CE226301 – CYW20819 BLE Throughput Measurement

Objective

This code example demonstrates measuring the BLE throughput at the GATT layer using the CYW20819 Bluetooth SoC with the 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 code example demonstrates measuring the BLE throughput at the GATT layer by continuously sending notifications on a custom characteristic. This application uses User Button 1 on the CYW920819EVB-02 kit to allow the user to select either the 1- Mbps or 2-Mbps PHY rate for the device. Two LEDs - red LED (LED2) and yellow LED (LED1) - on the CYW920819EVB-02 kit are used to indicate the device connected/disconnected state and the GATT congestion status respectively.

An iOS/Android mobile device or a PC can act as the BLE Central device which can connect to the Peripheral device, CYW920819EVB-02.

Hardware Setup

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

Software Setup

This code example consists of two parts: the GAP Central and the GAP Peripheral. For the GAP Central, 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 allow you to enable/disable GATT Notifications.

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.

   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; PUART is used for printing the Bluetooth stack and application trace messages.

2. 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.

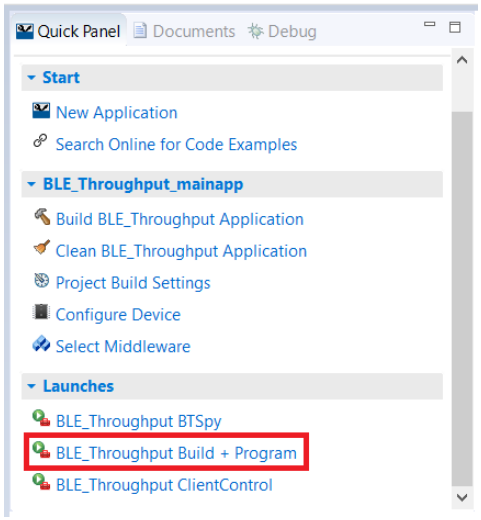
b. Once you have the repository on your local machine, the BLE throughput code example is located inside the repository folder at Code-Examples-BT-20819A1-1.0-for-ModusToolbox\CYW920819EVB02\apps\demo\throughput_test. Continue to follow the steps in KBA225201 to import this code example in ModusToolbox 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 **throughput_test** code example during the Import step in the Starter Application dialog window.

3. **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, or the application used a custom baud rate that the download process does not detect. 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).

4. To test using the CySmart mobile app, follow the steps below (see equivalent CySmart app screenshots in Figure 2). Refer to the user button description in the design implementation section for a description of how to switch from 1M to 2M PHY on the kit. This is useful when you want to measure throughput on 2M PHY if the mobile phone supports 2M PHY. The screenshots shown are from the Android version of the application; these will vary slightly with the iOS version.

   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.

   d. Swipe down on the CySmart app home screen to start scanning for BLE Peripherals; your device appears in the CySmart app home screen with the name “TPUT”. Select your device to establish a BLE connection. Once the connection is established, the red LED (LED2) will be ON.

