CE224734 - Bluetooth Low Energy (BLE) Object Transfer Client / Server

Objective

This BLE example project demonstrates how to create an object transfer and object managing system using Cypress BLE Component APIs and application layer callback of Object Transfer Profile (OTP) on the CY8CKIT-042-BLE PSoC® 4 Pioneer Kit.

Overview

This code example demonstrates the Object Transfer operation of the PSoC® Creator™ Bluetooth Low Energy (BLE) Component. It consists of two projects:

- Object Transfer Client
- Object Transfer Server

The Object Transfer Client project uses the BLE Object Transfer profile to manage and control the objects in the Server device. These projects support bulk data transfers which occur via a separate L2CAP connection-oriented channel. The Client is enabled to create and delete objects and to execute an action using the currently selected object. The selected object can be written, updated, or read via an Object Transfer Channel opened by the Client. This service provides a general method for a Client to select and initiate the transfer of any type of object.

This example supports all the GATT sub-procedures defined in the Object Transfer Service and Object Transfer Profile specifications. The device remains in Sleep mode between BLE connection intervals.

Requirements

Tool: PSoC Creator 4.2

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

Associated Parts: PSoC 4 BLE parts

Related Hardware: two parts of CY8CKIT-042-BLE PSoC 4 Pioneer Kit

Hardware Setup

This example uses the kit's default configuration. See the kit guide to ensure the kit is configured correctly.

Software Setup

Terminal Tool

This example uses a UART terminal window. User must have terminal software, such as Tera Term or PuTTY.

Operation

The Object Transfer Server can be connected to any BLE (Bluetooth 4.1 or later) - compatible device configured as GAP Central role and GATT Client which supports the Object Transfer Profile.

Advertising packets received during the scanning procedure from Peripheral devices are parsed and filtered. Only packets with Object Transfer service-specific data are handled and showed in the debug terminal. The Object Transfer Client device automatically connects to the Server devices. When a connection is established, the Client device automatically discovers and pairs with Server device. To disable automatic operations user should set value of `AUTOCONNECT` parameter to (0u) in the *common.h* file in the Object Transfer Client project.

The Object Transfer Client device can perform next operation on the Server: create a new object, delete an active object, write the active object, read the active object, rename the active object, or change properties of the active object. Client device can select then active object on the Server using next commands: First object, Last object, Next object, Previous object. Also, the Client device can read the number of objects form the Server Device and Objects list. All these operations can be executed by entering the appropriate command in the Terminal application. Table 1 contains all possible commands for the Client project and Table 2 lists commands for the Server project.

The green LED in the Object Transfer Server project blinks while the device is advertising. The green LED in the Object Transfer Client project blinks while the device is scanning. The blue LED is turned ON when a connection is established. The red LED is turned ON after disconnection.

Table 1. List of Commands of Object Transfer Client Project

OTS-Specified Commands	
l - Objects list	g - Number of objects
Selecting object:	Object Operations:
f – First	c - Create
t – Last	d - Delete
n – Next	w - Write
p – Previous	r - Read
Editing Active Object:	
y – Name	o - Properties
General Commands	
h - Commands list	z - Scan
x - Select peer device	i - Initiate connection
s - Start discovery	a - Authentication request
q – Disconnect	u - Unbond device

The commands listed in Table 2 are the procedures the user can perform in the Object Transfer Server Project.

Table 2. Commands of Object Transfer Server Project

General Commands	
h - Commands list	d – Disconnection
r - Remove device from white list and bond list	p - Enter passkey
y - Send accept displayed passkey command	n - Send reject displayed passkey command

Operation Steps

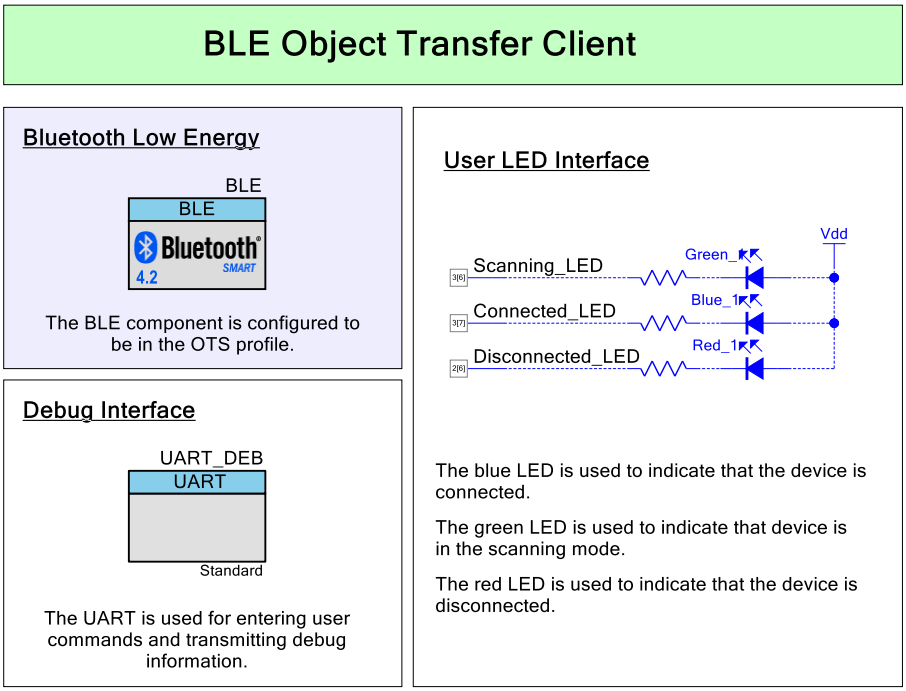
1. Plug the first CY8CKIT-042-BLE kit board into your computer's USB port.
2. Open a terminal window and perform the following configuration: Baud rate – 115200, Parity – None, Stop bits – 1, Flow control – XON/XOFF. These settings must match the configuration of the PSoC Creator UART Component in the project.
3. Build the BLE Object Transfer Client project and program it into the PSoC 4 device. Choose **Debug** > **Program**. For more information on device programming see PSoC Creator Help.
4. Plug the second CY8CKIT-042-BLE kit board into your computer's USB port.
5. Open a terminal window and perform following configuration: 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.
6. Build the BLE Object Transfer Server project and program it into the PSoC 4 device. Choose **Debug** > **Program**. For more information on device programming see PSoC Creator Help.

