

# BLE HID Keyboard

## 1.0

## Features

- BLE HID over GATT Profile in HID Device role operation
- Simulates keyboard pressing
- Simulates battery charging
- DeepSleep mode demonstration
- LED status indication

## General Description

This project demonstrates keyboard pressing in the boot and protocol mode. The example also demonstrates handling a suspend event from the central device and entering the low power mode when suspended.

## Development Kit Configuration

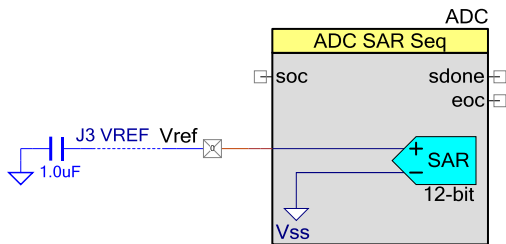
The device has the following configuration:

- A mechanical button (port 2 pin 7) is used to wake up the device, start re-advertising.
- The red LED (port 2 pin 6) is used to indicate a BLE disconnection state.
- The green LED (port 3 pin 6) is used to indicate an advertising state.
- The blue LED (port 3 pin 7) is used to indicate a Caps Lock state.

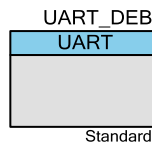
## Project Configuration

The example project consists of the following components: BLE, UART, digital output pin, digital input pin. ADC is not used by default. Battery voltage measurement could be enabled in `bas.h` header file. The UART is used for transmitting debug information. The output pins are used to reflect the line signal output on the LED. The input pin is configured to the resistive pull up mode and used as a Caps Lock on/off button. The top design schematic is shown in **Figure 1**.

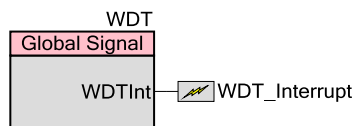
## BLE HID Keyboard Example Project



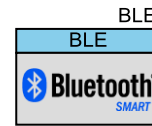
ADC measures battery voltage.



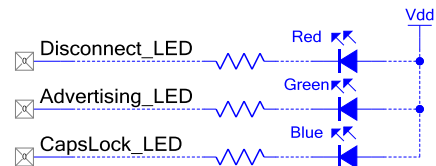
UART is used for transmitting the debug information.



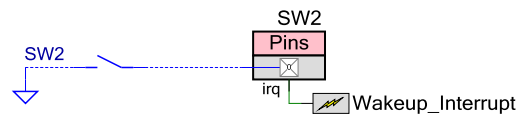
WDT is used as a generic timer for mouse event simulation.



BLE component is configured as HID device and simulates pressing keyboard keys.



The red LED is used to indicate that the device is disconnected. The green LED is used to indicate that the device is advertising. The blue LED is used to indicate a Caps Lock state.



The button is used as a Caps Lock on/off button and to wake the device up from the hibernate mode.

Figure 1. Top design schematic

The BLE component is configured as HID over 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.

