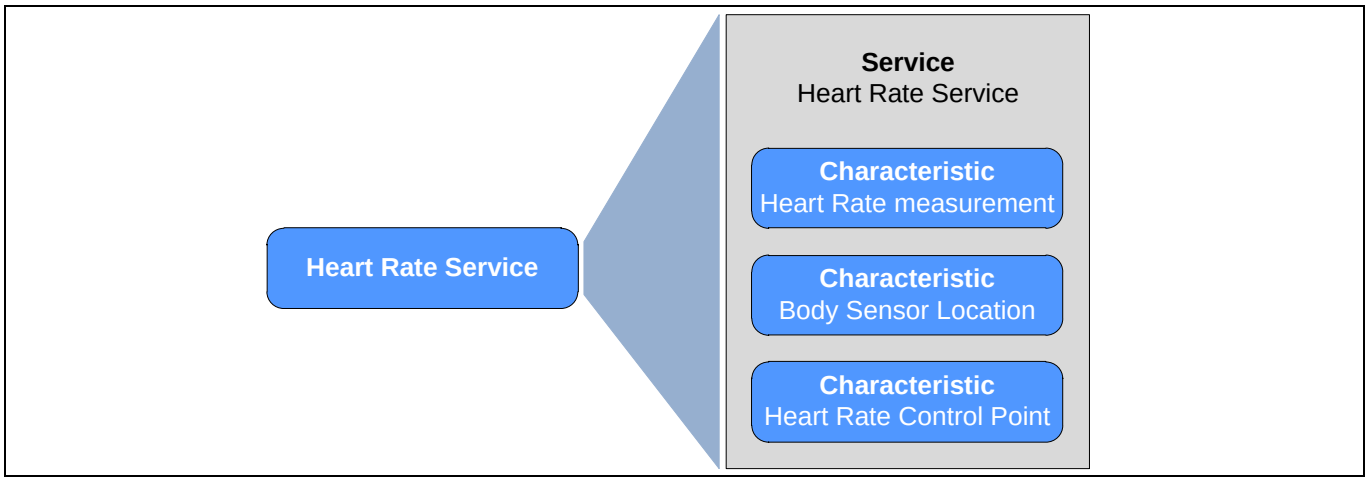


Figure 71 Bluetooth® LE heart rate service example

The word "Profile" in Bluetooth® LE is a collection of Services and their behavior that together perform a particular end application. A Heart Rate Profile (**HRP**) is an example of a Bluetooth® LE Profile that defines all the required Services for creating a heart-rate monitoring device. See the **Generic Access Profile (GAP)** section for details.

Figure 72 shows the data hierarchy using Attributes, Characteristics, Services, and Profiles defined previously in this section.