7. Observe the information about the device address, advertising, scanning, connecting, discovering, pairing, authentication and bonding in the Client Terminal and Server Terminal.
8. Observe the information in the Client Terminal window about the number of objects on the Server and the list of available commands.
9. Enter the [**l**] command in the Client Terminal to display list of object on the Server.
10. Enter the [**h**] command in the Client Terminal to display list of available commands.
11. Enter the [**c**] command in the Client Terminal to create new object on the Server. Enter the name of the object. For example, “Test”. Enter the [**g**] command to display number of objects on the Server. Observe that number of objects is one count more than before this operation. Enter the [**l**] command in the Client Terminal to display list of object on the Server. Observe that new object “Test” was created on the Server and this object is active.
12. Enter the [**w**] command in the Client Terminal to write active object. Enter the [**l**] command in the Client Terminal to display list of object on the Server. Observe change Current Size of object “Test”.
13. Enter the [**p**] command in the Client Terminal to select previous object as an active object. Enter the [**y**] command to change name of active object. Enter new name of active object. For example, “MyFile”. Enter the [**l**] command to display list of object on the Server. Observe that active object has name “MyFile”.
14. Enter the [**p**] command in the Client Terminal to select previous object as an active object. Enter the [**d**] command to delete active object. Enter the [**g**] command to display number of objects on the Server. Observe that number of objects is one count less than before this operation. Enter the [**l**] command to display list of object on the Server. Observe that object “File 4” has been deleted.
15. Enter the [**t**] command in the Client Terminal to select last object as an active object. Enter the [**r**] command to read active object. Observe read information on the Client Terminal.

Design and Implementation

The schematic of BLE Object Transfer Client Code Example is shown in Figure 1.

Figure 1. BLE Object Transfer Client Schematic



The project demonstrates the functionality of the BLE Component configured as Object Transfer Client.

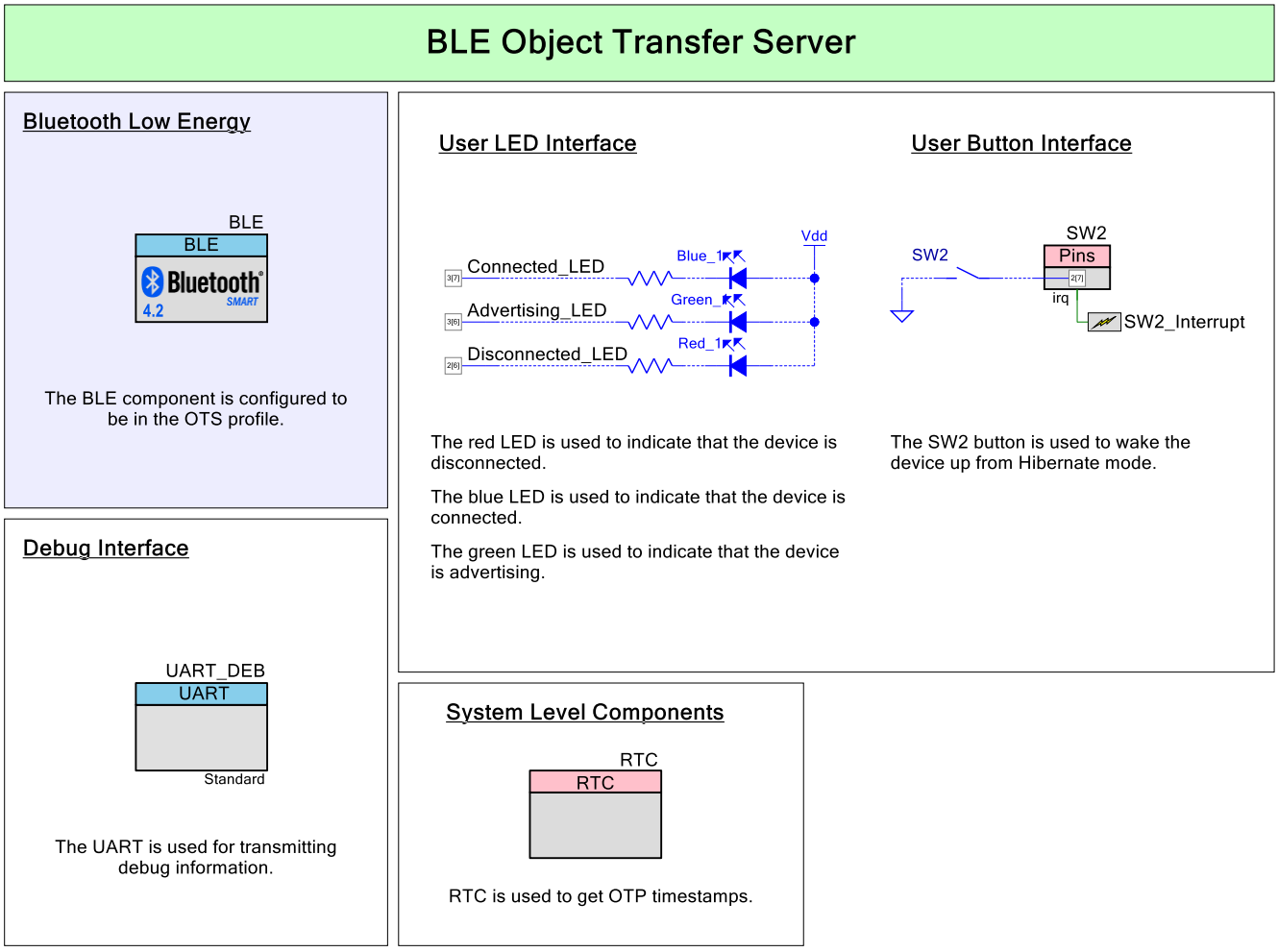
After startup, the device performs the BLE Component initialization. In this project, two callback functions are required for the BLE operation: `AppCallBack()` is required to receive generic events from the BLE Stack and the service-specific callback `OtsCallBack()` is required for Object Transfer Status service-specific events. The `CY_BLE_EVT_STACK_ON` event indicates successful initialization of the BLE Stack.