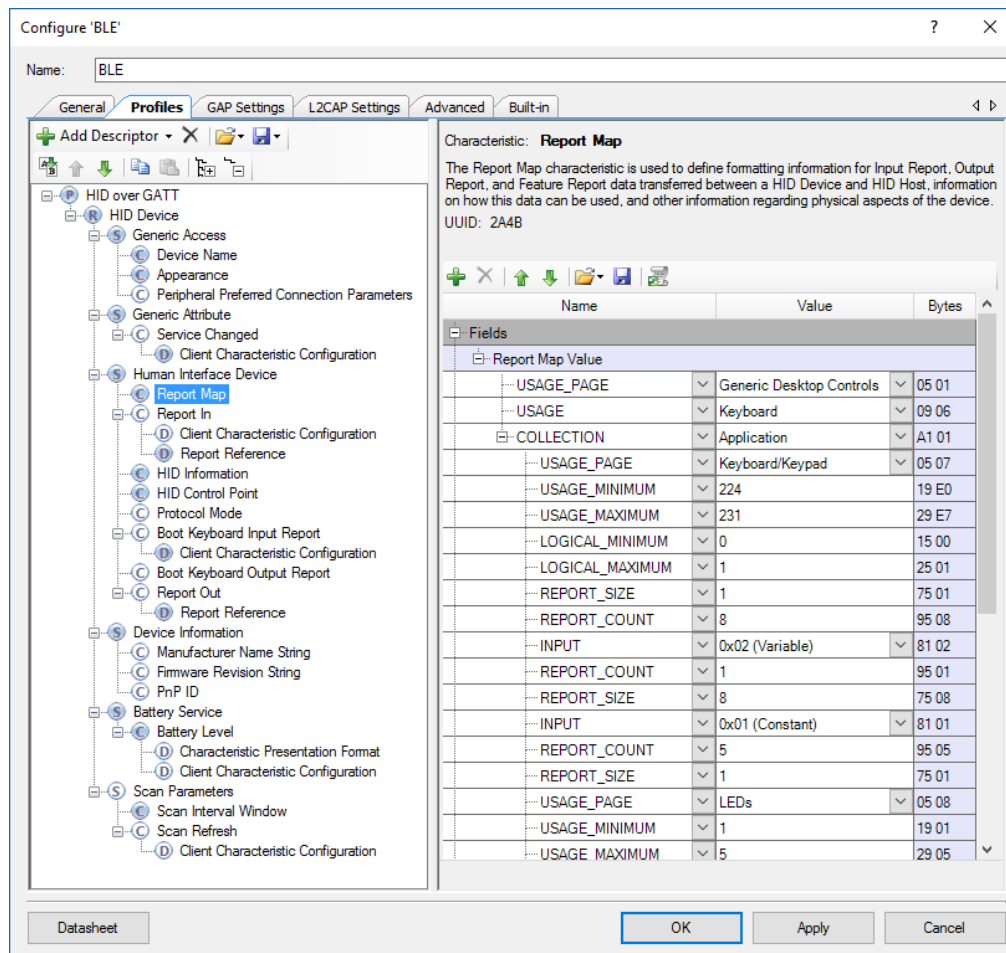


Figure 2. GATT settings

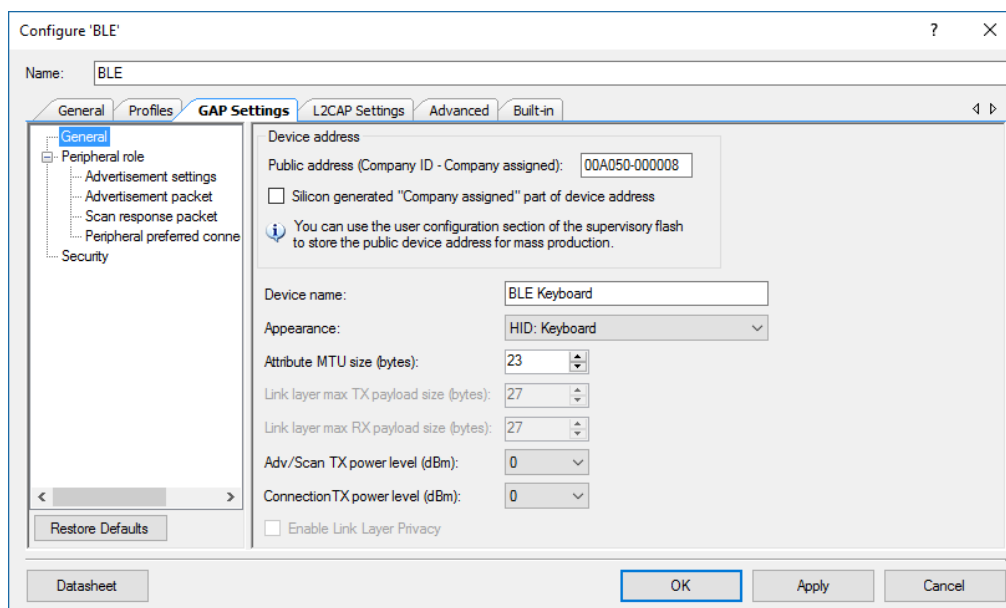


Figure 3. GAP settings

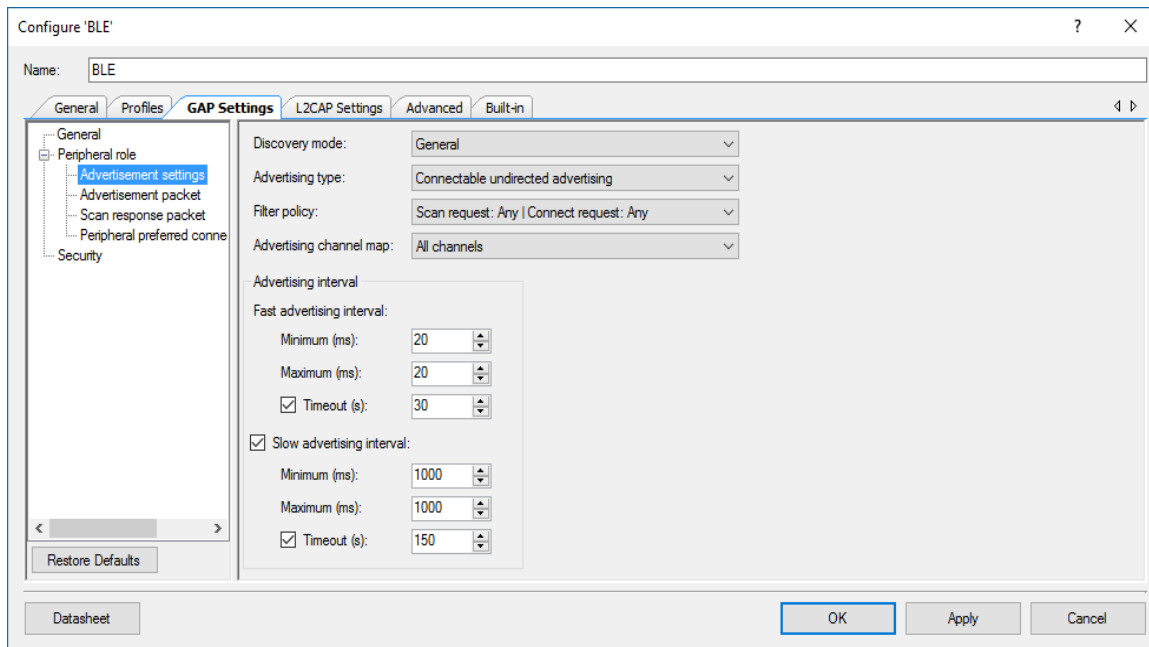


Figure 4. GAP settings -&gt; Advertisement settings

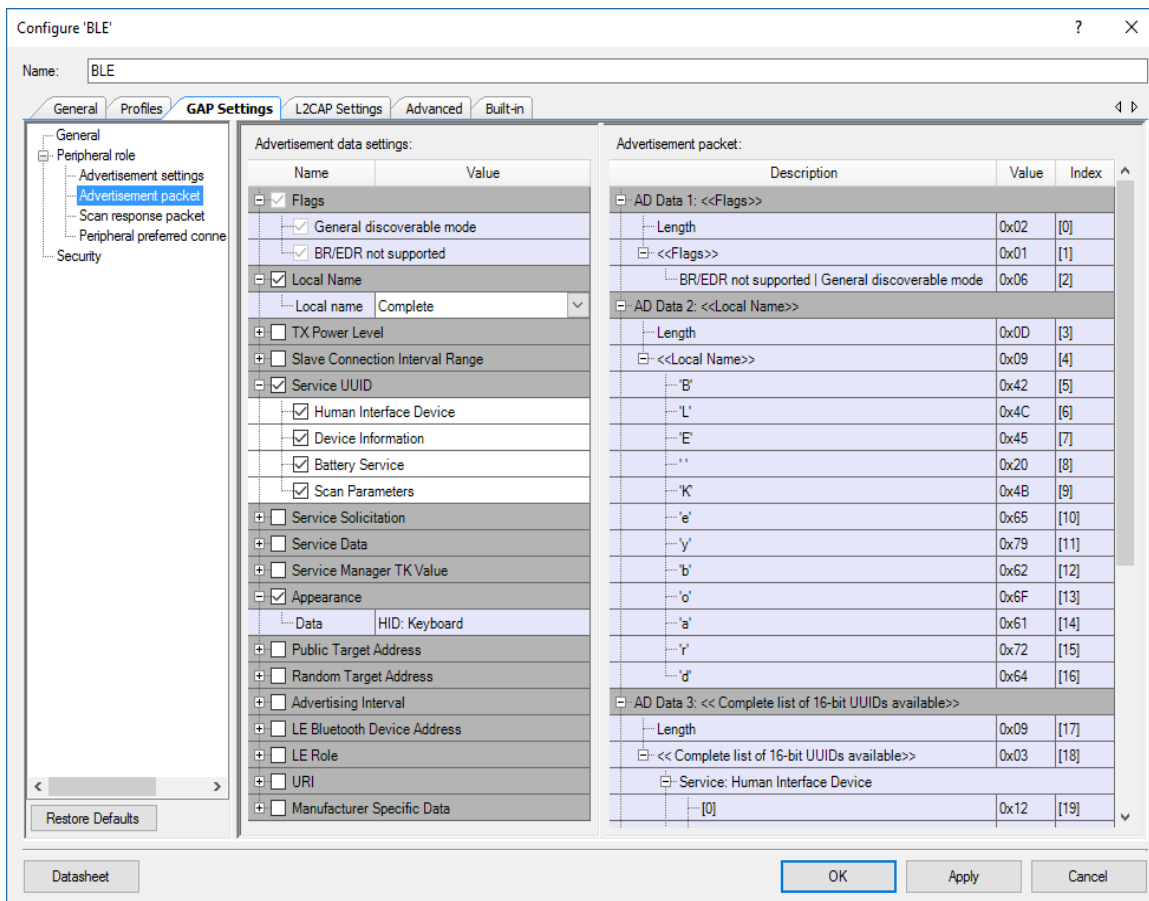


Figure 5. GAP settings -&gt; Advertisement packet

The screenshot shows the 'Configure BLE' dialog box with the 'Peripheral preferred connection parameters' settings. The 'Name' field is set to 'BLE'. The 'Peripheral role' section is expanded, showing 'Peripheral preferred connection parameters' selected. The 'Connection interval' settings are: Minimum (ms) 7.5, Maximum (ms) 15, Slave latency 25, and Connection supervision timeout (ms) 1000. The 'Restore Defaults' button is visible at the bottom left.

Parameter	Value
Minimum (ms)	7.5
Maximum (ms)	15
Slave latency	25
Connection supervision timeout (ms)	1000

Figure 6. GAP settings -&gt; Peripheral preferred connection parameters

