

Objective

This code example demonstrates the new BLE 4.2 and 5.0 features of the PSoC® Creator™ BLE Component.

Overview

The BLE 4.2 Data Length Security Privacy code example demonstrates the following Bluetooth 4.2 and 5.0 features:

- Data Length Extension (DLE)
- Authenticated Low Energy (LE) Secure Connection (SC) Pairing
- Link Layer (LL) Privacy
- Speed up to 2 Mbps

BLE increases the speed and reliability of data transfer between BLE devices creating bigger packets that allow transferring data by up to 2.5 times faster than with the previous versions. The theoretical maximum throughput is up to 800 kbps. Increased data transfer speeds and packet capacity reduce transmission errors and lead to lower battery consumption.

With Authenticated LE SC pairing, the device uses the Elliptic Curve Diffie-Hellman (ECDH) algorithm to generate keys, and a new pairing procedure to exchange keys. It also provides a new protection method called Numeric Comparison to protect the connection from man-in-the-middle (MITM) attacks when both devices have displays and Yes/No input/output capabilities.

The BLE LL Privacy feature provides a higher level of security for connections over a period of time. After two devices exchange keys as part of a pairing process and store their keys (called bonding), they can use a private address instead of a public address to secure their connection. A device with the encryption keys can resolve that private address and establish the identity of this device. A device in the Central role can convey whether it supports privacy with address resolution. The Peripheral device checks if the peer device supports address resolution by reading the Central Address Resolution Characteristic before using a directed advertisement where the initiator's address is set to a Resolvable Private Address (RPA).

There are two modes of privacy: Device Privacy and Network Privacy. In the Network Privacy mode, if the peer device address is added to the resolving list, the device will accept packets with the peer RPA. If the peer device uses its identity address (random or public) instead of RPA, the device will reject it. By default, Network Privacy mode is used.

The 2 Mbps feature enables a new Physical (PHY) modulation scheme that allows increasing the data throughput between the two devices that support this feature. Refer to [Bluetooth Core Version 5.0](#).

Requirements

Tool: [PSoC Creator](#) 4.2 or later

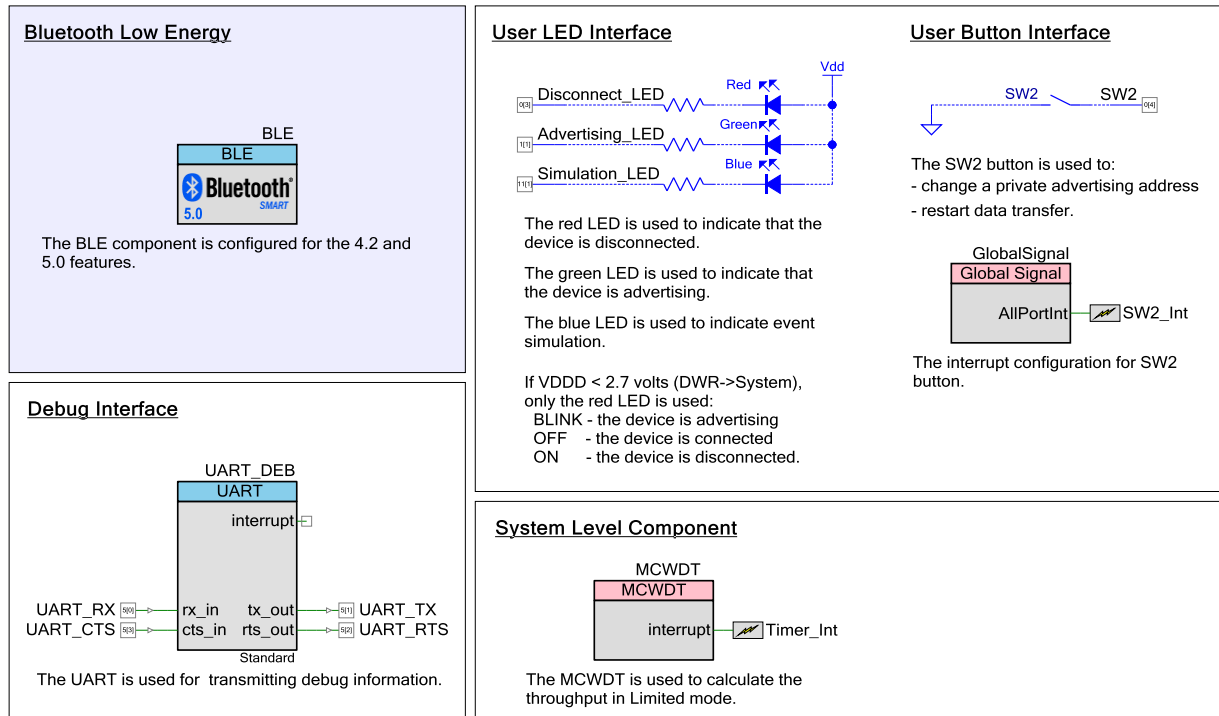
Programming Language: C (Arm® GCC 5.4-2016-q2-update or later)

Associated Parts: All [PSoC 6 MCU with BLE Connectivity \(PSoC 6 BLE\)](#) parts

Related Hardware: [CY8CKIT-062 PSoC 6 BLE Pioneer Kit](#)

Design

Figure 1. BLE 4.2 Data-Length Security Privacy Code Example Schematic



The project is configured to operate in the GAP Peripheral role and is intended to work with the GAP Central device: CySmart Central Emulation Tool and BLE Dongle.

When powered ON, the GAP Peripheral advertises with a public address. When it is connected to a BLE Dongle (GAP Central device), the GAP Peripheral initiates an authentication request; a key exchange happens between the GAP Central and GAP Peripheral devices. After the key exchange and authentication are complete, the GAP Peripheral stores the peer device information in the flash memory (bonding).

After bonding is complete and the GAP Peripheral is disconnected, the device starts advertising with a random address which is a resolvable private address. This address can be resolved by a GAP Central device if it has the required keys (specifically, the Identity Resolving Key or IRK). Only a GAP Central device which has the corresponding IRK can successfully resolve the private address of the GAP Peripheral and identify if this GAP Peripheral is the same as the one it connected to earlier. Thus, only a previously bonded GAP Central device can decipher this GAP Peripheral device.

A new GAP Central device can also connect to this GAP Peripheral (which advertises with a private address) without resolving its address, but then it would require a fresh authentication procedure. Thus, the connection between the GAP Peripheral and the original GAP Central device is kept private.

After authentication is complete, a Peripheral device reads a Central Address Resolution Characteristic value from the Central device and checks if a peer device supports the address resolution in the Link Layer. If address resolution is supported by a Central device, and the directed advertising mode is configured in the Configuration Characteristic, after disconnection, the Peripheral uses direct advertising with a resolvable private address.

You can clear the GAP Peripheral's bonded device list by pressing 'r' in the terminal program window. After the bonded device list is cleared, authentication on the next connection with the same GAP Central device will require a key exchange. To do this, the GAP Central device should not have this GAP Peripheral as part of its bonded device list. Therefore, you should also clear the GAP Central's bonded device list.