The device is in Sleep mode when it is connected to the Server and between connection intervals.

This example project uses the UART Component for displaying debug information and entering commands through the terminal emulator app. Freeware, such as HyperTerminal, PuTTY, and so on, is available on the web and can be used with this example.

The schematic of BLE Object Transfer Server Code Example is shown in Figure 2.

Figure 2. BLE Object Transfer Server Schematic



After startup, the device initializes the BLE Component. The Component requires several callback functions to receive events from the BLE Stack. `AppCallBack()` is used to receive general BLE events. Another callback, (`OtsCallBack()`), is used to receive events specific to the service's attribute operations.

The `CYBLE_EVT_STACK_ON` event indicates the successful initialization of the BLE Stack. After this event is received, the Component starts fast advertising with the packet structure, as configured in the BLE Component Customizer (Figure 12).

The device is in Sleep mode when it is connected to the server and between connection intervals.

Pin Assignments

The pin assignments for the Object Transfer Client Project are in Table 3 and for the Object Transfer Server Project are in Table 4.

Table 3. Pin Assignment Object Transfer Client Project

Pin Name	Development Kit	Comment
	CY8CKIT-062	
\UART_DEB:rx\	P1[4]	
\UART_DEB:tx\	P1[5]	
Connected_LED	P3[7]	The blue color of the RGB LED
Disconnected_LED	P2[6]	The red color of the RGB LED
Scanning_LED	P3[6]	The green color of the RGB LED

Table 4. Pin Assignment of Object Transfer Server Project

Pin Name	Development Kit	Comment
	CY8CKIT-062	
\UART_DEB:rx\	P1[4]	
\UART_DEB:tx\	P1[5]	
Connected_LED	P3[7]	The blue color of the RGB LED
Disconnected_LED	P2[6]	The red color of the RGB LED
Advertising_LED	P3[6]	The green color of the RGB LED
SW2	P2[7]	

Components and Settings

Table 5 lists the PSoC Creator Components used in Object Transfer Client example, how they are used in the design, and the non-default settings required for expected functionality.

Table 5. PSoC Creator Components used in Object Transfer Client Example

Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy (BLE)	BLE	The BLE Component is configured to demonstrate operation of the Object Transfer Status Client.	See the Parameter Settings for Object Transfer Client Project section
Digital Output Pins	Disconnected_LED Connected_LED Scanning_LED	These GPIOs are configured as firmware-controlled digital output pins that control LEDs.	**[General tab]** Uncheck HW connection Drive mode: Strong Drive
UART (SCB)	UART_DEBUG	This Component is used to print messages on a terminal program.	Default

Table 6 lists the PSoC Creator Components used in Object Transfer Server example, how they are used in the design, and the non-default settings required for expected functionality.

Table 6. PSoC Creator Components used in Object Transfer Server Example

Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy (BLE)	BLE	The BLE Component is configured to demonstrate operation of the Object Transfer Status Server device.	See the Parameter Settings for Object Transfer Server Project section
Digital Input Pin	SW2	This pin is used to generate interrupts when the user button (**SW2**) is pressed.	**[General tab]** Uncheck HW connection Drive mode: Resistive Pull Up
Digital Output Pins	Disconnect_LED Advertising_LED Connected_LED	These GPIOs are configured as firmware-controlled digital output pins that control LEDs.	**[General tab]** Uncheck HW connection Drive mode: Strong Drive
SysInt	SW2_Interrupt	This Component is configured to extract interrupts from SW2.	**[Basic tab]** DeepSleepCapable = true
UART (SCB)	UART_DEBUG	This Component is used to print messages on a terminal program.	Default
RTC	RTC	This Component is used to count date and time.	Default

For information on the hardware resources used by a Component, see the Component datasheet.

Parameter Settings for Object Transfer Client Project

BLE Component

The BLE Component is configured as the OTS Client in the GAP Central role with the settings shown in Figure 3