The screenshot shows the 'Configure BLE' dialog box with the 'Security' settings. The 'Name' field is set to 'BLE'. The 'Security' section is expanded, showing 'Security' selected. The 'Security mode' is set to 'Mode 1', 'Security level' is 'Unauthenticated pairing with encryption', 'Strict pairing' is 'No', 'I/O capabilities' is 'No Input No Output', 'Keypress notifications' is 'No', 'Bonding requirement' is 'Bonding', and 'Encryption key size (bytes)' is 16. The 'Restore Defaults' button is visible at the bottom left.

Parameter	Value
Security mode	Mode 1
Security level	Unauthenticated pairing with encryption
Strict pairing	No
I/O capabilities	No Input No Output
Keypress notifications	No
Bonding requirement	Bonding
Encryption key size (bytes)	16

Figure 7. Security settings

## Project Description

The project demonstrates the core functionality of the BLE component configured as a HID Device.

Right after the device is started, it performs the BLE component initialization as well as initialization of the UART components. Four callback functions are required in this project for BLE operation. One callback function (AppCallBack()) is required to receive generic events from BLE Stack and the others (HidsCallBack(), BasCallBack(), ScpsCallBack()) are required for receiving events from the services. The component has also buried a call to CyBle\_GappStartAdvertisement() on an execution of which the device will start advertising with the advertisement packet shown in **Figure 5**. As the BLE component is configured in the General Discovery mode, it will stop advertising after an advertisement period expires. On advertisement timeout, the system enters into the hibernate mode. Press the mechanical button on CY8CKIT-042 BLE (SW2) to wake up the system and start re-advertising. BLE subsystem and CPU enters into low power Deep-Sleep mode between connection and advertising intervals. BLE subsystem automatically wakes up to maintain connection and advertising data transfer.

To indicate that the device advertises, the green LED is blinking. The red LED will be lighted on after disconnection to indicate that no Client is connected to the device. When Client is connected successfully both red and blue LEDs will turn off. The blue Led will indicate a Caps Lock state sent from the Host through an output keyboard report characteristic.

