

Getting started with AIROC™ CYW20819/20/35 Bluetooth® & Bluetooth® LE

About this document

Scope and purpose

AN225684 introduces you to the AIROC™ CYW20819, CYW20820, and CYW20835 Bluetooth® MCUs, referred to as AIROC™ CYW208xx in the document. These are Bluetooth® 5.2-compliant, Arm® Cortex®-M4 CPU-based ultra-low-power MCUs with support for both Classic Bluetooth® and Bluetooth® Low Energy in case of CYW20819/20 and Bluetooth® Low Energy in CYW20835. This application note helps you get started with the CYW208xx devices with an overview of the device architecture, development kits, and software development tools. It shows how to create a simple Bluetooth® Low Energy application using Eclipse IDE for ModusToolbox™ and the Bluetooth® Software Development Kit (SDK). It also guides you to more resources available online to accelerate your learning on CYW208xx.

Note: AIROC™ CYW20819, CYW20820, and CYW20835 are referred to as CYW208xx in this document.

Table of contents

| | |
|--|-----------|
| About this document | 1 |
| Table of contents | 1 |
| 1 Introduction | 3 |
| 2 CYW20819, CYW20820 overview | 4 |
| 2.1 Device features | 4 |
| 3 CYW20835 overview | 6 |
| 3.1 CYW20835 device features | 6 |
| 3.2 Target applications | 7 |
| 4 Development ecosystem | 8 |
| 4.1 Software ecosystem - ModusToolbox™ | 8 |
| 4.1.1 Getting started with ModusToolbox™ | 8 |
| 4.1.2 Eclipse IDE for ModusToolbox™ | 8 |
| 4.1.3 AIROC™ Bluetooth® SDK for CYW208xx | 9 |
| 4.1.4 Configurators..... | 10 |
| 4.2 Development kits | 11 |
| 4.2.1 CYW920819M2EVB-01..... | 11 |
| 4.2.2 CYW920820M2EVB-01..... | 12 |
| 4.2.3 CYW920835M2EVB-01..... | 13 |
| 5 My first CYW208xx Bluetooth® Low Energy application | 14 |
| 5.1 Prerequisites..... | 14 |
| 5.1.1 Download the code example | 15 |
| 5.2 About the design | 16 |
| 5.3 Part 1: Create a new application..... | 17 |
| 5.3.1 Select a new workspace..... | 17 |
| 5.3.2 Create a new ModusToolbox™ application | 17 |
| 5.3.3 Select CYW20820-based target hardware | 18 |

Table of contents

| | | |
|----------|---|-----------|
| 5.3.4 | Import the Bluetooth® LE Find Me code example (Applicable only for the “Using CE directly” flow) | 19 |
| 5.3.5 | Select a starter application and create the application (Applicable only for “Working from Scratch” flow) | 20 |
| 5.4 | Part 2: Configure design resources | 21 |
| 5.4.1 | Configure hardware resources | 22 |
| 5.5 | Common application settings | 28 |
| 5.5.1 | BT_DEVICE_ADDRESS | 28 |
| 5.5.2 | UART | 28 |
| 5.5.3 | ENABLE_DEBUG | 28 |
| 5.6 | Library Manager..... | 29 |
| 5.7 | Part 3: Write the application code | 30 |
| 5.8 | Firmware description | 31 |
| 5.8.1 | Bluetooth® Low Energy GATT database | 31 |
| 5.8.2 | Bluetooth® stack configuration parameters | 31 |
| 5.8.3 | User application code entry..... | 32 |
| 5.8.4 | Bluetooth® stack events..... | 32 |
| 5.8.4.1 | Bluetooth® stack management events | 32 |
| 5.8.4.2 | GATT events..... | 34 |
| 5.8.5 | User interface logic | 35 |
| 5.9 | Part 4: Build, program, and test your design | 36 |
| 6 | Summary | 40 |
| | References..... | 41 |
| | Revision history..... | 42 |

1 Introduction

Applications featuring Bluetooth® connectivity are trending towards lower-power application-level features, and lower BoM cost and smaller board space. The CYW208xx Bluetooth® MCU enables you to meet all these critical requirements.

In terms of the core Bluetooth® functionality, the devices are:

- Bluetooth® 5.2-compliant with CYW20819/20
- Having support for dual-mode Bluetooth® operation
- Bluetooth® Low Energy (1 Mbps and 2 Mbps) and (EDR 2 Mbps and 3 Mbps) data rates.

CYW20835, on the other hand, supports single-mode Bluetooth® operation, Bluetooth® Low Energy (1 Mbps and 2 Mbps). CYW208xx devices fully implement the Bluetooth® Mesh 1.0 specification.

CYW208xx also supports value-added application features by having a powerful Arm® Cortex®-M4 CPU and a host of peripheral blocks, such as ADC, SPI, UART, and I²C that aid in interfacing external onboard sensors. These peripheral blocks, on-chip flash memory, and integrated buck and LDO regulators enable reduced BoM cost and smaller PCB footprint. Infineon's advanced CMOS manufacturing process and the support for various system power modes allow you to design battery-operated, low-power applications using CYW208xx.

The device is used in audio (source), sensors (medical, home, security), HID, and remote-control functionality, and a host of other IoT applications.

This application note will help you get started with the CYW208xx device, and covers the following:

- Overview of the [CYW20819 and CYW20820 device features and architecture](#)
- Overview of the [CYW20835 device features and architecture](#)
- Overview of the [development ecosystem](#) support for the CYW208xx device includes software development platforms, hardware evaluation platforms, code examples, application notes, and other related technical documents.
- Tutorial on how to [develop applications](#) using the CYW20819 device in Eclipse IDE for ModusToolbox™ by going through the step-by-step process of creating a simple Bluetooth® Low Energy-based application from scratch.

This application note does not cover the Bluetooth® Low Energy or classic Bluetooth® protocol. It assumes that the reader is already familiar with the basics of these protocols.

2 CYW20819, CYW20820 overview

CYW20819 is a Bluetooth® 5.2-compliant MCU with an integrated 2.4-GHz transceiver supporting both Bluetooth® Low Energy and Classic Bluetooth® (BR, EDR). **Figure 1** shows the high-level architectural block diagram of the device.

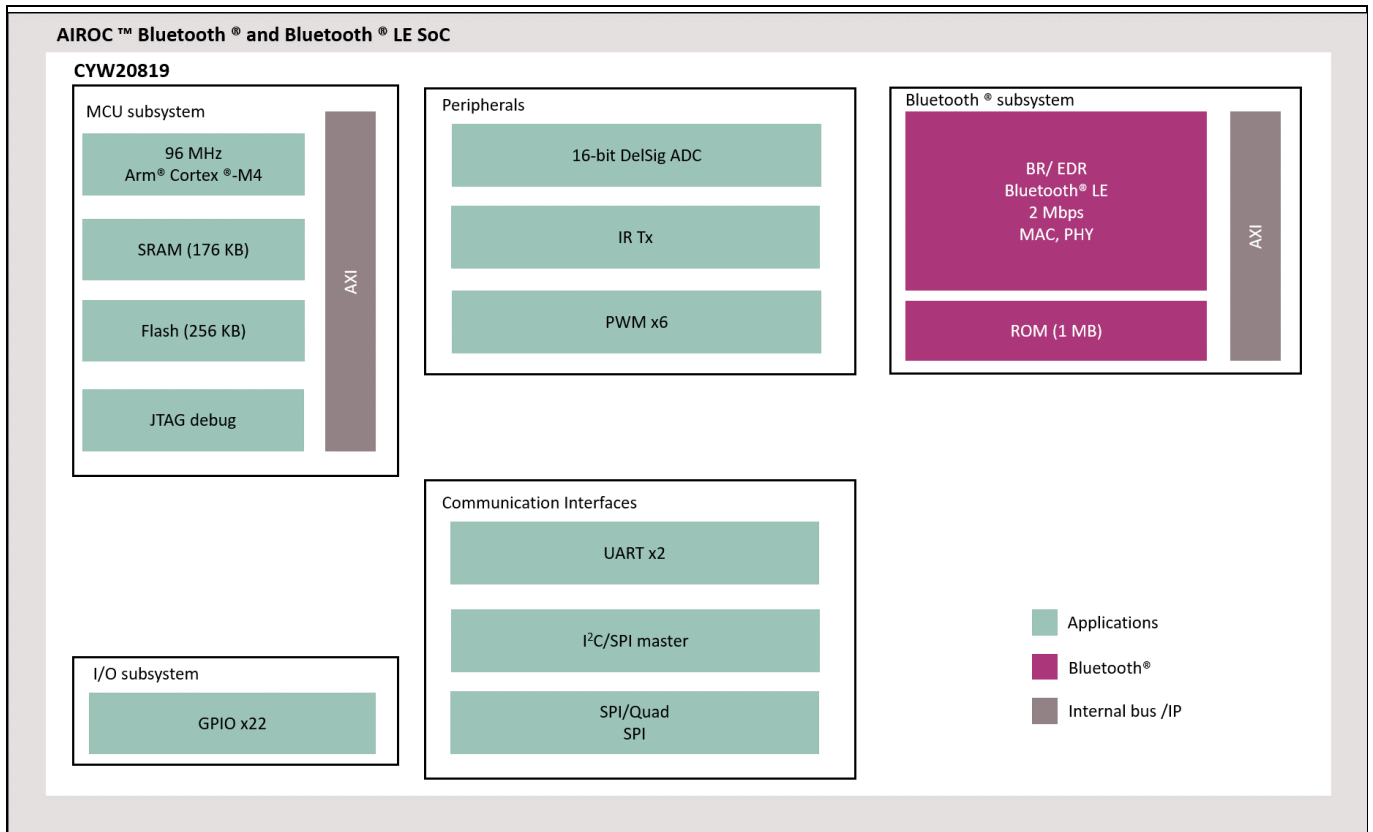


Figure 1 CYW20819 block diagram

2.1 Device features

- Bluetooth® subsystem
 - Complies with Bluetooth® core specification version 5.2
 - Includes support for BR, EDR 2 Mbps and 3 Mbps, eSCO, Bluetooth® LE, and LE 2 Mbps.
 - Programmable TX power up to +5 dBm (BR)
 - Excellent receiver sensitivity (-95 dBm for Bluetooth® LE 1 Mbps)
- Microcontroller
 - Powerful Arm® Cortex®-M4 core at up to 96 MHz
 - Bluetooth® stack in ROM allowing standalone operation without any external MCU
 - 256 KB on-chip flash
 - 176-KB on-chip RAM
 - Bluetooth® stack, peripheral drivers, and security functions built into ROM (1 MB), allowing the application to use on-chip flash efficiently
 - AES-128 and true random number generator (TRNG)
 - Security functions in ROM, including ECDSA signature verification
 - Over-the-air (OTA) firmware updates

CYW20819, CYW20820 overview

- Peripherals
 - Up to 22 GPIOs
 - I²C, I²S, UART, and PCM interfaces
 - Two Quad-SPI interfaces
 - Auxiliary ADC with up to 28 analog channels
 - Programmable key scan 20 x 8 matrix
 - 3-axis quadrature signal decoder
 - General-purpose timers and PWM
 - Real-time clock (RTC) and watchdog timers (WDT)
- Power management and clocking
 - On-chip power-on reset (POR), integrated buck (DC-DC), and LDO regulators
 - On-chip software-controlled power management unit (PMU)
 - Multiple system power modes to optimize power consumption
 - 24-MHz external crystal oscillator for device operation (±20 ppm accuracy)
 - On-chip 32-kHz low-power oscillator (LPO) with optional external 32-kHz crystal oscillator support
- Wi-Fi coexistence
 - Global Coexistence Interface (GCI) for use with Infineon Wi-Fi parts
 - Serial Enhanced Coexistence Interface (SECI) for use with SECI-compatible Wi-Fi parts
- CYW20820 has an internal power amplifier that allows it to achieve Tx output power up to +10 dBm as opposed to CYW20819, which goes up to +4 dBm.

For more information on the device, including the electrical specifications, see the device datasheet of [CYW20819](#) and [CYW20820](#) respectively.

3 CYW20835 overview

CYW20835 is a Bluetooth® 5.2-compliant MCU having a standalone baseband processor with an integrated 2.4-GHz transceiver supporting Bluetooth® Low Energy and includes an internal power amplifier (iPA) that provides 12-dBm output power to give an excellent range. **Figure 2** shows the high-level architectural block diagram of the device.

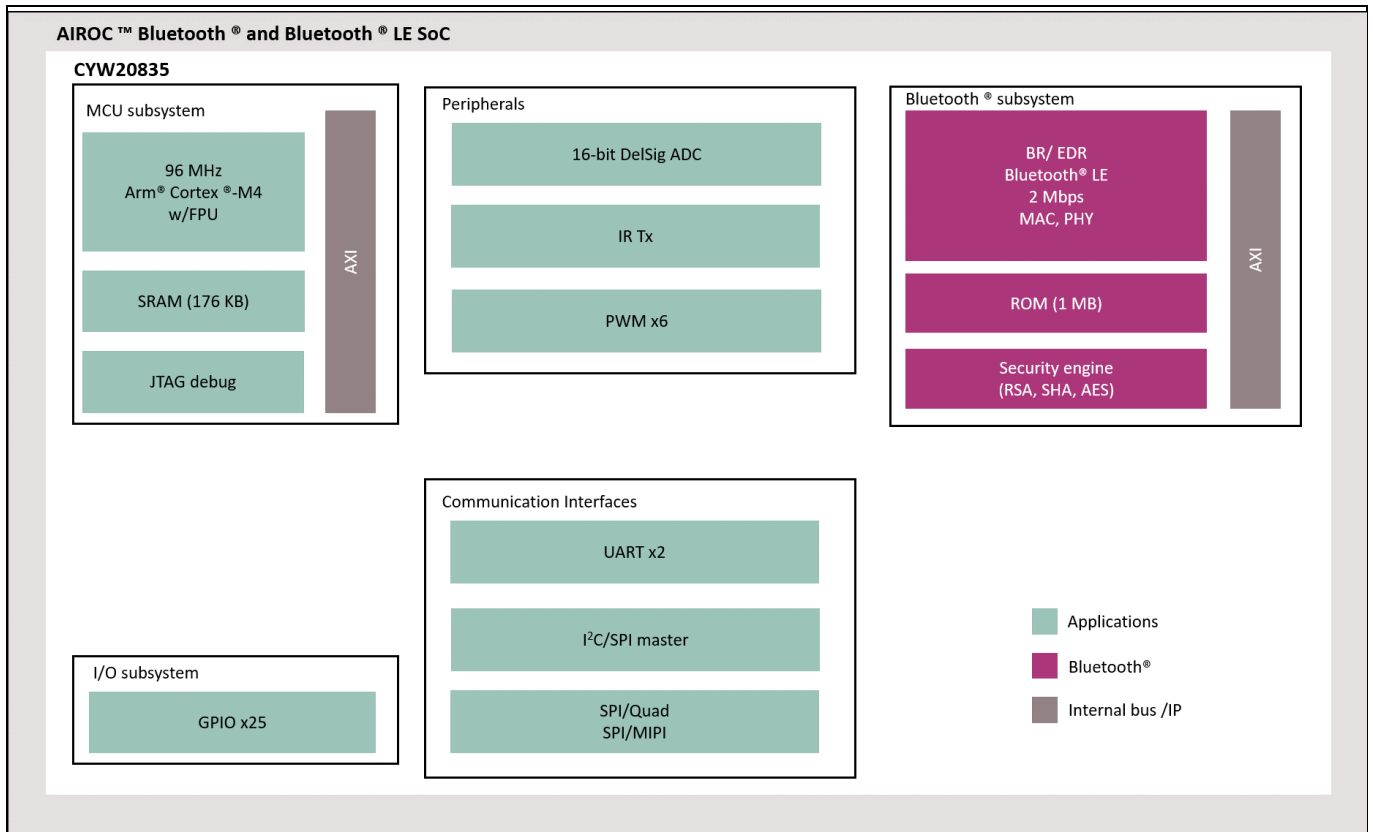


Figure 2 CYW20835 block diagram

3.1 CYW20835 device features

- Bluetooth® subsystem
 - Complies with Bluetooth® core specification version 5.2 with LE 2-Mbps support
 - Supports Adaptive Frequency Hopping (AFH)
 - Programmable Tx power up to 12 dBm
 - Rx sensitivity -94.5 dBm (Bluetooth® LE)
- Microcontroller
 - 96-MHz Arm® Cortex®-M4 microcontroller unit MCU with Floating-Point Unit (FPU)
 - Supports Serial Wire Debug (SWD)
 - Runs Bluetooth® stack and application
- Peripherals
 - 6x 16-bit PWMs
 - 24 GPIOs
 - Analog and digital microphone interfaces
 - I2C, I2S, UART, SPI, and PCM interfaces

CYW20835 overview

- Up to 8x20 programmable key-scanning matrix interface
- Wi-Fi coexistence
 - Support for global coexistence interface for easy coexistence implementation with select Infineon Wi-Fi devices
 - For more information on the device, including the electrical specifications, see the [Device datasheet](#).

3.2 Target applications

- Wearables and fitness bands
- Audio source applications
- Bluetooth® Low Energy Mesh home and Industrial automation
- Blood pressure monitors and other medical applications
- Proximity sensors
- Key fobs
- Thermostats and thermometers
- Toys
- Remotes

4 Development ecosystem

4.1 Software ecosystem - ModusToolbox™

ModusToolbox™ software provides support for many types of devices and environments. From a practical standpoint, ModusToolbox™ software delivers in various ways, such as installation resources, code examples, BSPs and libraries; you only use the resources you need. When you create applications, you use these resources and interact with the hardware through the Hardware Abstraction Layer (HAL) and/or the Peripheral Driver Library (PDL).

Figure 3 shows the high-level view of the tools included in the ModusToolbox™ software.

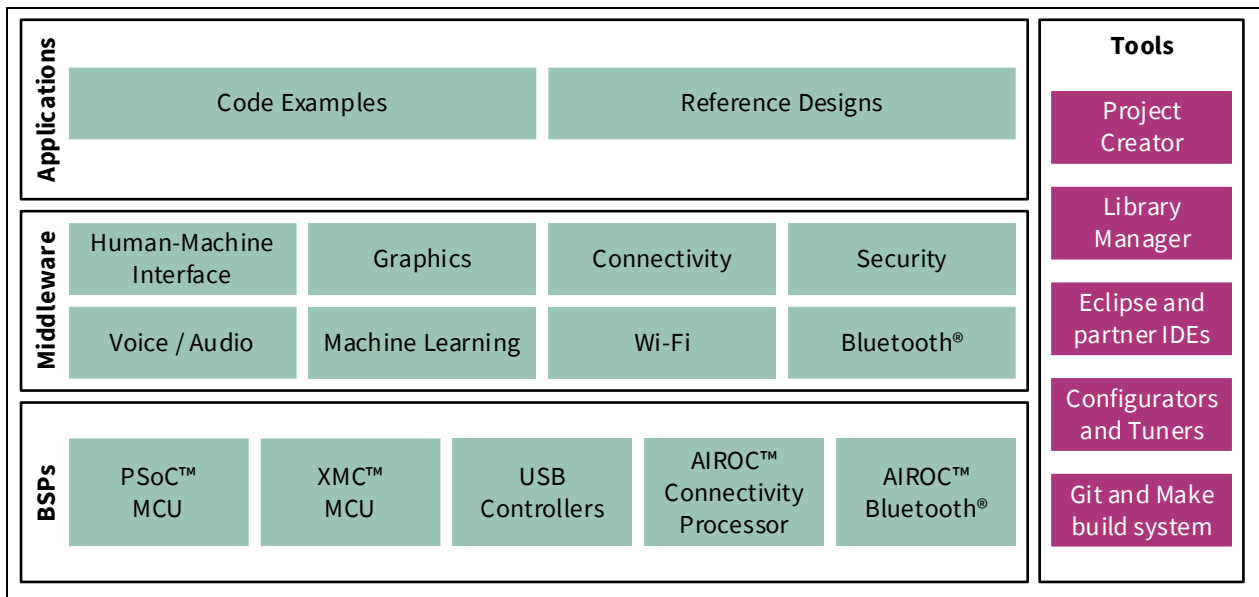


Figure 3 ModusToolbox™ software overview

4.1.1 Getting started with ModusToolbox™

Visit the [ModusToolbox™](#) home page to download and install the latest version of ModusToolbox™. See the ModusToolbox™ installation guide document in the Documentation tab of [ModusToolbox™](#) home page for information on installing the ModusToolbox™ software. After installing, launch ModusToolbox™ and navigate to the following items:

- **User Guide:** The detailed user guide at **Documentation > ModusToolbox™2.4 Documentation > ModusToolbox™ user guide.**
- These documents are also available in the **Documentation** tab of the [ModusToolbox™](#) home page.

