CE215121 - BLE HID Keyboard with PSoC 6 MCU with BLE Connectivity

Objective

This example demonstrates the implementation of the Bluetooth Low Energy (BLE) HID over GATT Profile where the device operates as a HID keyboard.

Overview

The design demonstrates the core functionality of the BLE Component configured as a HID Device (GATT Server). It simulates keyboard press in Boot and Protocol modes. Also, the design demonstrates how to handle a suspend event from the central device and enter Low-Power mode when suspended.

Requirements

Tool: PSoC Creator™ 4.2 or later

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

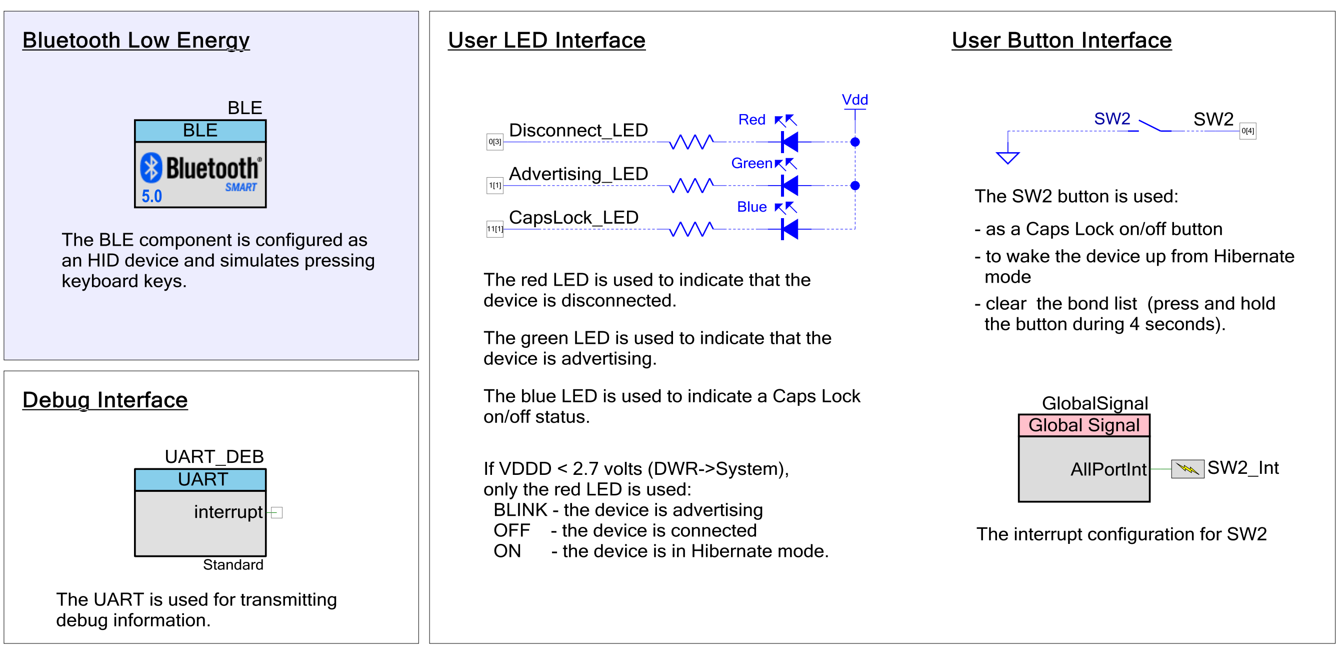
Associated Parts: All PSoC® 6 MCU with BLE Connectivity (PSoC 6 BLE) parts

Related Hardware: CY8CKIT-062 PSoC 6 BLE Pioneer Kit

Design

Figure 1 shows the top design schematic.

Figure 1. BLE HID Keyboard Code-Example Schematic



The BLE Component implements a HID over the GATT Profile in the HID Device role (GATT Server).

After a start, the device performs the BLE Component initialization. The four callback functions are required in this project for the BLE operation:

- `AppCallBack()` is required to receive generic events from the BLE Stack.
- `HidsCallBack()`, `BasCallBack()`, and `ScpsCallBack()` are required to receive events from the services.

The `CY_BLE_GAPP_StartAdvertisement()` function is called after the `CY_BLE_EVT_STACK_ON` event to start advertising with the packet shown in Figure 7. As the BLE Component is configured in the General Discovery mode, it stops advertising after an advertisement period expires. On an advertisement timeout, the system enters Hibernate mode. Press the mechanical button **SW2** on the PSoC 6 BLE Pioneer Kit to wake up the system and start advertising. The BLE subsystem and CPU enter Low-Power Deep Sleep mode between the connection and advertising intervals. The BLE subsystem automatically wakes up to maintain connection and advertise data transfer.