Figure 3. General Settings

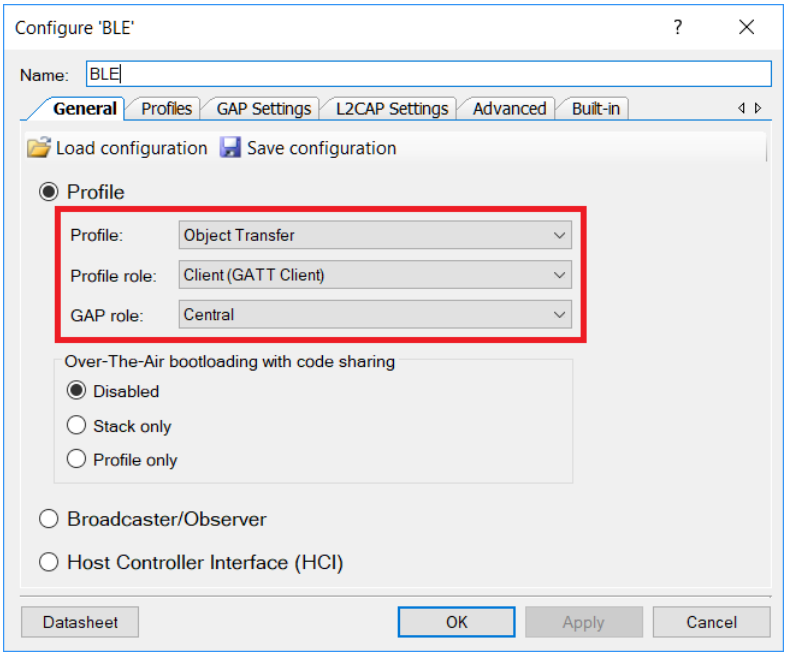


Figure 4. Profiles Settings

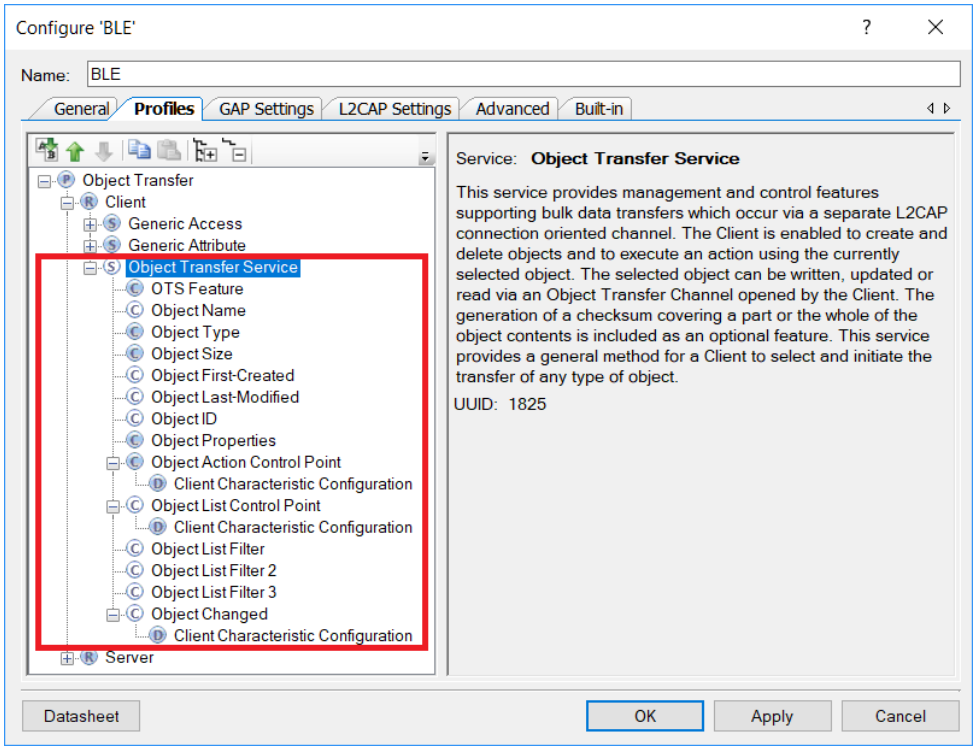


Figure 5. GAP Settings

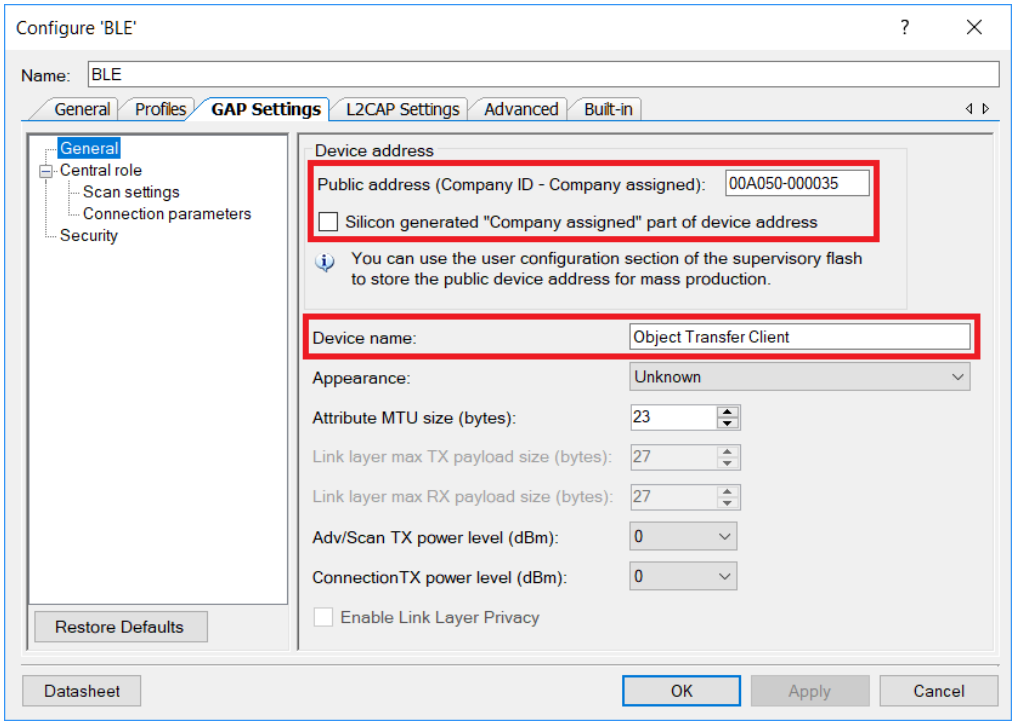


Figure 6. Security Settings

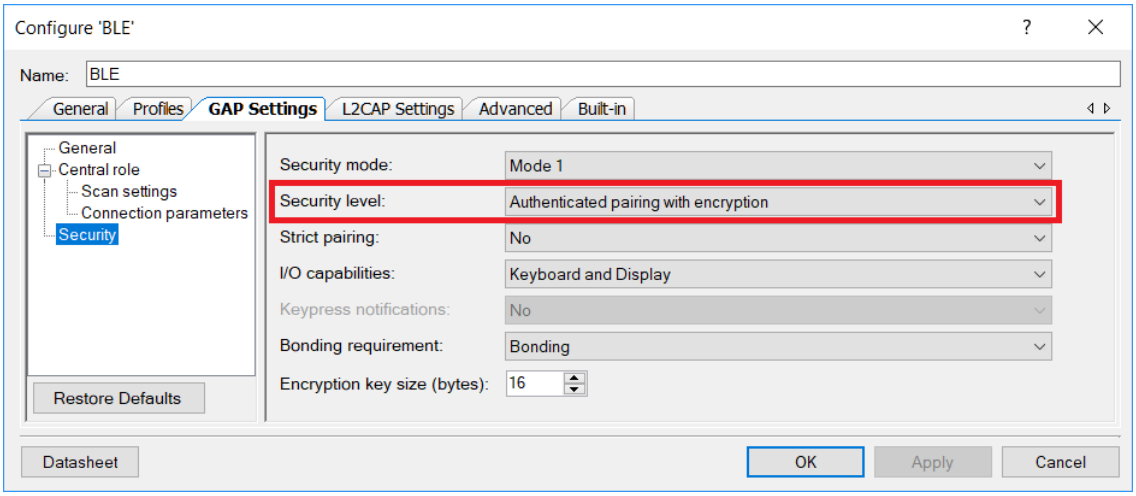
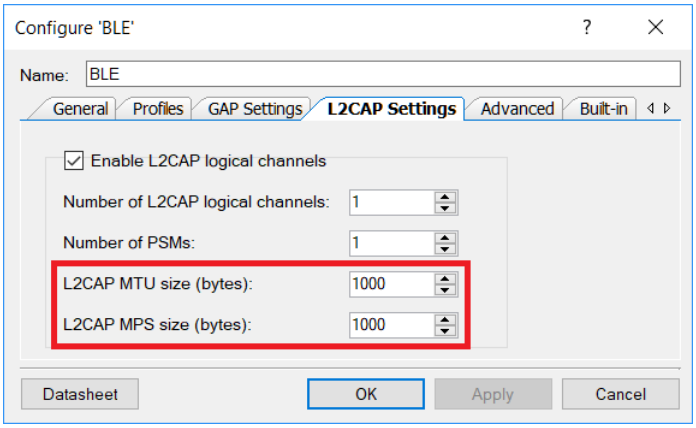


Figure 7. L2CAPSettings



Parameter Settings for Object Transfer Server Project

BLE Component

The BLE Component is configured as the Object Transfer Server in the GAP Peripheral role with the settings shown Figure 8 to Figure 14.