4.1.2 Eclipse IDE for ModusToolbox™

Eclipse IDE for ModusToolbox™ is based on the Eclipse IDE “Oxygen” version. It uses several plugins, including the Eclipse C/C++ Development Tools (CDT) plugin. The [Eclipse Survival Guide](#) provides tips and hints for using Eclipse IDE for ModusToolbox™.

The IDE contains Eclipse-standard menus and toolbars, plus various panes such as the Project Explorer, Code Editor, and Console, as shown in [Figure 4](#). One difference from the standard Eclipse IDE is the “ModusToolbox™ Perspective.” This perspective provides the “Quick Panel,” a “News View,” and adds tabs to Project Explorer.

Development ecosystem

The top-level entity that you ultimately program to a device is called an application in the IDE. The application consists of one or more Eclipse projects. The IDE handles all dependencies between projects automatically. It also provides hooks for launching various tools provided by the software development kits (SDKs).

With Eclipse IDE for ModusToolbox™, you can:

1. Create a new application based on a list of starter applications filtered by kit or device, or browse the collection of code examples online.
2. Configure device resources to build your hardware system design in the workspace.
3. Add software components or middleware.
4. Develop your application firmware.

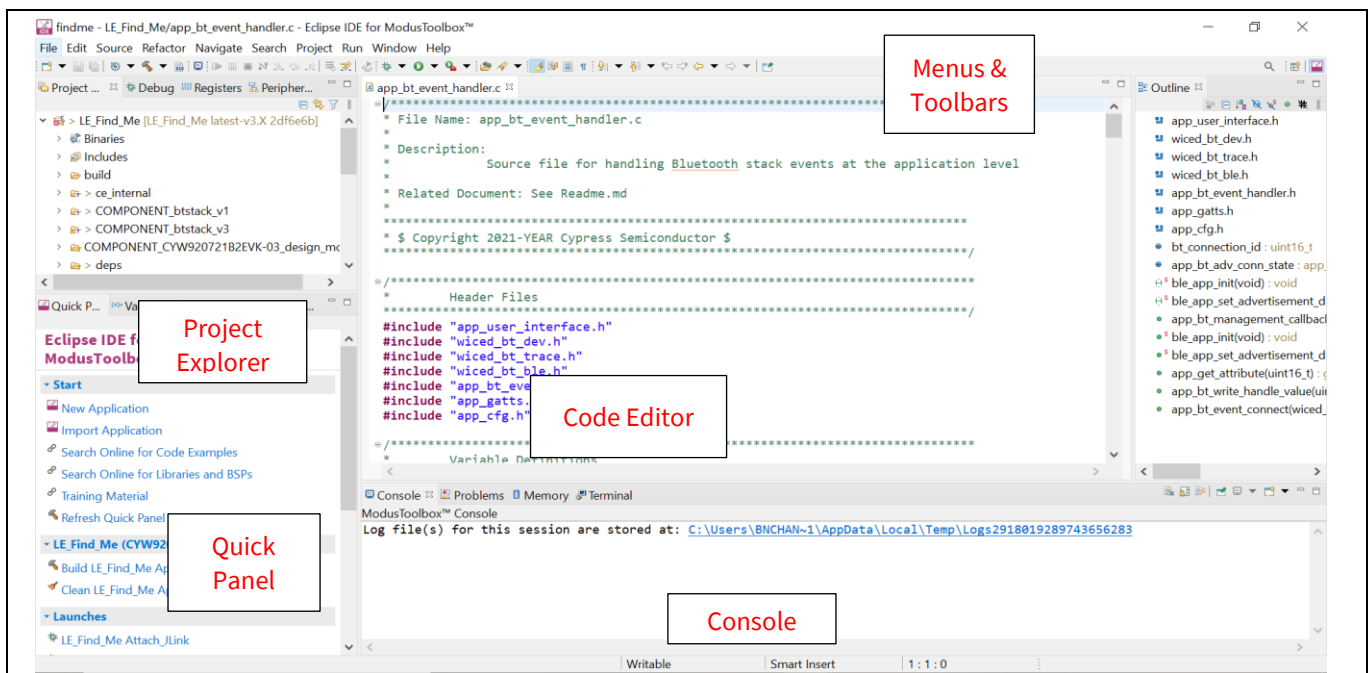


Figure 4 Eclipse IDE for ModusToolbox™ overview

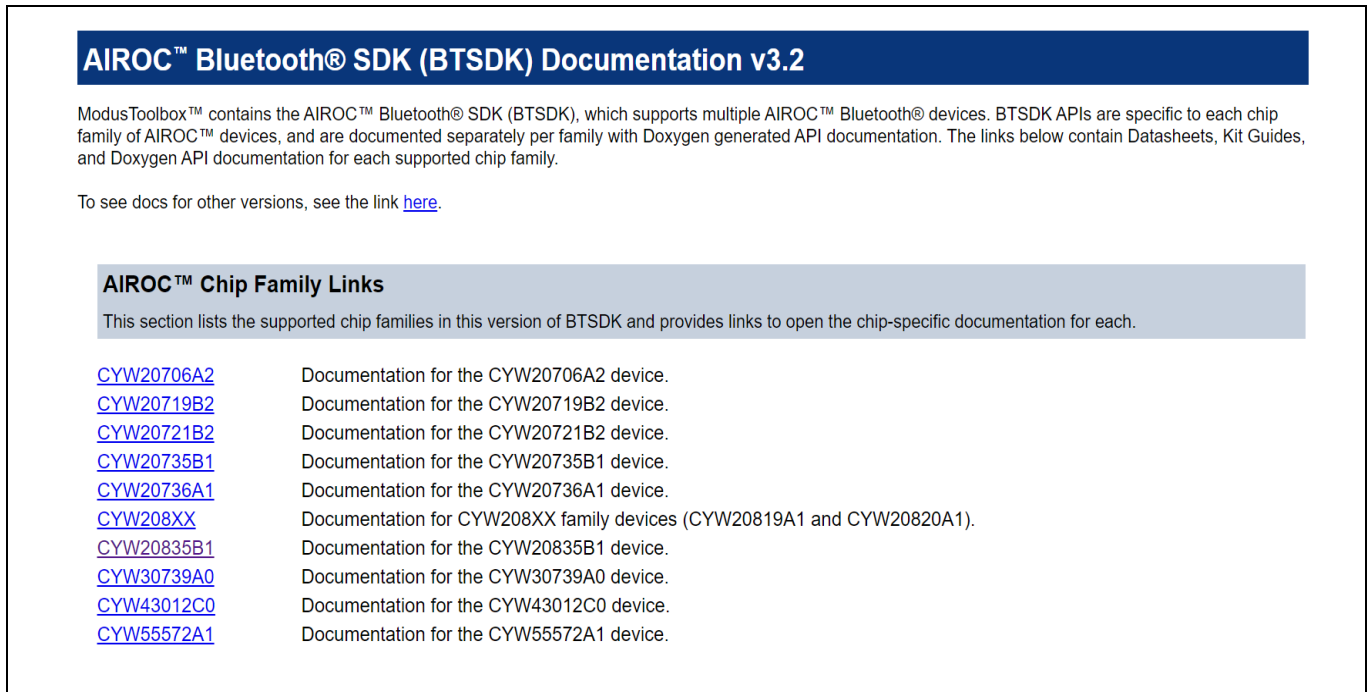
4.1.3 AIROC™ Bluetooth® SDK for CYW208xx

SDKs provide the central core of the ModusToolbox™ software. SDKs make it easier to develop firmware for supported devices without understanding all the intricacies of the device resources. An SDK contains configuration tools, drivers, libraries, middleware, various utilities, Makefiles, and scripts. BTSDK enables CYW208xx application development within ModusToolbox™. These are the following features of BTSDK:

- Dual-mode Bluetooth® stack included in the device ROM (BR/EDR and Bluetooth® Low Energy)
- Bluetooth® stack and profile level APIs for embedded Bluetooth® application development (Host and Controller on the same device)
- WICED HCI protocol to simplify developing applications that require interfacing with more powerful processors
- APIs and drivers to access on-chip peripheral blocks such as SPI, UART, and ADC.
- Bluetooth® protocols supported include Generic Access Profile (GAP), Generic Attribute Profile (GATT), Security Manager Protocol (SMP), Radio Frequency Communication protocol (RFCOMM), Service Discovery Protocol (SDP), and Bluetooth® Low Energy Mesh protocol.
- Bluetooth® Low Energy and BR/EDR profile APIs, libraries, and sample applications
- Support for over-the-air (OTA) upgrade

Development ecosystem

AIROC™ Bluetooth® SDK: In the IDE, choose **AIROC™ Bluetooth® SDK documentation > CYW208xx> WICED CYW208xx Documentation**. This HTML format guide gives you information on the SDK APIs related to the CYW20820 device. The API functions are logically grouped into components, as shown in **Figure 5**, for easy navigation and reference.



AIROC™ Bluetooth® SDK (BTSDK) Documentation v3.2

ModusToolbox™ contains the AIROC™ Bluetooth® SDK (BTSDK), which supports multiple AIROC™ Bluetooth® devices. BTSDK APIs are specific to each chip family of AIROC™ devices, and are documented separately per family with Doxygen generated API documentation. The links below contain Datasheets, Kit Guides, and Doxygen API documentation for each supported chip family.

To see docs for other versions, see the link [here](#).

AIROC™ Chip Family Links

This section lists the supported chip families in this version of BTSDK and provides links to open the chip-specific documentation for each.

| | |
|----------------------------|--|
| CYW20706A2 | Documentation for the CYW20706A2 device. |
| CYW20719B2 | Documentation for the CYW20719B2 device. |
| CYW20721B2 | Documentation for the CYW20721B2 device. |
| CYW20735B1 | Documentation for the CYW20735B1 device. |
| CYW20736A1 | Documentation for the CYW20736A1 device. |
| CYW208XX | Documentation for CYW208XX family devices (CYW20819A1 and CYW20820A1). |
| CYW20835B1 | Documentation for the CYW20835B1 device. |
| CYW30739A0 | Documentation for the CYW30739A0 device. |
| CYW43012C0 | Documentation for the CYW43012C0 device. |
| CYW55572A1 | Documentation for the CYW55572A1 device. |

Figure 5 Bluetooth® SDK API reference

See the **AIROC™ API reference** to know the detailed list of features supported in the SDK.

4.1.4 Configurators

ModusToolbox™ software provides graphical applications called “configurators” that make it easier to configure hardware resources. The configurators applicable for CYW208xx are listed below.

- The **Device Configurator** is used to enable/configure the peripherals and the pins used in the application.
- The **Bluetooth® Configurator** generates the Bluetooth® Low Energy GATT database for the application.

In the next section, we see a detailed look at using these configurators as part of a Bluetooth® Low Energy application creation exercise.

4.2 Development kits

4.2.1 CYW920819M2EVB-01

CYW920819M2EVB-01 shown in **Figure 6** is the development kit that supports prototyping and application development using CYW20819.

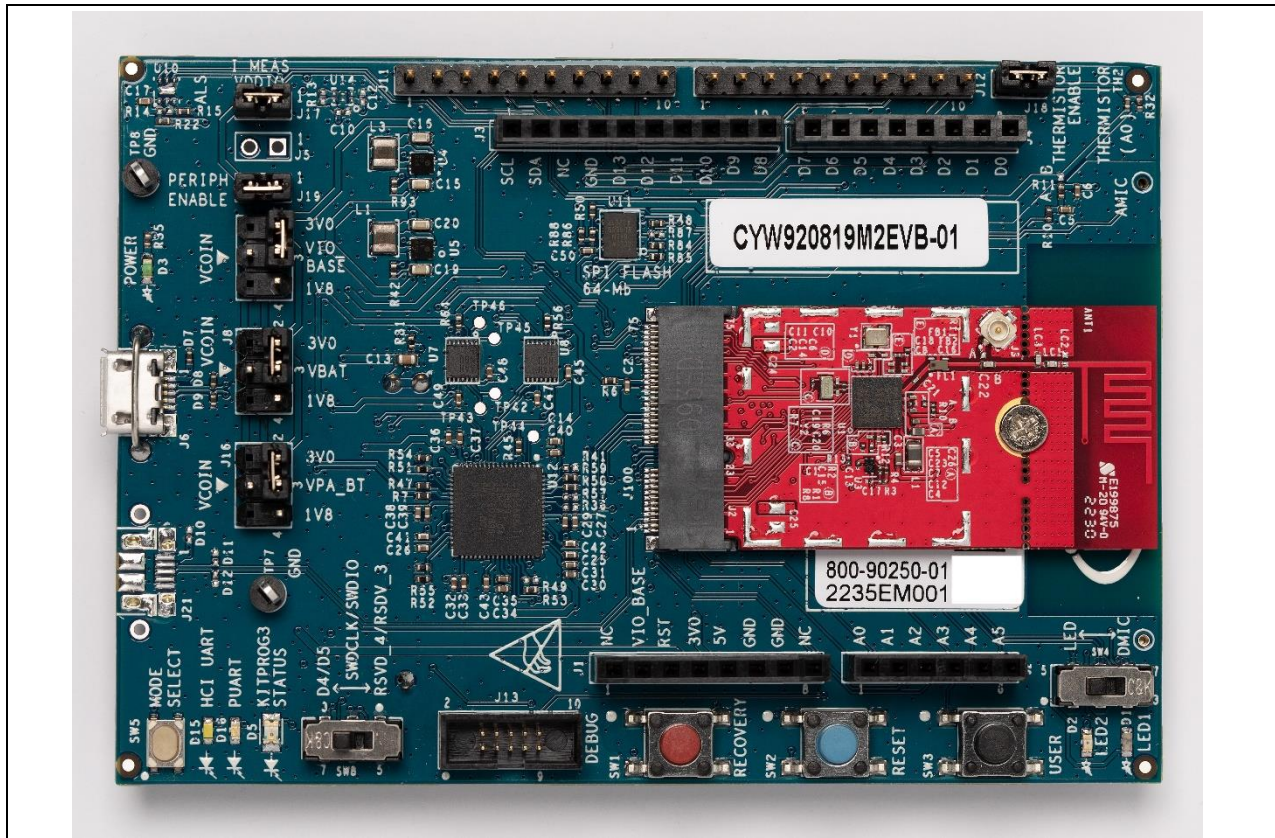


Figure 6 CYW920819M2EVB-01 evaluation board

The kit supports the following key features:

1. CYW20819 carrier board with onboard antenna
2. Expansion headers compatible with Arduino shields
3. Support for 1.8-V and 3.0-V operation of the CYW20819 device
4. Recovery button (orange), Reset button (blue), User button (black), two user LEDs, an analog thermistor, and a motion sensor
5. Onboard micro-USB connector that provides access to the two serial interfaces (HCI UART and Peripheral UART (PUART)) of the CYW20819 device through a USB-to-UART device. Typically, HCI UART uses downloads the application code, and PUART uses printing Bluetooth® stack and application trace messages or interfacing to external devices via UART.

For more information on the kit, including the user guide and schematics, see the **CYW920819M2EVB-01** web page.

4.2.2 CYW920820M2EVB-01

CYW920820M2EVB-01 shown in **Figure 7** is the development kit that supports prototyping and application development using CYW20820.

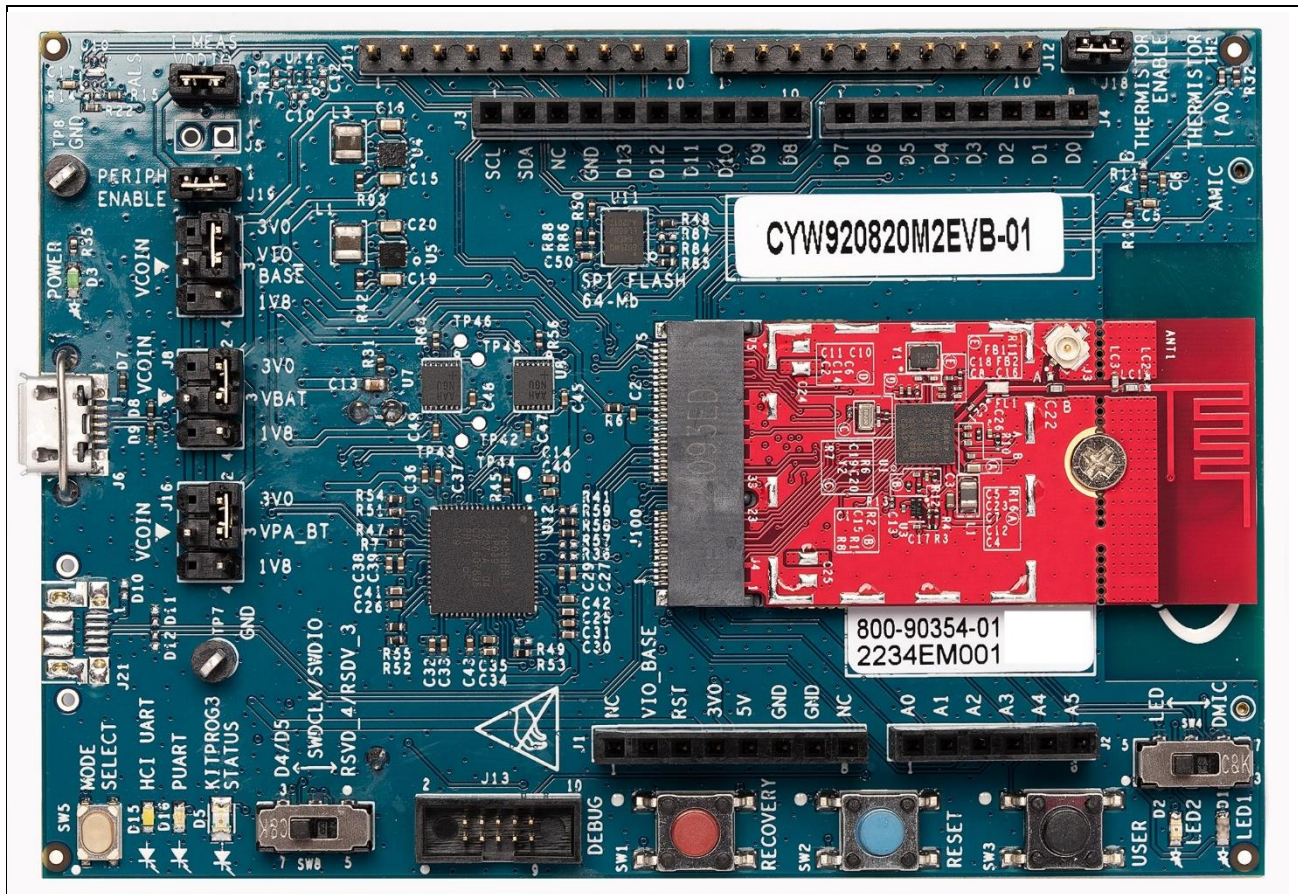


Figure 7 CYW920820M2EVB-01 evaluation board

The kit supports the following key features:

1. CYW20820 carrier board with onboard antenna
2. Expansion headers compatible with Arduino shields
3. Support for 1.8-V and 3.0-V operation of the CYW20820 device
4. Recovery button (orange), Reset button (blue), User button (black), two user LEDs, an analog thermistor, and a motion sensor
5. Onboard micro-USB connector that provides access to the two serial interfaces (HCI UART and Peripheral UART (PUART)) of the CYW20820 device through a USB-to-UART device. Typically, HCI UART uses downloading the application code, and PUART uses printing Bluetooth® stack and application trace messages or interfacing to external devices via UART.