The green LED blinks to indicate that the device is advertising. The red LED turns ON after disconnection to indicate that no client is connected to the device. When a client is connected successfully, the red and blue LEDs turn OFF. The blue LED indicates the Caps Lock state sent from the host through an output keyboard report characteristic.

Additionally, this project implements the Battery Service. By default, the battery level is simulated and changed from 2 to 20 percent.

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 the *common.h* file.

LED Behavior for V_{DDD} Voltage < 2.7 V

If the V_{DDD} voltage is set to less than 2.7 V in the DWR settings of the **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 Hibernate mode, the red LED stays ON.

Switching the CPU Cores Usage

This section describes how to switch between different CPU cores usage (Single core and 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.

Important to remember:

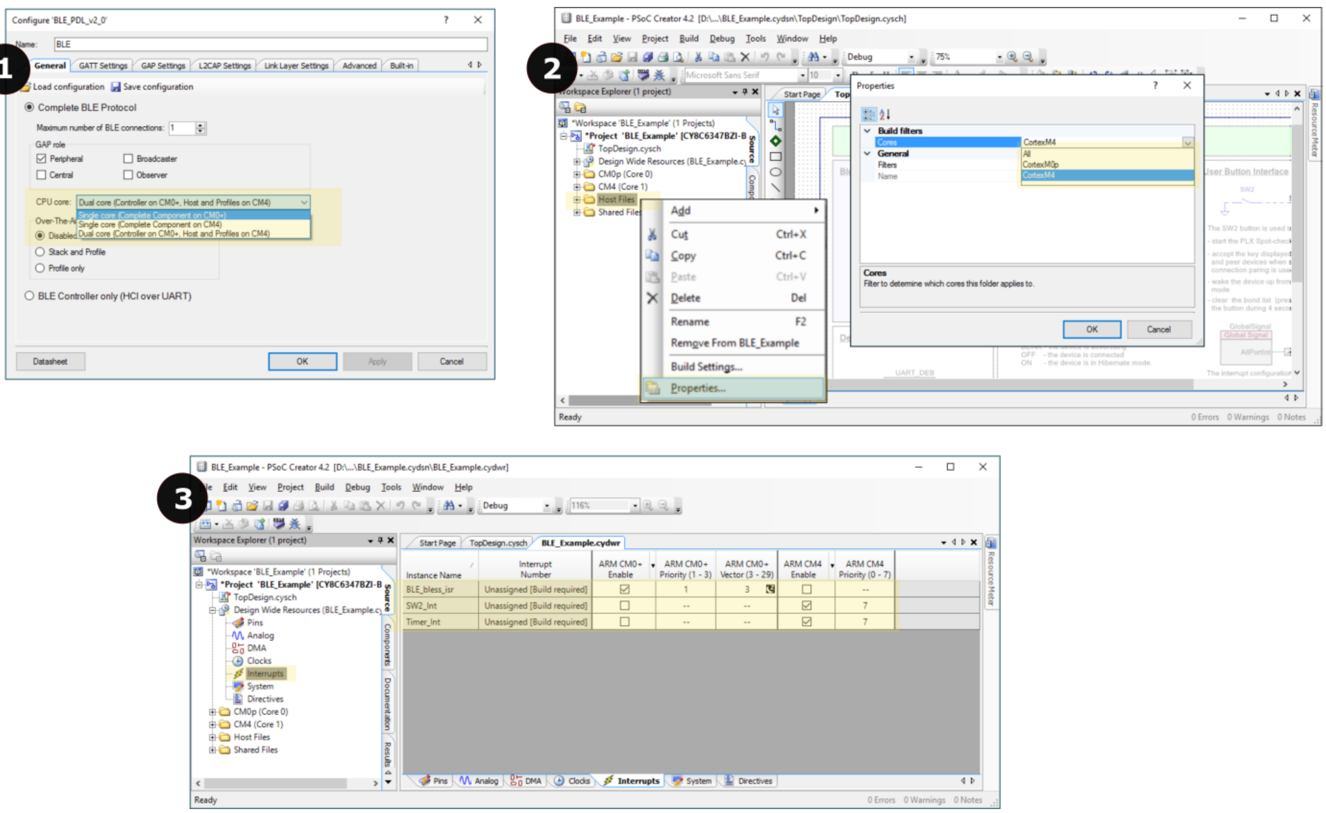
- All application host-files must be run on the host core.
- The BLE Subsystem (BLESS) interrupt must be assigned to the core where the controller runs.
- All additional interrupts (SW2, MCWDT, etc.) used in the example must be assigned to the host core.

Do the following to switch the CPU cores usage:

1. In the BLE Component Customizer **General** tab, select appropriate CPU core option.

2. Change the core properties to CortexM4 or CortexC0p for the project folder Host Files based on the CPU core option selected in step 1. It should be:
   - 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 the 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 2. Steps for Switching the CPU Cores Usage



Hardware Setup

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

Table 1. The pin assignment and connections required on the development board for the supported kits.