The project continuously sends data over BLE using GATT notifications on a custom TX Data Characteristic after the notification is enabled by the GATT Client, and calculates the amount of data transmitted in 10 seconds. This is then converted into a throughput value in kbps terms and displayed on a debug terminal. The data transmission is stopped when notification is disabled by the peer device.

The device can operate in Limited Data Transfer mode when the corresponding value is written into the Configuration characteristic. In this mode, the amount of data sent by the device is limited to the Data Length field value. The data sending transaction can be triggered by enabling notification and indication, and later by pressing the **SW2** key.

After the device is connected, the Client can send an MTU Exchange Request to let the Server know about its MTU size. When the Client enables notifications on the Server, the Server sends (MTU - 3) bytes of data continuously whenever its BLE Stack is free. Here, '3' is size of the notification packet header (1-byte attribute opcode and 2-byte attribute handle).

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

Refer to Bluetooth Specification version 4.2 [Vol 3, Part A] Section 2.4 for the L2CAP packet format and [Vol 3, Part F] Section 3.47.1 for the Notification packet format.

If no MTU Exchange is done, a default MTU of 23 bytes is taken and 20 bytes of data is continuously sent. The maximum MTU size can be 512 bytes. The maximum PDU is 251 bytes; therefore, the maximum notification is divided into three packets during transmission.

The maximum throughput depends on the packet fragmentation, modulation scheme, connection interval, and how quickly the CPU calls the `Cy_BLE_ProcessEvents()` API to push fragmented packets. 70 MHz is the optimal clock frequency for SlowClk to reach the maximum throughput in a worst packet fragmentation case.

Figure 2 is an example of the relationship between the notification length and the throughput at a 120-ms connection interval with the CPU clock frequency set to 70 MHz.

Figure 2. Throughput Example

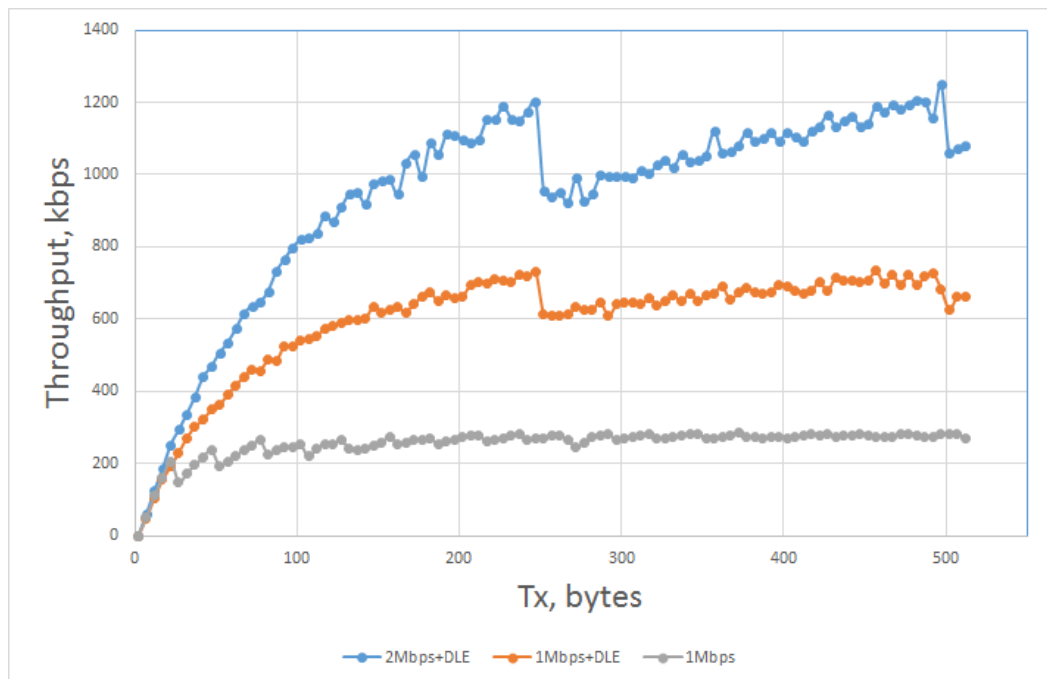


Table 1. Commands List

Command	Description
R	Remove the bond.
S	Display the bond list.

Press any key to disable Deep Sleep and prompt the above list to the Terminal emulator. Deep Sleep mode will resume after the command that is entered completes.

Design Considerations

Using UART for Debugging

Download and install a serial port communication program. Freeware such as Bray's Terminal and PuTTY are available on the web.

1. Connect the PC and kit with a USB cable.
2. Open the Device Manager program in your PC, find a COM port that the kit is connected to, and note the port number.
3. Open the serial port communication program and select the previously noted COM port.
4. Configure the Baud rate, Parity, Stop bits, and Flow control information in the PuTTY configuration window. The default settings: Baud rate – 115200, Parity – None, Stop bits – 1, Flow control – XON/XOFF. These settings must match the configuration of the PSoC Creator UART Component in the project.
5. Start communicating with the device as explained in the [Operation](#) section.

The UART debugging can be disabled by setting the `DEBUG_UART_ENABLED` to `DISABLED` in `common.h`.

LED Behavior for VDDD Voltage < 2.7 Volts

If the VDDD voltage is set to less than 2.7 volts in the DWR settings **System** tab, only the red LED is used. The red LED blinks to indicate that the device is advertising. The red LED is OFF when the device is connected to a peer device. When the device is in disconnected, the red LED stays ON.

Switching the CPU Cores Usage

This section describes how to switch between different CPU cores usage (Single core/ Dual core) in the BLE Peripheral Driver Library (PDL) examples.

The BLE Component has the CPU Core parameter that defines the cores usage. It can take the following values:

- **Single core (Complete Component on CM0+)** – only CM0+ core will be used.
- **Single core (Complete Component on CM4)** – only CM4 core will be used.
- **Dual core (Controller on CM0+, Host and Profiles on CM4)** – both cores will be used: CM0+ for the Controller and CM4 for the Host and Profiles.