For more information on the kit, including the user guide and schematics, see the [CYW920820M2EVB-01](#) web page.

4.2.3 CYW920835M2EVB-01

CYW920835M2EVB-01 shown in **Figure 8** is the development kit that supports prototyping and application development using CYW20835.

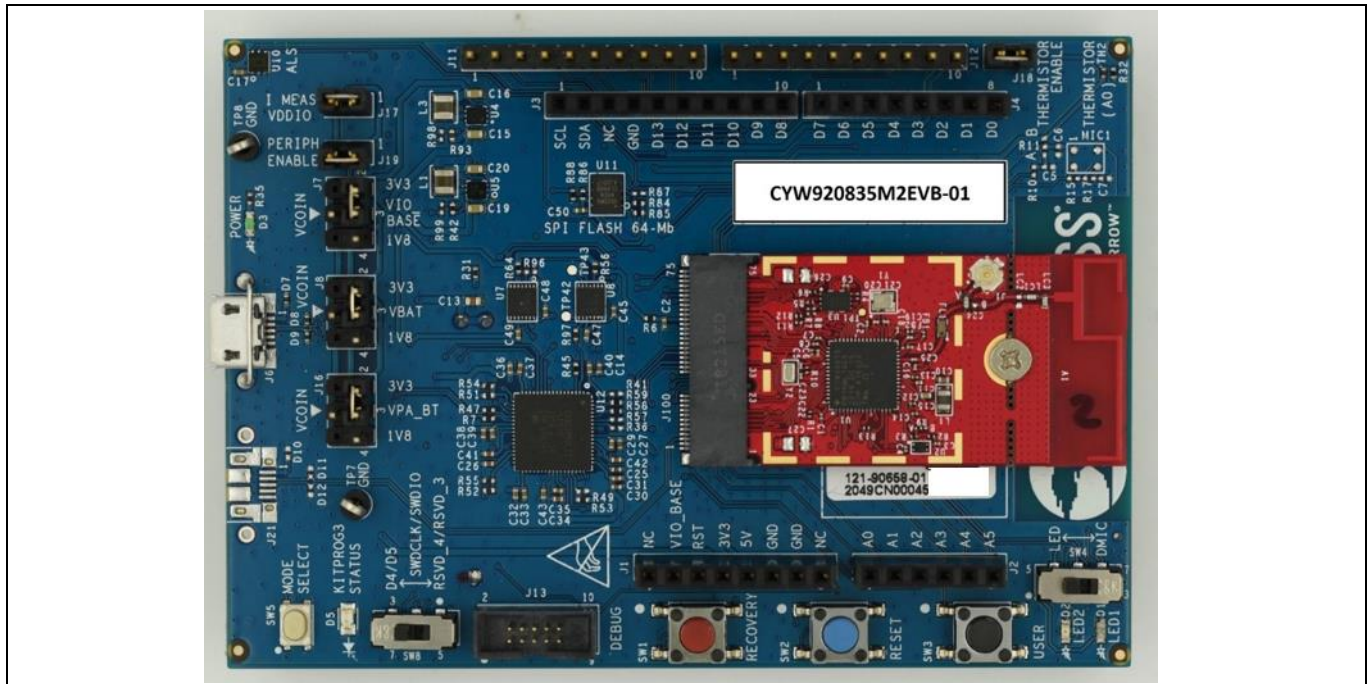


Figure 8 CYW920835M2EVB-01 evaluation kit

The kit supports the following key features:

1. 60-QFN CYW20835 Bluetooth® Low Energy 5.2-compliant MCU
2. Onboard sensors – an ambient light sensor, thermistor, footprint for the analog microphone, and footprint for Infineon digital microphone
3. User switches and LEDs
4. USB connector for power, programming, and USB-UART bridge

For more information on the kit, including the user guide and schematics, see the [CYW920835M2EVB-01](#) web page.

5 My first CYW208xx Bluetooth® Low Energy application

This section provides step-by-step instructions to build a simple Bluetooth® Low Energy-based application for the CYW20820 device using the Eclipse IDE for ModusToolbox™. A Bluetooth® SIG-defined standard profile called **Find Me Profile (FMP)** is implemented in the design. The steps covered in this section are:

- Part 1: Create a new application
- Part 2: Configure design resources
- Part 3: Write the application code
- Part 4: Build, program, and test your design

These instructions require that you use a particular code example (`BLE_FindMe` in this case). However, the extent to which you use the code example (CE) depends on the path you follow through these instructions. Note that the terms Code Example (CE) and application mean the same thing in the context of this document. A Code Example (CE) is simply an existing ModusToolbox™ application that serves a specific purpose or functionality.

We have defined two paths through these instructions depending on what you want to learn:

| Path | “Using CE directly” path (Evaluate existing code example (CE) directly) | “Working from Scratch” path (Use existing code example (CE) as a reference only) |
|----------|---|--|
| Best For | Someone is new to the tool or device and wants to see how it all works quickly. | Someone wants hands-on experience to learn to develop CYW208xx-based Bluetooth® applications in ModusToolbox™. |

What to do for each path is clearly defined at the start of each part of the instructions.

If you start from scratch and follow all instructions in this application note, you use the code example as a reference while following the instructions. Working from scratch helps you learn the design process and takes more time. Alternatively, you can evaluate the existing code example directly to get acquainted with the CYW208xx application development flow in a short time.

It would help if you started by reading Prerequisites and About the design in both cases.

5.1 Prerequisites

Ensure that you have the following items for this exercise.

- **ModusToolbox™** 2.4 or later version installed on your PC.
- **CYW920819M2EVB-01/CYW920820M2EVB-01/ CYW920835M2EVB-01** evaluation board.
- LightBlue **iOS/Android** app or any Android or iOS app that supports the Immediate Alert Service (IAS).

Scan the following QR codes from your mobile phone to download the LightBlue app.



5.1.1 Download the code example

Code examples are available on GitHub. Clone or download the `mtb-example-btsdk-ble-findme` code example repository from [GitHub](#) to a location on your PC. There are three ways you can do this.

1. Download the code examples repository as a zip file by going to the [mtb-example-btsdk-ble-findme](#) and using the clone or download option, as shown in [Figure 9](#). Unzip the downloaded file to a location on your hard drive. This approach is beneficial if you are not familiar with the Git version control system.

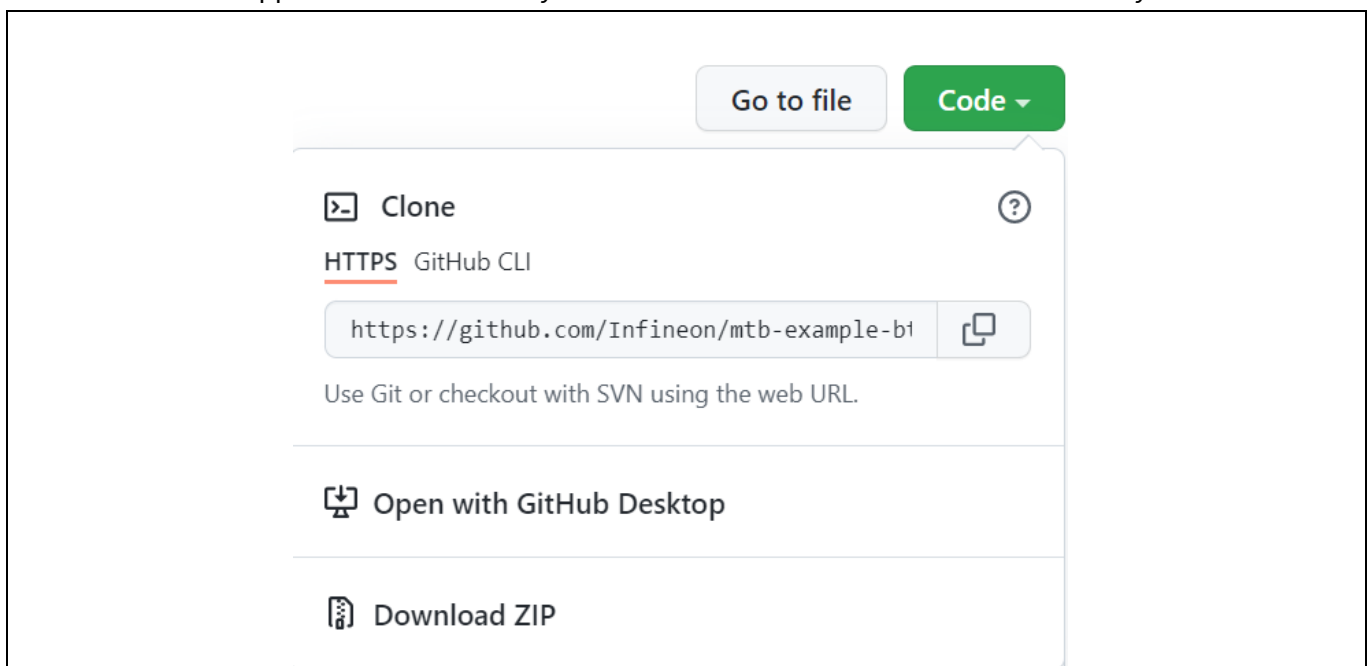


Figure 9 Download code example repository from GitHub

2. In Eclipse IDE for ModusToolbox™, select **File > Import > Git > Projects from Git** and follow the instructions in the wizard. This is standard Eclipse functionality.
3. Use the `git clone` command or the GitHub desktop client.

Once you have the repository on your PC, the Bluetooth® LE Find Me profile code example located inside the repository folder at `mtb-example-btsdk-ble-findme`. You will be using this code example as a reference or directly depending on which development flow you choose to follow in the next sections.

Note: Eclipse IDE for ModusToolbox™ also provides a link to access the online code example in the **Quick Panel**, as shown in [Figure 10](#). Click the **Search Online for Code Examples** link. This opens a web

My first CYW208xx Bluetooth® Low Energy application

page to the GitHub repository to select and download the appropriate repository. You can then use the steps mentioned above to download the required repository.

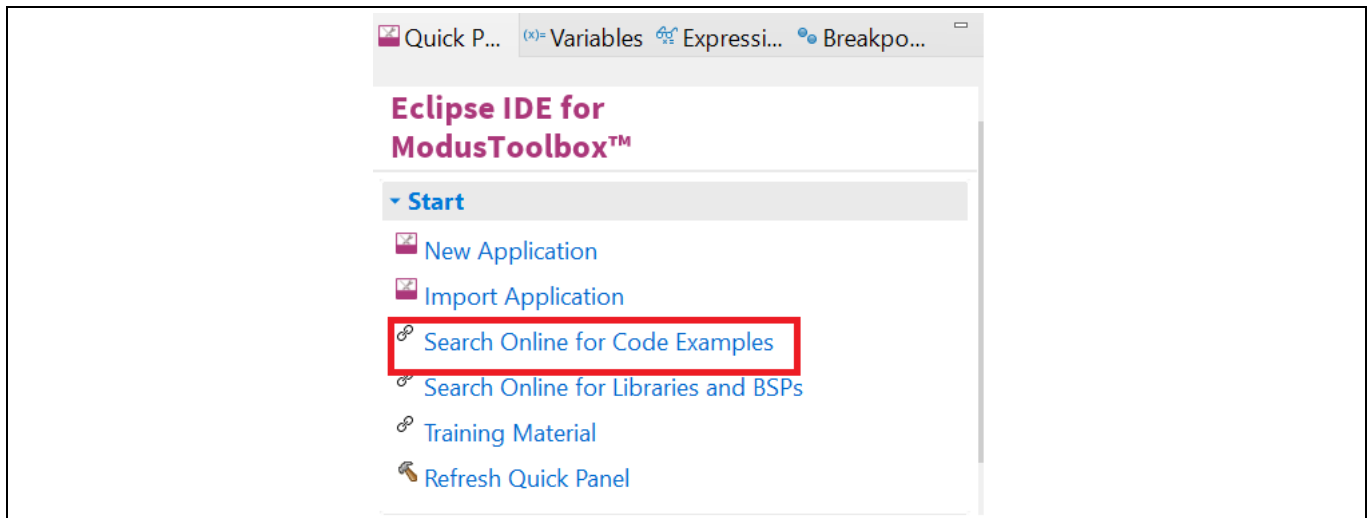


Figure 10 Quick Launch option to search online for code examples

5.2 About the design

This design implements a Bluetooth® Low Energy **Find Me Profile (FMP)** that consists of an Immediate Alert Service (IAS). FMP and IAS are a Bluetooth® Low Energy standard Profile and Service respectively, as defined by the **Bluetooth® SIG**.

The design uses the two LEDs (red LED, yellow LED) on the CYW920820M2EVB-01 kit. The red LED (LED2) displays the IAS alert level – no alert (LED OFF), mild alert (LED blinking), high alert (LED ON). The yellow LED (LED1) indicates whether the Peripheral device (CYW20820) is advertising (LED blinking), connected (LED ON), or disconnected (LED OFF). In addition, a debug UART interface uses sending the Bluetooth® stack and application trace messages.

An iOS/Android mobile device or a PC can act as the Bluetooth® Low Energy Central device, connecting to the Peripheral device.

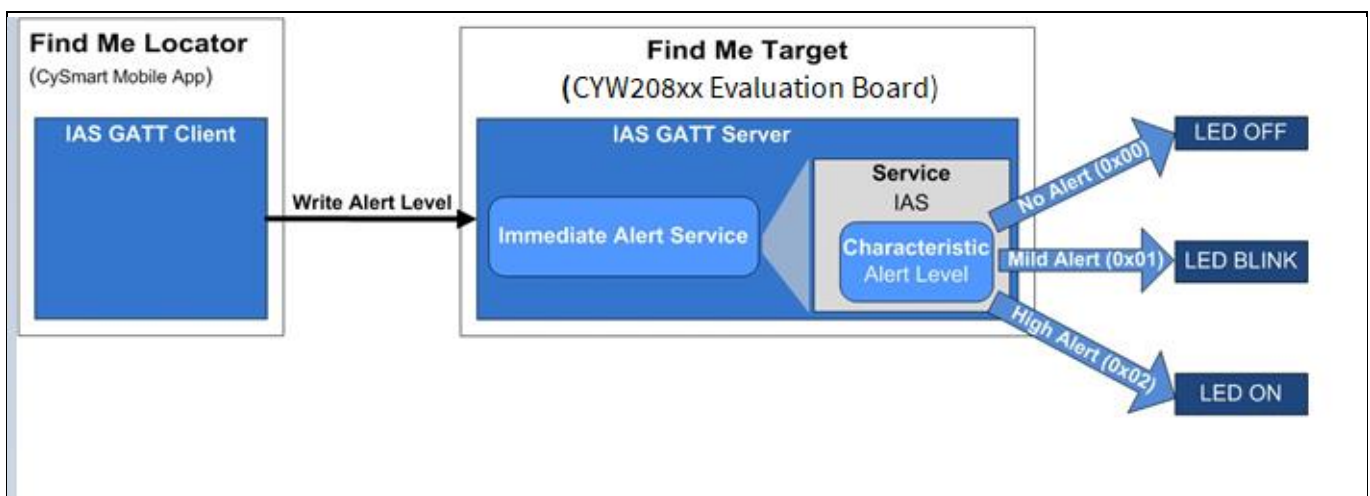


Figure 11 Find Me Profile (FMP) application using CYW208xx

5.3 Part 1: Create a new application

This part takes you step-by-step through creating a new ModusToolbox™ application. Before performing the steps in this section, decide whether you want to import and run the code example as-is or you would instead learn how to create an application from scratch. Depending on your choice, the steps you need to follow are as shown below:

| Path | “Using CE directly” path (Evaluate existing code example (CE) directly) | “Working from Scratch” path (Use existing code example (CE) as reference only) |
|---------|--|--|
| Actions | Follow the sections 5.3.1 , 5.3.2 , 5.3.3 , and 5.3.4 . Ignore section 5.3.5 . | Follow the sections 5.3.1 , 5.3.2 , 5.3.3 , and 5.3.5 . Ignore section 5.3.4 . |

Launch ModusToolbox™ and get started.

5.3.1 Select a new workspace

At launch, ModusToolbox™ presents a dialog to choose a directory for use as the workspace directory. The workspace directory is used to store workspace preferences and development artifacts such as device configuration and application source code.

You can choose an existing empty directory by clicking the **Browse** button, as [Figure 12](#) shows. Alternatively, you can type in a directory name to be used as the workspace directory along with the complete path, and ModusToolbox™ will create the directory for you.

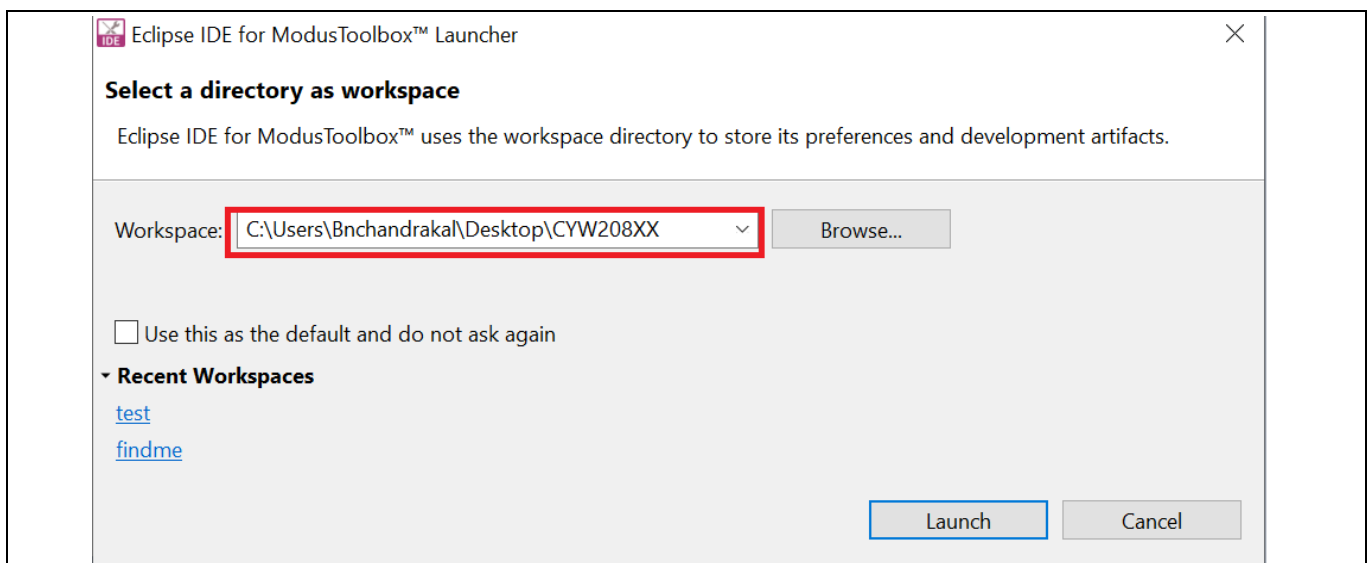


Figure 12 Select a directory as workspace

5.3.2 Create a new ModusToolbox™ application

Click **New Application** in the Start group of the Quick Panel. Alternatively, you can choose **File > New > ModusToolbox™ Application** ([Figure 13](#)).

The Eclipse IDE for ModusToolbox™ Application window appears.