Table 1. Pin Assignment

Pin Name	Development Kit CY8CKIT-062	Comment
\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
CapsLock_LED	P11[1]	The blue color of the RGB LED
SW2	P0[4]	

Components

Table 2 lists the PSoC Creator Components used in this example and the hardware resources used by each of the components.

Table 2. PSoC Creator Components List

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

Parameter Settings

Bluetooth Low Energy (BLE)

The BLE Component is configured as a HID over a GATT Profile in the HID device role (GATT Server). The HID Device has one instance of the HID Service, Battery Service, Device Information Service, and Scan Parameters Service.

Figure 3. General Settings

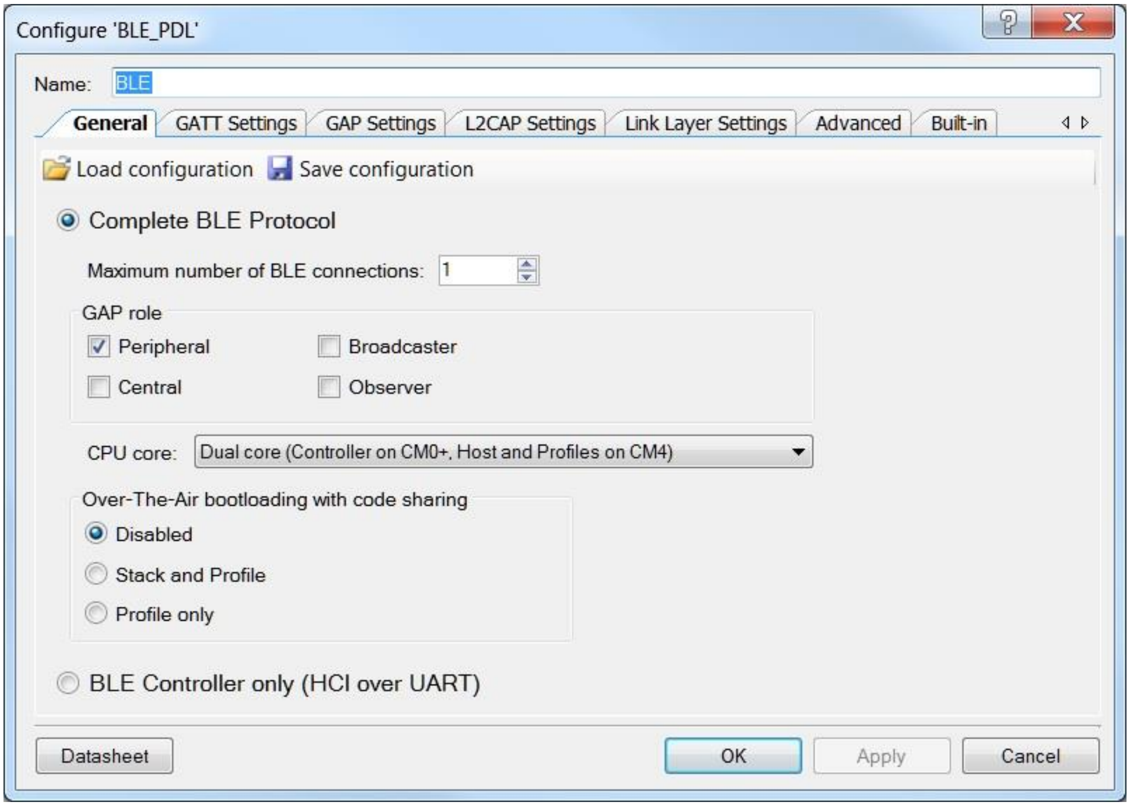


Figure 4. GATT Settings

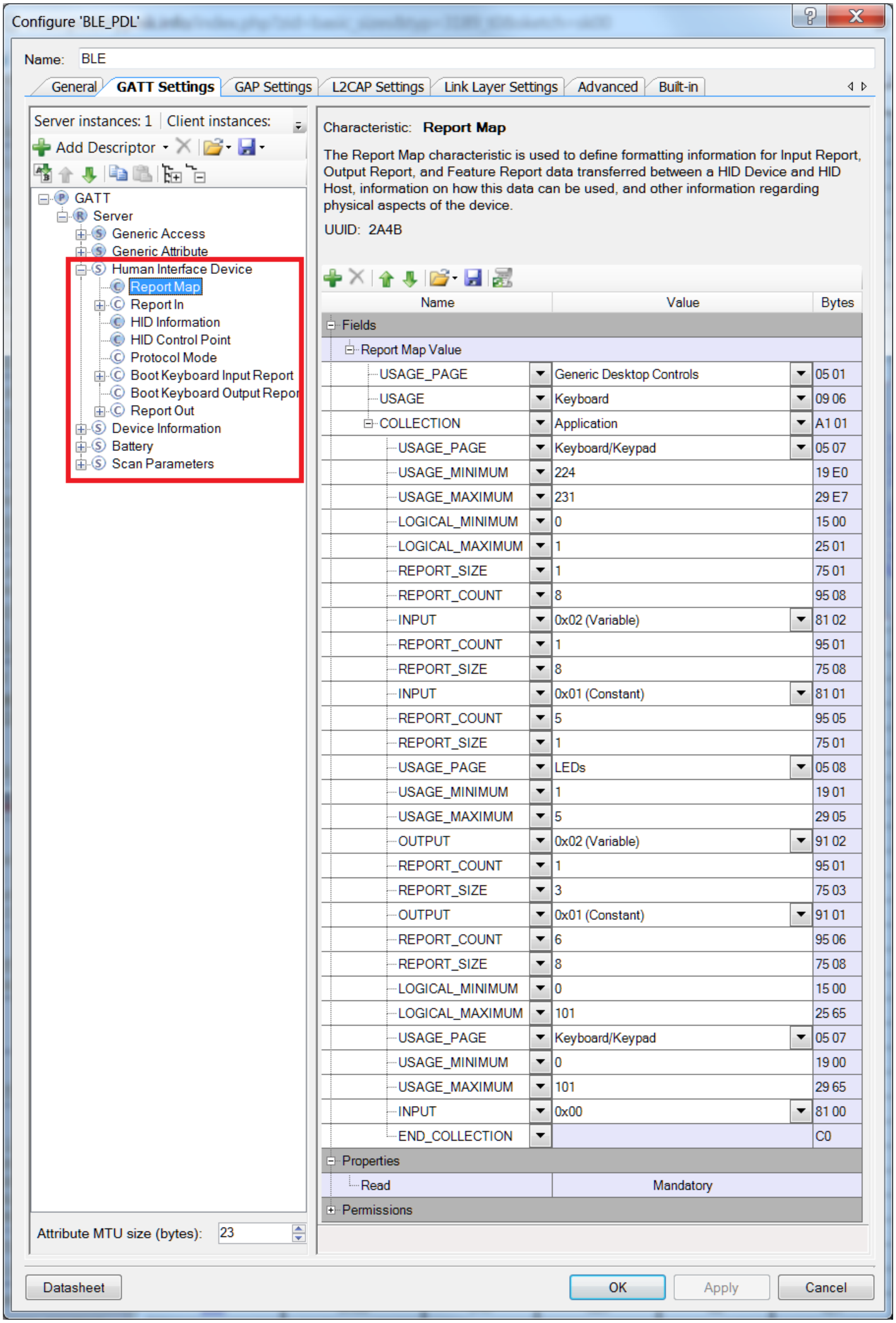