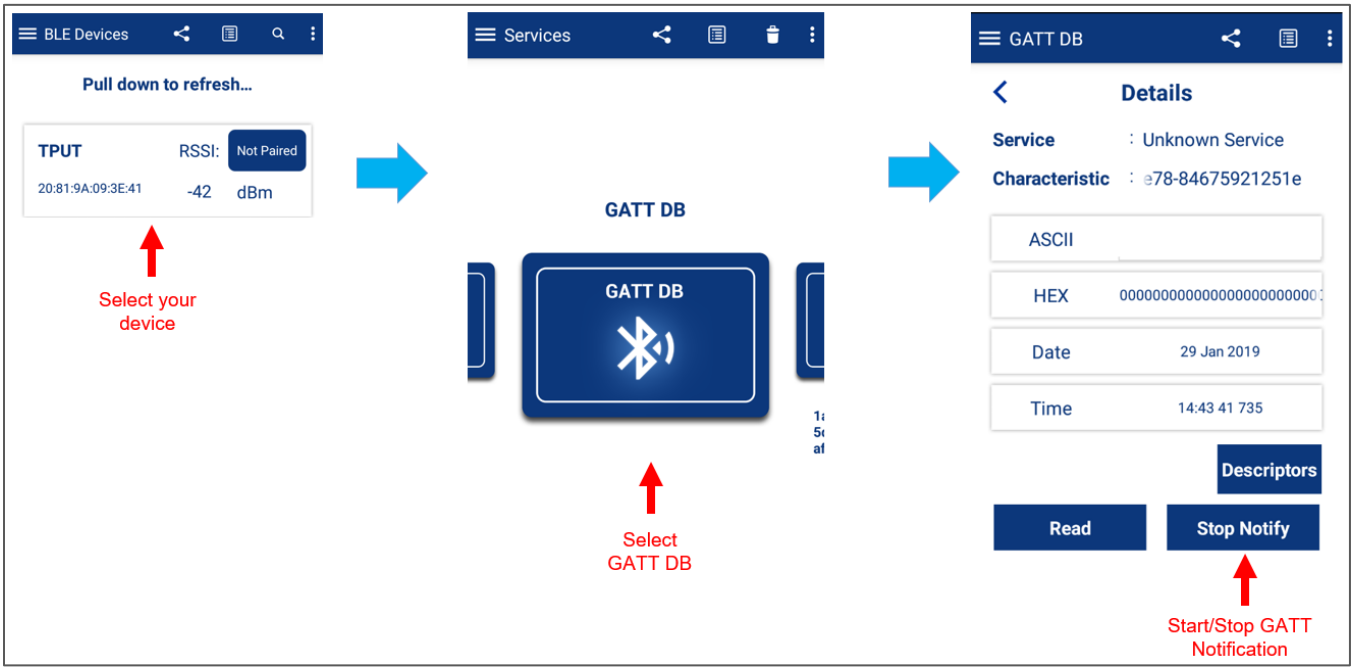
   e. Select **GATT DB** from the carousel view. (Swipe left or right to change carousel selections).

   f. Select **Unknown Service** and then select the **Characteristic.**

   g. Select **Notify**. CYW920819EVB-02 will start sending GATT notifications to the mobile. The yellow LED (LED1) will be ON while the device is sending notifications and will be OFF intermittently indicating GATT packet congestion.

   h. The device calculates the throughput, and the results are displayed on the PUART terminal as shown in Figure 3.

Figure 2. Testing with the CySmart App on Android



5. Open the **WICED Peripheral UART** (PUART) to view the Bluetooth stack and application traces:
   a. Use a serial terminal application and connect to the **PUART** port. Configure the terminal application to access the COM port using settings listed in Table 1.

Table 1. WICED Peripheral UART Settings

WICED Peripheral UART Serial Port Configuration	Value
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

Figure 3. Bluetooth stack and application traces on WICED PUART Serial Port

```
[application_start]  ***** THROUGHPUT APPLICATION *****
Advertisement State Change: BTM_BLE_ADVERT_UNDIRECTED_HIGH
Advertisement State Change: BTM_BLE_ADVERT_OFF
####### TPUT: GATT Notifications are disabled #######
Advertisement State Change: BTM_BLE_ADVERT_OFF
New connection parameters:
Connection interval = 26.25ms
####### TPUT: GATT Notifications enabled #######
TPUT: ######### GATT TPUT: 6344 Bytes per second. PHY: 1M ######
TPUT: ######### GATT TPUT: 80764 Bytes per second. PHY: 1M ######
TPUT: ######### GATT TPUT: 79544 Bytes per second. PHY: 1M ######
TPUT: ######### GATT TPUT: 74420 Bytes per second. PHY: 1M ######
TPUT: ######### GATT TPUT: 81984 Bytes per second. PHY: 1M ######
TPUT: ######### GATT TPUT: 79300 Bytes per second. PHY: 1M ######
TPUT: ######### GATT TPUT: 73444 Bytes per second. PHY: 1M ######
TPUT: ######### GATT TPUT: 80520 Bytes per second. PHY: 1M ######
TPUT: ######### GATT TPUT: 76616 Bytes per second. PHY: 1M ######
TPUT: ######### GATT TPUT: 75640 Bytes per second. PHY: 1M ######
####### TPUT: GATT Notifications are disabled #######
```

6. Do the following to test using the CySmart desktop application on PC:
   a. Plug the CYW920819EVB-02 (Peripheral device) into your computer.
   b. Build and download the application to the board. The CYW920819EVB-02 will start advertisements.
   c. Open the CySmart desktop application and connect to the CySmart CY5677 dongle (Central device). See the CySmart User Guide to learn how to use the desktop application.
   d. To measure GATT throughput:
      i. **Scan** and **Connect** to 'TPUT' device.
      ii. If asked, click 'Yes' to update the connection parameters. Refer to the Connection Interval section to understand how the connection interval affects the throughput.
      iii. Go to the device tab and click **Discover all attributes**.
      iv. Click on **Enable all Notifications**.