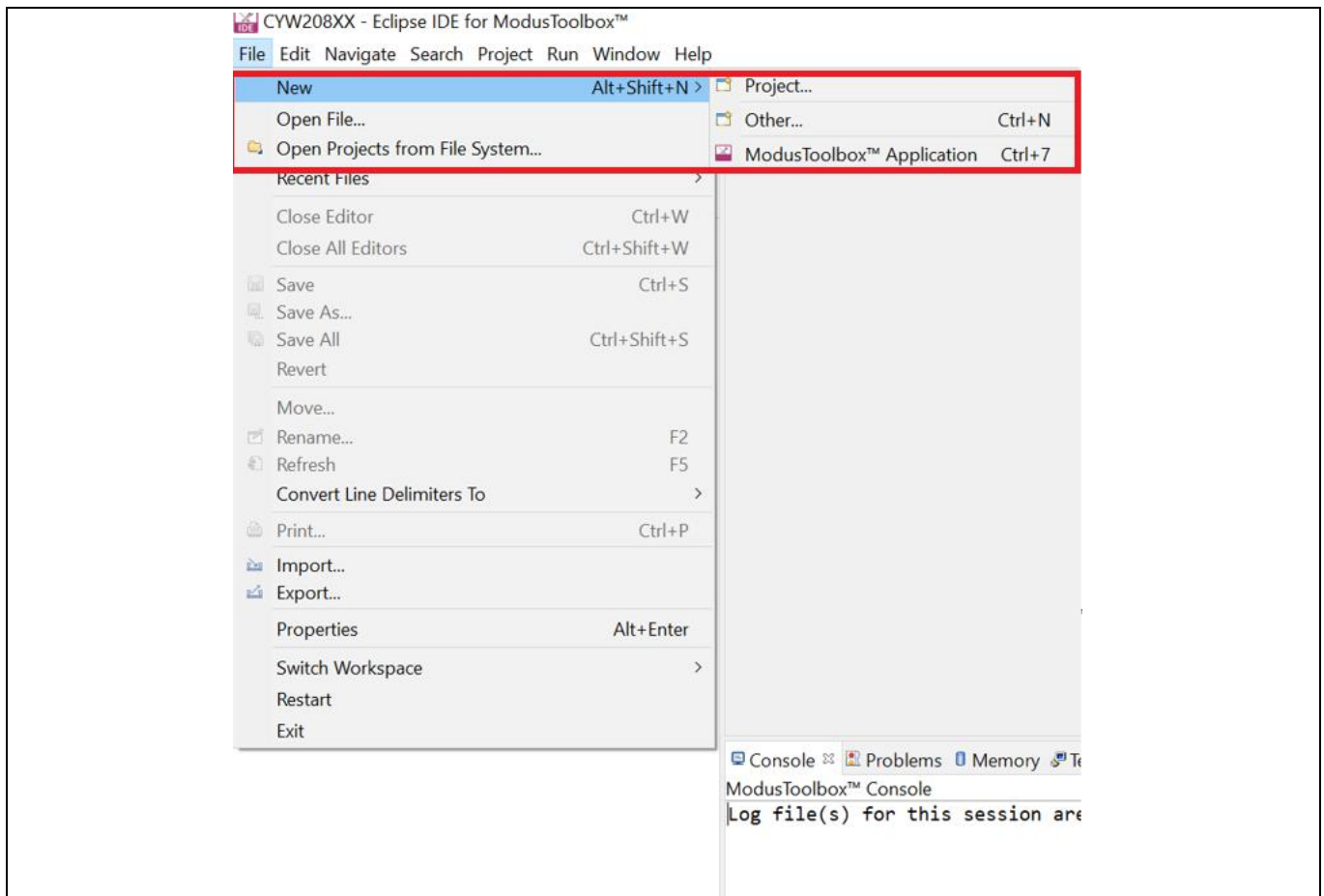


Figure 13 Create a new ModusToolbox™ application

5.3.3 Select CYW20820-based target hardware

ModusToolbox™ presents the list of Infineon kits to start your application development. In this case, we want to develop an application on the CYW920820M2EVB-01 evaluation board that uses the CYW20820 device. Select **CYW920820M2EVB-01**, and click **Next** (Figure 14).

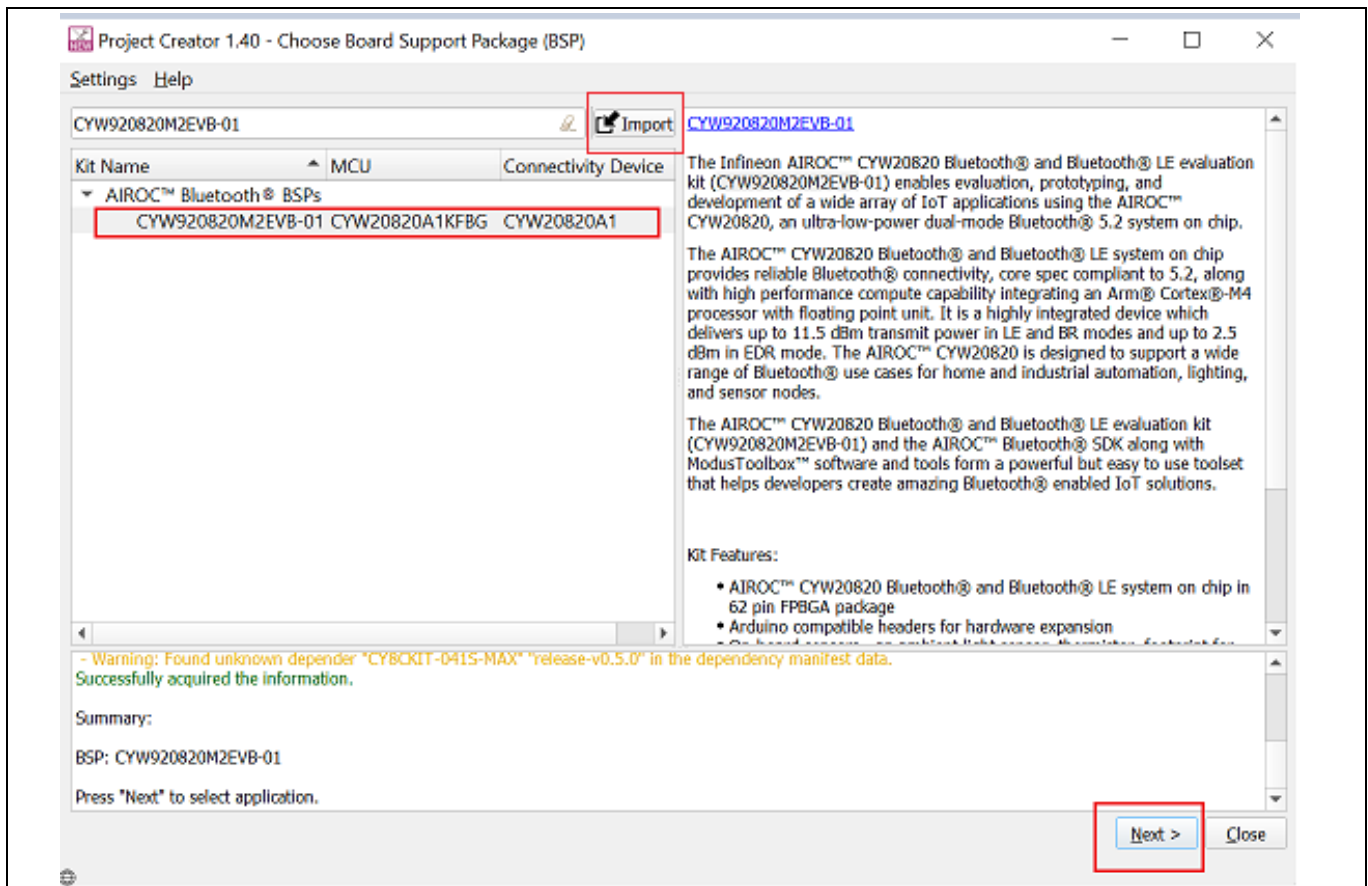


Figure 14 Choose target hardware

Note: **Custom BSP option:** If you have a custom board based on the CYW20820 device, you need to create a Custom BSP and import it via the Import button as shown in [Figure 14](#). To know the steps for creating a Custom BSP refer the document [Creating Custom BSP](#). Even though this application note does not use the “Custom BSP” development flow, the steps that follow can be used as a guide for the “Custom BSP” development flow.

5.3.4 Import the Bluetooth® LE Find Me code example (Applicable only for the “Using CE directly” flow)

The **Import** option is useful to import any code example you downloaded from a repository or received from a colleague into Eclipse IDE for ModusToolbox™. Use this feature to import the Bluetooth® LE Find Me code example for the Using CE directly flow. [Figure 15](#) shows the **Import Bluetooth® LE Find Me Code Example** dialog, click **Import** and navigate to the Bluetooth® LE Find Me code example location in the CE repository you downloaded earlier (mtb-example-btsdk-ble-findme). Select the code example folder. The Bluetooth® LE Find Me example name and description appear in the **Starter Application** dialog. Modify the application name if required. Click **Next** and then **Finish** to complete the application creation process.

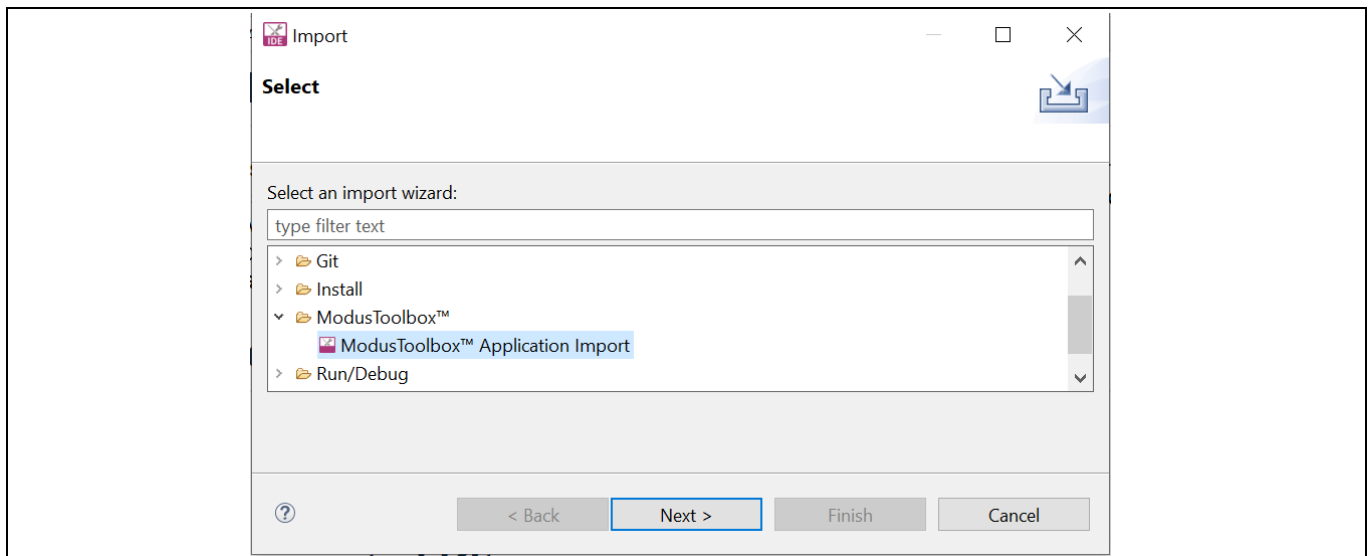


Figure 15 Import Bluetooth® LE Find Me code example

You have successfully imported a new ModusToolbox™ application for CYW20820.

5.3.5 Select a starter application and create the application (Applicable only for “Working from Scratch” flow)

You use an existing template application as the starting point for the Working from Scratch development flow. In the **Starter Application** dialog shown in [Figure 16](#), select **Empty_BTSDK_App**. In the **Name** field, type in a name for the application and click **Next**; the application summary dialog appears. Click **Finish** to let ModusToolbox™ create the application for you.

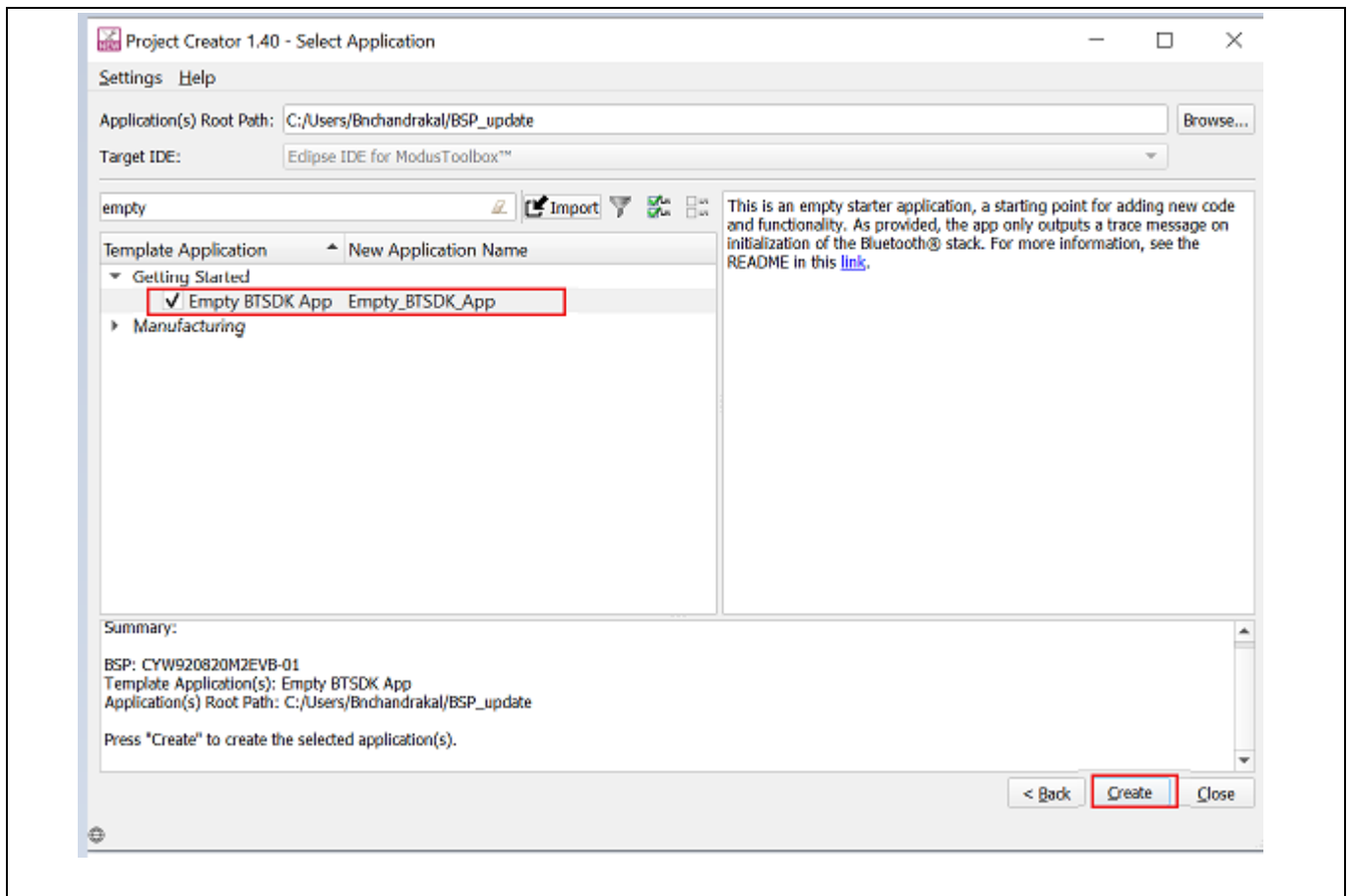


Figure 16 Starter application window

You have successfully created a new ModusToolbox™ application for CYW20820.

5.4 Part 2: Configure design resources

In this step, you will configure the design resources for your application and generate the configuration code.

| Path | “Using CE directly” path (Evaluate existing Code Example (CE) directly) | “Working from Scratch” path (Use existing Code Example (CE) as reference only) |
|---------|---|---|
| Actions | Read and understand all steps. The CE has the resource configurations done, so you need not perform any of the steps in this section. | Perform all steps |

The **Empty_BTSDK_App** application template has all the resources available on the CYW920820M2EVB-01 kit pre-configured and ready for use. These resources include user LEDs, push buttons, and communication peripherals (Bluetooth®, UART, I²C, and SPI). The template application also contains a default application code snippet that initializes the device and the Bluetooth® stack and prints a status message on the Peripheral UART (PUART) interface.

Before proceeding further, a quick tour of the ModusToolbox™ Project Explorer is in order. **Figure 17** shows the ModusToolbox™ Project Explorer view after the template application is created.

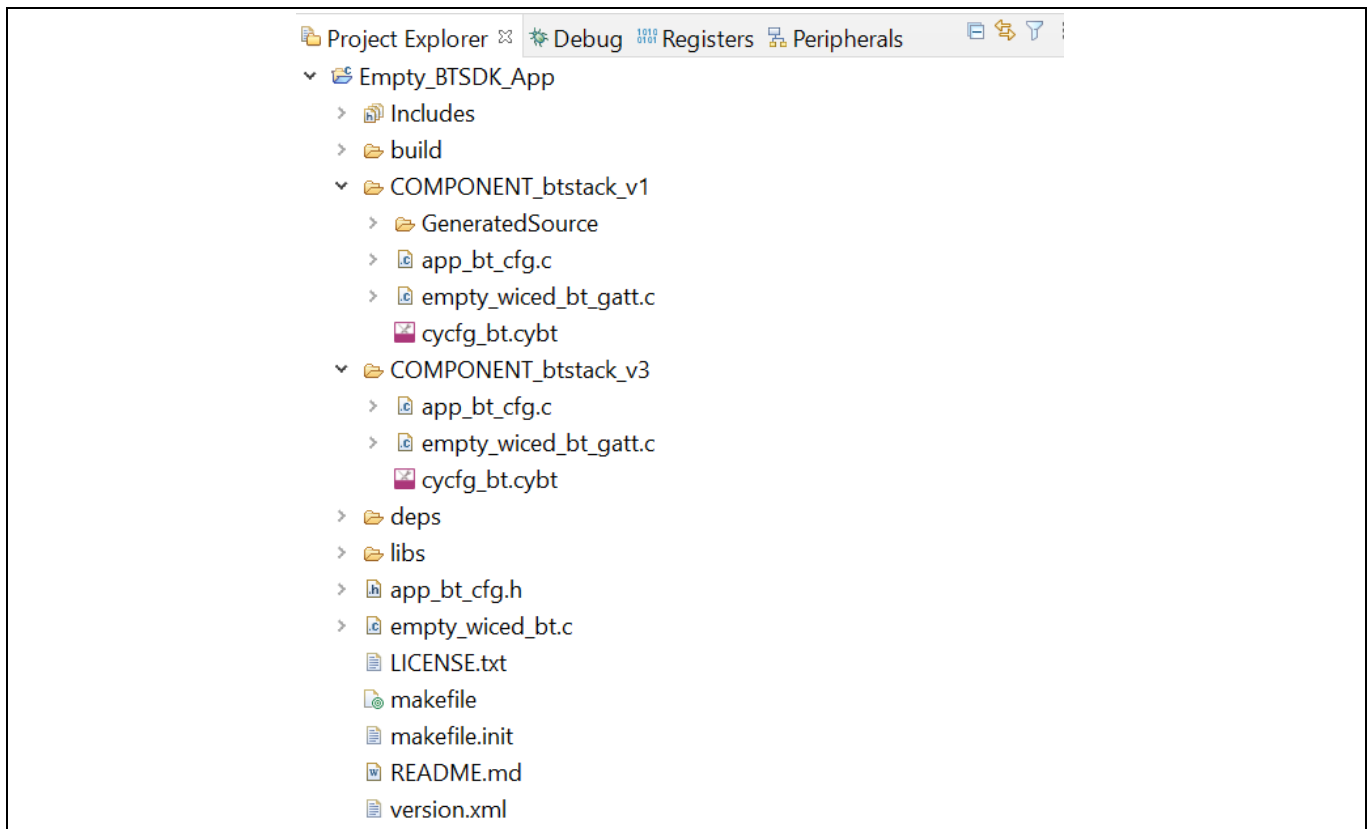


Figure 17 Project Explorer view

- The *Includes* folder contains the header files related to the Bluetooth® stack, Hardware Abstraction Layer (HAL), hardware platform, etc. The API function prototypes, constants, and defaults are contained in these header files. The application code you write will use these API functions and constants as required.
- The *GeneratedSource* folder contains the design configuration for this application. This includes the list of resources enabled/disabled for the application and configuration settings for the different resources. We will discuss more about the contents of this folder in the next step of the application development flow.
- The *app_bt_cfg.c* and *app_bt_cfg.h* files contain runtime Bluetooth® stack configuration parameters like device name, and advertisement/connection settings files.
- The *empty_wiced_bt_gatt.c* files contain functions for all devices with BTSTACK versions earlier than 3.0.
- The *readme.txt* file contains information that explains what the application does (in this case, *Empty_BTSDK_App*).
- *makefile.init* is used as part of the build process, and it allows advanced users to specify new rules and variables in the build flow.
- *Makefile* contains basic and advanced configuration path info.