Figure 8. General Settings

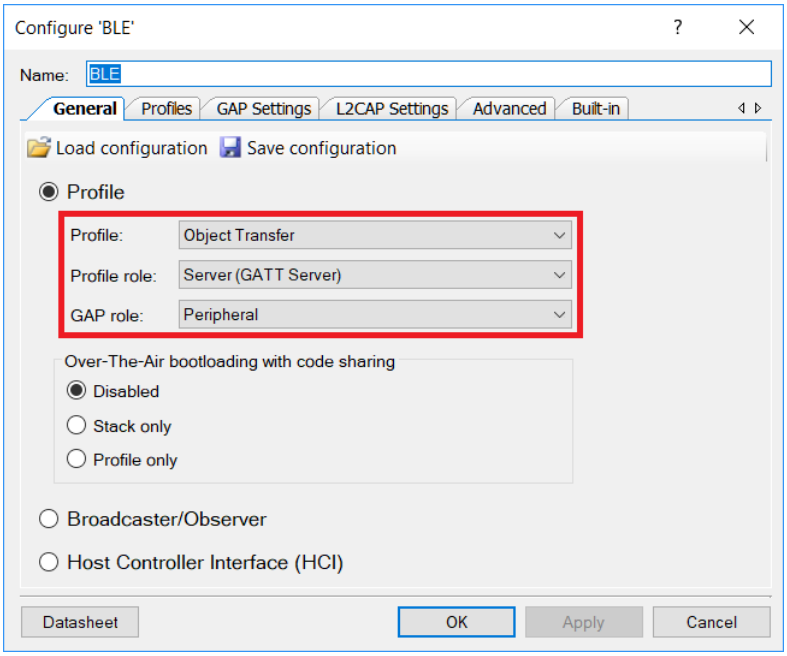


Figure 9. Profiles Settings

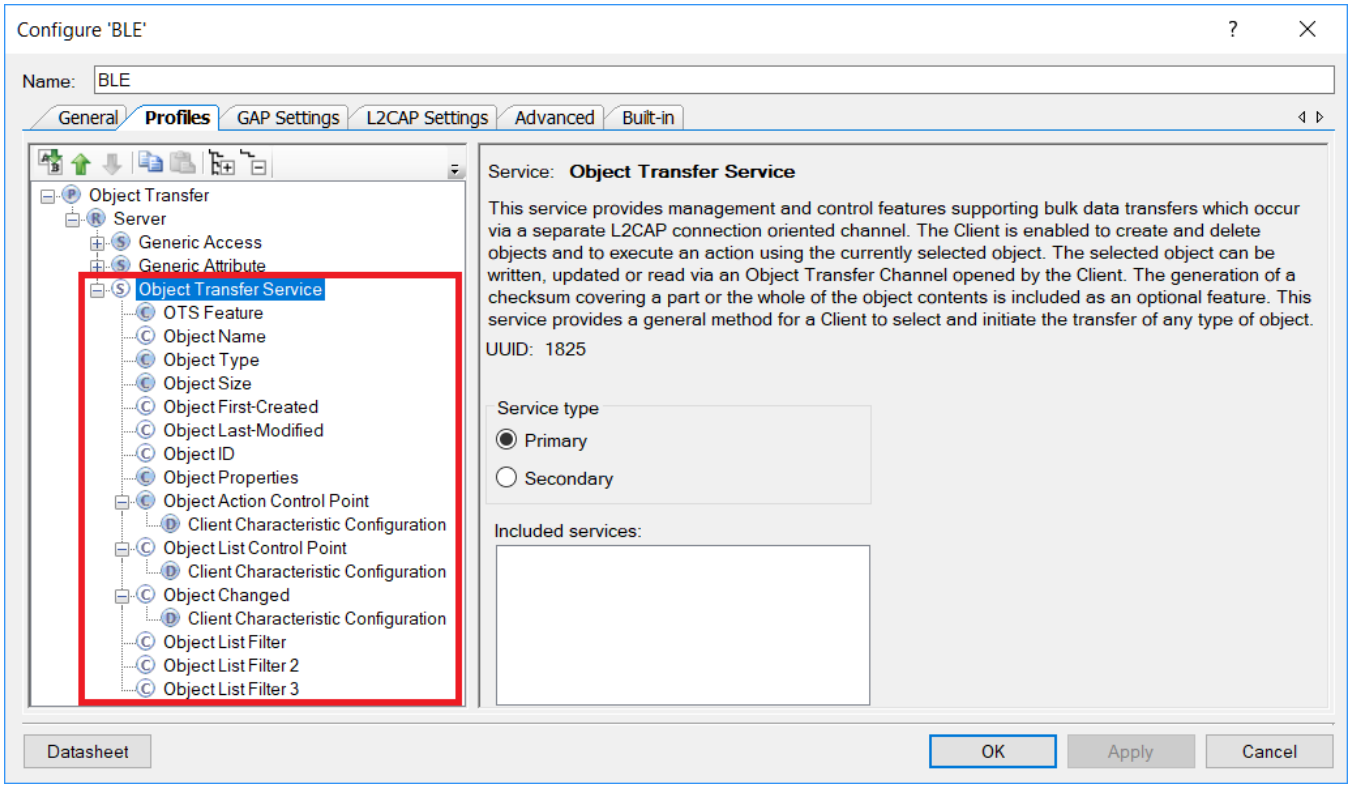


Figure 10. GAP Settings

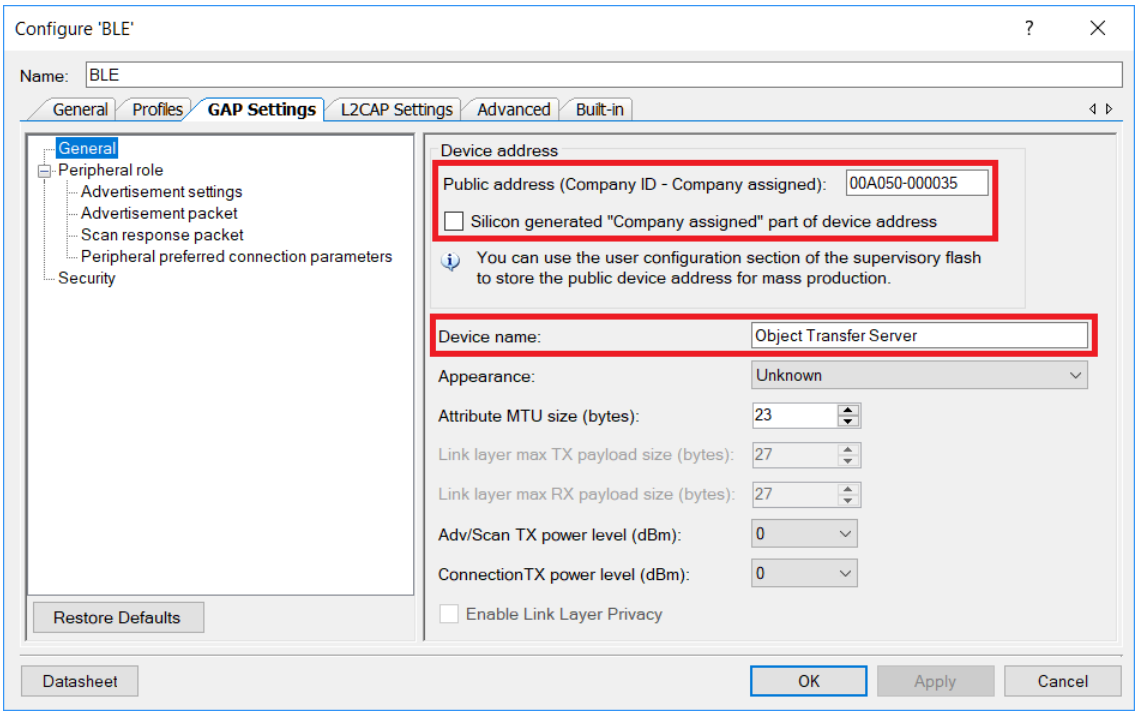


Figure 11. GAP Settings: Advertisement Settings

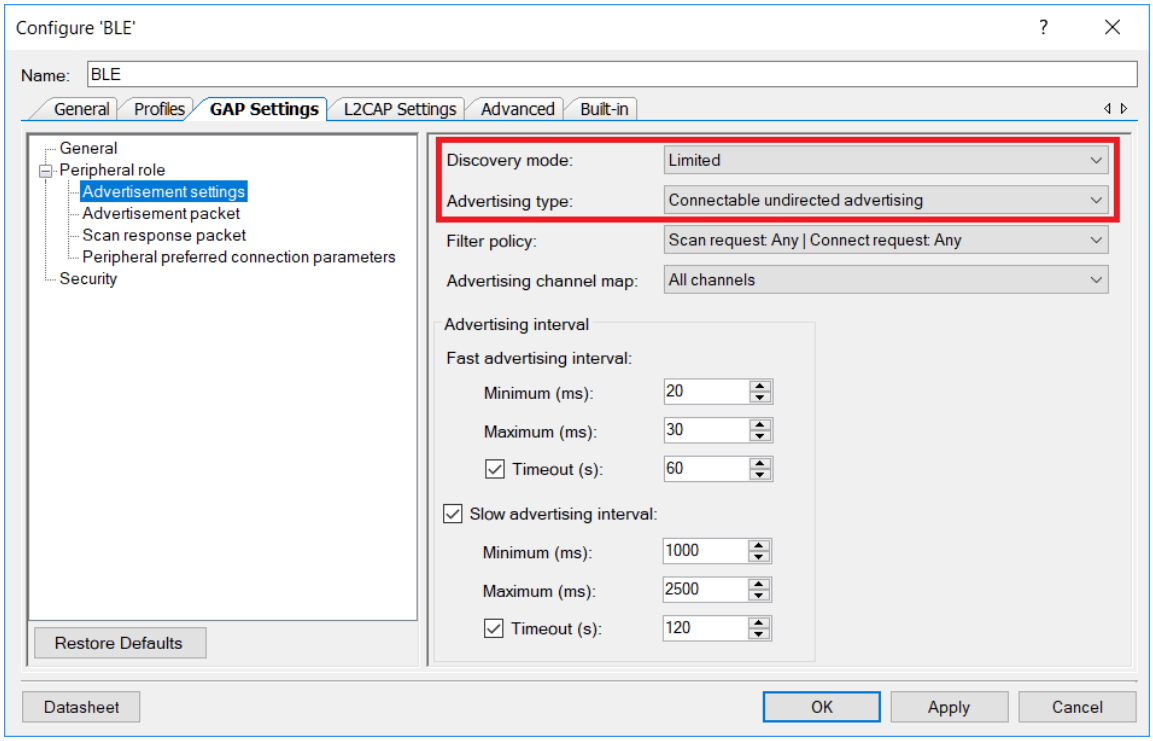


Figure 12. GAP Settings: Advertisement Packet

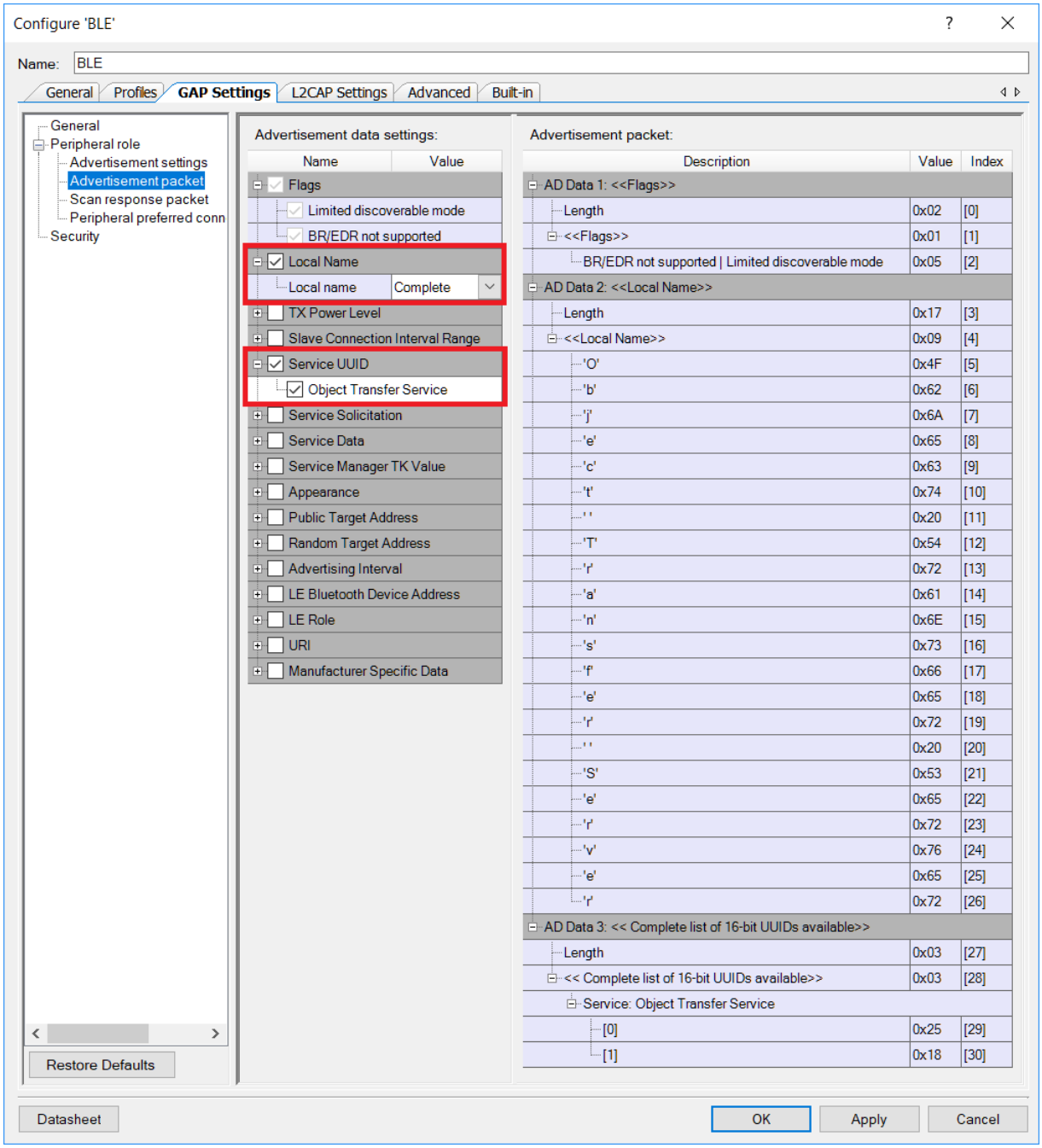


Figure 13. Security Settings

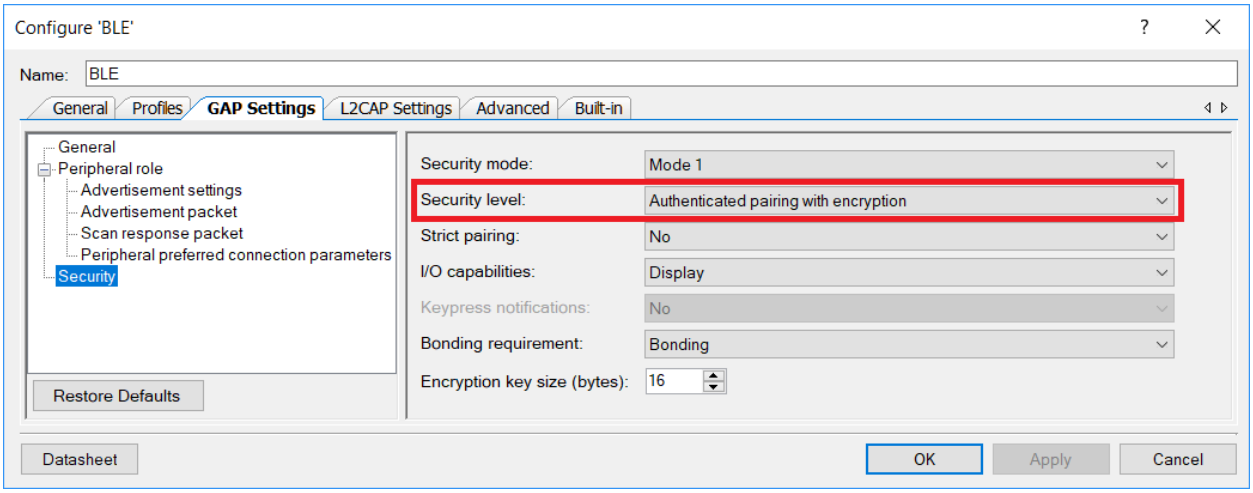
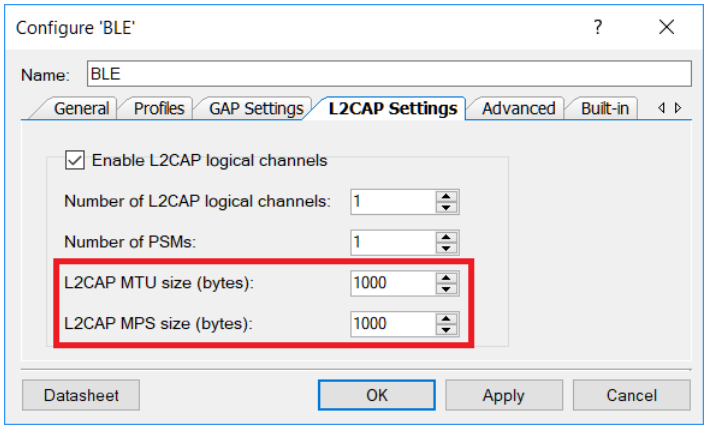


Figure 14. L2CAP Settings



Reusing This Example

This example is designed for the CY8CKIT-042-BLE pioneer kit. To port the design to a different PSoC 4 device, kit, or both, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed.

Related Documents

Application Notes		
AN91267	Getting Started with PSoC™ 4 BLE	Introduces to PSoC BLE, an Arm® Cortex®-M0 based programmable radio-on-chip with Bluetooth Low Energy.
AN91184	PSoC 4 BLE - Designing BLE Applications	Shows how to design the Bluetooth Low Energy (BLE) application based on PSoC 4 BLE, using standard profiles defined by the Bluetooth SIG included in the BLE Component in PSoC Creator. Demonstrates how to build an application with the BLE Health Thermometer Profile on the CY8CKIT-042-BLE kit.
Videos		
PSoC 4 BLE 101: Intro to Bluetooth Low Energy		This is the first installment of a series of getting-started videos on Cypress Bluetooth Low Energy solutions.