You can connect HID Device to Windows 8. Windows 7 and older OS don't have HOGP drivers. Make sure that a PC with Windows 8 has Bluetooth 4.0 installed. To connect to HID Device, click on "Add a device" in the Devices and Printers window of the Control panel. Select the "BLE Keyboard" device and click the Next button. Setup will automatically install the necessary files in the system and start receiving simulated keys "abcdef..."

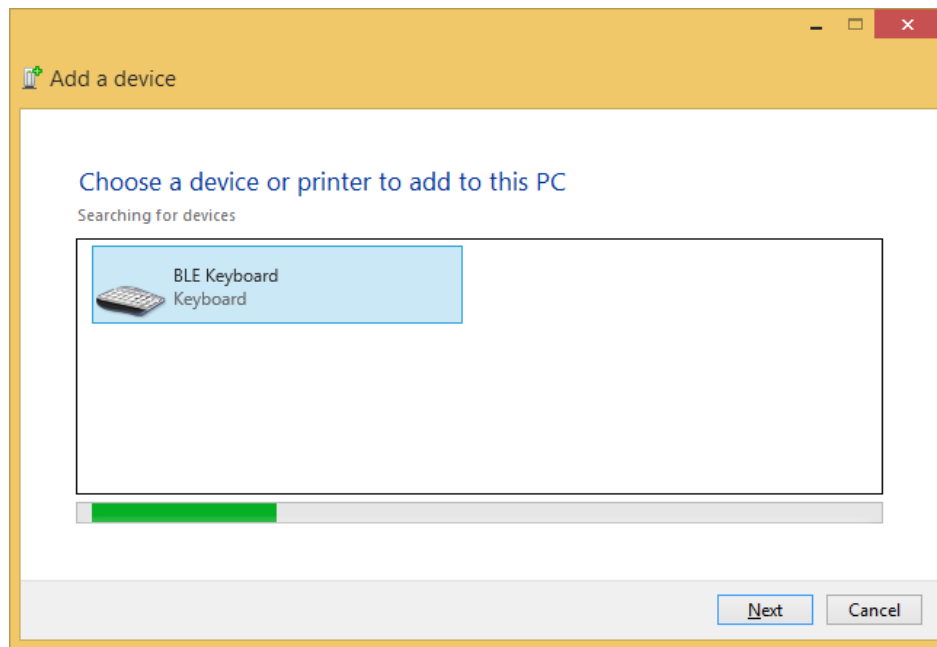


Figure 8. Pairing with Windows 8 PC

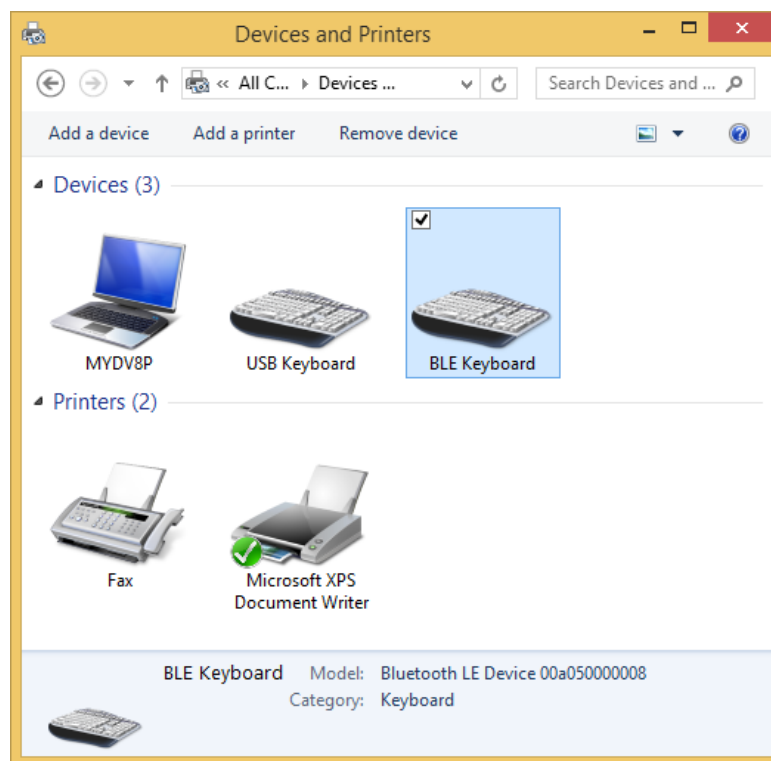


Figure 9. BLE Keyboard is recognized as HID device

Also, you can connect HID Device to Android or iOS device with Bluetooth 4.0 support. To do this, go to phone's Bluetooth settings and pair it with CY8CKIT-042 BLE device (it should be recognized as BLE Keyboard).