Now, let's get into how to configure the design resources in the template application.

5.4.1 Configure hardware resources

Now that you have an application set up from the **Empty_BTSDK_App** template application, it's time to configure the hardware resources required for this application and generate the corresponding configuration code. Eclipse IDE for ModusToolbox™ stores the application's configuration settings in the *design.modus* file.

Getting started with AIROC™ CYW20819/20/35 Bluetooth® & Bluetooth® LE



My first CYW208xx Bluetooth® Low Energy application

This file uses by the graphical configurators, which generate the configuration firmware. This firmware is stored in the application's `GeneratedSource` folder.

The Device Configurator is used to enable/configure the peripherals and the pins used in the application. To launch the Device Configurator, double-click the `design.modus` file or click on **Device Configurator** in the Quick Panel, as shown in **Figure 18**. Any time you change the Device Configurator, click **File > Save** to save the updated configuration.

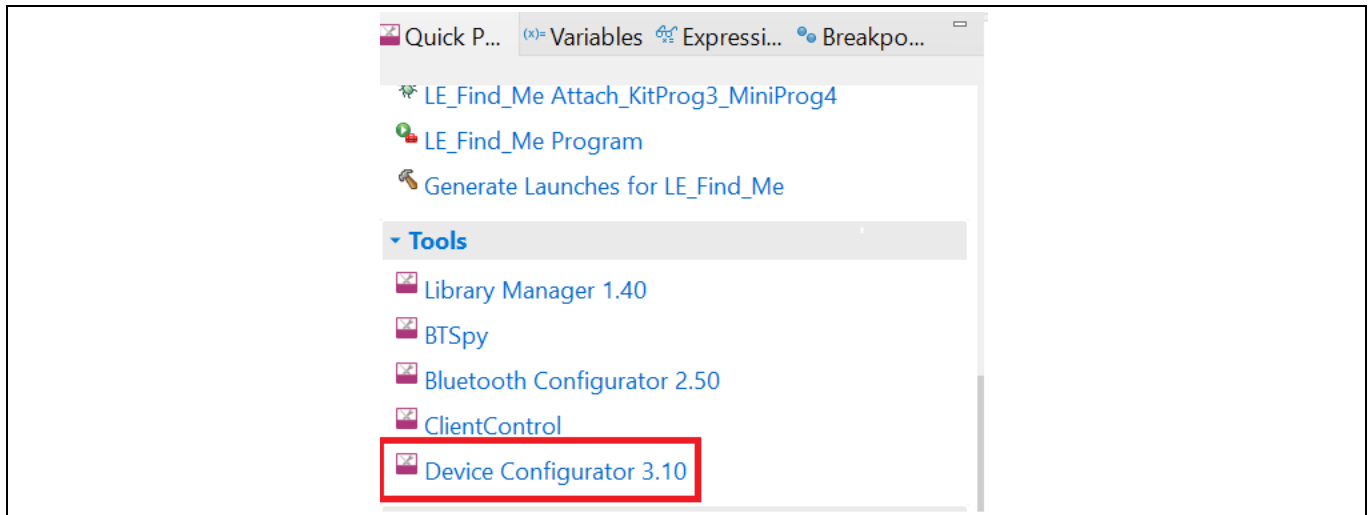


Figure 18 Quick Panel view

Figure 19 shows the Device Configurator view showing the Peripherals view for this application.

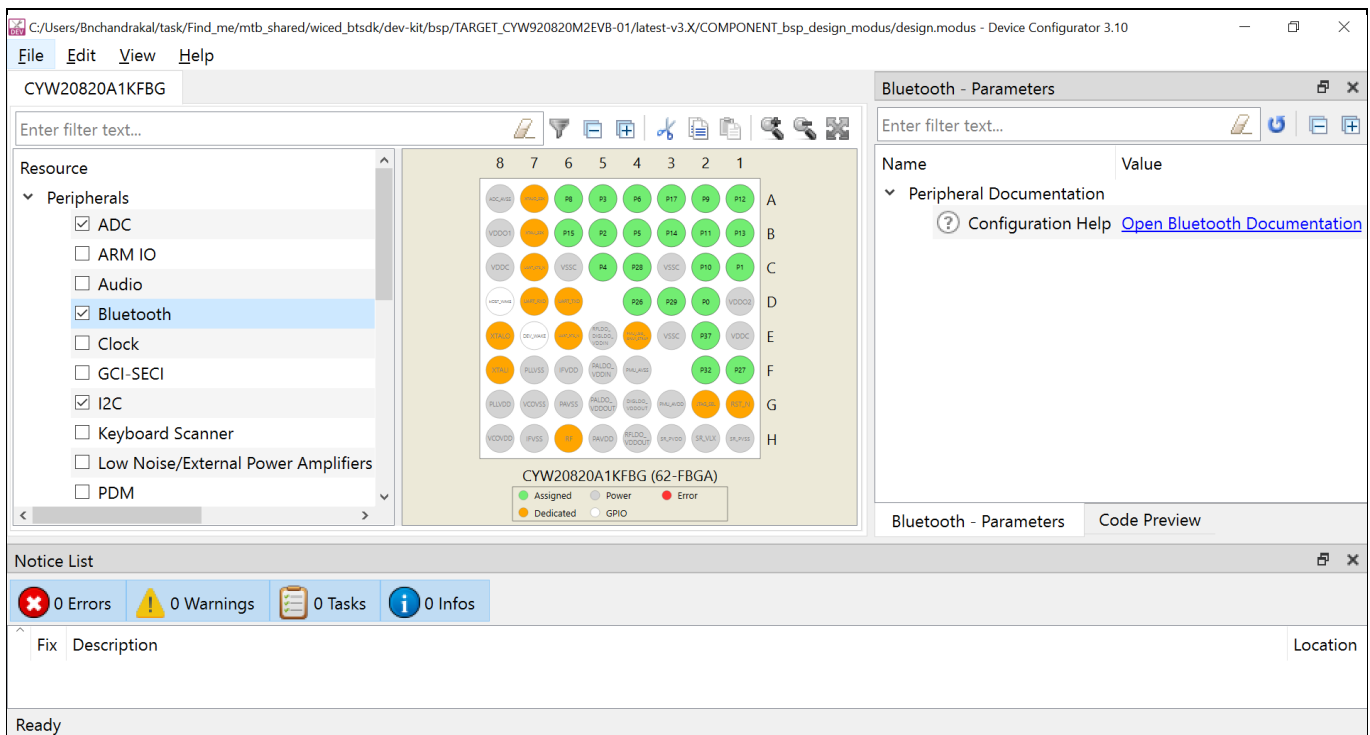


Figure 19 Device Configurator - peripherals

- **Peripherals:** The template application has the peripherals on the kit enabled by default, as shown in **Figure 19** (Bluetooth®, I²C, SPI 1, UART 2). For the Find Me application, only the Bluetooth® and PUART (UART 2)

My first CYW208xx Bluetooth® Low Energy application

peripherals are used. However, we can leave the default peripheral to enable/disable settings as they are in the template application – the I²C and SPI will not be enabled in the code, so having them enabled in the configurator doesn't matter.

To configure a peripheral, click on the peripheral name on the **Resource** pane; the **Parameters** pane shows the configuration options provided for that peripheral. To open the configurator documentation for more information, select **Help** -> **View Help** on the Device Configurator.

- **Bluetooth® Configurator:** The Bluetooth® peripheral has an additional configurator called the Bluetooth® Configurator that is used to create the Bluetooth® Low Energy GATT database. For the Find Me Profile application, we need to generate a GATT database corresponding to the Find Me Target role of the CYW20820 device. Before doing that we will move `cycfg_bt.cybt` and `app_bt_cfg.c` files from the `COMPONENT_btstack_v1` folder to the top-level folder of the project from IDE or from your PC file explorer. We are copying these files because CYW20820/CYW20819/CYW20835 devices support btstack v1. Once done, delete the `COMPONENT_btstack_v1` folder and `COMPONENT_btstack_v3` folder either from IDE or from your PC file explorer. Now to configure the GATT database, launch the Bluetooth® Configurator by double-clicking `cycfg_bt.cybt` file or by selecting **Bluetooth Configurator** in the **Quick Panel**, as shown in **Figure 20**. The default view of the Bluetooth® Configurator is shown in **Figure 21**.

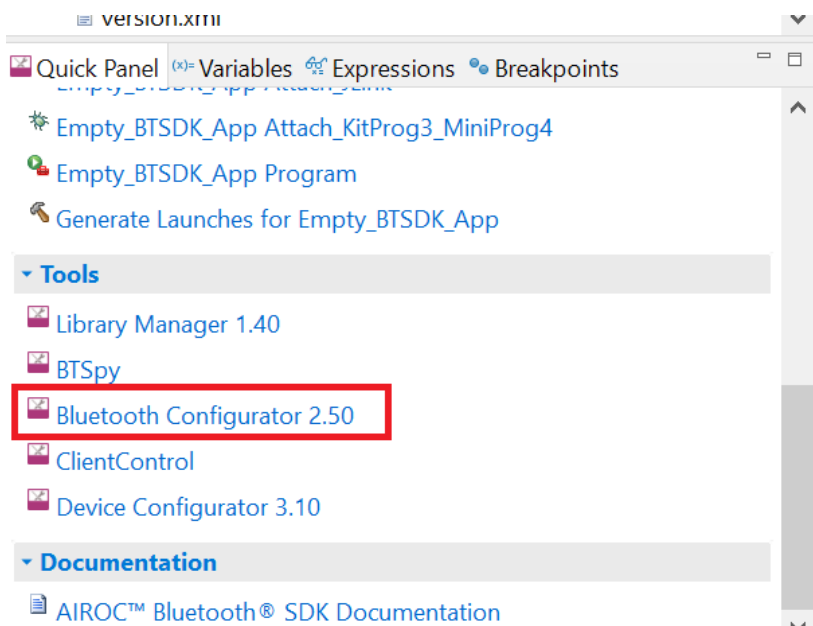


Figure 20 Quick Panel View - Bluetooth Configurator

Getting started with AIROC™ CYW20819/20/35 Bluetooth® & Bluetooth® LE



My first CYW208xx Bluetooth® Low Energy application

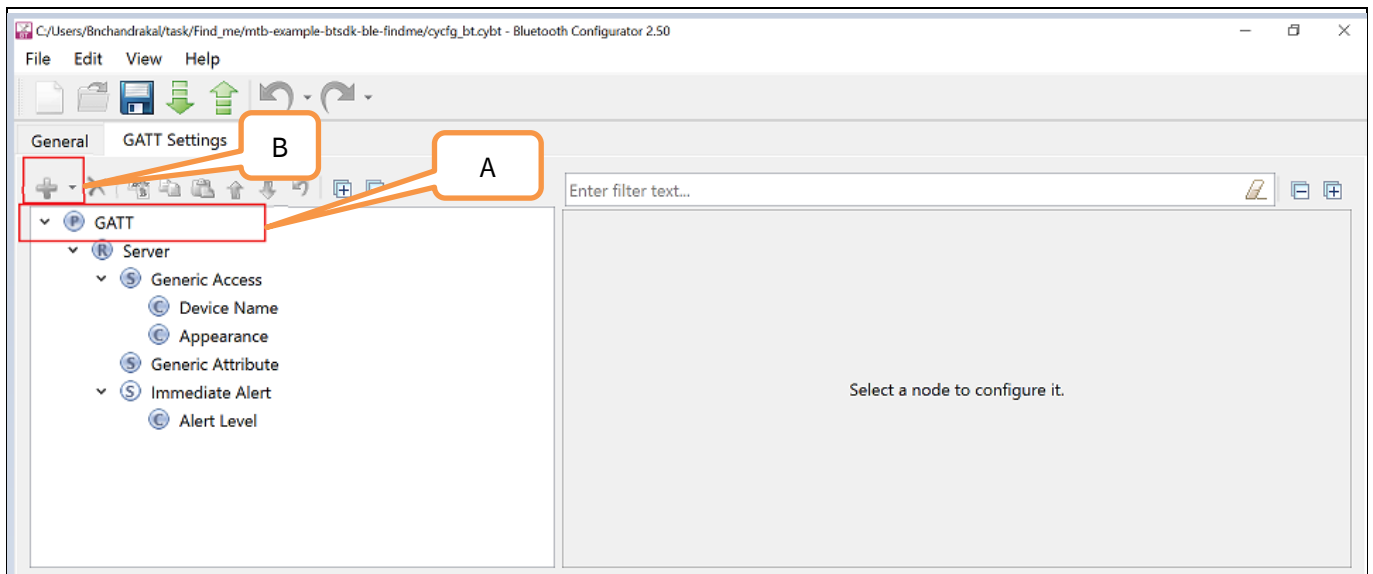


Figure 21 Bluetooth® configurator

To add the Find Me Target profile, click on **GATT** profile (A in [Figure 21](#)), and then click the **+** icon (B in [Figure 21](#)). Select the **Find Me Target (GATT Server)** profile from the drop-down menu, as shown in [Figure 22](#)), and then click the **+** icon (B). Select the **Find Me Target (GATT Server)** profile from the drop-down menu, as shown in [Figure 22](#).

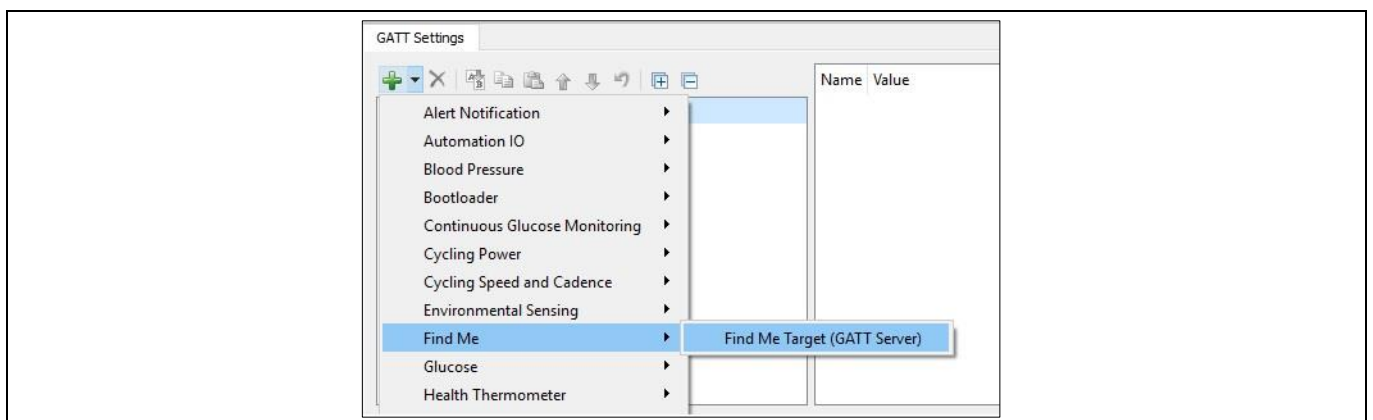


Figure 22 Adding Find Me Target profile

[Figure 23](#) shows the GATT database view once the Find Me Target Server has been added. Note that the Immediate Alert Service corresponding to the Find Me Target profile has been added to the [Figure 23](#). Click **File** > **Save** in the Configurator window or click the Save icon. The configurator stores the GATT database in the source files `cycfg_gatt_db.h` and `cycfg_gatt_db.h` in the `GeneratedSource` folder. In the same folder, the `cycfg_bt.h` file contains the Bluetooth® configurator settings in XML format, which the configurator uses to load the settings when launched again.

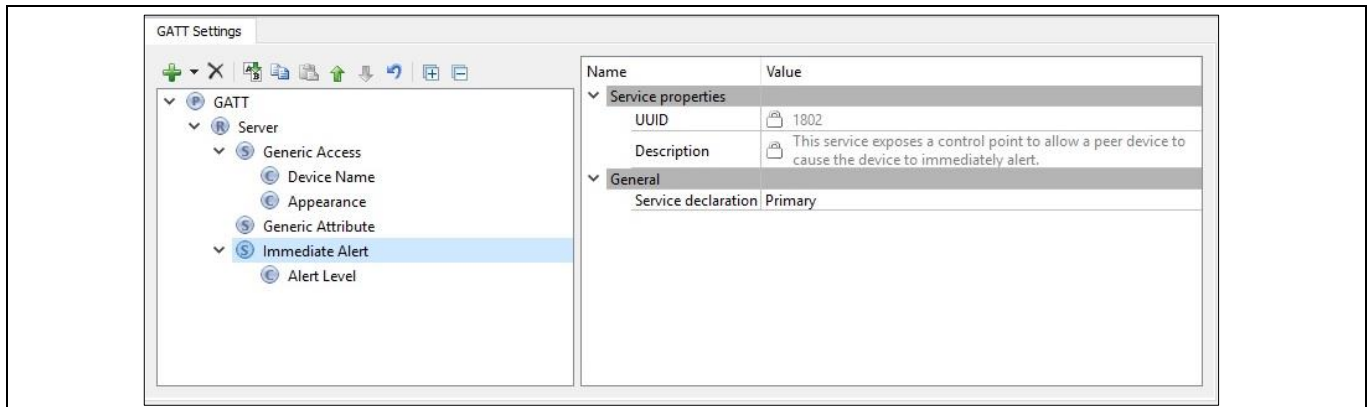


Figure 23 Final GATT database view

- Pins:** The pins used in the application are enabled in the **Pins** section of the Device Configurator, as shown in **Figure 24**. The template application already has some pins enabled and configured corresponding to the kit features, such as the pins used for LEDs, buttons, and peripherals like I²C, SPI, and PUART. For the Find Me application, the pins of interest are the two LED pins (LED1, LED2) and the PUART interface pins (UART_TX, UART_RX) (see **Table 1**). The parameter settings for these pins are left at the default settings as shown in **Figure 25**, **Figure 26**, **Figure 27**, and **Figure 28**. We leave the SPI and I²C pins unchanged – we won't be using those pins for anything else, so it doesn't hurt to have the pins assigned to the default peripherals for the board we are using. Any time you change the Device Configurator, click **File > Save** or click the Save icon to save the updated configuration.