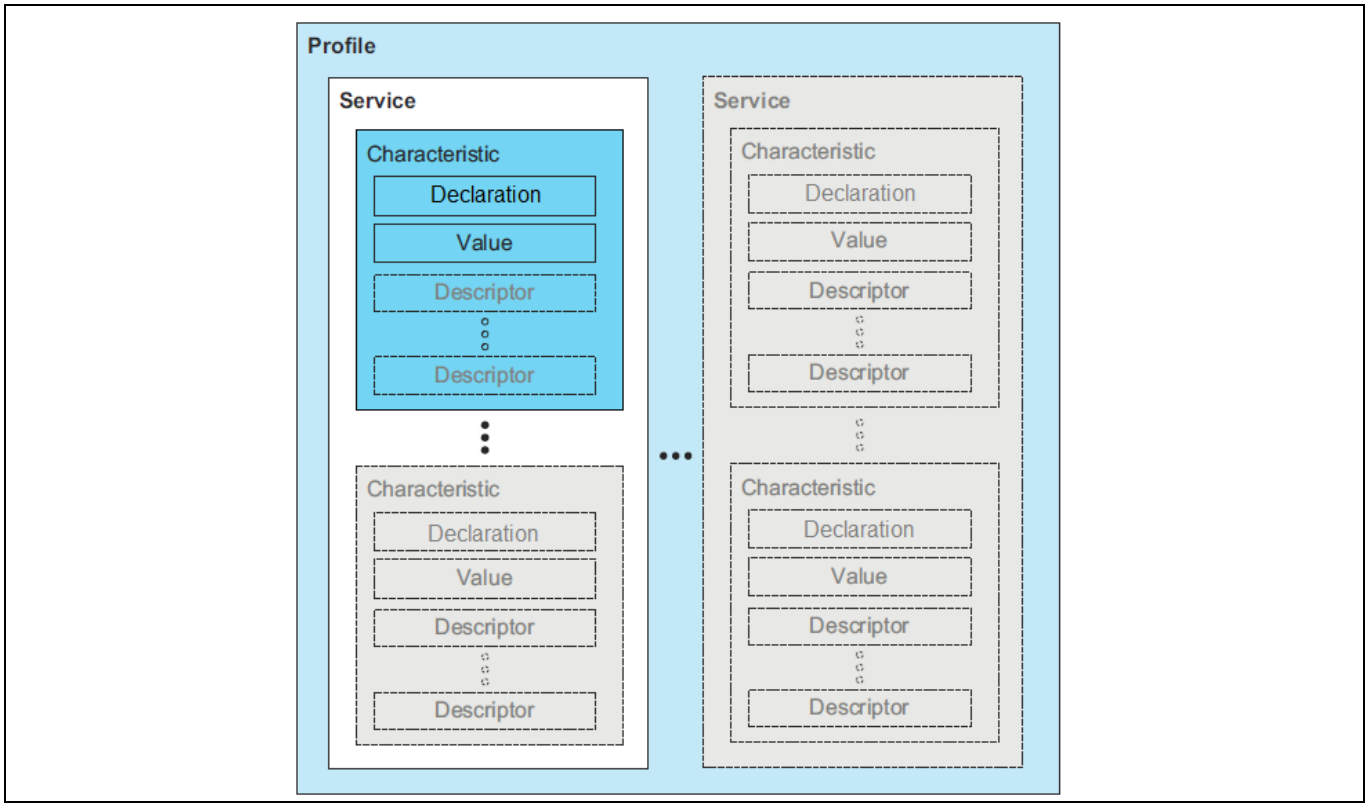


Figure 72 Bluetooth® LE data hierarchy*

* Image courtesy of Bluetooth® SIG

13.7.2 Attribute operations

Attributes defined in the previous section are accessed using the following five basic methods:

- **Read Request:** The GATT client sends this request to the GATT server to read an Attribute value. For every request, the GATT server sends a response to the GATT client. A smartphone reading the Battery-Level Characteristic of a heart-rate monitor device (see **Figure 70**) is an example of a Read Request.
- **Write Request:** The GATT client sends this request to the GATT server to write an Attribute value. The GATT server responds to the GATT client, indicating whether the value was written. A smartphone writing a value of 0x01 to the CCCD of a Battery Level Characteristic to enable notifications is an example of a Write Request.
- **Write Command:** The GATT client sends this command to the GATT server to write an Attribute value. The GATT server does not send any response to this command. For example, the Bluetooth® LE Immediate Alert Service (**IAS**) uses a Write Command to trigger an alert (turn on an LED, ring a buzzer, drive a vibration motor, and so on) on an IAS Target device (for example, a Bluetooth® LE key fob) from an IAS locator (for example, a smartphone).
- **Notification:** The GATT server sends this to the GATT client to notify it of a new value for an Attribute. The GATT client does not send any confirmation for a notification. For example, a heart-rate monitor device sends heart-rate measurement notifications to a smartphone when its CCCD is written with a value of 0x01.
- **Indication:** The GATT server sends this type of message. The GATT client always confirms it. For example, a Bluetooth® LE Health Thermometer Service (**HTS**) uses indications to reliably send the measured temperature value to a health thermometer collector, such as a smartphone.

13.8 Generic Attribute Profile (GATT)

The GATT defines the ways in which Attributes can be found and used. The GATT operates in one of two roles:

- **GATT client:** The device that requests the data (for example, a smartphone).
- **GATT server:** The device that provides the data (for example, a heart-rate monitor)

Figure 73 shows the client-server architecture in the GATT layer using a heart-rate monitoring device as an example. The heart-rate monitoring device exposes multiple Services (HRS, BAS, and Device Information Service); each Service consists of one or more Characteristics with a Characteristic value and descriptor, as shown in **Figure 70**.