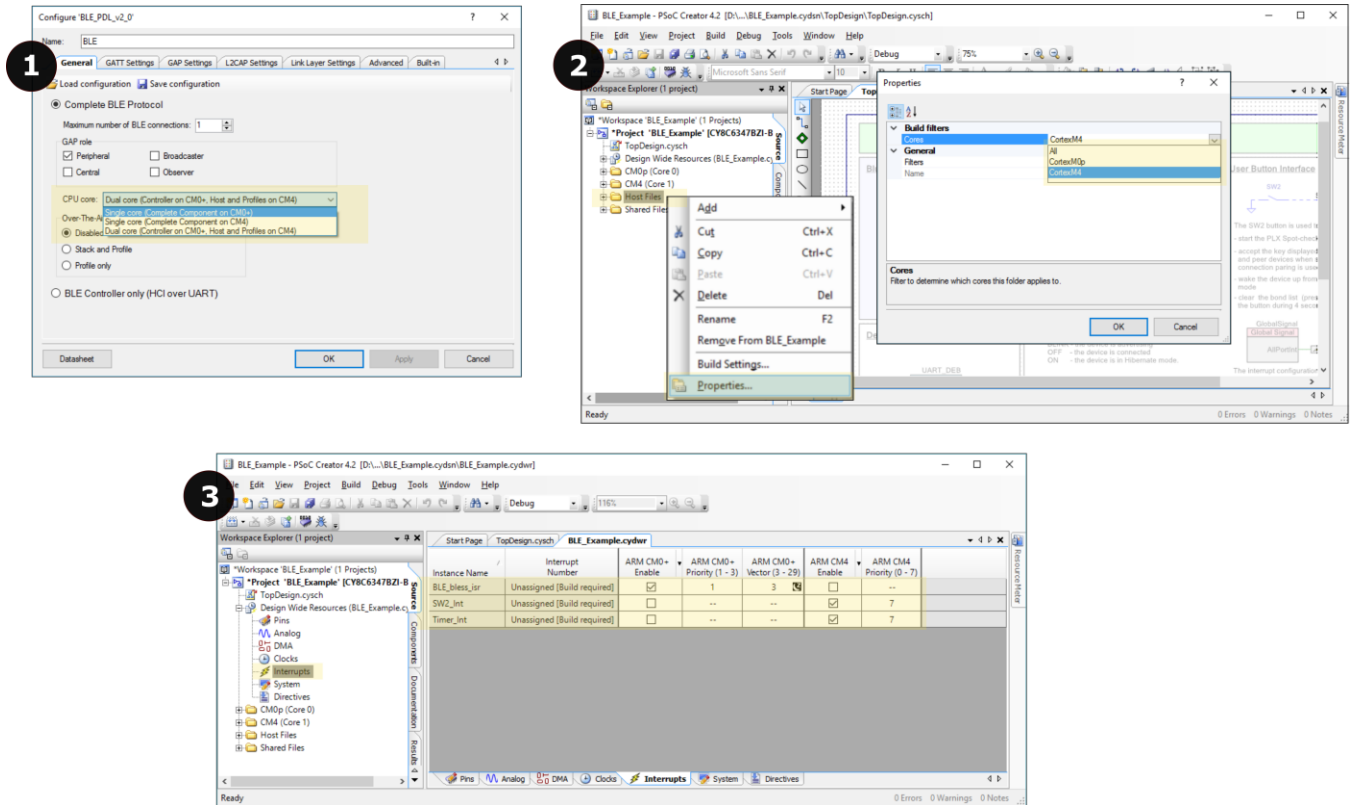
The BLE examples' structure allows easy switching between different CPU cores options. Keep in mind the following:

- All application host files must be run on the host core.
- The BLE Subsystem (BLESS) interrupt must be assigned to the core where the controller runs.
- All additional interrupts (SW2, MCWDT, etc.) used in the example must be assigned to the host core.

Do the following to switch the CPU Cores usage:

1. In the BLE Component Customizer **General** tab, select the appropriate CPU core option.
2. Change the cores Properties to CortexM4 or CortexC0p for the project folder Host Files in dependence of which CPU core was chosen in Step 1 as follows:
 - For **Single core (Complete Component on CM0+)** option – **CM0+**
 - For **Single core (Complete Component on CM4)** option – **CM4**
 - For **Dual core (Controller on CM0+, Host and Profiles on CM4)** option – **CM4**
3. Assign BLE_bless_isr and other peripheral (button – SW2, timer(s), etc.) interrupts to appropriate core in **DWR > Interrupts** tab:
 - For **Single core (Complete Component on CM0+)** option: BLE_bless_isr and peripheral interrupts on **CM0+**
 - For **Single core (Complete Component on CM4)** option: BLE_bless_isr and peripheral interrupts on **CM4**
 - For **Dual core (Controller on CM0+, Host and Profiles on CM4)** option: BLE_bless_isr interrupt on **CM0+**, other peripheral interrupts on **CM4**

Figure 3. Steps for Switching the CPU Cores Usage



Hardware Setup

The code example was created for the **CY8CKIT-062 PSoC® 6 BLE Pioneer Kit**.

Pin assignment and connections required on the development board for the supported kits are in [Table 2](#).

Table 2. Pin Assignment

Pin Name	Development Kit	Comment
	CY8CKIT-062	
\UART_DEB:rx\	P5[0]	
\UART_DEB:tx\	P5[1]	
\UART_DEB:rts\	P5[2]	
\UART_DEB:cts\	P5[3]	
Advertising_LED	P1[1]	The green color of the RGB LED
Disconnect_LED	P0[3]	The red color of the RGB LED
Simulation_LED	P11[1]	The blue color of the RGB LED
SW2	P0[4]	

Components

Table 3 lists the PSoC Creator Components used in this example and the hardware resources used by each Component.

Table 3. PSoC Creator Components List

Component	Hardware Resources
BLE	1 BLE, 1 Interrupt
UART_DEB	1 SCB
MCWDT_1	
SW2	1 pin
Wakeup_Interrupt	1 interrupt
Disconnect_LED, Advertising_LED, Simulation_LED	3 pins

Parameter Settings

BLE Component

The BLE Component is configured as a Custom Profile in the GATT Server role and GAP Peripheral role. The Throughput Service that is built using the Custom Service exposes control and data for proper coordination between the Client and Server. Table 4 shows the UUID definitions for the Throughput Service including two custom characteristics.

Table 4. Throughput Service UUID

UUID Value	Description
00090000-F8CE-11E4-ABF4-0002A5D5C51B	Throughput Service
00090001-F8CE-11E4-ABF4-0002A5D5C51B	TX Data Characteristic
00090002-F8CE-11E4-ABF4-0002A5D5C51B	Configuration Characteristic

The TX Data Characteristic is configured with the Notification and Indication properties and a Client Characteristic Configuration descriptor to enable data transmission. The characteristic length is set to the maximum possible value for notifications – 509 octets (MTU - 3).

The Configuration Characteristic has Read and Write properties. The Configuration Characteristic value fields are described in Table 5. The length of the characteristic value is 5 octets.

Table 5. Configuration Characteristic

Field	Octets	Bit Number	Description
Mode	1	0	Defines the mode of data transfer. 0: configures consecutive data transfer, throughput calculation, and the display process. 1: configures Limited data-transfer mode with a length set in the Data Length field. The last packet is sent by Indication and the throughput is calculated after the Indication confirmation is received.
		1	Defines the private advertising mode after bonding. 0: undirected advertising mode. 1: low duty-cycle directed advertising mode.
		2-7	Reserved for future use (RFU).
Data Length	4		Defines the data length in bytes to be transmitted in Limited mode.

The BLE Component is configured to have the following properties:

- Public Device Address: 00A050-00001D
- MTU: 512 bytes
- Link Layer maximum TX payload size: 251 bytes

- Link Layer maximum RX payload size: 251 bytes
- Enable LE 2 Mbps

The security level is set to Authenticated LE SC pairing with encryption, but individual attribute permissions for the Client Characteristic Configuration descriptor and Configuration Characteristic are set to Authentication required only. This allows the project to work with the SC (mode 1 level 4) and fallback to the legacy authenticated mode (mode 1 level 3) if SC is not supported by the peer device.