Figure 5. GAP Settings

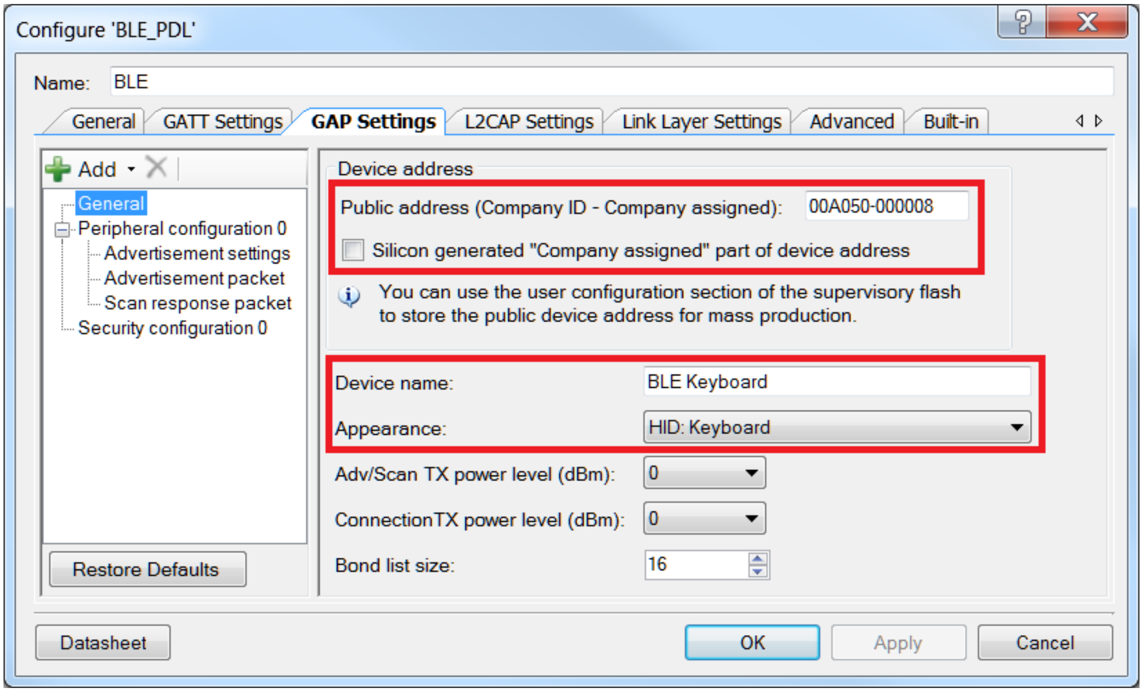


Figure 6. GAP Settings: Advertisement Settings

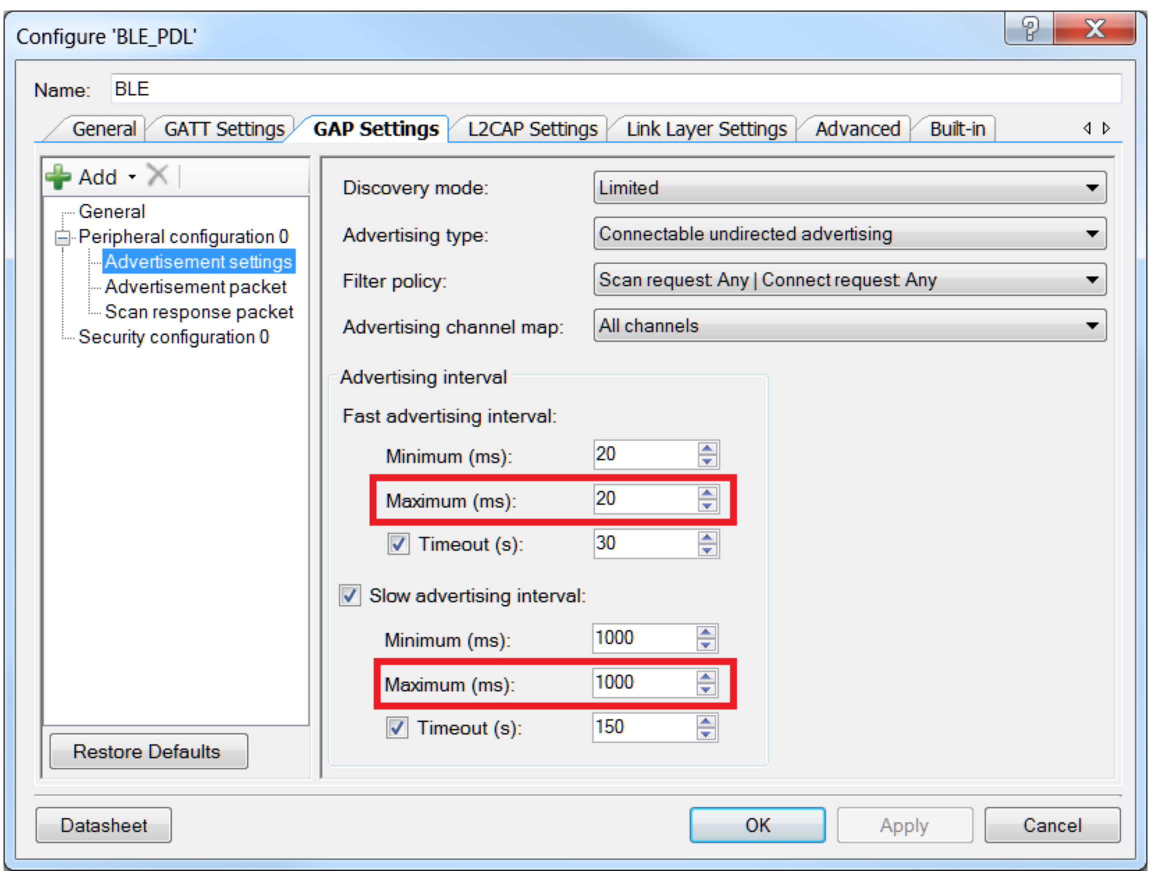


Figure 7. GAP Settings: Advertisement Packet

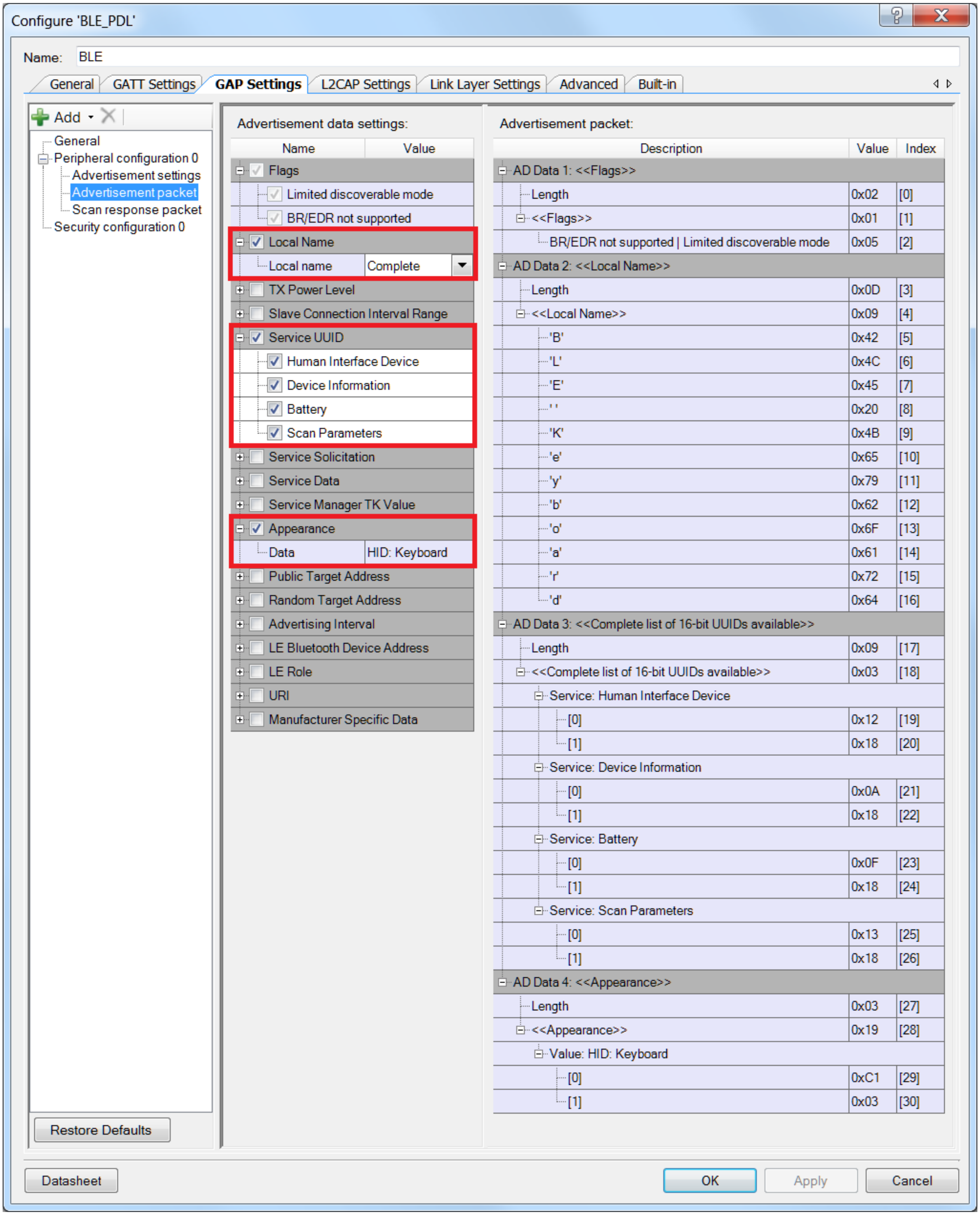
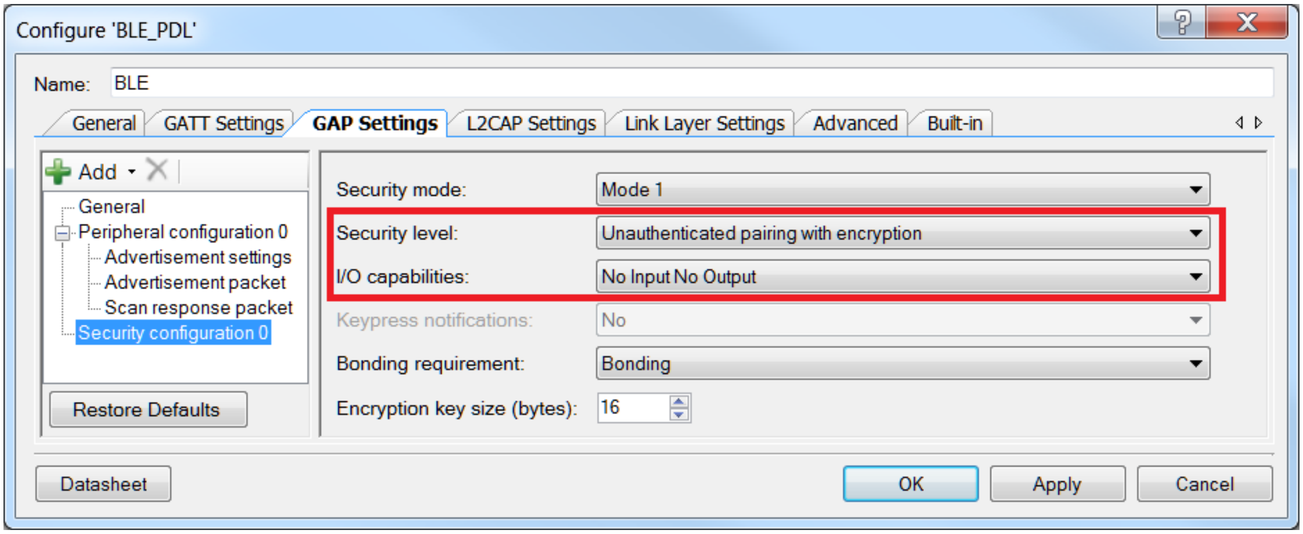