      The GATT throughput results (in bytes-per-second) will be displayed on the PUART terminal as shown in Figure 3.
      v. Click on **Disable All Notifications** to stop measuring GATT throughput.
      vi. Click **Disconnect** to disconnect from the Central device.

Design and Implementation

In this code example, the CYW20819 device on the kit acts as a GATT Server and GAP Peripheral. Once the device is connected, the Client can send a maximum transmission unit (MTU) Exchange Request to let the Server know about its MTU size. When the Client enables notifications on the Server, the Server sends a maximum Attribute protocol (ATT) payload of 244 bytes of data continuously. If no MTU Exchange is made by the Client, then the Server assumes the default MTU of 23 bytes and thus 20 bytes of data is continuously received by the Client.

CYW920819EVB-02 sends the data over BLE using GATT notifications on a custom characteristic after the notification is enabled by the GATT Client. The throughput is calculated every one second based on the amount of data pushed down successfully from the GATT layer. The throughput results are displayed in the PUART terminal in bytes-per-second.

The Link Layer sends the notification data divided into packets with the Protocol Data Unit (PDU) payload length. The maximum amount of data in one packet can be (251 – 7) bytes where 251 bytes is the maximum PDU payload size, and the 7 bytes consists of the L2CAP header (2-byte Length and 2-byte Channel ID and a 3-byte notification packet header or the ATT header).

The application code and the Bluetooth stack runs on the Arm® Cortex®-M4 core of the CYW20819 SoC. The application-level source files for this code example are listed in Table 2.

Table 2. Application Source Files

File Name	Comments
tput.c	Contains the `application_start()` function, which is the entry point for execution of the user application code after device startup. This file also contains the Bluetooth stack event handler, GATT event handler and timer functions.
tput_bt_cfg.c	These files contain the runtime Bluetooth stack configuration parameters like device name, advertisement settings, and connection settings.
tput_util.c, *tput_util.h*	These files contain the utility functions such as mapping event ID to String.
cycfg_bt.h *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.
cycfg_pins.c, *cycfg_pins.h* *cycfg_connectivity.h* *cycfg_notices.h*	These files reside in the *GeneratedSource* folder under the application folder. They contain the pin and peripheral configuration information that is generated using the device configurator.

File Name	Comments
tput_le_coc.c *tput_le_coc.h*	These files reside in the *le_coc* folder under the application folder. They enable throughput measurement at the LE Connection-Oriented-Channel (CoC) level. This is disabled by default in this code example. If you want to measure throughput at the L2CAP layer, enable LE COC in the ModusToolbox IDE by right-clicking on the project, and selecting **Change Application Settings** > **LE_COC_SUPPORT**.

User Button1 (Button D2 on the kit)

Switch between 1-Mbps and 2-Mbps PHY via user button1. On power up, CYW920819EVB-02 comes up in 1M PHY mode. When the BLE connection is established, if the peer is 2M PHY capable, it may request the device to set 2M PHY. If the peer allows both 1M and 2M PHY, then on every D2 button press on the kit, CYW920819EVB-02 will switch between the 1M and 2M PHY alternately.

LED1 (LED D1 on the kit)

LED1 indicates an active data transmission. It will be OFF when there is congestion or when data transfer is disabled. It will be ON when the device is actively transferring the data.

