CE226123 – CYW20819 BLE Find Me Profile

Objective

This code example demonstrates the implementation of a simple BLE Immediate Alert Service (IAS)-based Find Me Profile (FMP) using the CYW20819 Bluetooth SoC, using ModusToolbox™ integrated development environment (IDE).

Requirements

Tool: ModusToolbox IDE 1.1 or later version

Programming Language: C

Associated Parts: CYW20819

Related Hardware: CYW920819EVB-02 Evaluation Kit

Overview

This design implements a BLE FMP that consists of an IAS. FMP and IAS are BLE standard Profile and Service respectively, as defined by the Bluetooth SIG. The design uses the two LEDs (red LED, yellow LED) on the CYW920819EVB-02 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 (CYW20819) is advertising (LED blinking), connected (LED ON), or disconnected (LED OFF).

An iOS/Android mobile device or a PC can act as the BLE Central device, which connects to the Peripheral device.

Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide, if required, to ensure the kit is configured correctly.

Software Setup

This code example consists of two parts: a locator and a target. For the locator, download and install the CySmart™ app for iOS or Android. You can also use the CySmart Host Emulation Tool Windows PC application if you have access to the CY5677 CySmart BLE 4.2 USB Dongle. You can also use other Android or iOS apps that support the IAS service.

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

iOS

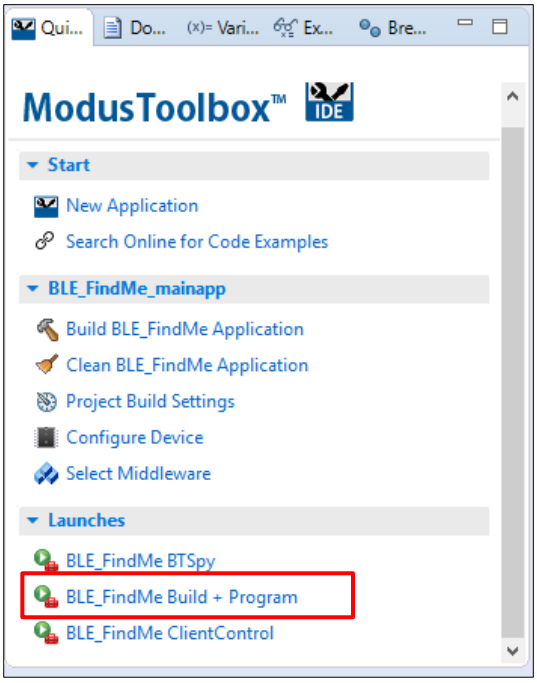
Android

Operation

1. Connect the kit to your PC using the provided USB cable.
2. The USB Serial interface on the kit provides access to the two UART interfaces of the CYW20819 device – WICED HCI UART, and WICED® Peripheral UART (PUART). The HCI UART is used only for downloading the application code in this code example and the PUART is for printing 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 bit; Parity : None; Stop : 1 bit; Flow control – None; New line for receive data : Line Feed(LF) or Auto setting
3. Import the code example into a new or existing workspace. This involves the below steps.
   a. Follow the steps in KBA225201 to clone or download the CYW20819 code examples repository from Cypress GitHub portal.
   b. Once you have the repository on your local machine, the BLE Find Me profile code example is located inside the repository folder at *Code-Examples-BT-20819A1-1.0-for-ModusToolbox\CYW920819EVB-02\apps\demo\BLE_FindMe*. Continue to follow the steps in KBA225201 to import this code example in ModusToolbox IDE with the below changes.
      i. Select the *CYW920819EVB-02* evaluation kit in the *Choose Target Hardware* dialog window.
      ii. Select the *modus.mk* file of the *BLE_FindMe* code example during the **Import** step in the *Starter Application* dialog window.
4. **Build and Program the Application**: In the project explorer, select the **<App Name>_mainapp** project. In the Quick Panel, scroll to the **Launches** section, and click the **<App Name> Build + Program** configuration as shown in Figure 1.

Figure 1. Programming the CYW20819 device from ModusToolbox



Note: If the download fails, it is possible that 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. In that case, it may be necessary to put the board in recovery mode, and 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).

5. Observe the yellow LED (LED1) starts blinking after the device starts advertising.

6. To test using the CySmart mobile app, follow the steps below (see equivalent CySmart app screenshots in Figure 2, Figure 3):
   a. Turn ON Bluetooth on your Android or iOS device.
   b. Launch the CySmart app.
   c. Press the reset switch on the CYW920819EVB-02 kit to start sending advertisements. The yellow LED (LED1) starts blinking to indicate that advertising has started. Advertising will stop after 90 seconds if a connection has not been established.
   d. Swipe down on the CySmart app home screen to start scanning for BLE Peripherals; your device ("Find Me Target") appears in the CySmart app home screen. Select your device to establish a BLE connection. Once the connection is established, the yellow LED (LED1) changes from blinking state to always ON state.
   e. Select the 'Find Me' Profile from the carousel view.
   f. Select an Alert Level value on the Find Me **Profile** screen. Observe the state of the red LED (LED2) on the device changes based on the alert level.