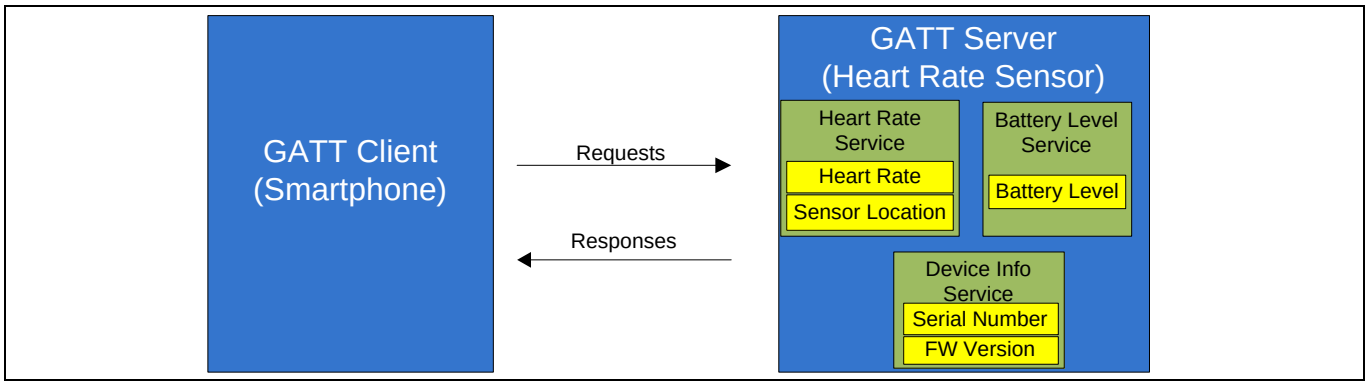


Figure 73 GATT Client-Server architecture

After the Bluetooth® LE connection is established at the link-layer level, the GATT client (which initially knows nothing about the connected Bluetooth® LE device) initiates a process called “service discovery.” As part of the

service discovery, the GATT client sends multiple requests to the GATT server to get a list of all the available Services, Characteristics, and Attributes in the GATT server. When service discovery is complete, the GATT client has the required information to modify or read the information exposed by the GATT server using the Attribute operations described in the previous section.

13.9 Generic Access Profile (GAP)

The GAP layer provides device-specific information such as the device address; device name; and the methods of discovery, connection, and bonding. The Profile defines how a device can be discovered, connected, the list of Services available, and how the Services can be used. **Figure 75** shows an example of a Heart Rate Profile.

The GAP layer operates in one of four roles:

- **Peripheral:** This is an advertising role that enables the device to connect with a GAP Central. After a connection is established with the Central, the device operates as a slave. For example, a heart-rate sensor reporting the measured heart-rate to a remote device operates as a GAP Peripheral.
- **Central:** This is the GAP role that scans for advertisements and initiates connections with Peripherals. This GAP role operates as the master after establishing connections with Peripherals. For example, a smartphone retrieving heart-rate measurement data from a Peripheral (heart-rate sensor) operates as a GAP Central.
- **Broadcaster:** This is an advertising role that is used to broadcast data. It cannot form Bluetooth® LE connections and engage in data exchange (no request/response operations). This role works similar to a radio station in that it sends data continuously whether or not anyone is listening; it is a one-way data communication. A typical example of a GAP Broadcaster is a beacon, which continuously broadcasts information but does not expect any response.
- **Observer:** This is a listening role that scans for advertisements but does not connect to the advertising device. It is the opposite of the Broadcaster role. It works similar to a radio receiver that can continuously listen for information but cannot communicate with the information source. A typical example of a GAP Observer is a smartphone app that continuously listens for beacons.

Figure 74 shows a generic Bluetooth® LE system with Bluetooth® Low Energy Pioneer Kit as the Peripheral and a smartphone as the Central device. The interaction between Bluetooth® LE protocol layers and their roles on the Central and the Peripheral devices are also shown.