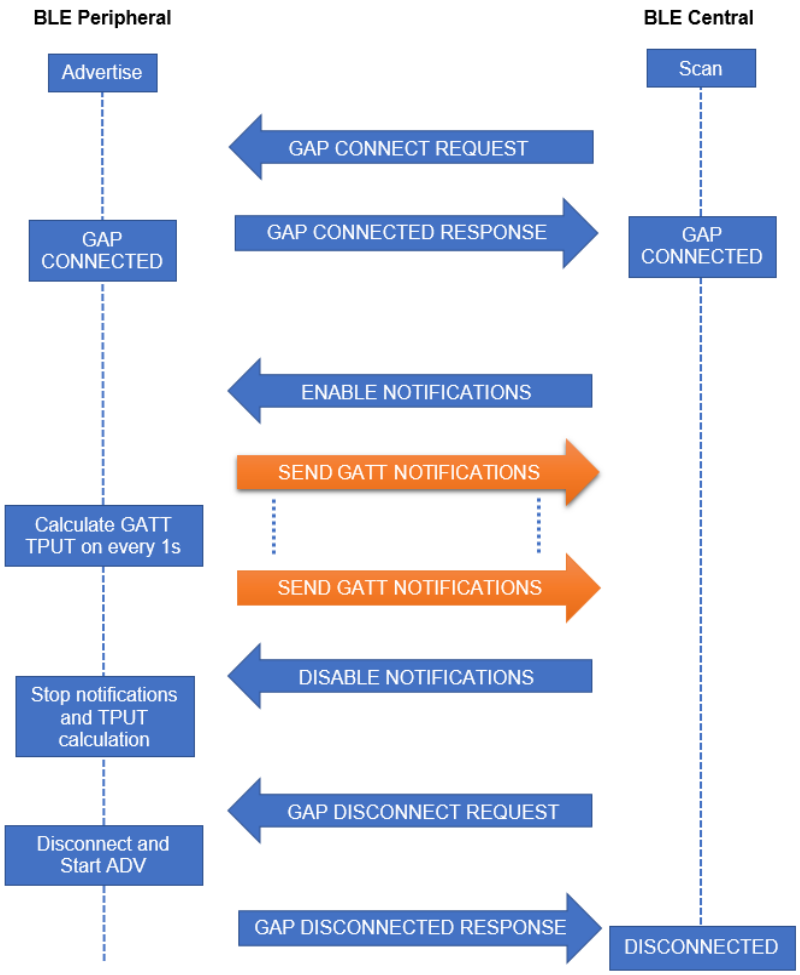
LED2 (LED D2 on the kit)

LED2 is ON when a GAP connection is established; OFF otherwise.

Application Flow

The flow of the application is depicted in Figure 4.

Figure 4. Throughput Application Flow Diagram



Optimize for maximum throughput

Some of the known factors that impact the data throughput are explained below.

PHY

The PHY rate being used will have direct impact on the maximum data throughput. You can select either 1-Mbps or 2-Mbps PHY on CYW920819EVB-02. In this code example, you have the option to switch between 1M and 2M PHY via User Button 1 on CYW920819EVB-02 after the BLE connection is established. Not all Central devices support both modes, so it may not be possible to switch for all Central devices.

Connection Interval

A BLE Connection Interval is the time between two data transfer events between the Central and the Peripheral device (in simple words, how often the devices talk). It ranges from 7.5 ms to 4 seconds (with increments of 1.25 ms). It also impacts how many packets can be transferred during one connection event. The higher the value, the more packets can be sent in one connection event. Choosing the connection interval value is purely application-specific. If your application sends large amounts of data at a time and needs good data throughput, then it makes sense to have larger connection interval.

The BLE connection is established with the connection interval value set by the Central device. However, the Peripheral may request a different value. The Central device makes the final decision and chooses a value that may be different from, but closer to, the requested value. In this code example, the Peripheral device requests a connection interval value of 26.25 ms but the value you get will depend on the Central device that you use.

The connection interval differs between iOS and Android. It also changes depending on the version of the OS running on the device. This is because the BLE radio may have to attend to other events from the OS and the number of packets sent per connection event may not reach the maximum possible by the BLE stack.

Note that the CySmart desktop application has an option to change the Connection Interval but the CySmart mobile app doesn't support that option. Refer to **Configuring Master Settings** section on the CySmart user guide for the detailed instructions on connection parameters.

The PUART log will indicate the connection interval once a connection is established whenever it is changed.