   Figure 2 and Figure 3 show the steps for using CySmart App on iOS and Android respectively.

Figure 2. Testing with the CySmart App on iOS

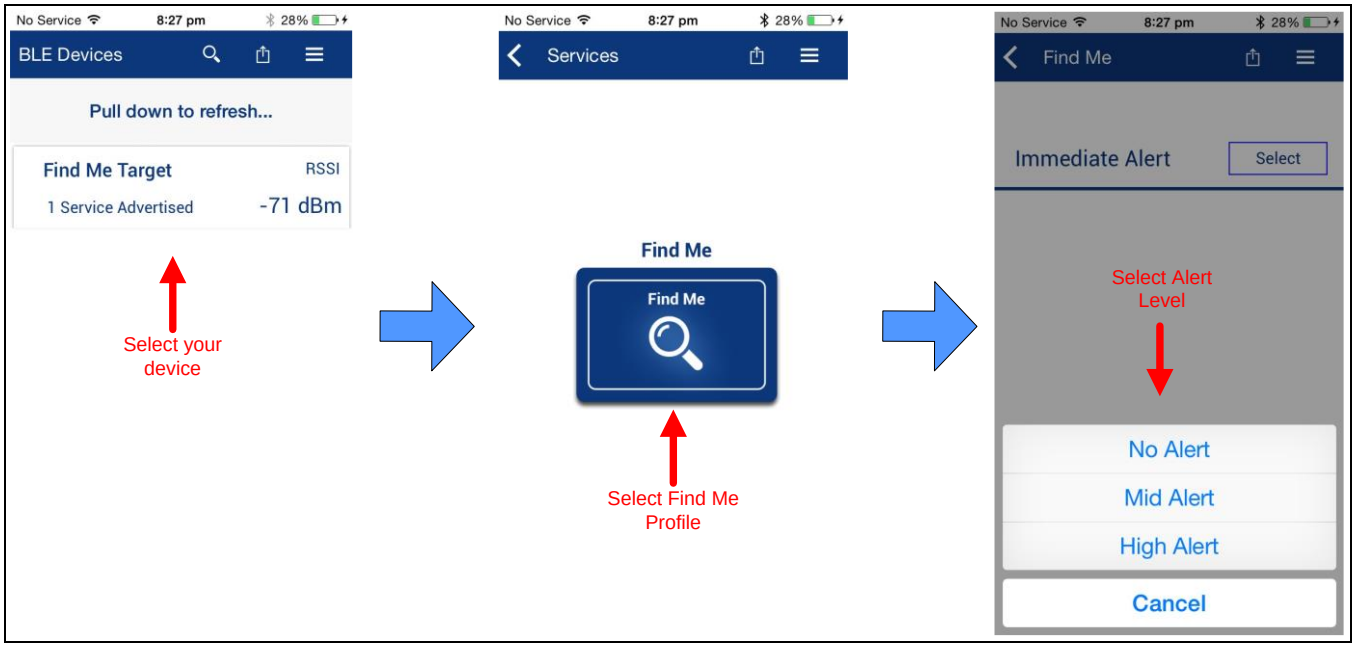
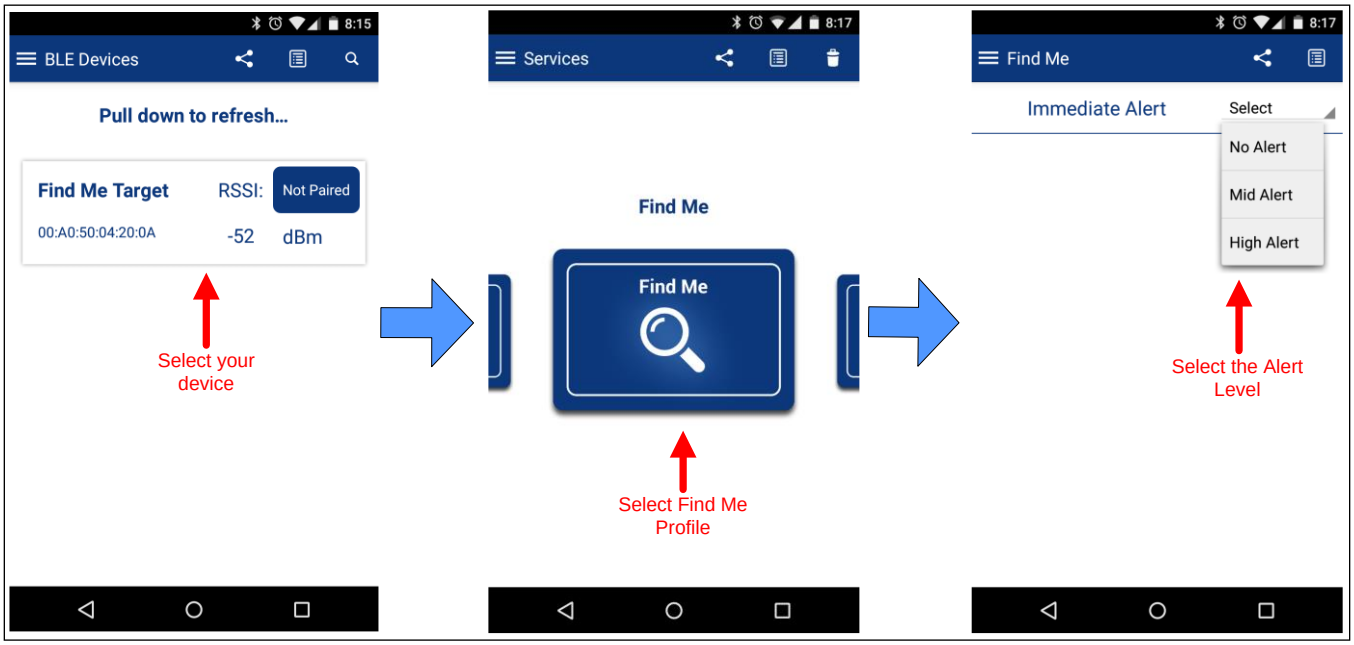
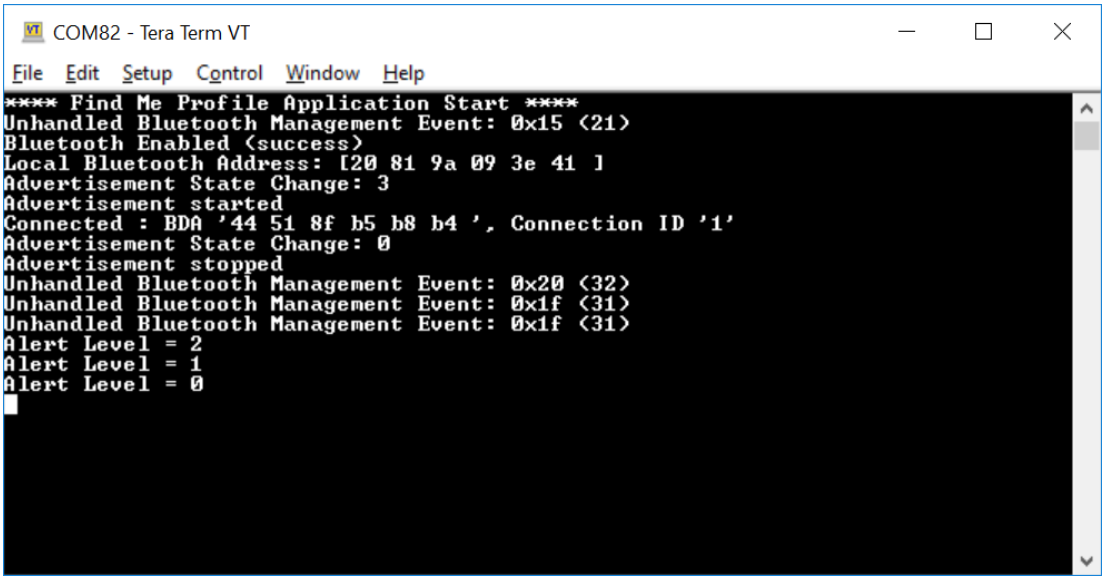