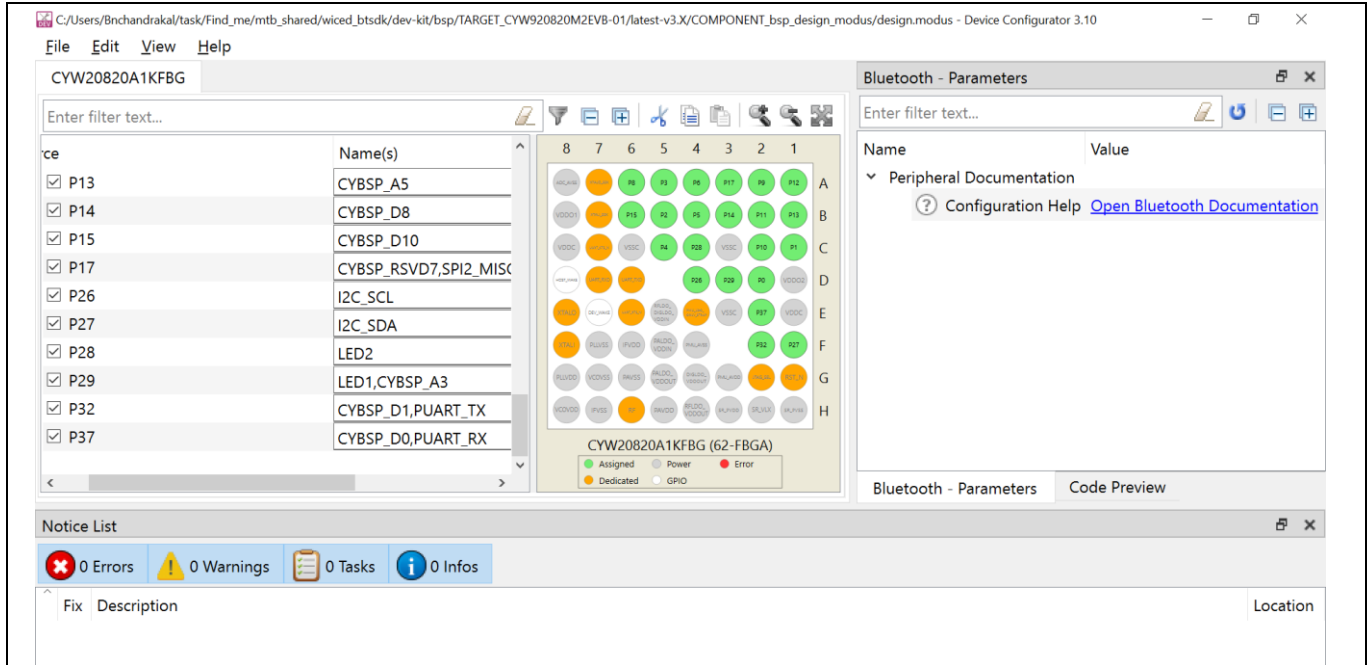


Figure 24 Device configurator – pins

Table 1 Pin mapping details

| Pin | Alias | Purpose | Settings |
|-----|-------|---|---|
| P28 | LED2 | Mapped to the red LED (LED2) on the kit. Indicates IAS Alert Level. | See Figure 25 and Figure 26 . Ensure that the Index setting for |

My first CYW208xx Bluetooth® Low Energy application

| Pin | Alias | Purpose | Settings |
|-----|----------|---|---|
| P29 | LED1 | Mapped to the yellow LED (LED1) on the kit. Indicates the Advertising/Connected state of the Bluetooth® Low Energy peripheral device. | each LED is different as shown in the figures – the configurator does not do this checking. The Index value should also start from 1 (not 0). Refer <code>cycfg_pins.h</code> to see how the Index value is used by the configuration code. |
| P32 | PUART_TX | Tx pin of PUART peripheral | See Figure 27 . |
| P37 | PUART_RX | Rx pin of PUART peripheral | See Figure 28 . |

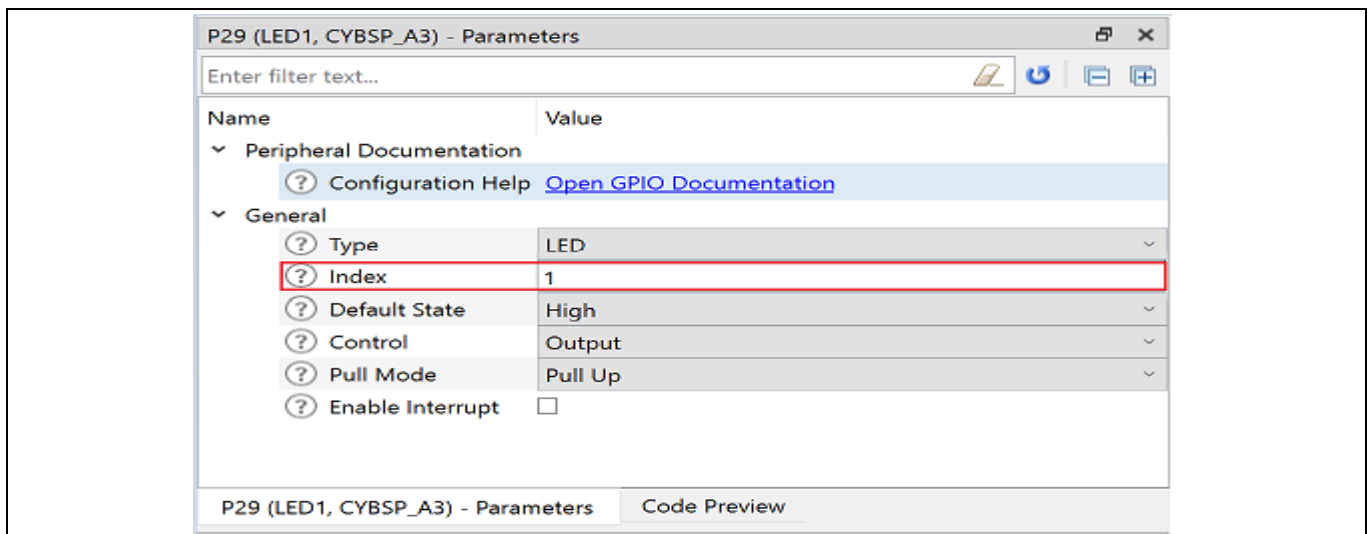


Figure 25 LED1 pin settings

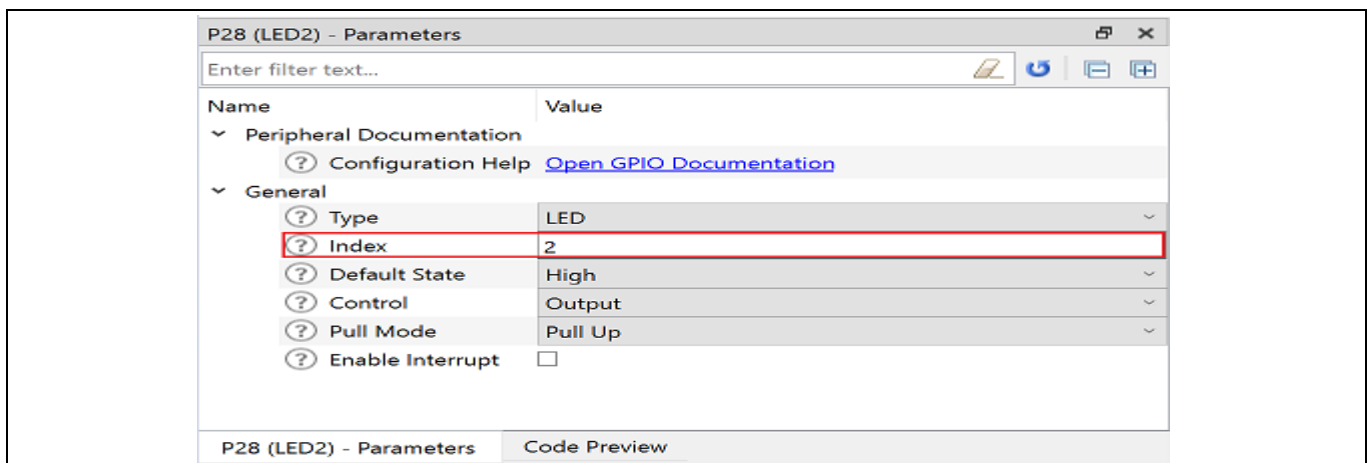


Figure 26 LED2 pin settings

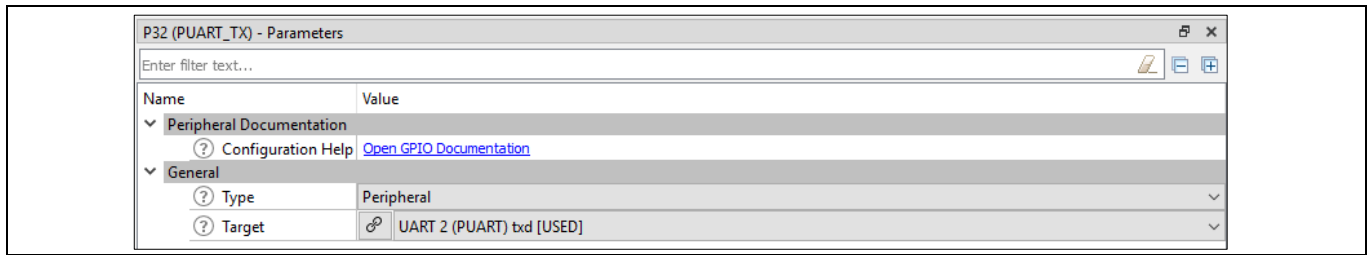


Figure 27 PUART Tx pin settings

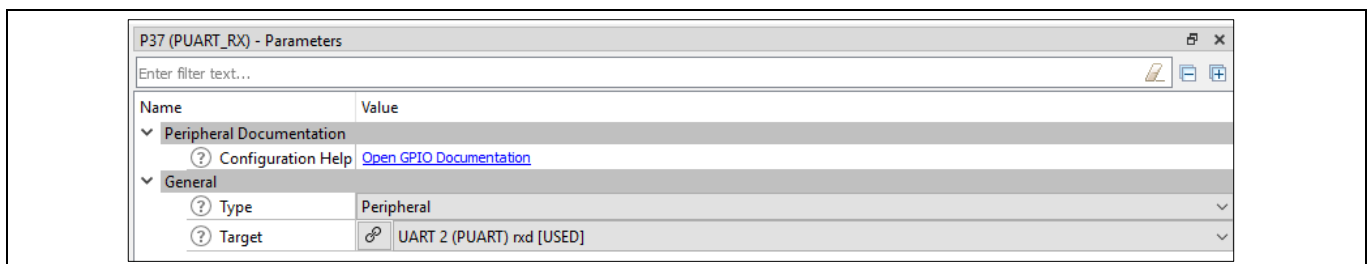


Figure 28 PUART Rx pin settings

You have completed the resource configuration for the application. The required configuration code generates in the `GeneratedSource` folder files, and the settings have also been stored in the `design.modus` file.

5.5 Common application settings

The following application settings are common for all BTSDK applications and can be configured via the application's makefile or passed in via the command line.

5.5.1 BT_DEVICE_ADDRESS

Set the BDA (Bluetooth® Device Address) for your device. The address is 6 bytes, for example, 20819A10FFEE. By default, the SDK will set a BDA for your device by combining the seven hex digit device ID with the last five hex digits of the host PC MAC address.

5.5.2 UART

Set to the UART port you want to use to download the application. For example, 'COM6' on Windows or '/dev/ttyWICED_HCI_UART0' on Linux or '/dev/tty.usbserial-000154' on macOS. By default, the SDK will auto-detect the port.

5.5.3 ENABLE_DEBUG

For hardware debugging, configure `ENABLE_DEBUG=1`. This setting configures GPIO for SWD. See the document [AIROC™-Hardware-Debugging](#) for more information.

- CYW920819M2EVB-01/CYW920820M2EVB-01: SWD signals are shared with D4 and D5; see SW9 in schematics.
- CYBT-213043-MESH/CYBT-213043-EVAL/CYBT-253059-EVAL: SWD signals are routed to P12=SWDCK and P13=SWDIO. Use expansion connectors to connect VDD, GND, SWDCK, and SWDIO to your SWD debugger probe.

My first CYW208xx Bluetooth® Low Energy application

- CYBT-223058-EVAL/CYW920835M2EVB-01/CYBT-243053-EVAL/CYBLE-343072-EVAL-M2B/CYBLE-333074-EVAL-M2B/CYBLE-343072-MESH: SWD signals are routed to P02=SWDCK and P03=SWDIO. Use expansion connectors to connect VDD, GND, SWDCK, and SWDIO to your SWD debugger probe.
- CYBT-263065-EVAL/CYBT-273063-EVAL: SWD signals are routed to P02=SWDCK and P04=SWDIO. Use expansion connectors to connect VDD, GND, SWDCK, and SWDIO to your SWD debugger probe.
- SWD hardware debugging is not supported on the following:
 - CYW920835M2EVB-01
 - CYW920819M2EVB-01/CYW920820M2EVB-01

5.6 Library Manager

The Bluetooth® SDK in ModusToolbox™ provides a ‘Library Manager’ dialog to select various middleware components for developing Bluetooth® applications. To launch the Library Manager dialog, right-click the top-level project name in the Project Explorer and select the **ModusToolbox™ > Library Manager** option (refer to [Figure 29](#) for this option). Alternatively, you can click the **Select Library Manager** link in the Quick Panel.

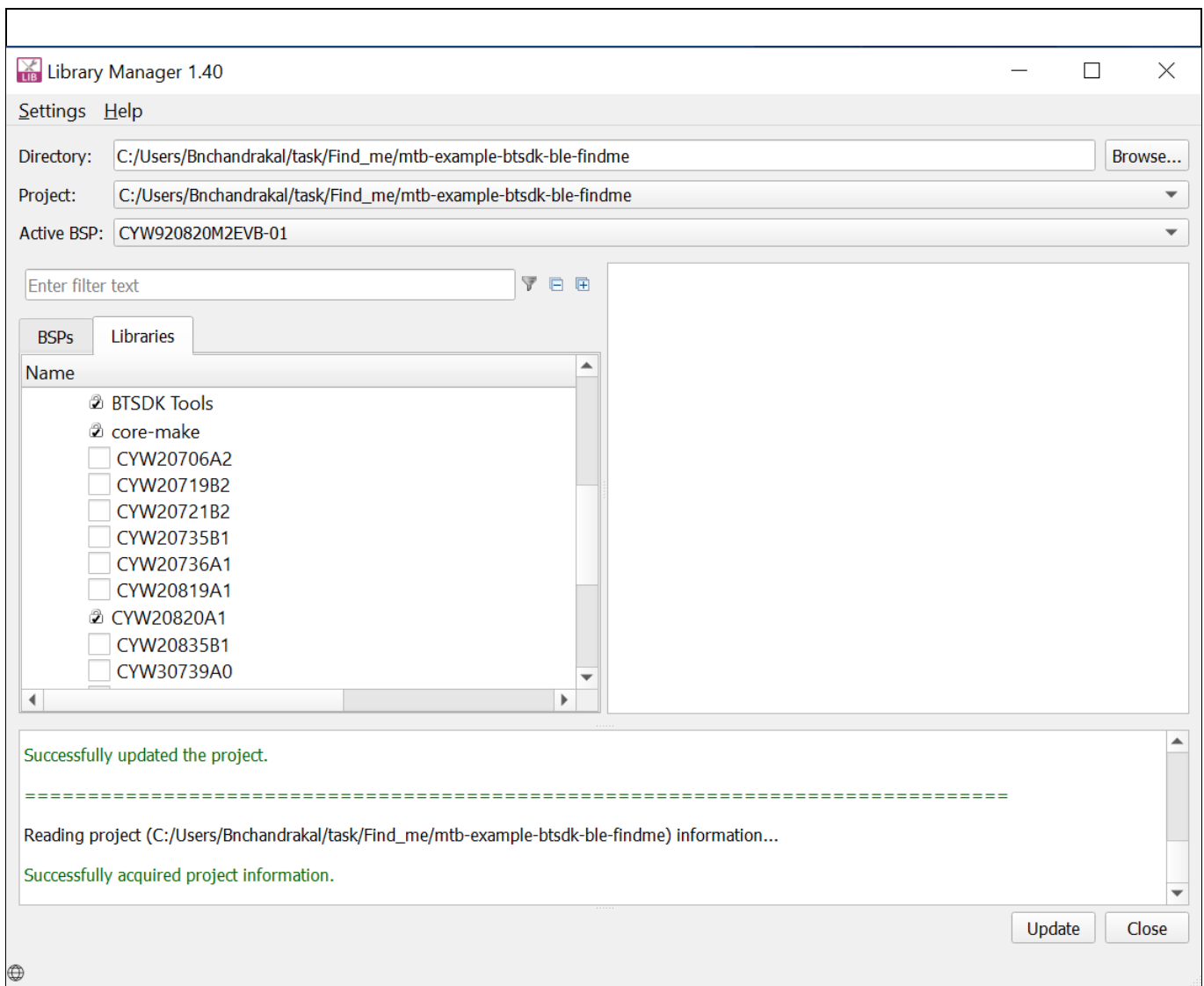


Figure 29 Library Manager

5.7 Part 3: Write the application code

At this point in the development process, you have created an application, configured the hardware resources, and generated the configuration code, including the Bluetooth® Low Energy GATT database. This part examines the application code that implements the Find Me Target functionality.

| Path | “Using CE directly” path (Evaluate existing code example (CE) directly) | “Working from scratch” path (Use existing code example (CE) as reference only) |
|---------|---|--|
| Actions | Ignore Step 1 - the CE already has all the necessary source files added. Read through the Firmware description section to understand the firmware design. | Perform Step 1. Read through the Firmware description section to understand the firmware design. |

The application code must do the significant tasks as follows:

- Perform system initialization, including the Bluetooth® stack.
- Implement Bluetooth® stack event handler functions for different events, such as an advertisement, connection, and attribute read/write requests.
- Implement user interface logic to update the LED state on the kit based on the events triggered.

We use a modular firmware development approach where each of the above tasks are contained in separate source/header files to understand the code and enable code-reuse easily.

1. Add files to your project (Required only for “Working from Scratch” flow)

- Locate the ***mtb-example-btsdk-ble-findme*** code example that you downloaded from the repository. See the Prerequisites section for the repository download process and code example location.
- Copy the following files from the `mtb-example-btsdk-ble-findme` code example top-level folder to your `mtb-example-btsdk-ble-findme` folder inside the ModusToolbox™ workspace folder.
 - `find_me.c`
 - `app_bt_event_handler.c`
 - `app_bt_event_handler.h`
 - `app_gatts.c`
 - `app_gatts.h`
 - `app_user_interface.c`
 - `app_user_interface.h`
- Delete the `empty_wiced_bt.c` file either from Eclipse IDE for ModusToolbox™ Project Explorer or from your PC file explorer. You no longer need this file because the equivalent `application_start ()` function is now part of the `find_me.c` file copied in the previous step.

5.8 Firmware description

This section explains the application firmware of the Find Me application. The important source files relevant for the user application level code for this code example are listed in [Table 2](#).

Table 2 Important user application-related source files

| File name | Comments |
|--|--|
| <code>cycfg_gatt_db.c</code> , <code>cycfg_gatt_db.h</code> | These files reside in the <i>GeneratedSource</i> folder under the application folder. They contain the GATT database information generated using the Bluetooth® Configurator tool. |
| <code>app_bt_cfg.c</code> <code>app_bt_cfg.h</code> | These files contains the runtime Bluetooth® stack configuration parameters, such as device name and advertisement/connection settings. |
| <code>Find_me.c</code> | Contains the <code>application_start ()</code> function which is the entry point for execution of the user application code after device startup. |
| <code>app_bt_event_handler.c</code> <code>app_bt_event_handler.h</code> | These files contain the code for the Bluetooth® stack event handler functions. |
| <code>app_user_interface.c</code> <code>app_user_interface.h</code> | These files contain the code for the application user interface (in this case, the LED) functionality. |
| <code>app_gatts.c</code> <code>app_gatts.h</code> | These files contain the code for GATT functions to handle GATT events from the stack. |

5.8.1 Bluetooth® Low Energy GATT database

The `cycfg_gatt_db.c` and `cycfg_gatt_db.h` files contain the Bluetooth® Low Energy GATT database definitions for the Find Me Target profile generated in the previous step using the Bluetooth® Configurator tool. The GATT database is accessed by both the Bluetooth® stack and the application code. The stack will directly access the attribute handles, UUIDs, and attribute permissions to process some of the Bluetooth® events. The application code will access the GATT database to perform attribute read/write operations. The relevant database structures are listed below.