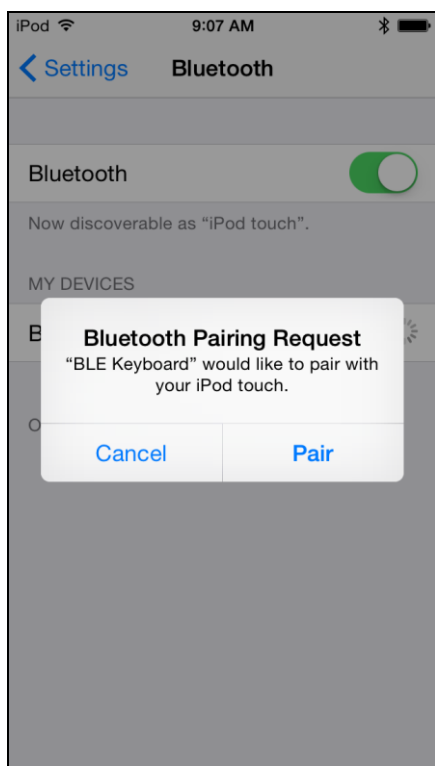


Figure 10. iOS Bluetooth pairing

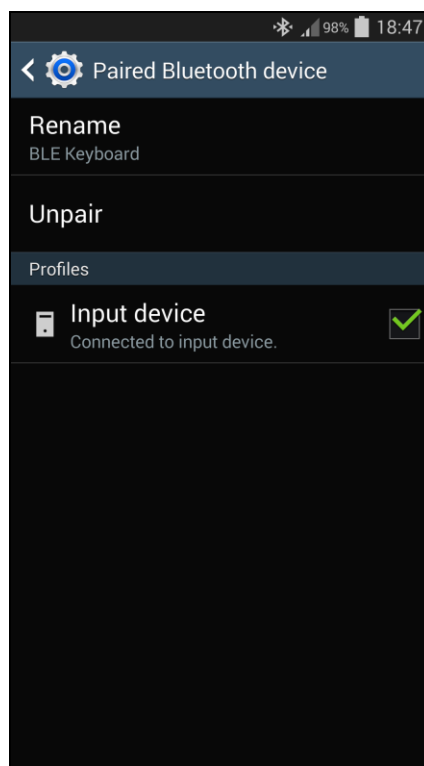


Figure 11. Android settings for paired Bluetooth device

Additionally, this project implements Battery Level Service. By default, the battery level is simulated and is changed from 2 to 20 percent. To enable battery level measurement, set `BAS_MEASURE_ENABLE` to 1, `BAS_SIMULATE_ENABLE` to 0 and connect J2 pin P3[0] to J3 pin VREF. For instructions on how to use this service, refer to BLE\_Battery\_Level example project datasheet.

## Expected Results

After pairing with peer device (Windows 8 PC, Android/iOS phone or tablet), focus input to some editable field (open text editor, take a note, etc). Simulated keys "abcdef..." will fill the document. When SW2 button is pressed, Caps Lock led on the keyboard will be turned on/off. Blue led on KIT will indicate Caps Lock state, received from HID client. Note that old Android OS doesn't send a Caps Lock state backward to device, that's why led will not be turned on/off.