Figure 8. Security Settings

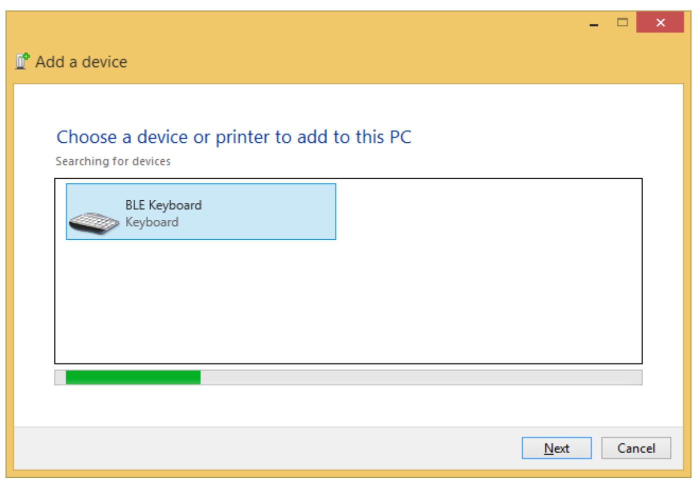


Operation

You can connect the HID Device to Windows 8. Windows 7 and older OS do not have HOGP drivers.

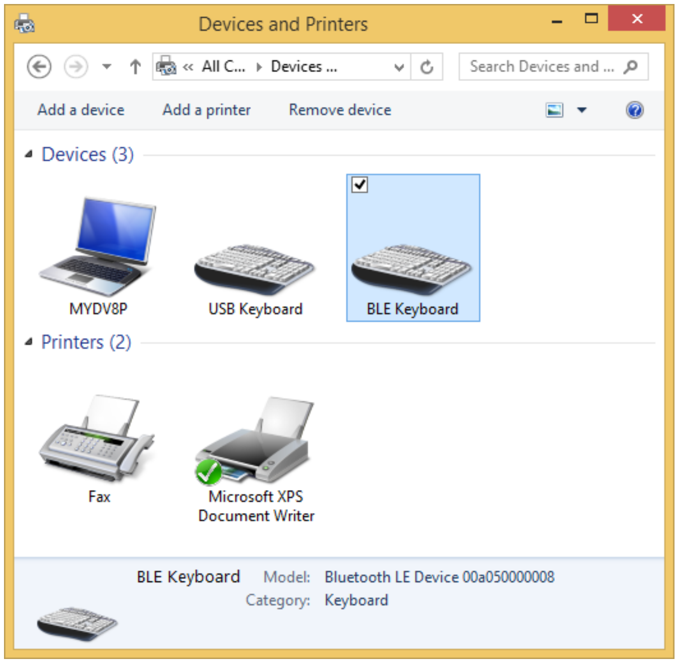
1. Make sure that a PC with Windows 8 has Bluetooth 4.0 installed.
2. To connect to a HID device, click **Add a device** in the **Devices and Printers** window of the Control Panel.
3. Select the **BLE Keyboard** device and click **Next**.

Figure 9. Pairing with Windows 8 PC



The setup will automatically install the necessary files in the system.

Figure 10. BLE Keyboard is Recognized as HID Device



4. Focus the input to an editable field (open text editor, take a note, and so on).
5. Observe that simulated keys "abcdef…" fill the document.
6. When **SW2** is pressed, the Caps Lock LED on the keyboard is turned ON/OFF. The blue LED on the kit indicates the Caps Lock state received from the HID Client.

Note: Earlier versions of Android OS does not send a Caps Lock state back to the device, so the LED will not be turned ON/OFF.