Figure 3. Testing with the CySmart App on Android



7. Use the PUART serial port to view the Bluetooth stack and application trace messages in the terminal window as shown in Figure 4.

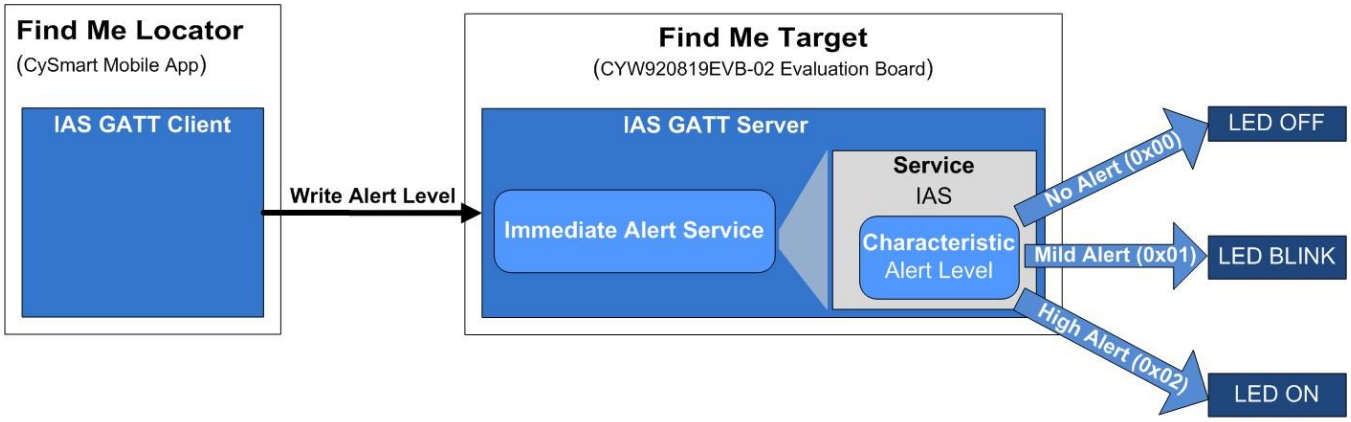
Figure 4. Log Messages on WICED PUART COM Port



Design and Implementation

The ‘Find Me Locator’ (the BLE Central device) is a BLE GATT Client. The ‘Find Me Target’ (the Peripheral device) is a BLE GATT Server with the IAS implemented, as Figure 5 shows.

Figure 5. Find Me Profile (FMP) Application using CYW20819



The BLE Find Me profile defines what happens when the locating Central device broadcasts a change in the alert level.

The Find Me locator performs service discovery using the 'GATT Discover All Primary Services' procedure. The BLE Service Characteristic discovery is done by the 'Discover All Characteristics of a Service' procedure. When the Find Me Locator wants to cause an alert on the Find Me Target, it writes an alert level in the Alert Level Characteristic of the IAS. When the Find Me Target receives an alert level, it indicates the level using the red LED: OFF for no alert, blinking for mild alert, and ON for high alert.

The application code and Bluetooth stack runs on the Arm® Cortex®-M4 core of the CYW20819 SoC. The important source files relevant for the user application level code for this code example are listed in Table 1.

Table 1. Important User-Application-Related Source Files

File Name	Comments
main.c	Contains the `application_start()` function which is the entry point for execution of the user application code after device startup.
app_bt_cfg.c, app_bt_cfg.h	These files contain the runtime Bluetooth stack configuration parameters like device name, advertisement/ connection settings, etc. Note that the name that the device uses for advertising (“Find Me Target”) is defined in *app_bt_cfg.c*.
app_bt_event_handler.c, app_bt_event_handler.h	These files contain the code for the Bluetooth stack event handler functions.
app_user_interface.c, app_user_interface.h	These files contain the code for the application user interface (in this case, the LED) functionality.
cycfg_gatt_db.c, cycfg_gatt_db.h	These files reside in the *GeneratedSource* folder under the application folder. They contain the GATT database information generated using the Bluetooth Configurator tool.