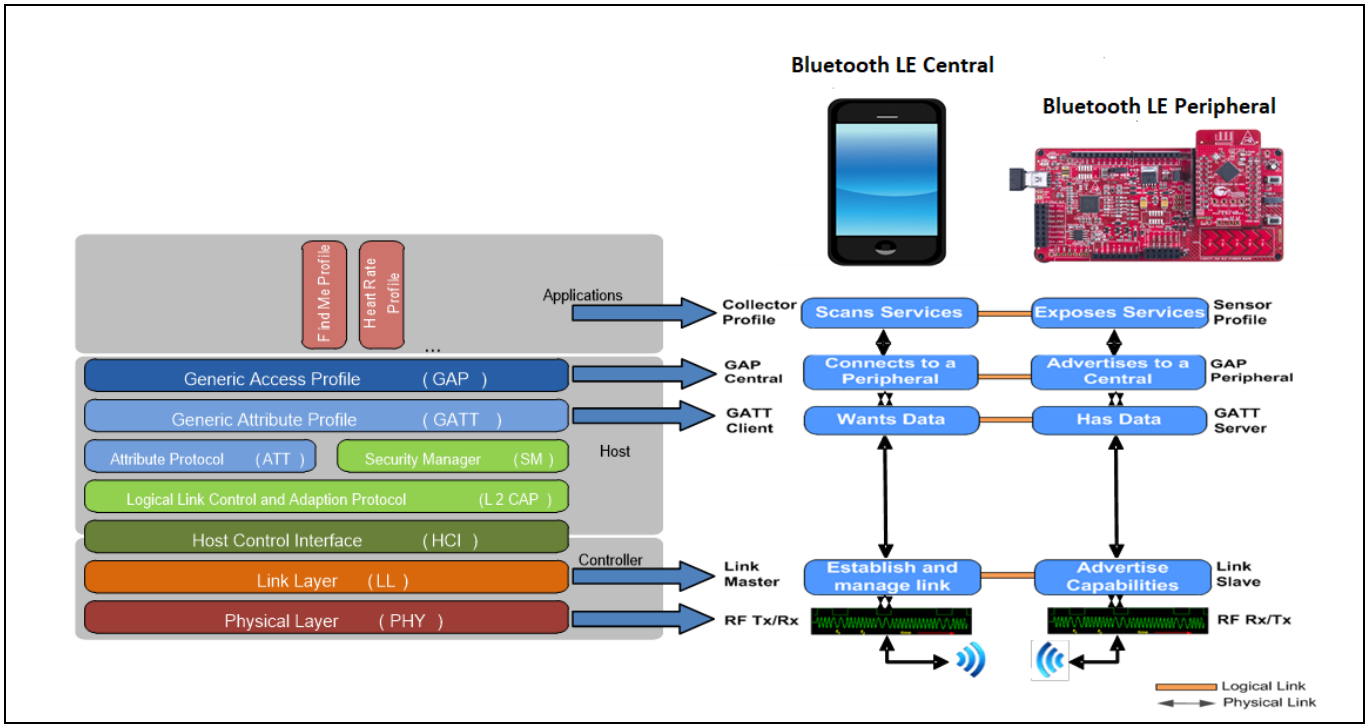


Figure 74 Bluetooth® LE system design

Figure 75 shows an example where a smartphone with a heart-rate app operates as a Central and a heart-rate sensor operates as a Peripheral. The heart-rate monitoring device implements the Heart-Rate Sensor Profile, while the smartphone receiving the data implements the Heart-Rate Collector Profile.

In this example, the Heart-Rate Sensor Profile implements two standard Services. The first is a Heart Rate Service that comprises three Characteristics (the Heart Rate Measurement Characteristic, the Body Sensor Location Characteristic, and the Heart Rate Control Point Characteristic). The second Service is a Device Information Service. At the link layer, the heart-rate measurement device is the slave and the smartphone is the master. See the Bluetooth developer portal for a detailed description of the Heart Rate **Service** and **Profile**.