PSoC 4 BLE 101: 1 Configuring a Find Me Profile with BLE		Using Cypress Pioneer kit with a PSoC 4 Radio module. Alan Hawse walks you through a simple example for a find-me tag application.
PSoC 4 BLE 101: 2 Finishing the Find Me Application with Firmware		In this lesson, we take the Find Me profile you configured in the previous video and add the firmware required to make it work on the PSoC 4 BLE device.
PSoC 4 BLE 101: 3 Adding Battery Level Service and Testing with CySmart		This lesson takes the Find Me profile built in the first two lessons and adds a Battery Level service.
PSoC 4 BLE 101: 5 Extending Battery Life with PSoC Low Energy Modes		Adds power savings into your BLE designs easily using PSoC and PSoC Creator. In the last lesson, we created Find Me peripheral with the Battery Level service.
Software and Drivers		
CySmart – Bluetooth® LE Test and Debug Tool		CySmart is a Bluetooth® LE host emulation tool for Windows PCs. The tool provides an easy-to-use Graphical User Interface (GUI) to enable customers to test their Bluetooth LE peripheral applications.
PSoC Creator Component Datasheets		
Bluetooth Low Energy (BLE) Component		The Bluetooth Low Energy (BLE) Component provides a comprehensive GUI-based configuration window to facilitate designing applications requiring BLE connectivity.