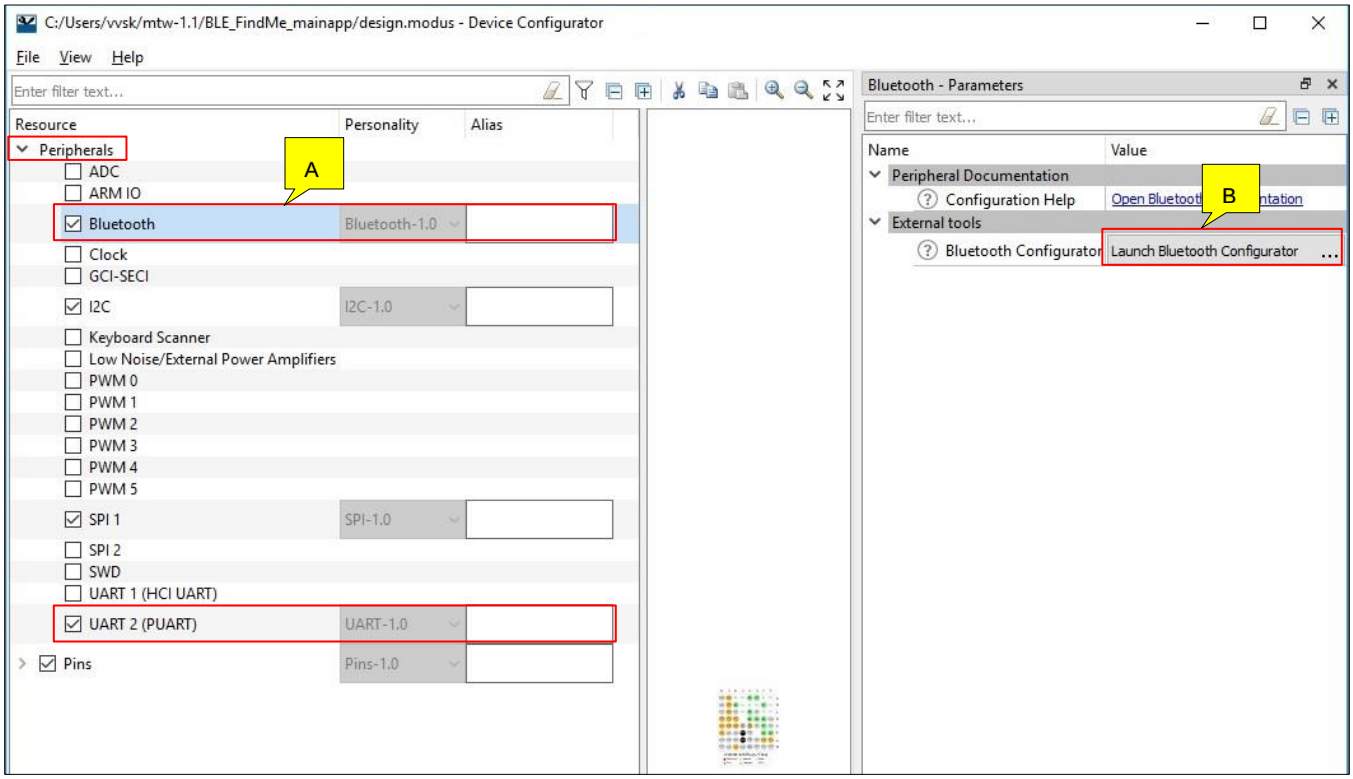
See AN225684 – Getting Started with CYW20819 application note to understand the firmware design for this code example.

Resources and Settings

This section explains the ModusToolbox resources and their configuration as used in this code example. Note that all the configuration explained in this section has already been done in the code example. The ModusToolbox IDE stores the configuration settings of the application in the *design.modus* file. This file is used by the graphical configurators, which generate the configuration firmware. This firmware is stored in the application's *GeneratedSource* folder.

Figure 6 shows the Device Configurator settings highlighting the peripherals used for this code example. 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 **Configure Device** in the Quick Panel. Any time you make a change in the Device Configurator, click **File > Save** to save the updated configuration.

Figure 6. Device Configurator - Peripherals



- **Peripherals**: The design uses the Bluetooth and PUART peripherals which are enabled under the **Peripherals** section as shown in Figure 6. Other chip peripherals that are applicable for using the kit are also enabled by default in Figure 6, but they are not used in this application.
- **Bluetooth Configurator**: The Bluetooth Configurator is used for generating/modifying the BLE GATT database. To launch the Bluetooth Configurator from the Device Configurator window, click on the **Bluetooth** resource under the **Peripherals** section, and then click on the **Launch Bluetooth Configurator** shortcut under the **Bluetooth-Parameters** window (as shown in callouts A, B in Figure 6).

  The Bluetooth Configurator for this code example is shown in Figure 7. Note that the Immediate Alert Service corresponding to the Find Me Target profile has been added in Figure 7. Figure 8 shows how the Find Me Target profile was added to create the GATT database shown in Figure 7. Click on **GATT** profile in the GATT database tree and follow the steps in Figure 8 to add the Find Me Target profile. Any time you make a change in the Bluetooth Configurator, ensure that you save the changes (click **File** > **Save** or click the Save icon).