Figure 11. HID Keyboard Emulation on iOS Device

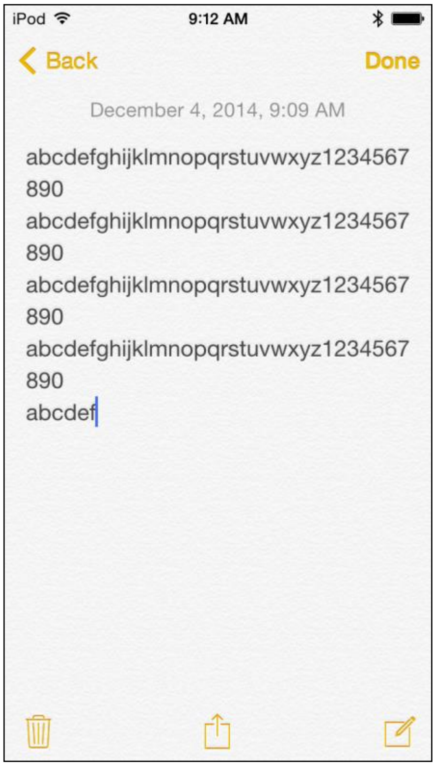


Figure 12. HID Keyboard Emulation on Android Device

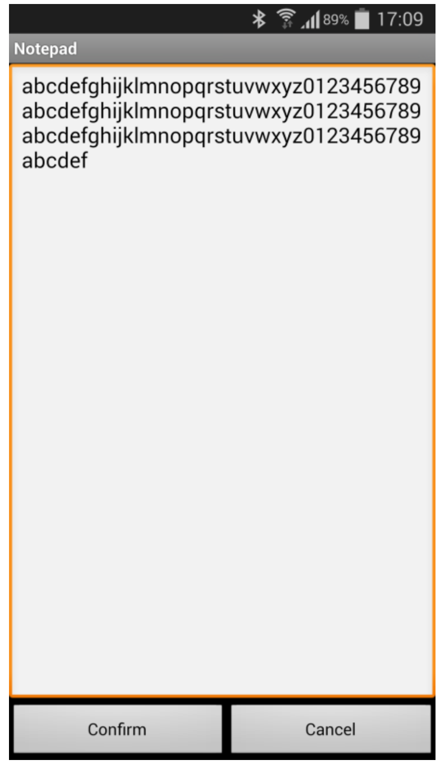
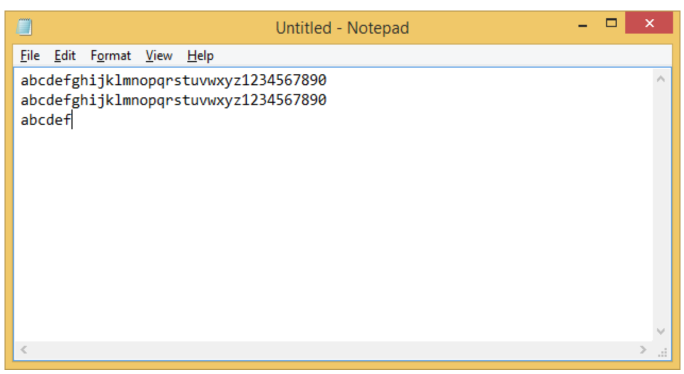


Figure 13. HID Keyboard Emulation on Windows 8 PC



Also, you can connect a HID Device to an Android or iOS device with Bluetooth 4.0 support: go to the phone's Bluetooth settings and pair it with your device (it should be recognized as BLE keyboard).

Figure 14. iOS Bluetooth Pairing

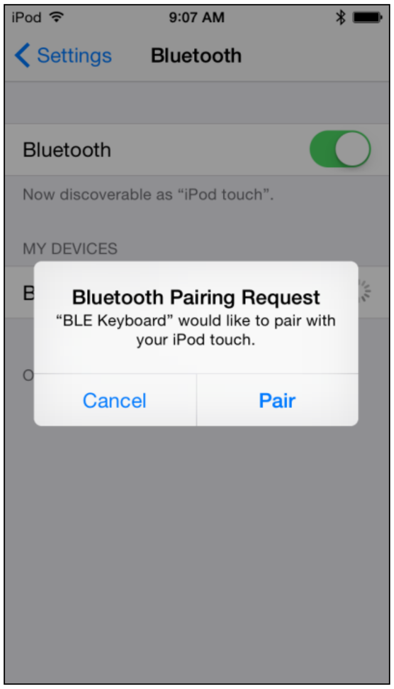
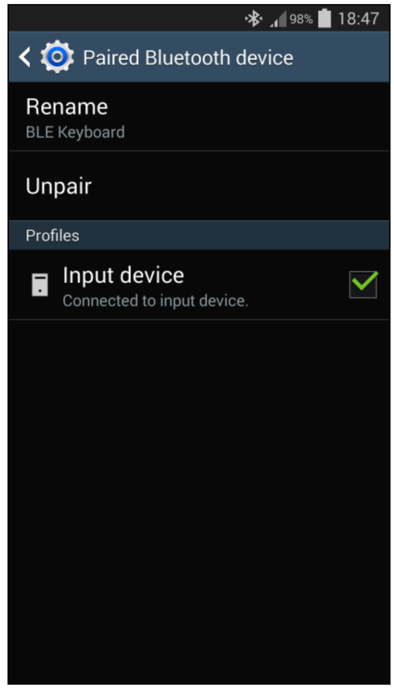


Figure 15. Android Settings for Paired Bluetooth Device



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 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: CE215121 - BLE HID Keyboard with PSoC 6 MCU with BLE Connectivity

Document Number: 002-15121

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5968177	NPAL	11/15/2017	New spec

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

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