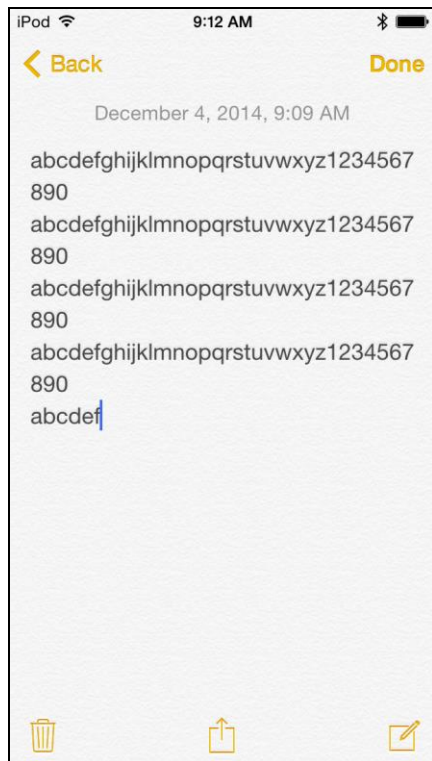


Figure 12. HID Keyboard emulation on iOS device

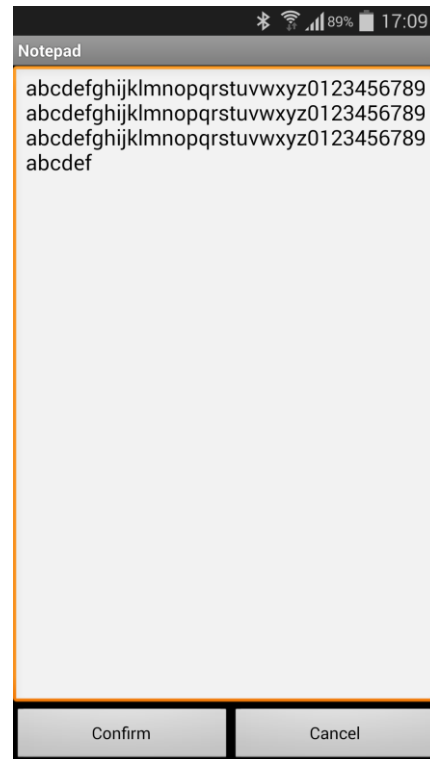


Figure 13. HID keyboard emulation on Android device

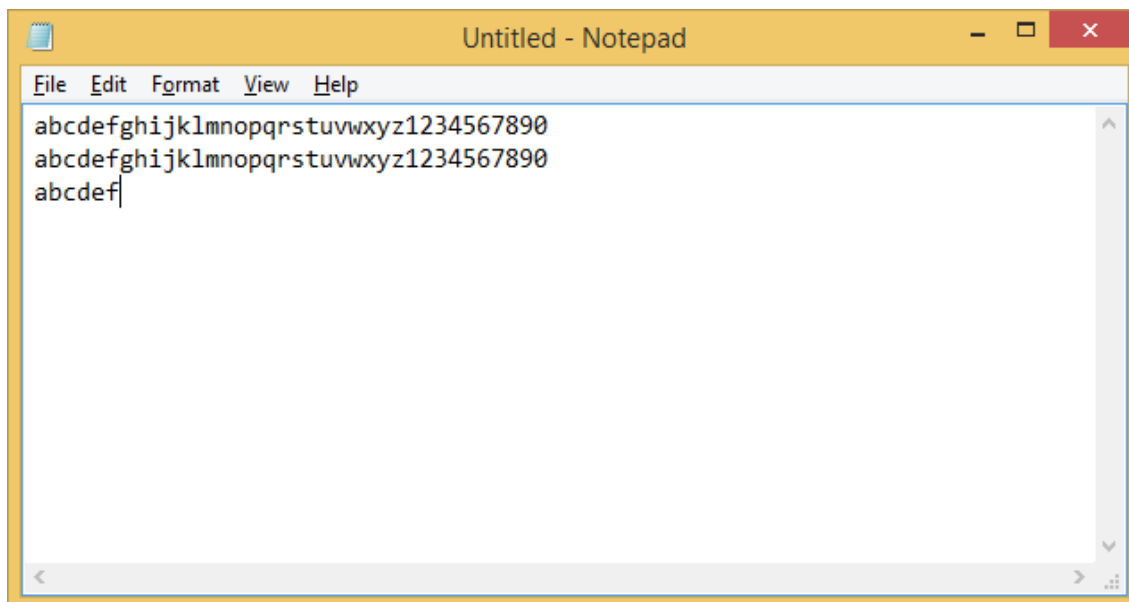


Figure 14. HID Keyboard emulation on Windows 8 PC

© Cypress Semiconductor Corporation, 2009-2016. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.