Figure 7. GATT Database View in BLE Configurator

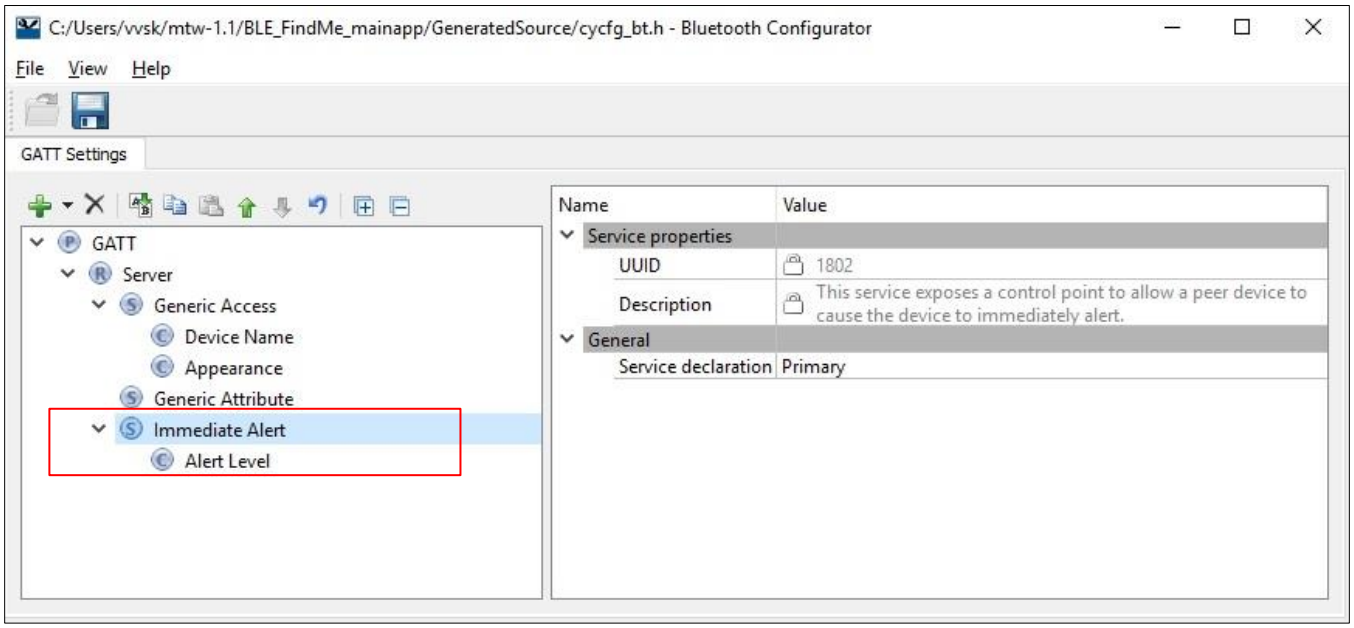
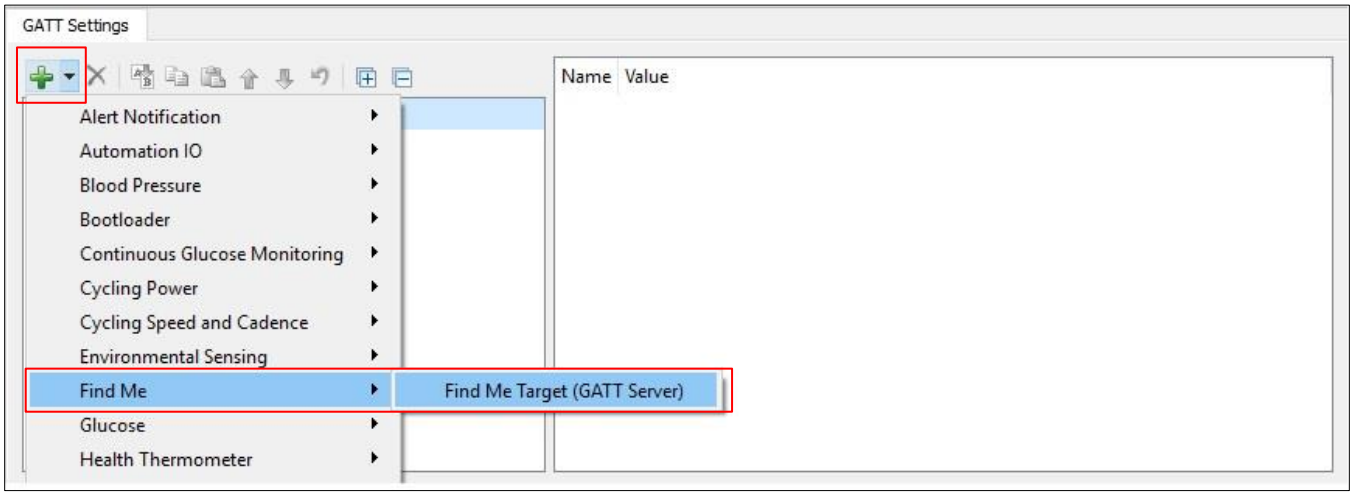


Figure 8. Adding Find Me Target Profile



- **Pins**: The pins used in the application are enabled in the **Pins** section of the Device Configurator as shown in Figure 9. Other pins that are applicable for using the kit are also enabled by default in Figure 9, but they are not used in this application. Table 2 provides more information on the pins used in this application.