Figure 4. General Settings

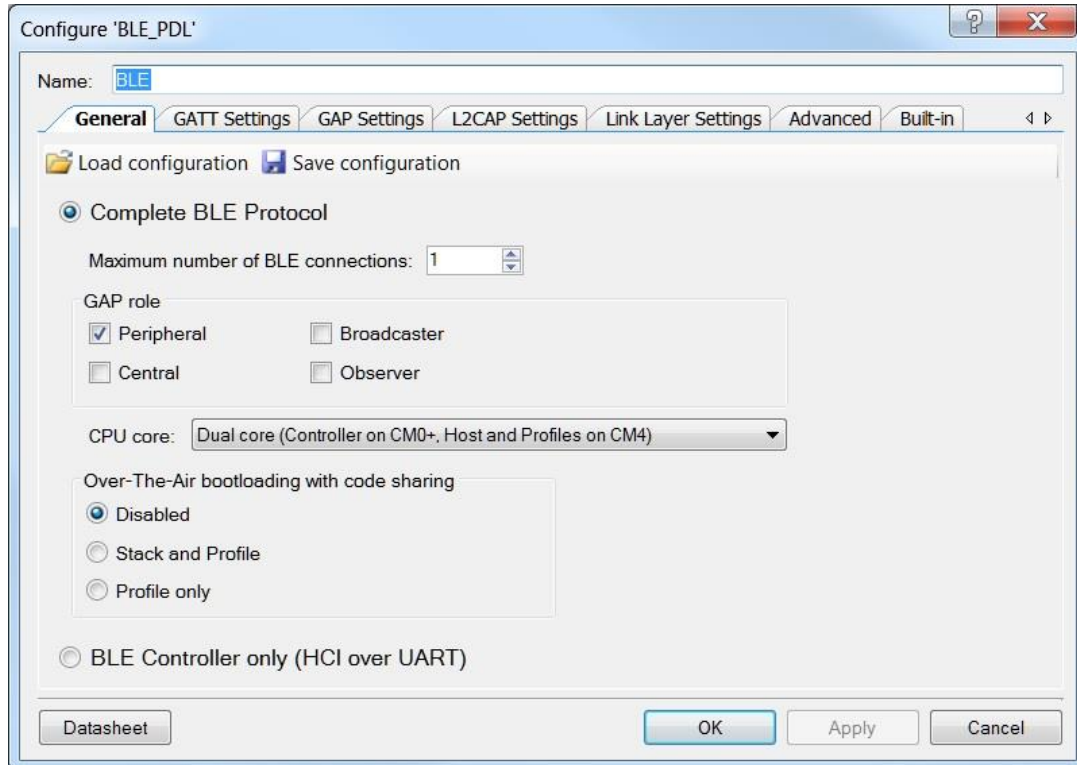
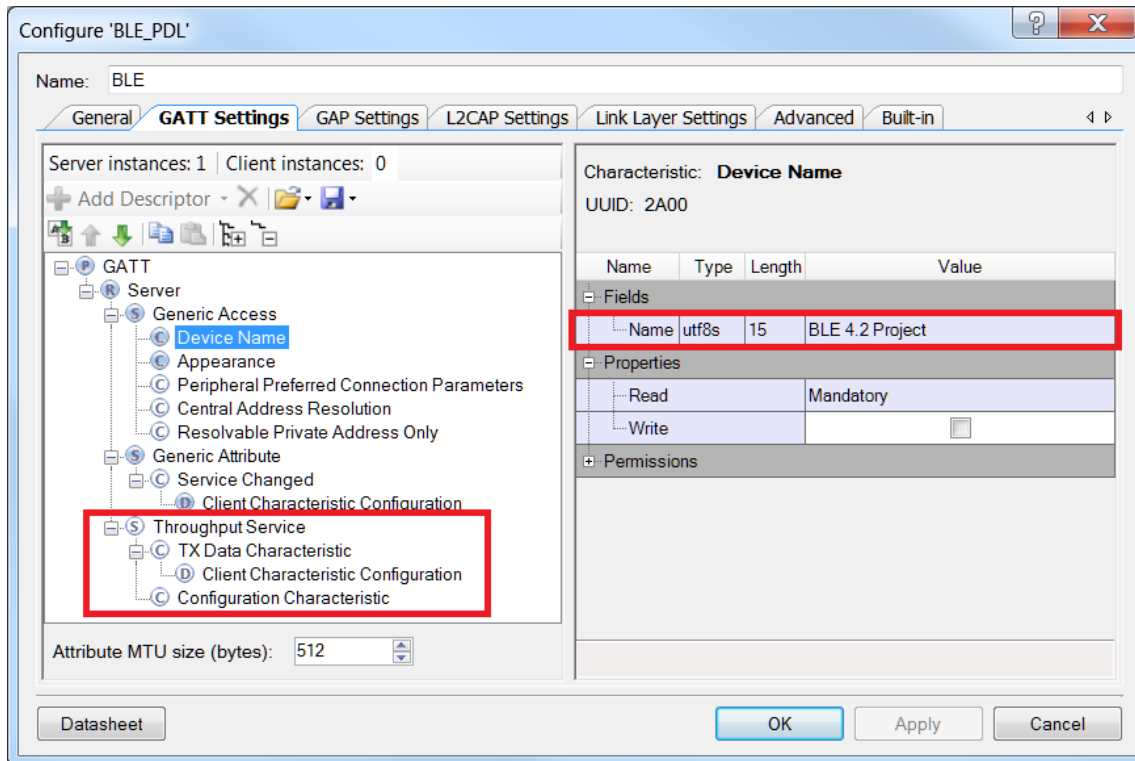


Figure 5. GATT Settings



Configure 'BLE_PDL'

Name: BLE

General | **GATT Settings** | GAP Settings | L2CAP Settings | Link Layer Settings | Advanced | Built-in

Server instances: 1 | Client instances: 0

+ Add Descriptor

GATT

- Server
 - Generic Access
 - Device Name**
 - Appearance
 - Peripheral Preferred Connection Parameters
 - Central Address Resolution
 - Resolvable Private Address Only
 - Generic Attribute
 - Service Changed
 - Client Characteristic Configuration
 - Throughput Service
 - TX Data Characteristic
 - Client Characteristic Configuration
 - Configuration Characteristic