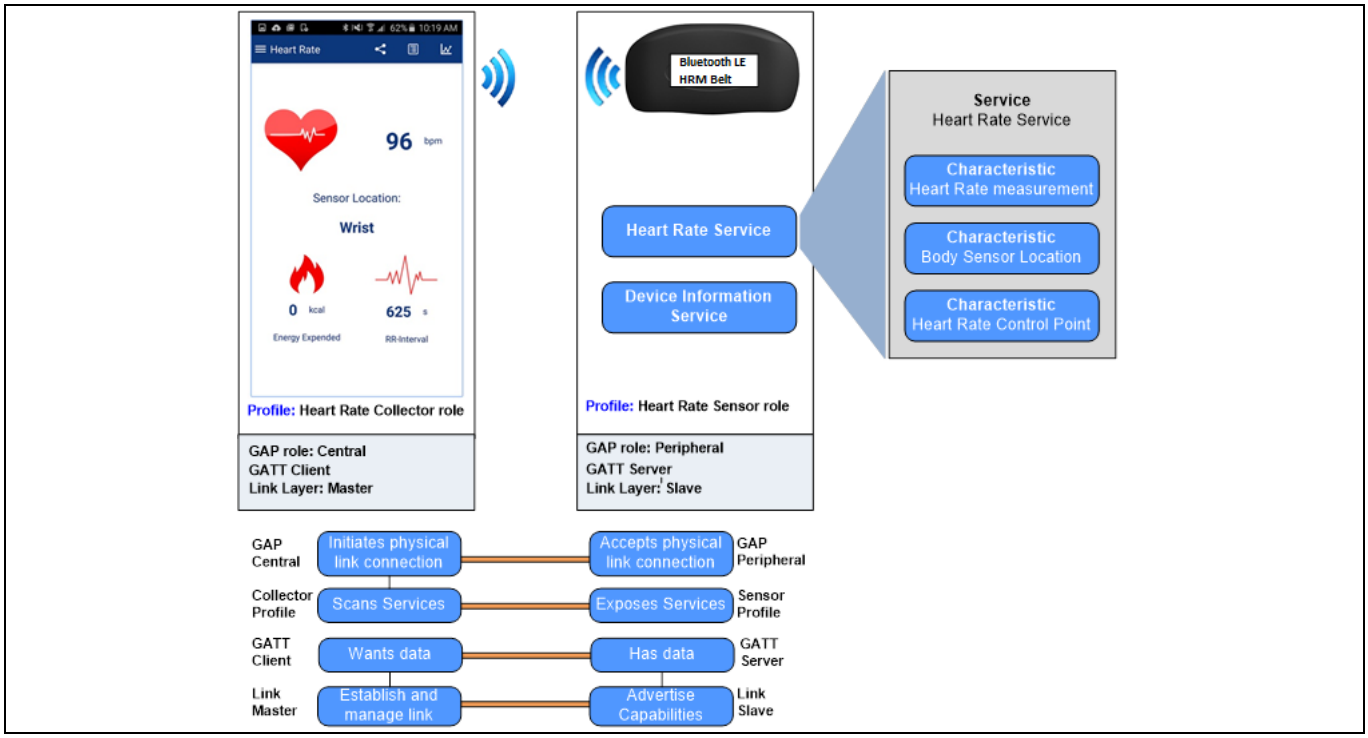


Figure 75 Bluetooth® LE heart-rate monitor system

Related application notes

[1] **AN94020 - Getting started with PRoC Bluetooth® Low Energy**

[2] **AN96841 - Getting started with EZ-Bluetotth® LE Creator modules**

[3] **AN91184 - PSoC™ 4-BLE - Designing Bluetooth® Low Energy applications**

[4] **AN91162 - Creating a Bluetooth® Low Energy custom profile**

[5] **AN92584 - Designing for low power and estimating battery life for Bluetooth® Low Energy applications**

[6] **AN95089 - PSoC™ 4/PRoC Bluetooth® Low Energy crystal oscillator selection and tuning techniques**

[7] **AN91445 - Antenna design and RF layout guidelines**

[8] **AN97060 - PSoC™ 4-BLE and PRoC-Bluetooth® LE - Over-The-Air (OTA) Device Firmware Upgrade (DFU) guide**

Revision history

Document version	Date of release	Description of changes
**	2014-11-07	New Application Note
*A	2014-11-12	Fixed hyperlinks and document formatting
*B	2015-03-27	Major reformatting including moving the Bluetooth® LE Protocol and PSoC™ 4-BLE device details to Appendix. Example design simplified and added Deep-Sleep low-power mode.
*C	2015-05-11	Updated Software Version as “PSoC™ Creator™ 3.1 SP3” in page 1. Replaced “PSoC™ Creator 3.1 SP1” with “PSoC™ Creator 3.1 SP3” in all instances across the document. Updated My first PSoC™ 4 with AIROC™ Bluetooth® LE design Updated Stage 1: Create the design Updated description
*D	2015-05-15	Added support for PSoC™ 4-BLE 256K Updated Software Versions as “PSoC™ Creator 3.2” Updated Bluetooth® Low Energy component GUI screenshots Updated CySmart iOS/Android App screenshots Updated Table 2 in Appendix A with WLCSP dimensions
*E	2015-10-15	Rearranged/updated contents of multiple sections to address customer feedback Added PSoC™ is more than a Bluetooth® LE MCU section Updated Bluetooth® LE overview section to clarify GATT roles and data representation Updated to support PSoC™ Creator 3.3 and CySmart 1.1 tool Updated PSoC™ Creator and CySmart screenshots
*F	2015-11-06	Added Bluetooth 4.2 device details Updated to support PSoC™ Creator 3.3 SP1 and CySmart 1.2 tool
*G	2017-04-19	Updated logo and copyright
*H	2018-01-16	Updated template
*I	2021-06-04	Updated to Infineon template Replaced BLE with Bluetooth® LE/Bleutooth® Low Energy and PSoC™ 4 BLE to PSoC™ 4 with AIROC™ Bluetooth® LE across the document Updated kit information in Appendix C (13.1) Updated software version as PSoC™ Creator 4.4 Fixed hyperlinks

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-06-04
Published by
Infineon Technologies AG
81726 Munich, Germany

© 2021 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?
Go to www.cypress.com/support

Document reference
001-91267 Rev. *I

IMPORTANT NOTICE
The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.