Figure 9. Device Configurator - Pins

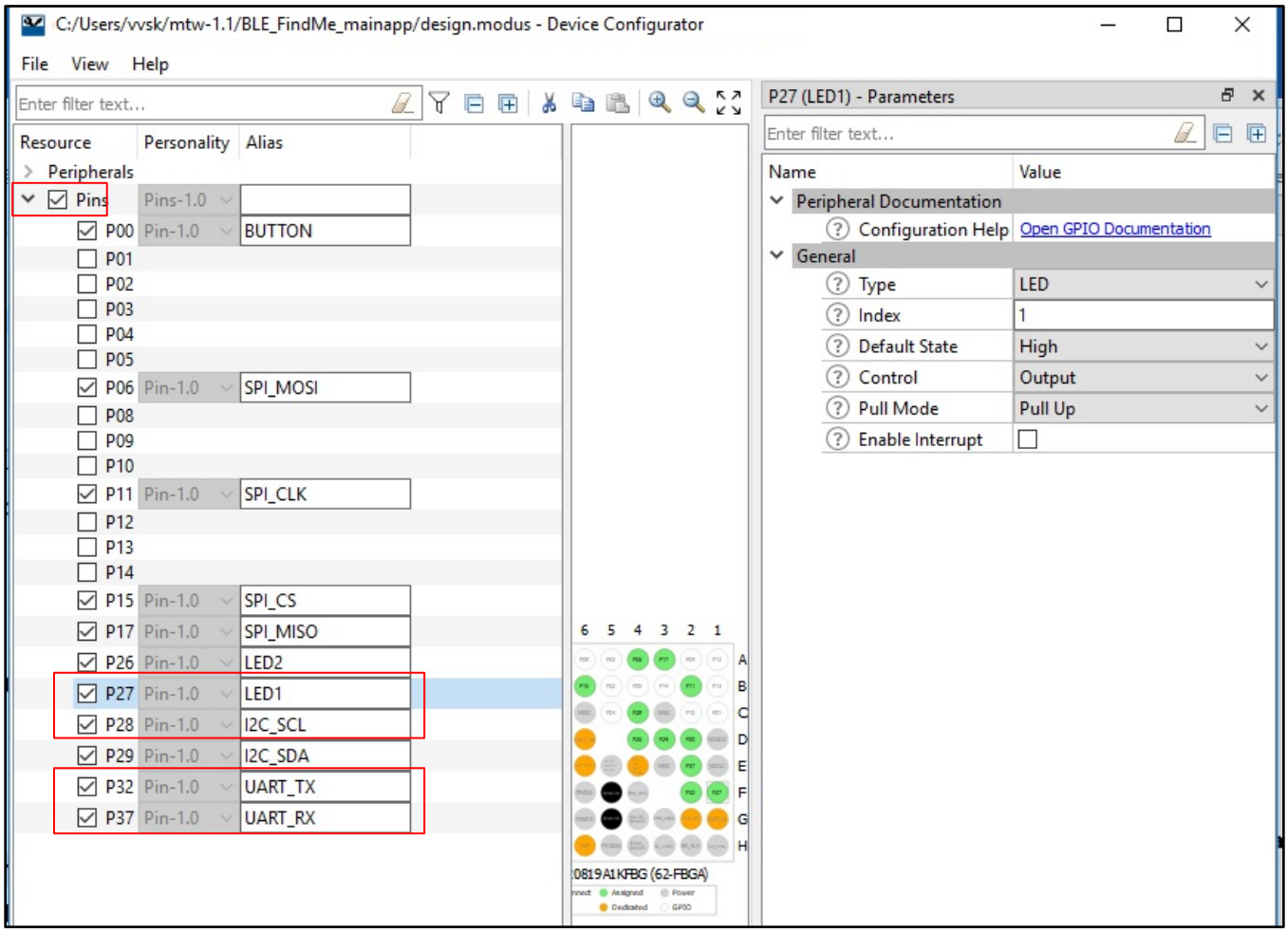


Table 2. Pin Mapping Details

Pin	Alias	Purpose	Settings
P26	LED2	Mapped to the red LED (LED2) on the kit. Indicates IAS Alert Level.	See Figure 10
P27	LED1	Mapped to the yellow LED (LED1) on the kit. Indicates the Advertising/Connected state of the BLE peripheral device.	
P32	PUART_TX	Tx pin of PUART peripheral	See Figure 11
P37	PUART_RX	Rx pin of PUART peripheral	See Figure 12