- `gatt_database []`: This array contains the attribute handles, types and permissions. Note that this array does not have the actual attribute values, maintained as separate arrays as explained below.
- `GATT Value Arrays`: The actual GATT database containing the attribute values is declared as a series of `uint8_t` arrays under the section `GATT Initial Value Arrays` in `cycfg_gatt_db.c`. These arrays are also exposed as extern variables for application code access in the `cycfg_gatt_db.h` file. The FMP target application has these arrays defined by the name's `app_gap_device_name []`, `app_gap_appearance []`, and `app_ias_alert_level []`. `app_ias_alert_level []` is the Alert Level characteristic corresponding to the IAS service that the client will write to set the alert level. The application code performs the actual write to this attribute.
- `app_gatt_db_ext_attr_tbl []`: This array of structures is a GATT lookup table that conveys the mapping of the attribute handles defined in `gatt_database []` to the GATT value arrays. The application code uses this lookup table to perform the attribute read/write operations on the actual GATT arrays.

5.8.2 Bluetooth® stack configuration parameters

The `app_cfg.c` and `app_cfg.h` files contain the runtime Bluetooth® stack configuration parameters like device name (`app_gap_device_name`), core stack configuration parameters (`wiced_bt_cfg_settings []`).

My first CYW208xx Bluetooth® Low Energy application

In the scope of this application note, we will not be covering these parameters. However, you can refer to the comments in the source files to learn about these parameters. Note that the device name defined in the `app_gap_device_name` variable is the one that will be used on the peer device side to identify the device to establish a connection (“Find Me Target” in this case). The Device Name can be configured in the Bluetooth configurator under GATT settings.

5.8.3 User application code entry

The `main.c` file contains the `application_start ()` function. This function is the entry point for executing the user application code after device initialization is complete. In this code example, this function does two things:

- Selects the UART serial port as the debug UART to view the trace messages and prints a startup message on the debug UART using the `WICED_BT_TRACE` function. The `WICED_BT_TRACE` function is used to send messages using `sprintf`-type formatting.
- Registers a Bluetooth® stack management callback function by calling `wiced_bt_stack_init ()`. The stack management callback function then typically controls the rest of the application based on Bluetooth® events. Typically, only a minimal application initialization is done in the `application_start ()` function. Most application initialization is done in the stack callback function once the Bluetooth® stack has been enabled. The stack callback function `app_bt_management_callback` is defined in `app_bt_event_handler.c`. A callback function is a function that is called by another function when a particular event happens.

5.8.4 Bluetooth® stack events

The `app_bt_event_handler.c` and `app_bt_event_handler.h` files contain the application code logic to handle the different types of events generated by the stack. At a high level, two categories of events need to be handled:

- Bluetooth® stack management events
- GATT events

5.8.4.1 Bluetooth® stack management events

The callback function `app_bt_management_callback` handles events like Stack Enabled, Advertisement State Change, Security-related Events like Pairing, and Key Exchange. This callback function is registered as a part of the `application_start ()` function in `main.c`. See the `wiced_bt_management_evt_t` definition in `wiced_bt_dev.h` for the list of management events. It is not required for the application code to handle all the management events. The events handled depend on the application requirements. **Figure 30** shows the execution logic for the stack management event handler in this code example.

The figure shows that only two management events (`BTM_ENABLED_EVT` and `BTM_BLE_ADVERT_STATE_CHANGED_EVT`) are handled in the stack management callback function.

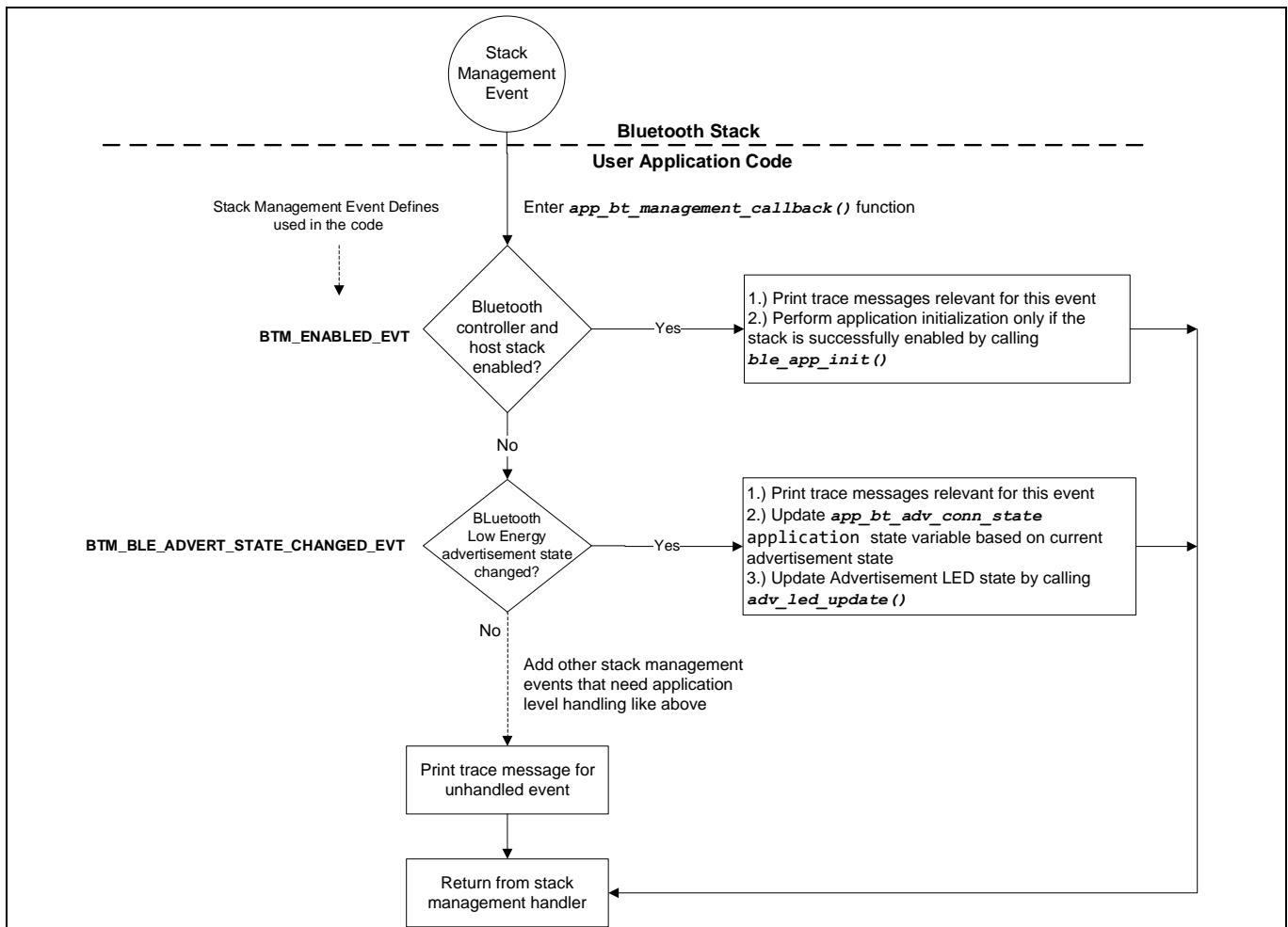


Figure 30 Bluetooth® stack management event handler function flow

At this point, it is pertinent to discuss the `BTM_ENABLED_EVT` event, which is an essential management event that should be handled in all CYW20820-based applications. It signifies that the Bluetooth® stack has been enabled. All the application code initialization is done only after the Bluetooth® stack has been enabled successfully by calling the `ble_app_init()` function as **Figure 30** shows.

The `ble_app_init()` function, defined in `app_bt_event_handler.c`, performs the initialization tasks listed below. For any CYW20820-based application that you create, you should add the required initialization code in this function.

- Initializes the user interface (LED) logic by calling `app_user_interface_init()` defined in `app_user_interface.c`.
- Disables pairing by calling `wiced_bt_set_pairable_mode()`. For this application, the pairing feature is not used.
- Configures the advertisement packet data by calling `ble_app_set_advertisement_data()` defined in `app_bt_event_handler.c`. Look at this function definition in the code example to understand how to configure the elements of an advertisement packet.
- Registers the callback function to handle GATT events (`ble_app_gatt_event_handler()`) by calling `wiced_bt_gatt_register()`.
- Initializes the GATT database (`gatt_database`) defined in `cycfg_gatt_db.c` by calling `wiced_bt_gatt_db_init()`.

My first CYW208xx Bluetooth® Low Energy application

- As the final step of the initialization process, the device starts advertising by calling `wiced_bt_start_advertisements ()`.

5.8.4.2 GATT events

The `ble_app_gatt_event_handler ()` function handles GATT events like connection and attribute request events. This function is registered with a call to `wiced_bt_gatt_register ()` from the `ble_app_init ()` function in `app_bt_event_handler.c`. Refer to the `wiced_bt_gatt_evt_t` definition in `wiced_bt_gatt.h` for the list of GATT events. It is not required for the application code to handle all the GATT events. The events handled depend on the application requirements. **Figure 31** shows the execution logic for the GATT event handler in this code example. The figure shows that only two GATT events (`GATT_CONNECTION_STATUS_EVT` and `GATT_ATTRIBUTE_REQUEST_EVT`) are handled in the function.

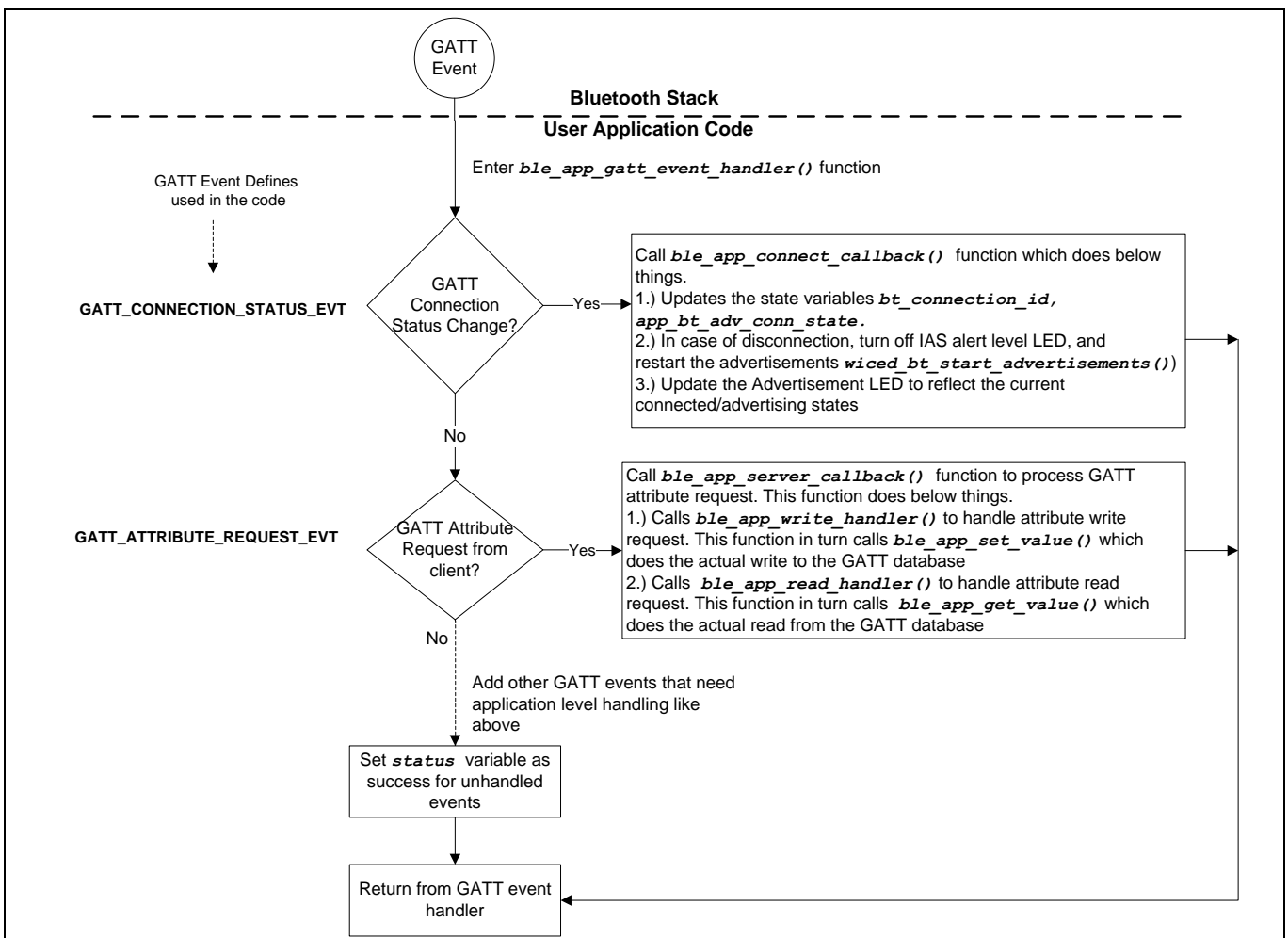


Figure 31 GATT event handler

At this point, it is pertinent to discuss the `GATT_ATTRIBUTE_REQUEST_EVT` event, which is used to process the GATT Attribute read/write operations. **Figure 32** gives information on the functions called in the case of a read or write operation. In this code example, when the Find Me Locator updates the IAS Alert Level characteristic on the CYW20820 device, `GATT_ATTRIBUTE_REQUEST_EVT` is triggered, which in turn calls the series of functions related to the attribute write request. At the end of the write operation, the `app_ias_alert_level []` function in the GATT database in `cycfg_gatt_db.c` gets updated with the alert level set by the Find Me Locator, and the LED is set appropriate to the alert level.

My first CYW208xx Bluetooth® Low Energy application

Figure 32 shows the function call chart summarizing the sequence of function calls for different stack events for this application. All these functions (except `adv_led_update ()`) are defined in `app_bt_event_handler.c`. Refer to the source code to understand the implementation details of these functions.

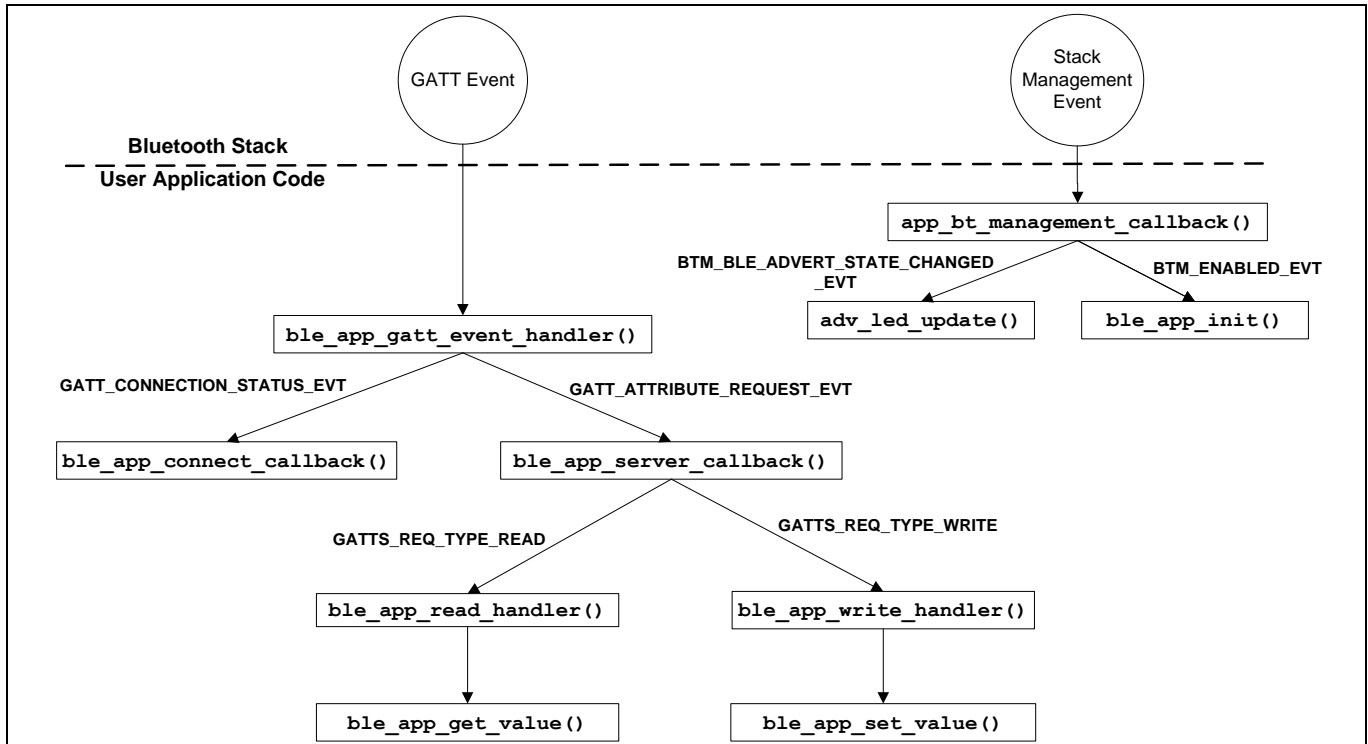


Figure 32 Bluetooth® stack events function call chart

5.8.5 User interface logic

The `app_user_interface.c` and `app_user_interface.h` files contain the application code to handle the user interface logic. The design uses two LEDs for the user interface, whose details are as follows:

- LED1 (yellow LED) on the kit indicates the advertising/connected state of the Bluetooth® Low Energy peripheral device. LED1 is in the OFF state when the device is not advertising, blinking state while advertising, and always in ON state when connected to the peer device. Refer to the `adv_led_update ()` function for implementation details. A global state variable `app_bt_adv_conn_state` is used to update the LED state. The `adv_led_update ()` function is called from two places in the application code:
 - The `app_bt_management_callback ()` function updates LED1 when the advertisement state changes (stack management event `BTM_BLE_ADVERT_STATE_CHANGED_EVT`)
 - The `ble_app_connect_callback ()` function updates LED1 when the connection state changes (GATT event `GATT_CONNECTION_STATUS_EVT`)
- LED2 (red LED) on the kit indicates the IAS alert level characteristic when the device is connected to a peer device. When connected to a peer device, LED2 is in the OFF state for low alert, blinking state for mid alert, and ON state for high alert. When the device is not connected to any peer device, LED2 is in the OFF state. Refer to the `ias_led_update ()` function for implementation details. The `adv_led_update ()` function is called from two places in the application code:
 - The `ble_app_set_value ()` function updates LED2 when an attribute write request to the IAS Alert Level characteristic is done from the client side.

My first CYW208xx Bluetooth® Low Energy application

- The `ble_app_connect_callback ()` function drives LED2 to the OFF state when a disconnection occurs (GATT event `GATT_CONNECTION_STATUS_EVT`)

5.9 Part 4: Build, program, and test your design

This section shows how to build the application and program the CYW20820 device on the CYW920820M2EVB-01 kit. It also explains how to test the Find Me Profile Bluetooth® Low Energy design using the LightBlue mobile app, and the PUART serial interface to view the Bluetooth® stack and application trace messages.

At this point, it assumes that you have followed the previous steps in this application note to develop the Find Me Profile application.

| Path | “Using CE directly” path (Evaluate existing Code Example (CE) directly) | “Working from Scratch” path (Use existing Code Example (CE) as reference only) |
|---------|--|---|
| Actions | Perform all the steps in this section | Perform all the steps in this section |

Connect the kit to your PC using the provided USB cable.