Attribute MTU size (bytes): 512

Datasheet

OK Apply Cancel

Characteristic: **Device Name**
 UUID: 2A00

Name	Type	Length	Value
Name	utf8s	15	BLE 4.2 Project

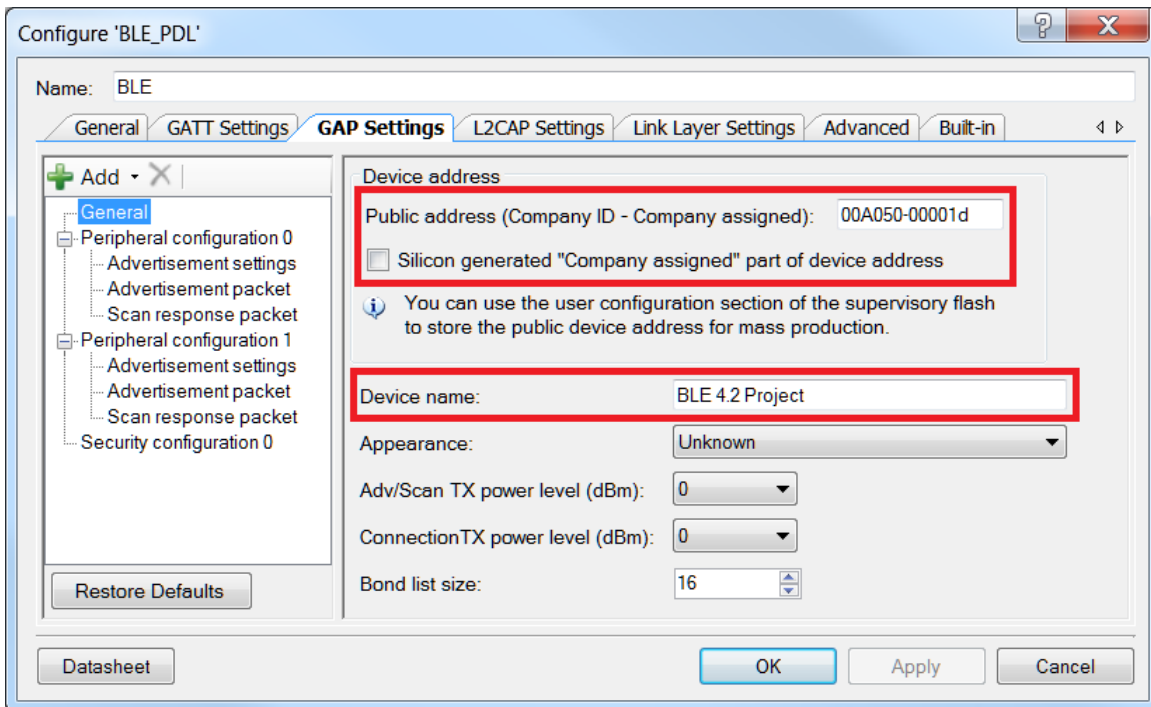
Fields

Properties

Read	Write
Mandatory	

Permissions

Figure 6. GAP Settings



Configure 'BLE_PDL'

Name: BLE

General | GATT Settings | **GAP Settings** | L2CAP Settings | Link Layer Settings | Advanced | Built-in

+ Add

General

- Peripheral configuration 0
 - Advertisement settings
 - Advertisement packet
 - Scan response packet
- Peripheral configuration 1
 - Advertisement settings
 - Advertisement packet
 - Scan response packet
- Security configuration 0

Restore Defaults

Datasheet

OK Apply Cancel

Device address

Public address (Company ID - Company assigned): 00A050-00001d

☐ Silicon generated "Company assigned" part of device address

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

Device name: BLE 4.2 Project

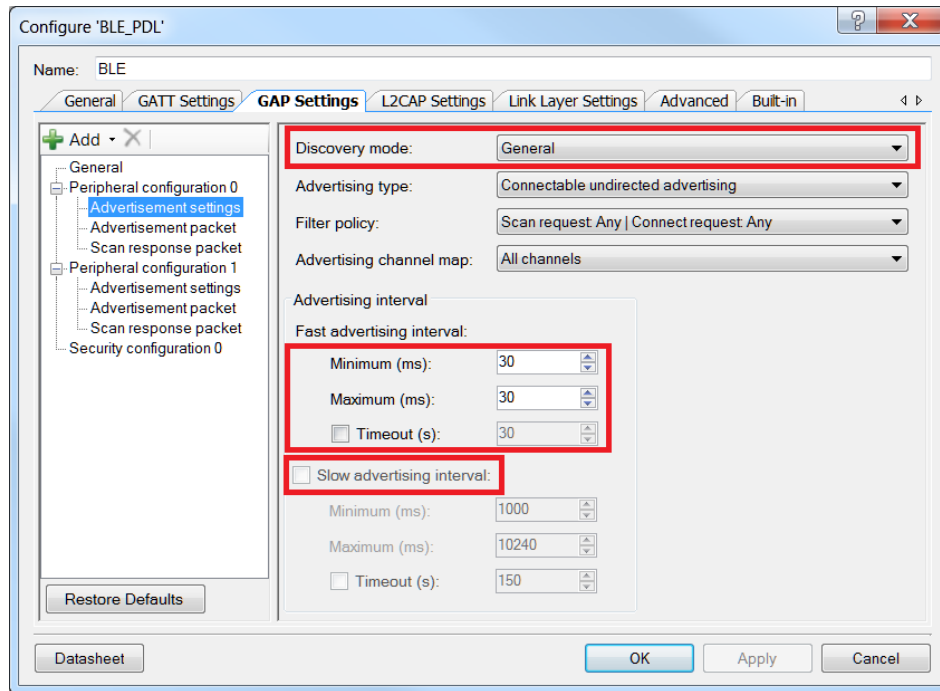
Appearance: Unknown

Adv/Scan TX power level (dBm): 0

ConnectionTX power level (dBm): 0

Bond list size: 16

Figure 7. GAP Settings: Peripheral configuration 0 > Advertisement Settings



Configure 'BLE_PDL'

Name: BLE

General | GATT Settings | **GAP Settings** | L2CAP Settings | Link Layer Settings | Advanced | Built-in

Discovery mode: General

Advertising type: Connectable undirected advertising

Filter policy: Scan request Any | Connect request Any

Advertising channel map: All channels

Advertising interval

Fast advertising interval:

Minimum (ms): 30

Maximum (ms): 30

Timeout (s): 30

Slow advertising interval: ☐

Minimum (ms): 1000

Maximum (ms): 10240

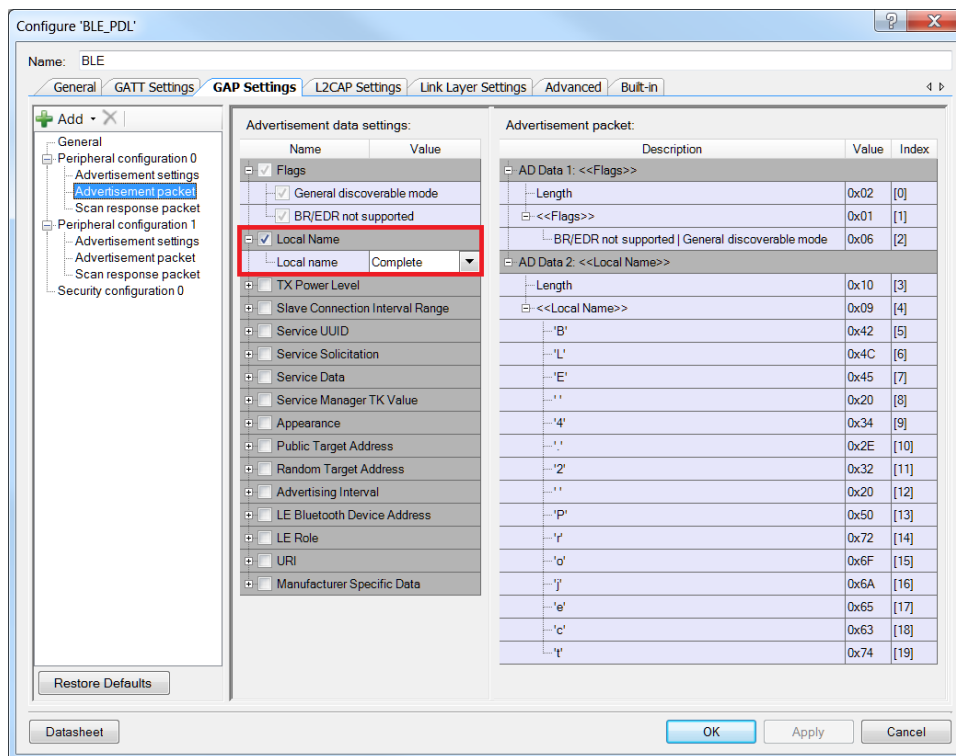
Timeout (s): 150

Restore Defaults

Datasheet

OK Apply Cancel

Figure 8. GAP Settings: Peripheral configuration 0 > Advertisement Settings



Configure 'BLE_PDL'

Name: BLE

General | GATT Settings | **GAP Settings** | L2CAP Settings | Link Layer Settings | Advanced | Built-in

Advertisement data settings:

Name	Value
<input checked="" type="checkbox"/> Flags	
<input checked="" type="checkbox"/> General discoverable mode	
<input checked="" type="checkbox"/> BR/EDR not supported	
<input checked="" type="checkbox"/> Local Name	Complete
<input type="checkbox"/> TX Power Level	
<input type="checkbox"/> Slave Connection Interval Range	
<input type="checkbox"/> Service UUID	
<input type="checkbox"/> Service Solicitation	
<input type="checkbox"/> Service Data	
<input type="checkbox"/> Service Manager TK Value	
<input type="checkbox"/> Appearance	
<input type="checkbox"/> Public Target Address	
<input type="checkbox"/> Random Target Address	
<input type="checkbox"/> Advertising Interval	
<input type="checkbox"/> LE Bluetooth Device Address	
<input type="checkbox"/> LE Role	
<input type="checkbox"/> URI	
<input type="checkbox"/> Manufacturer Specific Data	

Advertisement packet:

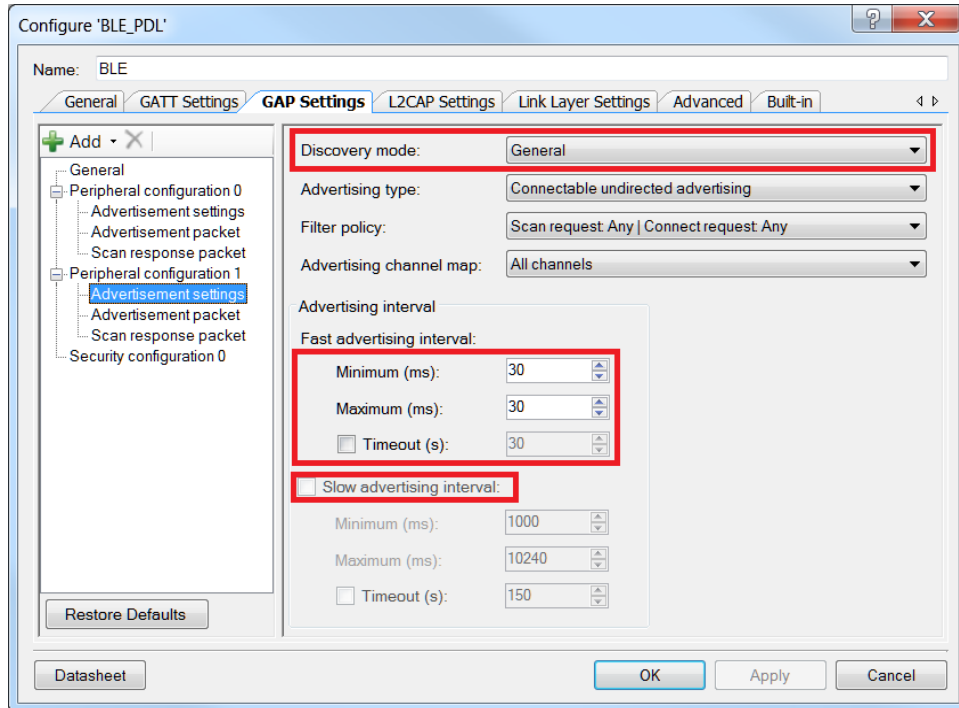
Description	Value	Index
AD Data 1: <<Flags>>		
Length	0x02	[0]
<<Flags>>	0x01	[1]
BR/EDR not supported General discoverable mode	0x06	[2]
AD Data 2: <<Local Name>>		
Length	0x10	[3]
<<Local Name>>	0x09	[4]
'B'	0x42	[5]
'L'	0x4C	[6]
'E'	0x45	[7]
'.'	0x20	[8]
'4'	0x34	[9]
'.'	0x2E	[10]
'2'	0x32	[11]
'.'	0x20	[12]
'P'	0x50	[13]
'r'	0x72	[14]
'o'	0x6F	[15]
'j'	0x6A	[16]
'e'	0x65	[17]
'c'	0x63	[18]
't'	0x74	[19]

Restore Defaults

Datasheet

OK Apply Cancel

Figure 9. GAP Settings: Peripheral configuration 1 > Advertisement Settings



Configure 'BLE_PDL'

Name: BLE

General | GATT Settings | **GAP Settings** | L2CAP Settings | Link Layer Settings | Advanced | Built-in

General

- Peripheral configuration 0
 - Advertisement settings
 - Advertisement packet
 - Scan response packet
- Peripheral configuration 1
 - Advertisement settings**
 - Advertisement packet
 - Scan response packet
- Security configuration 0

Restore Defaults

Discovery mode: General

Advertising type: Connectable undirected advertising

Filter policy: Scan request: Any | Connect request: Any

Advertising channel map: All channels

Advertising interval

Fast advertising interval:

Minimum (ms): 30

Maximum (ms): 30

Timeout (s): 30

Slow advertising interval:

Minimum (ms): 1000

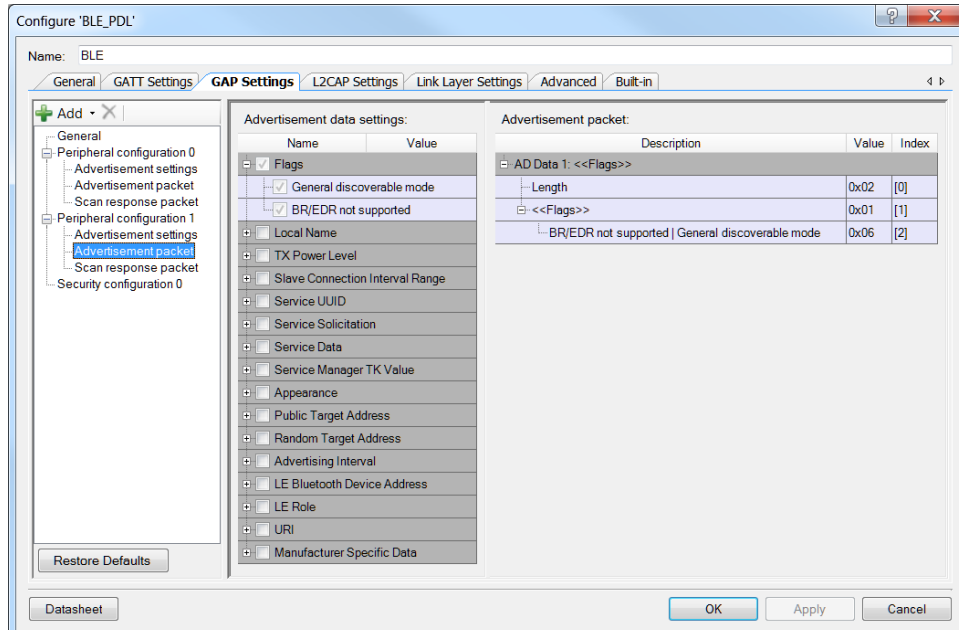
Maximum (ms): 10240

Timeout (s): 150

Datasheet

OK Apply Cancel

Figure 10. GAP Settings: Peripheral configuration 1 > Advertisement Settings



Configure 'BLE_PDL'

Name: BLE

General | GATT Settings | **GAP Settings** | L2CAP Settings | Link Layer Settings | Advanced | Built-in

General

- Peripheral configuration 0
 - Advertisement settings
 - Advertisement packet
 - Scan response packet
- Peripheral configuration 1
 - Advertisement settings**
 - Advertisement packet
 - Scan response packet
- Security configuration 0

Restore Defaults

Advertisement data settings:

Name	Value
Flags	
General discoverable mode	
BR/EDR not supported	
Local Name	
TX Power Level	
Slave Connection Interval Range	
Service UUID	
Service Solicitation	
Service Data	
Service Manager TK Value	
Appearance	
Public Target Address	
Random Target Address	
Advertising Interval	
LE Bluetooth Device Address	
LE Role	
URI	
Manufacturer Specific Data	

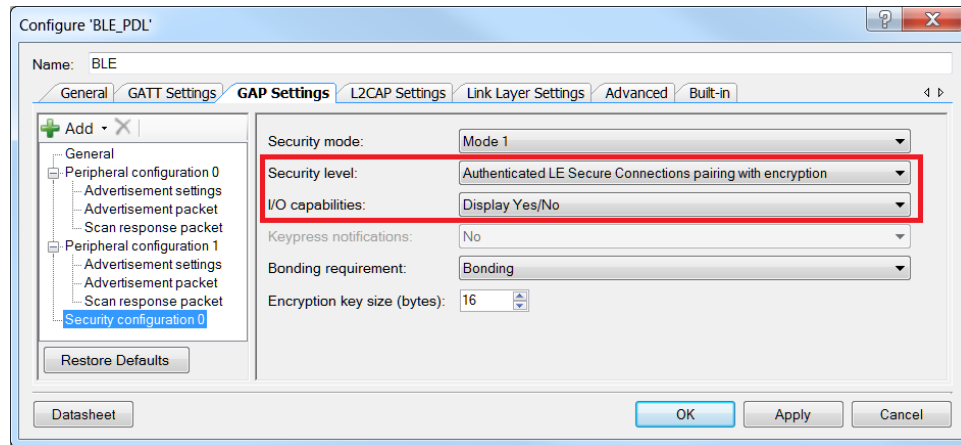
Advertisement packet:

Description	Value	Index
AD Data 1: <<Flags>>		
Length	0x02	[0]
<<Flags>>	0x01	[1]
BR/EDR not supported General discoverable mode	0x06	[2]

Datasheet

OK Apply Cancel

Figure 11. Security Settings



Operation

You can use the CySmart Central Emulation tool on a [Windows PC](#), [Android](#), or [iOS](#) BLE 4.2-compatible device as a Client for connection to this project.

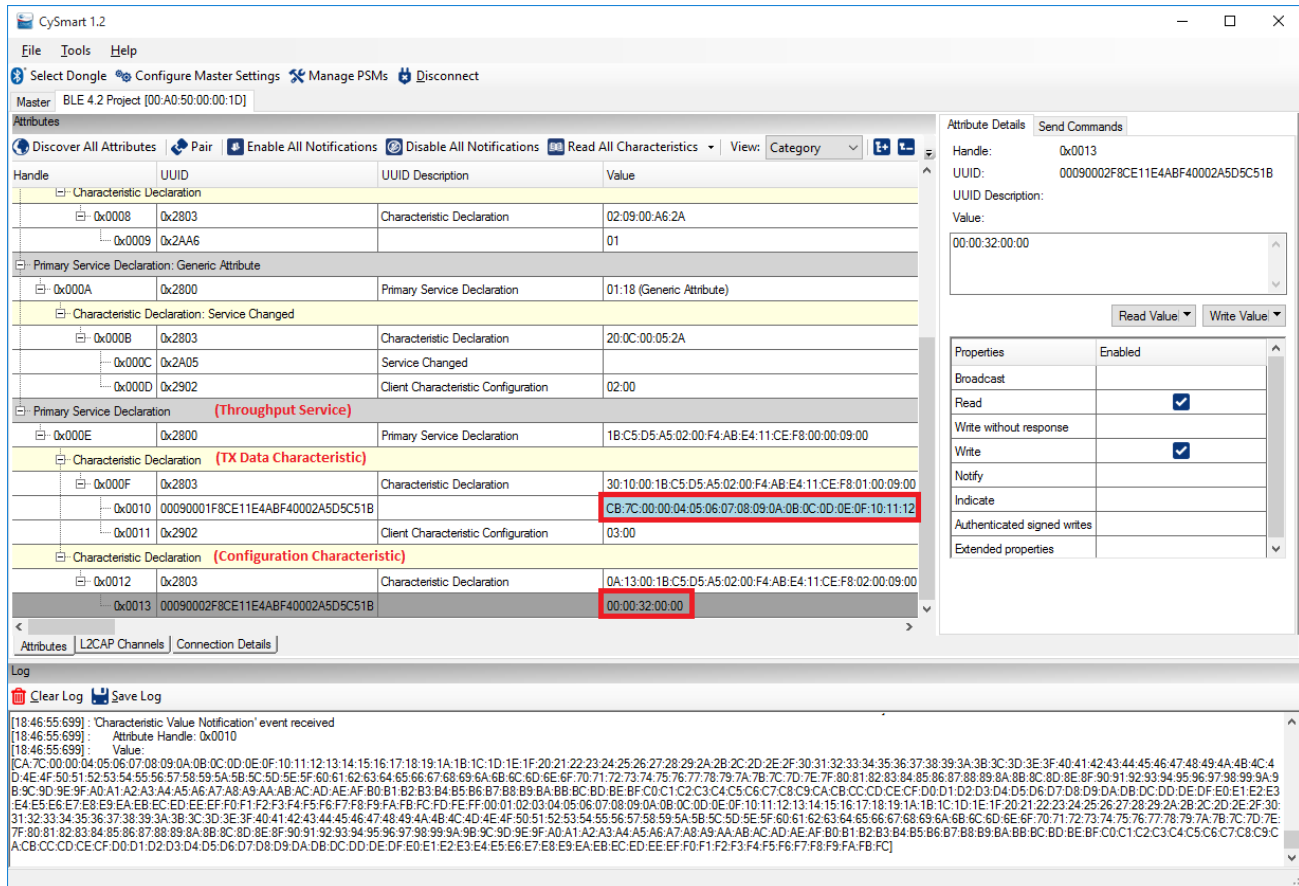
Do the following to use the CySmart Windows application as a Client:

1. Connect the CySmart BLE Dongle to a USB port on the PC.
2. Launch the CySmart Central Emulation Tool and select the connected dongle in the start window.
3. Make sure that the following Master Configuration Settings are set in the CySmart:
 - Connection Parameters/Own Bluetooth Address Type: Public
 - Connection Parameters/Connection Interval Minimum and Maximum: 7.5 ms
 - Security Parameters/Security Level: Secure Connection
 - Security Parameters/Bonding: Bondable
 - Privacy 1.2/Address resolution: Enable
 - Data Length Extension/Suggested maximum Tx octet: 251
 - Other/GATT MTU size: 512
4. Connect the BLE Pioneer board to a USB port on the PC, open Device Manager, and note the COM port number for the KitProg USB-UART device in the **Ports (COM & LPT)** branch of the tree.
5. Build and program the BLE 4.2 Data Length Security Privacy project into the [CY8CKIT-062 PSoC 6 BLE Pioneer Kit](#).
6. Run a serial port communication program (Bray's Terminal, PuTTY, etc.) and make a new connection to the COM port that you noted in Step 4.
7. Reset the development kit to start advertising by pressing the **SW1** button. Observe in the terminal log that the device advertises with a public address:

Advertising with a public address: 00 a0 50 00 00 1d
8. Click the **Start Scan** button to discover available devices.
9. Select **BLE 4.2 Project** in the list of available devices and connect to it.
10. Respond **Yes** to a pairing request received from the peer device.
11. Compare the displayed passkeys on both devices. Click **Yes** on CySmart and press 'y' on the terminal (or press the **SW2** button) to confirm the Numeric comparison pairing procedure.

12. Respond **Yes** to add the device to the resolving list request from CySmart or add the device manually as described in Step 18.
13. Click **Discover All Attributes**, and then click **Enable All Notifications**. Observe the received characteristic values.

Figure 12. CySmart Central Emulation Tool (Windows)



14. After each 10 seconds, observe the calculated throughput values displayed on the terminal. Note that the BLE Dongle has the 115200-bps baud rate limitation; therefore the BLE throughput will be limited by this value.
15. Click **Disable All Notifications** to stop data transfer.
16. Select the **Configuration** Characteristic and click **Read Value** to observe the default configuration, write '0x01' (limited mode) to the first byte, and click **Write Value**. **Enable All Notifications** and observe that the throughput is calculated after the data amount configured in the Data Length (Table 5) field is sent. Press **SW2** to initiate transfer.
17. Click **Disconnect** and observe on the terminal log that the device advertises with a private address.
Advertising with a new private address
18. Configure CySmart for LL Privacy in Network mode because the project automatically uses RPA and does not advertise the device's name after bonding:
 - Select the device with the Public Device Address 00A050-00001D in the Device list window, click **Add > To Resolving List > Selected Device**. In the pop-up window, confirm the device details by clicking **Add**. You may skip this step if you had responded **Yes** in Step 12.
 - Connection Parameters/Own Bluetooth Address Type: RPA or Public.
19. Click **Start Scan** to discover available devices. CySmart automatically resolves a private address using the resolving list and shows an equivalent public address.
20. Select **Peer Device** with the Public Identity Address: 00A050-00001D in the list of available devices and connect to it.

21. Respond **Yes** to a pairing request received from the peer device and observe that pairing with a peer device completed successfully.
22. Select the **Configuration** characteristic and click **Read Value** to observe the default configuration, write '0x02' (Low duty-cycle directed advertising mode) to the first byte and click **Write Value**.
23. Click **Disconnect** and observe on the terminal log that the device advertises with the low duty-cycle directed advertising mode with a private address. For example:

Directed Advertising to: 00 a0 50 7e c2 0e Advertising with a new private address

24. Click **Start Scan** to discover available devices.
25. Select **Peer Device** with the Public Identity Address: 00A050-00001D in the list of available devices and connect to it.
26. Respond **Yes** to a pairing request received from the peer device and observe that pairing with a peer device completed successfully.

For more information about the CySmart Central Emulation tool, refer to [CySmart User Guide](#).

The CySmart mobile app does not have Throughput Service implementation, but it still can be used in the GATT DB mode.

1. Launch the CySmart mobile app ([Android/iOS](#)), and swipe down to refresh the list of the discovered BLE devices.
2. Connect to the **BLE 4.2 Project** device and open Unknown Service in GATT DB.
3. Enable Notification for the first unknown characteristic (TX Data Characteristic) and notice the calculated throughput values displayed on the terminal.
4. Before verifying BLE 4.2 features, make sure that your gadget supports this feature on the official qualifications web site: <https://www.bluetooth.org/tpg/listings.cfm>.

Note: The following happens if the end product does not support BLE 4.2 features:

- **LE Secure Connection:** The reject automatically falls back to the legacy authenticated mode.
- **LL Privacy:** You should manually clear the bonding information on both sides after each connection.
- **DLE:** The data length is automatically negotiated to the minimum payload value.

Related Documents

Application Notes		
AN210781	Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes the PSoC 6 MCU with BLE Connectivity, and how to build a basic code example.
AN215656	PSoC 6 MCU Dual-Core CPU System Design	Presents the theory and design considerations related to this code example.
Software and Drivers		
CySmart – BLE Test and Debug Tool		CySmart is a BLE host emulation tool for Windows PCs. The tool provides an easy-to-use GUI to enable the user to test and debug their BLE Peripheral applications.
PSoC Creator Component Datasheets		
Bluetooth Low Energy (BLE_PDL) Component		The Bluetooth Low Energy (BLE_PDL) Component provides a comprehensive GUI-based configuration window to facilitate designing applications requiring BLE connectivity.
Device Documentation		
PSoC 6 MCU: PSoC 63 with BLE Datasheet Programmable System-on-Chip (PSoC®)		PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual (TRM)
Development Kit (DVK) Documentation		
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit		

Document History

Document Title: CE212742 - BLE 4.2 Data Length Security Privacy with PSoC 6 MCU with BLE Connectivity

Document Number: 002-12742

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5855492	NPAL	11/21/2017	Initial release

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](#)

Cypress Developer Community

[Forums](#) | [WICED IOT Forums](#) | [Projects](#) | [Videos](#) | [Blogs](#) | [Training](#) | [Components](#)

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.



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

© Cypress Semiconductor Corporation, 2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.