Data Length Extension (DLE)

DLE allows the packet to hold a larger payload up to 251 bytes. The DLE feature was introduced in version 4.2 of the Bluetooth SIG specification. Older versions of BLE can support the maximum payload of 27 bytes. The Peripheral (CYW920819EVB-02 of BLE version 5.0) has DLE enabled by default. If the Central device has DLE enabled (in BLE version >4.1) it will get a higher throughput.

ATT Maximum Transmission Unit (MTU)

ATT MTU determines the max amount of data that can be handled by the transmitter and receiver as well as how much they can hold in their buffers. The minimum ATT MTU allowed is 27 bytes. This allows a maximum of 20 bytes of ATT payload (3 bytes are used for the ATT header and 4 bytes for the L2CAP header). There is no limit on the maximum MTU value.

Because CYW920819EVB-02 has DLE enabled by default, it can transfer up to 251 – 4 (L2CAP header) = 247 bytes. So, the maximum ATT payload data will be 247 – 3 (ATT header) which comes up to 244 bytes. If the ATT MTU is exactly 247 bytes, then the ATT data will fit into a single packet. If the MTU is greater than 247 bytes, then the MTU will span multiple packets causing the throughput to go down because of packet overhead and timing in between the packets. This example application sets the MTU to 244 by default, but it can be changed by varying the value for the macro `GATT_NOTIFY_BYTES_LEN` in *tput.c*.

Figure 5. LE Packet Format

<table>
<tr><th>Preamble</th><th>Access Address</th><th colspan="5">PDU (2-257 bytes)</th><th>CRC</th></tr>
<tr><td rowspan="4">1 byte
(1M PHY)
2 bytes
(2M PHY)</td><td rowspan="4">4 bytes</td><td>LL Header</td><td colspan="3">Payload (0-251 bytes)</td><td rowspan="2">MIC
(Optional)</td><td rowspan="4">3 bytes</td></tr>
<tr><td rowspan="3">2 bytes</td><td>L2CAP Header</td><td colspan="2">ATT Data (0-247 bytes)</td></tr>
<tr><td rowspan="2">4 bytes</td><td colspan="2">ATT Header</td><td>ATT Payload</td><td rowspan="2">4 bytes</td></tr>
<tr><td>Op Code</td><td>Attribute Handle</td><td rowspan="2">Up to 244 bytes</td></tr>
<tr><td></td><td>1 byte</td><td>2 bytes</td><td></td></tr>
</table>

Packet Overhead

As shown in Figure 5, the LE packet includes many packet header bytes which get added up in each layer that are not accounted for in the application data throughput. To minimize the packet overhead, try to configure the ATT MTU size in such a way that at any time the ATT payload data will fit in a single LE packet. In this code example, the ATT MTU size used is 244 bytes, which exactly matches with the ATT payload data size of 244 bytes.