1. The USB serial interface on the kit provides access to the two UART interfaces of the CYW20820 device – WICED HCI UART and WICED Peripheral UART (PUART). The HCI UART interface is used only for downloading the application code in this code example; the PUART interface prints the Bluetooth® stack and application trace messages. Use your favorite serial terminal application and connect to the PUART serial port. Configure the terminal application to access the serial port using the following settings:
Baud rate: 115200 bps; Data: 8 bits; Parity: None; Stop: 1 bit; Flow control – None; New line for receive data: Line Feed (LF) or Auto setting
4. Build and program the application: In the Project Explorer, select the **<App Name>_LE_Find_Me** project. In the Quick Panel, scroll to the **Launches** section, and click the **<App Name> Attach_KitProg3_MiniProg4** configuration as shown in **Figure 33**.

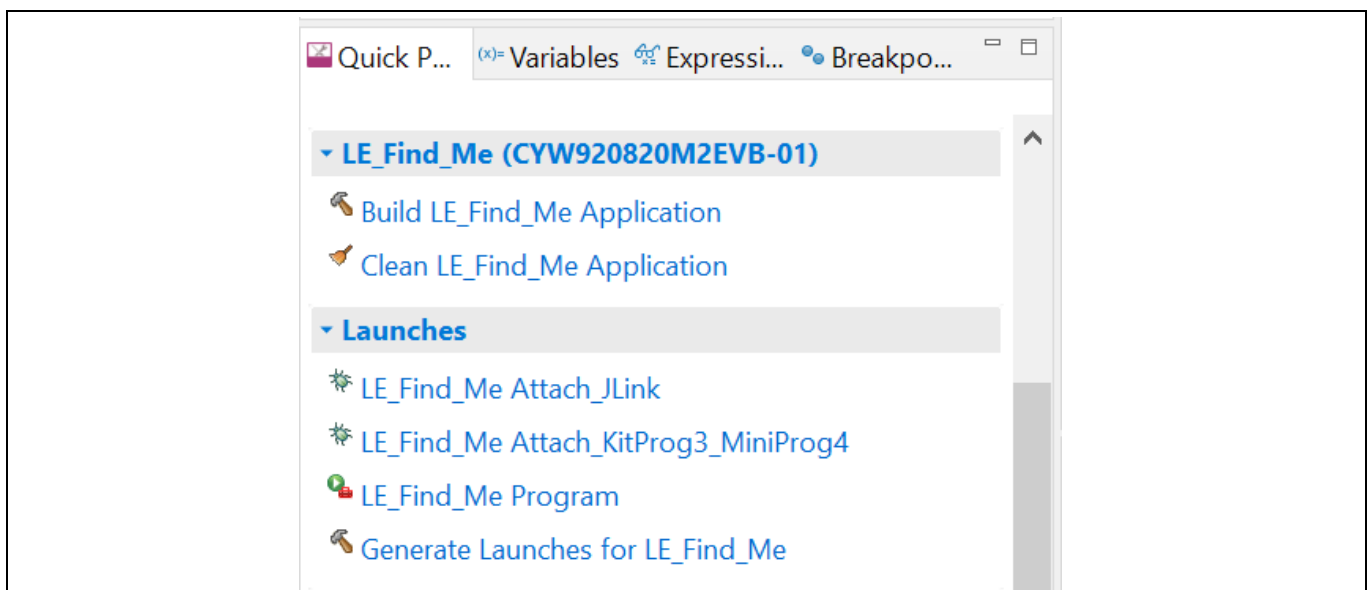


Figure 33 Programming the CYW20820 device from ModusToolbox™

Note: If the download fails, a previously loaded application is preventing programming. For example, the application may use a custom baud rate that the download process does not detect, or the device may be in a low-power mode. It may be necessary to put the board in recovery mode and

My first CYW208xx Bluetooth® Low Energy application

then try the programming operation again from the IDE. To enter recovery mode, first press and hold the **Recover** button (SW1), then press the **Reset** button (SW2), release the **Reset** button (SW2), and then release the **Recover** button (SW1).

Note: If you encounter errors in the application build process, read the error messages in the IDE console window, and revisit the relevant previous steps in this document to check if you have missed or incorrectly done any of those steps.

5. To test using the LightBlue mobile app, follow these steps (see equivalent LightBlue app screenshots in [Figure 34](#) for iOS and [Figure 35](#) for Android).
 - a) Turn ON Bluetooth® on your Android or iOS device.
 - Launch the LightBlue app.
 - Press the reset switch on the CYW920820M2EVB-01 kit to start sending advertisements. The yellow LED (LED1) starts blinking to indicate that advertising has begun. Advertising will stop after 90 seconds if a connection has not been established.
 - Swipe down on the LightBlue app home screen to start scanning for Bluetooth® Low Energy peripherals; your device appears in the LightBlue app home screen. Select your device to establish a Bluetooth® Low Energy connection. Once the connection is established, the yellow LED (LED1) changes from blinking state to always ON state.
 - Select the 'Alert Level' Option.
 - Select an Alert Level value on the Find Me Profile screen. Observe that the state of the red LED (LED2) on the device changes based on the alert level.

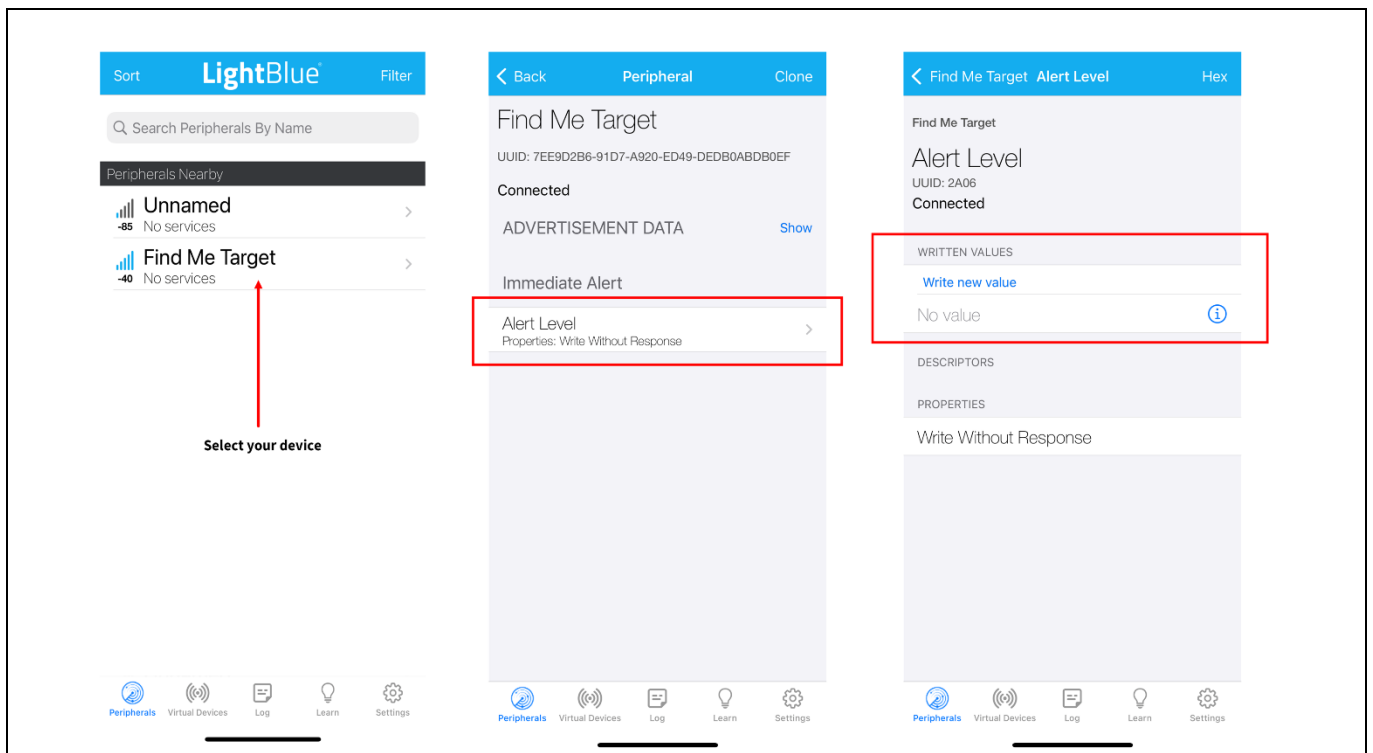


Figure 34 Testing with the LightBlue app on iOS

My first CYW208xx Bluetooth® Low Energy application

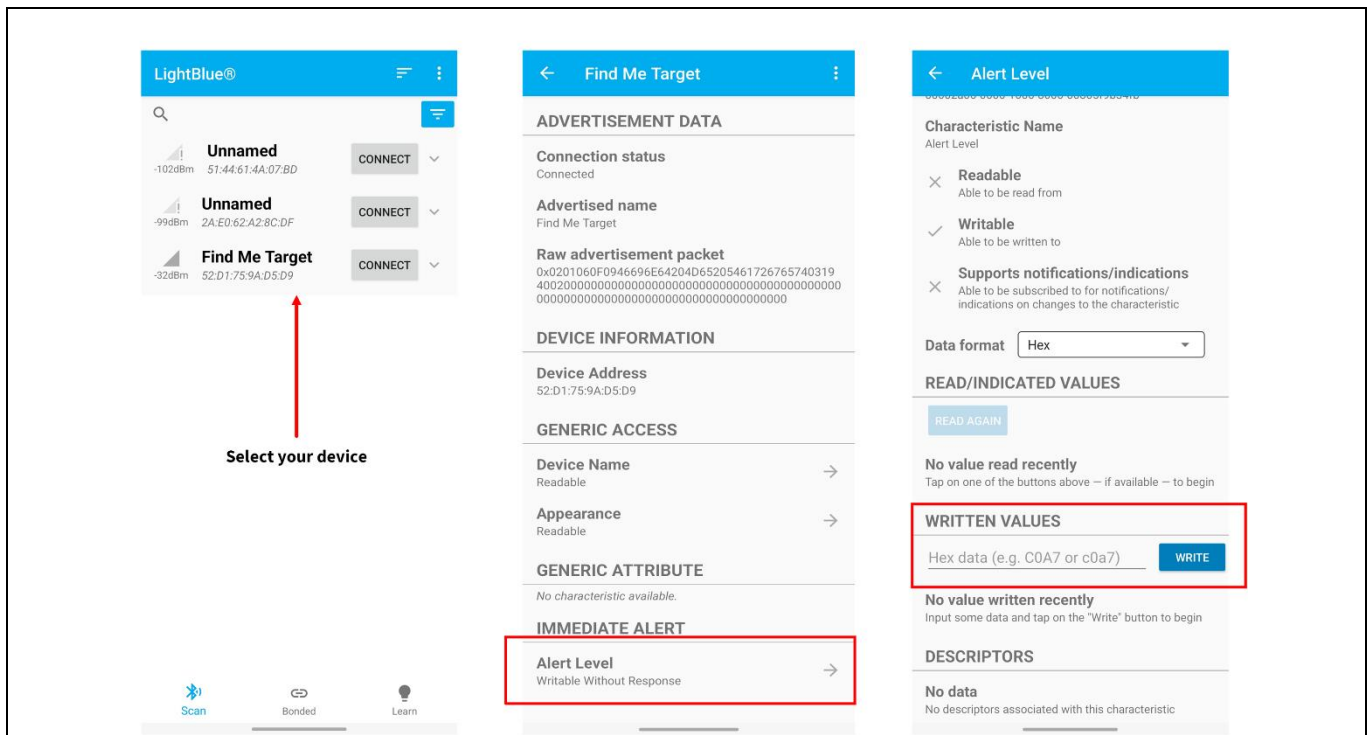


Figure 35 Testing with the LightBlue app on Android

- Use the PUART serial port to view the Bluetooth® stack and application trace messages in the terminal window as shown in [Figure 36](#).

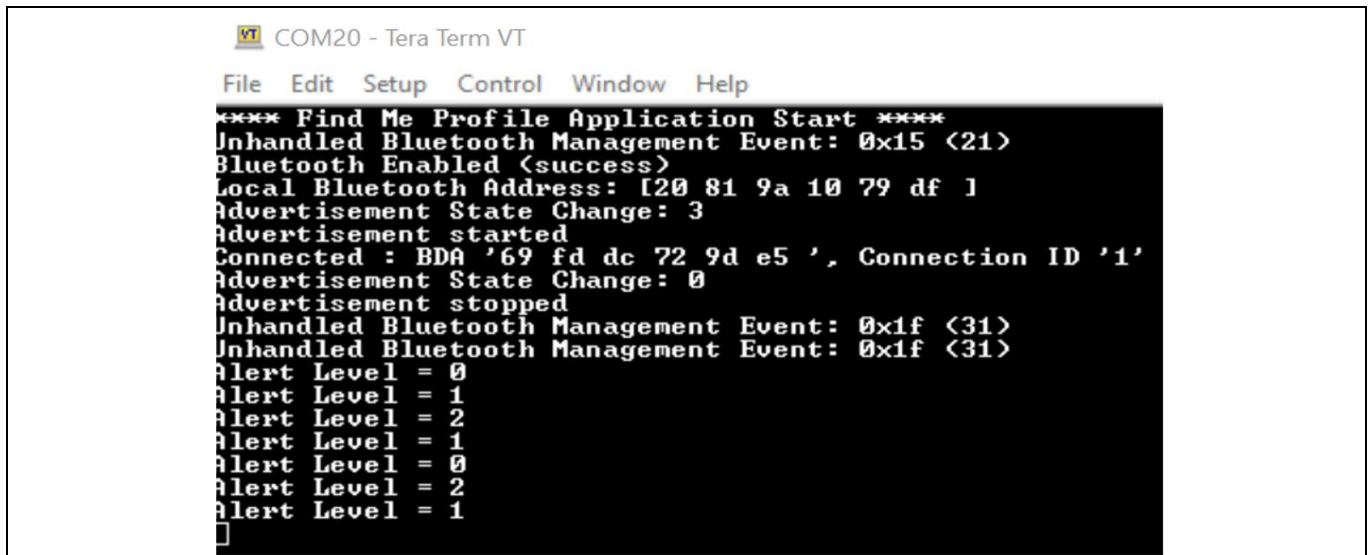


Figure 36 Log messages on WICED PUART serial port

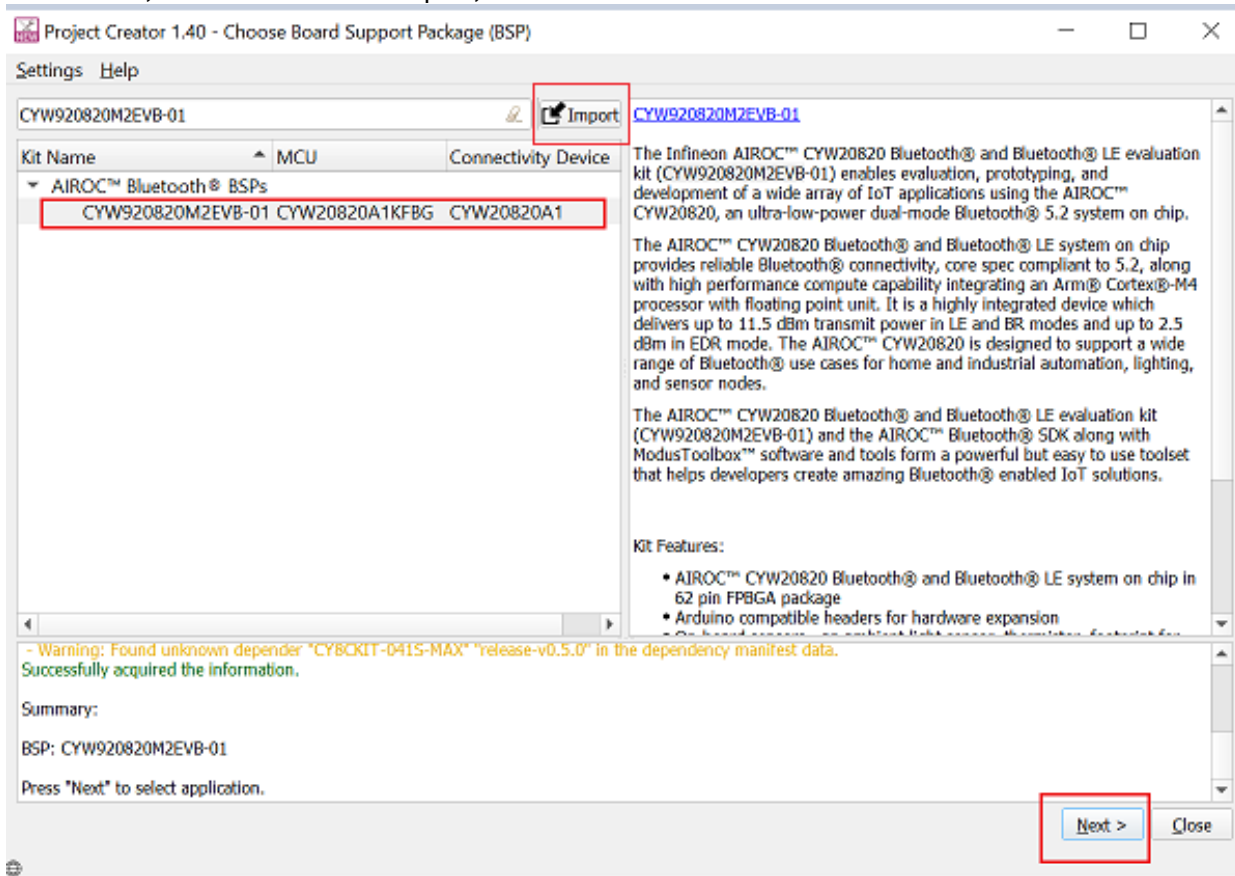
Now, you have successfully developed a simple Bluetooth® Low Energy application for the CYW208xx device using Eclipse IDE for ModusToolbox™. For further learning about CYW208xx device including technical

Getting started with AIROC™ CYW20819/20/35 Bluetooth® & Bluetooth® LE



My first CYW208xx Bluetooth® Low Energy application

documents, additional code examples, see the



section.

Summary

6 Summary

This application note explored the CYW208xx Bluetooth® MCU device architecture, the associated development tools, and the steps to create a simple Bluetooth® Low Energy application for CYW208xx using ModusToolbox™. CYW208xx is a Bluetooth® 5.0-compliant, standalone baseband processor with an integrated 2.4-GHz transceiver with support for both Bluetooth® Low Energy and Classic Bluetooth®. The device is intended for use in audio (source), sensors (medical, home, security), HID and remote-control functionality as well as a host of other IoT applications.

A wealth of code examples, application notes, and other technical documents are available to help you quickly develop CYW208xx based Bluetooth® applications that meets your end application requirements. See the References section to continue learning more about the CYW208xx device and develop Bluetooth® applications.

References

References

- [1] Application notes
 - [ModusToolbox™ 2.4 User Guide](#)
- [2] Code examples
 - [Code Examples for ModusToolbox™ Software](#) – Visit this code example for a comprehensive collection of code examples using ModusToolbox™ IDE.
- [3] Device documentation
 - [CYW20835 device datasheet](#)
 - [CYW20819 device datasheet](#)
 - [CYW20820 device datasheet](#)
- [4] Development kits
 - [CYW920820M2EVB-01 evaluation kit](#)
 - [CYW920819M2EVB-01 evaluation kit](#)
 - [CYW920835M2EVB-01 evaluation kit](#)
- [5] Tool documentation
 - [Eclipse IDE for ModusToolbox™](#) - The Infineon IDE for IoT designers

Revision history

Revision history

| Document version | Date of release | Description of changes |
|------------------|-----------------|--|
| ** | 2019-02-21 | Initial release |
| *A | 2021-04-26 | Updated to Infineon template |
| *B | 2022-09-27 | Updated for MTB 2.4 and added support for CYW920835 and M.2 based 20819/20820 kits |

Trademarks

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

Edition 2022-09-27

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2022 Infineon Technologies AG.

All Rights Reserved.

Do you have a question about this document?

Go to www.infineon.com/support

Document reference

002-25684 Rev. *B

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.