Figure 10. LED Pin Settings

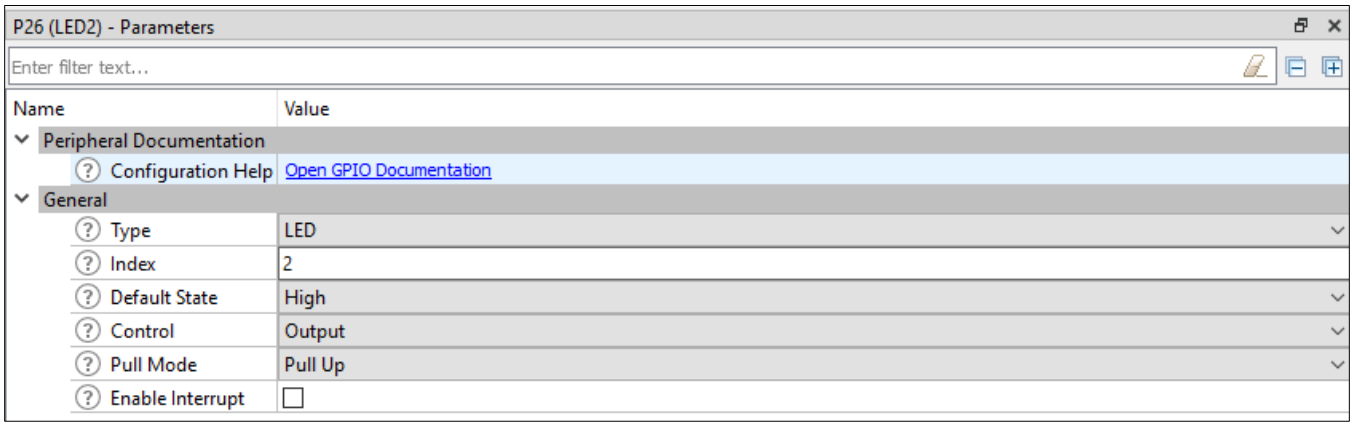


Figure 11. PUART Tx Pin Settings

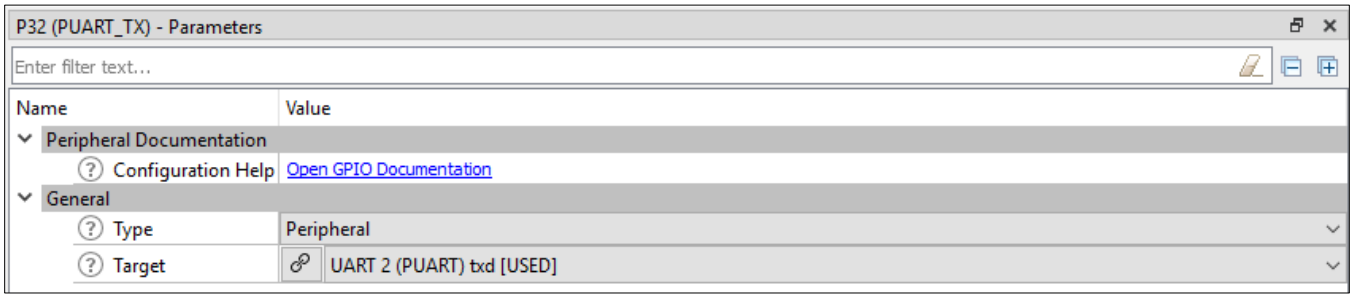
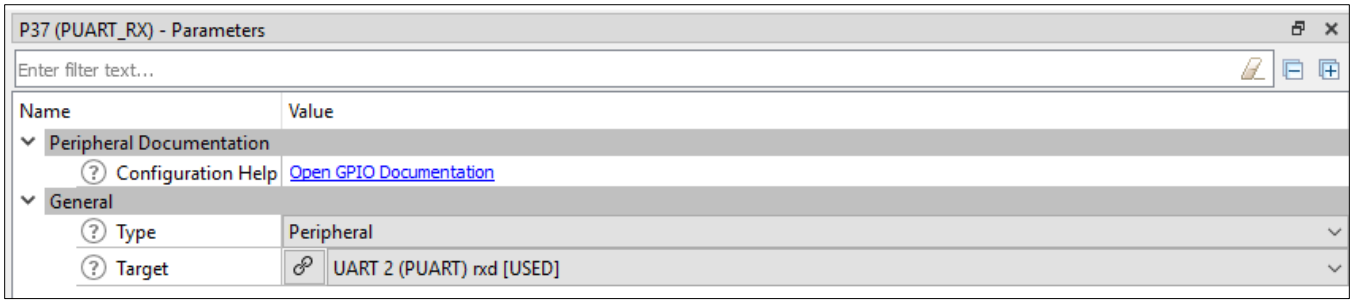


Figure 12. PUART Rx Pin Settings



Related Documents

Application Notes	
AN225684 – Getting Started with CYW20819	Describes CYW20819 Bluetooth SoC, software/hardware development ecosystem, and how to build your first BLE application using the device in ModusToolbox IDE
Code Examples	
Visit the CYW20819 code examples repository in Cypress GitHub portal for a comprehensive collection of code examples using ModusToolbox IDE	
Device Documentation	
CYW20819 Device Datasheet	
Development Kits	
CYW920819EVB-02 Evaluation Kit	
Tool Documentation	
ModusToolbox IDE	The Cypress IDE for IoT designers

Document History

Document Title: CE226123 – CYW20819 BLE Find Me Profile

Document Number: 002-26123

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6488517	VVSK	02/20/2019	New code example

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Code Examples | Projects | Videos | Blogs | Training| Components

Technical Support

cypress.com/support

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