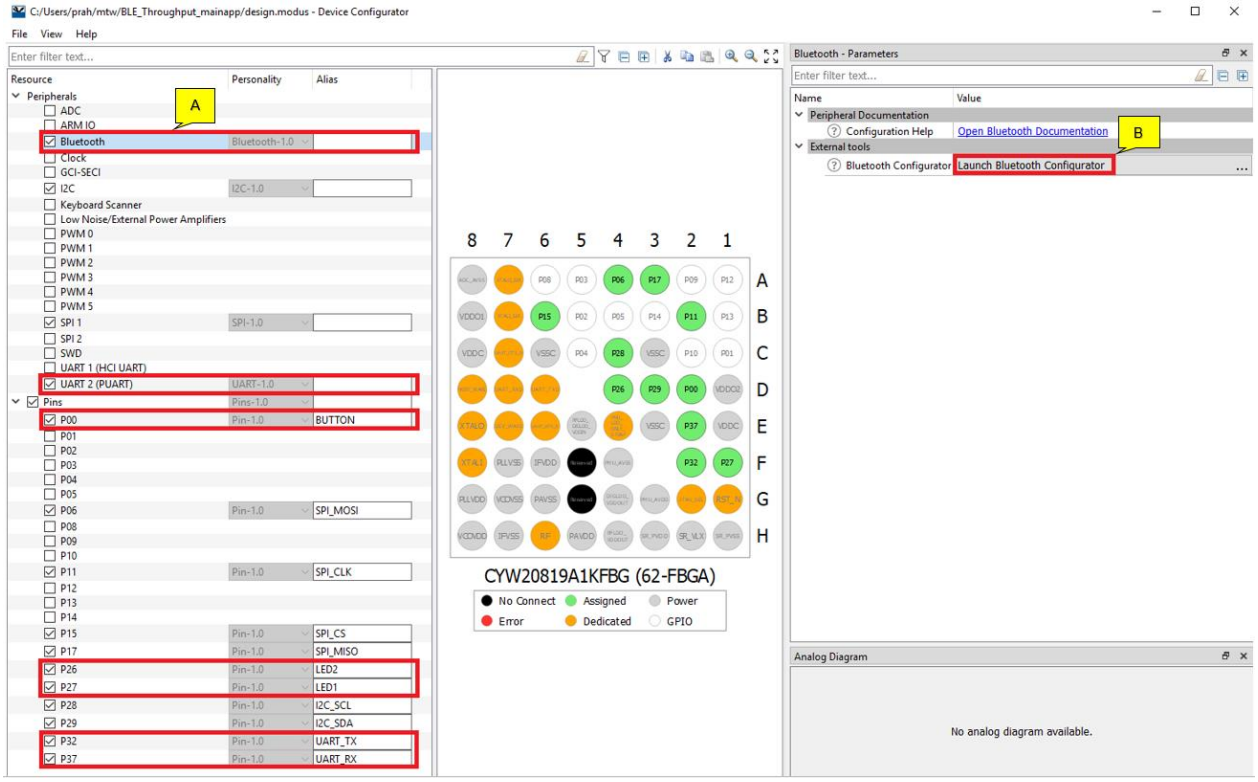
Resources and Settings

This section explains the ModusToolbox resources and their configuration as used in this 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 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, ensure that you save the updated configuration.

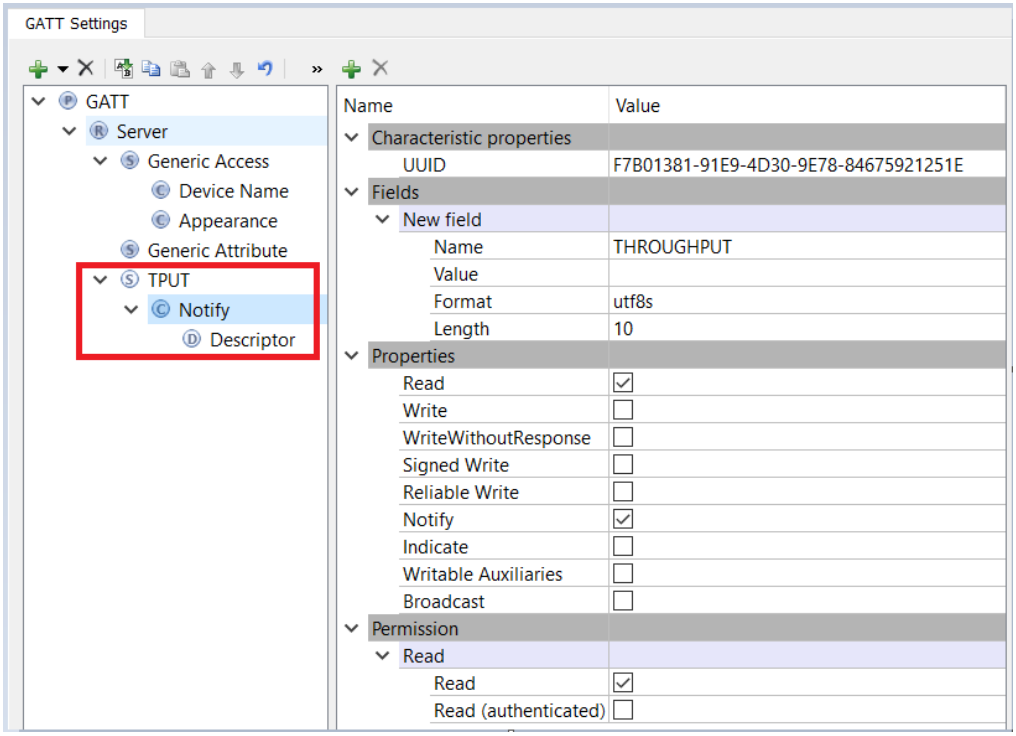
The code example uses only the highlighted options in Figure 6 for the application.