PSoC 4 Serial Communication Block (SCB) Component		Supports a PSoC 4 multifunction hardware block that implements I^2C, SPI, UART, and EZI2C communications
Device Documentation		
PSoC® 4: PSoC 4200 BLE Family Datasheet Programmable System-on-Chip (PSoC®)		
Development Kit (DVK) Documentation		
Bluetooth® Low Energy Pioneer Kit (CY8CKIT-042-BLE)		

Document History

Document Title: CE224734 - Bluetooth Low Energy (BLE) Object Transfer Client / Server

Document Number: 002-24734

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6271061	AZOV	03/22/2019	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 MCU

Cypress Developer Community

Community | 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, 2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries ("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. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress shall have no liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. CYPRESS DOES NOT REPRESENT, WARRANT, OR GUARANTEE THAT CYPRESS PRODUCTS, OR SYSTEMS CREATED USING CYPRESS PRODUCTS, WILL BE FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (collectively, "Security Breach"). Cypress disclaims any liability relating to any Security Breach, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any Security Breach. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. 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. "High-Risk Device" means any device or system whose failure could cause personal injury, death, or property damage. Examples of High-Risk Devices are weapons, nuclear installations, surgical implants, and other medical devices. "Critical Component" means any component of a High-Risk Device whose failure to perform can be reasonably expected to cause, directly or indirectly, the failure of the High-Risk Device, 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 any use of a Cypress product as a Critical Component in a High-Risk Device. You shall indemnify and hold Cypress, its directors, officers, employees, agents, affiliates, distributors, and assigns harmless from and against all claims, costs, damages, and expenses, arising out of any claim, including claims for product liability, personal injury or death, or property damage arising from any use of a Cypress product as a Critical Component in a High-Risk Device. Cypress products are not intended or authorized for use as a Critical Component in any High-Risk Device except to the limited extent that (i) Cypress's published data sheet for the product explicitly states Cypress has qualified the product for use in a specific High-Risk Device, or (ii) Cypress has given you advance written authorization to use the product as a Critical Component in the specific High-Risk Device and you have signed a separate indemnification agreement.

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.