Figure 6. Device Configurator



- **Peripherals**: The design uses the Bluetooth and PUART peripherals which are enabled under the **Peripherals** section as shown in Figure 6.
- **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** button 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 custom GATT Service named **TPUT** has been added in Figure 7. Any time you make a change in the Bluetooth Configurator, ensure you save the changes.

Figure 7. GATT Database View in BLE Configurator



- **Pins**: The pins used in the application are enabled in the **Pins** section of the Device Configurator as shown in Figure 6. Table 3 provides more information on these pins.

Table 3. Pin Mapping Details

Pin Number	Alias	Purpose	Settings
P26	LED2	Mapped to the red LED (LED2) on the kit. Indicates BLE Advertising/Connected state of the BLE peripheral device.	See Figure 8
P27	LED1	Mapped to the yellow LED (LED1) on the kit. Indicates the GATT Notifications being sent to the Central.	
P00	BUTTON1	Mapped to the User Button (D2) on the kit.	See Figure 9
P32	PUART_TX	Tx pin of PUART peripheral.	See Figure 10
P37	PUART_RX	Rx pin of PUART peripheral.	See Figure 11

Figure 8. LED Pin Settings

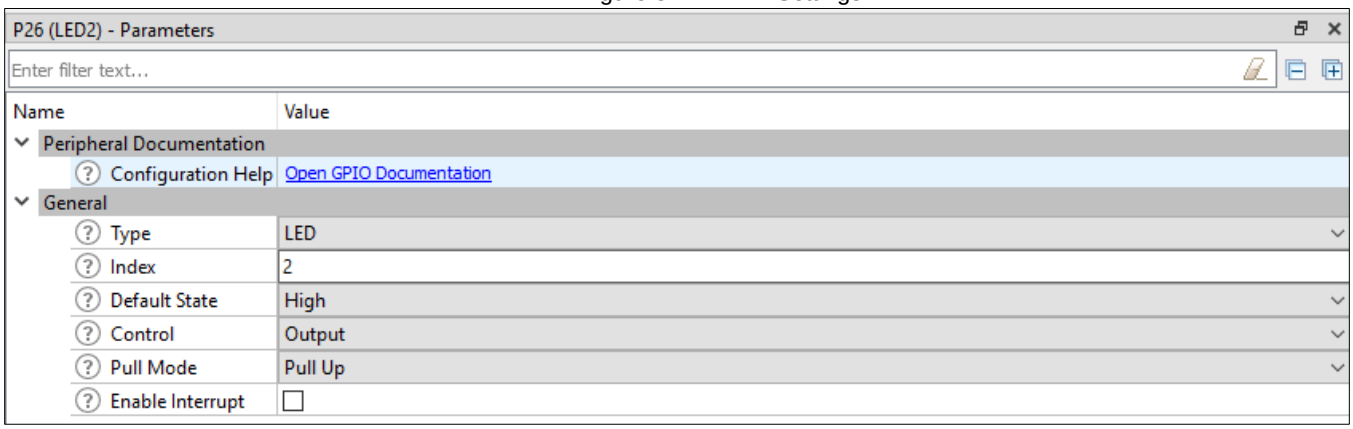


Figure 9. BUTTON1(D2) Settings

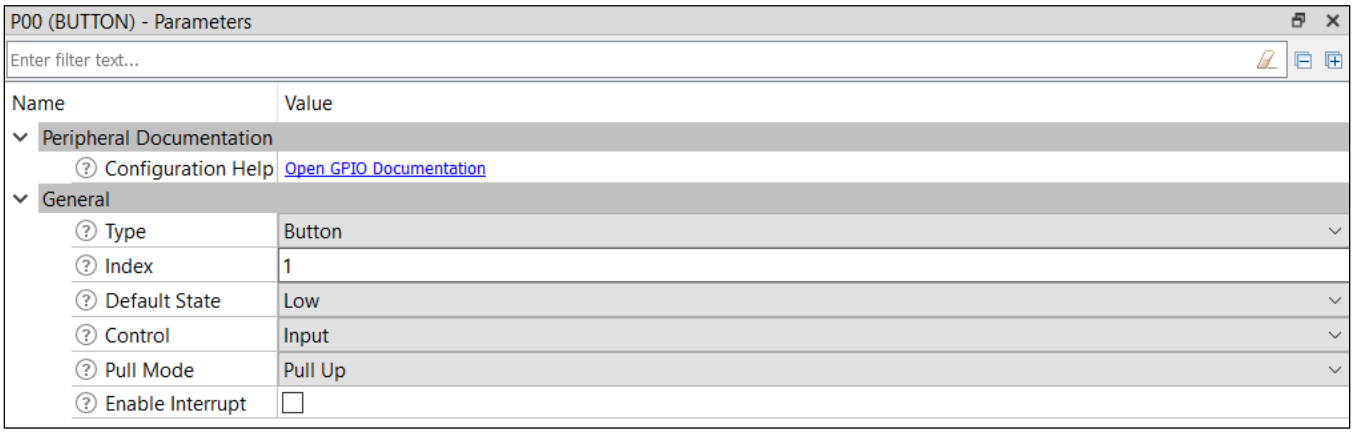


Figure 10. PUART Tx Pin Settings

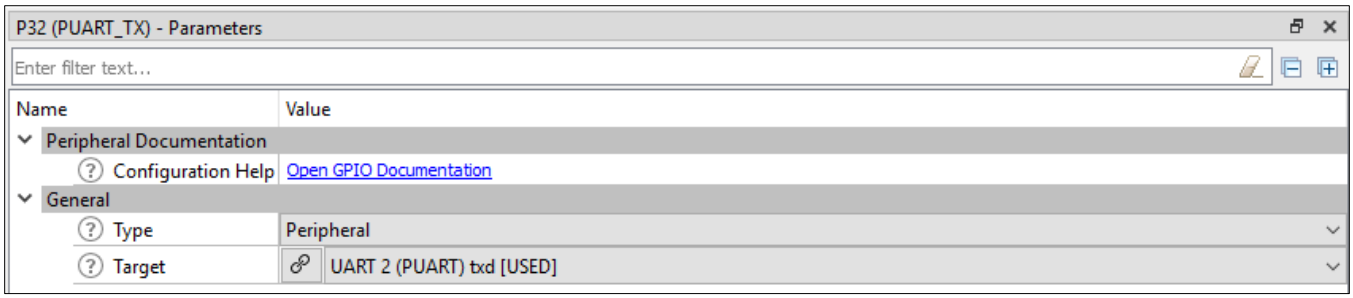
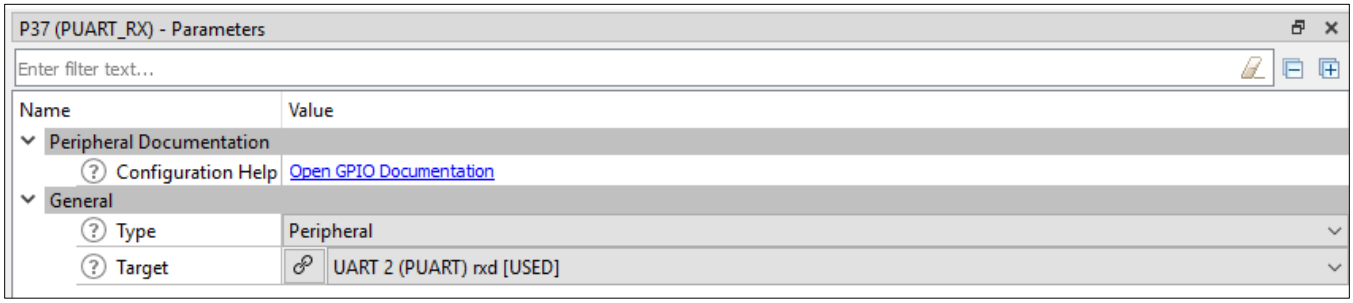


Figure 11. PUART Rx Pin Settings



Related Documents

Application Notes	
AN225684 – Getting Started with CYW20819	Describes CYW20819 Bluetooth SoC and how to build your first BLE application using the device in ModusToolbox IDE
Code Examples	
Visit the Cypress GitHub site 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: CE226301 – CYW20819 BLE Throughput Measurement

Document Number: 002-26301

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6489921	PRAH	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