Please note that Cypress is an Infineon Technologies Company.

The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers

Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

HID Reference Keyboard User Guide

Document Number: 002-29203 Rev. *A

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

Contents

1 Introduction 3

2 Hardware Setup 4

2.1 Power and Ground 5
2.2 Reset and Recovery Buttons 6
2.3 Connecting HCI UART 8
2.4 Connecting PUART 9

3 Programming 11

3.1 Auto Baud Recovery Mode for Programming 11
3.2 Building and Downloading Firmware 11
3.3 TESTING_USING_HCI Option 12
3.4 SLEEP_ALLOWED Option 13
3.5 LED Options 14

4 Testing the Reference Keyboard CYW920819REF-KB-01 Platform 15

4.1 BR/EDR Link Test 16
4.2 LE Link Test 16

Document Revision History 17

Worldwide Sales and Design Support 18

1 Introduction

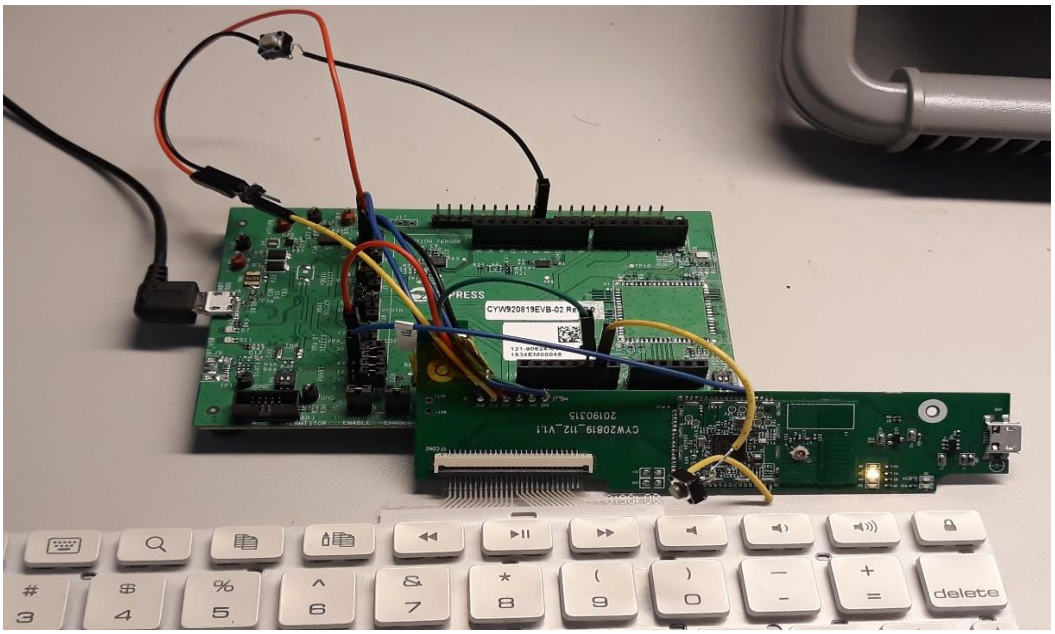
The CYW920819REF-KB-01 platform is designed as a Cypress HID Reference Keyboard, using the CYW920819EVB-02 Evaluation Kit connected to the keyboard hardware as described in this document. It is supported in ModusToolbox™ 2.0 with BTSDK 2.1 (or higher). It can be programmed with BR/EDR and/or LE Bluetooth applications to demonstrate a standard Bluetooth keyboard device.

BTSDK 2.1 supplies the “dual_mode_keyboard” Code Example (“HID_20819REF_KB.dual_mode_keyboard” as shown in the ModusToolbox IDE), a sample application that demonstrates both BR/EDR and LE Bluetooth keyboard functionality on the platform. It can be paired with BR/EDR HID hosts or LE Bluetooth HID Over GATT Protocol (HOGP) host devices.

The keyboard is powered by the USB port; however, only power and ground are connected to the keyboard. The port is purely used as a power supply. The keyboard is designed to upgrade firmware over the air. When OTA firmware upgrade is not available, the only way to program firmware is through direct wire connections. This photo shows the keyboard, USB power connection, and necessary programming signals brought out to a connector for wired programming.

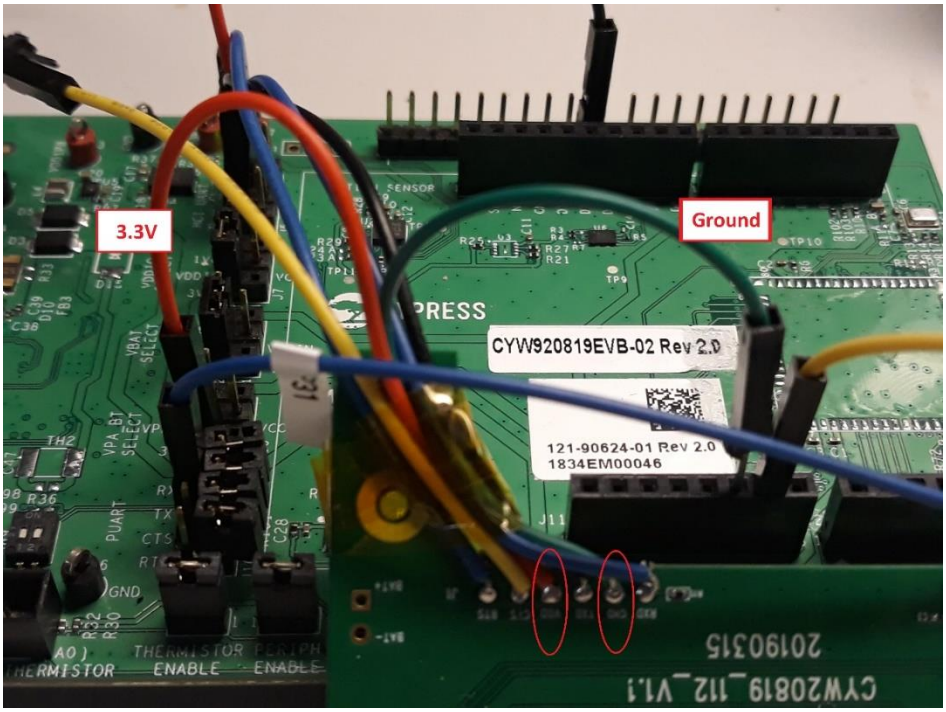
2 Hardware Setup

This chapter describes setting up the hardware using the CYW9208xxEVB-02 Board (EVB) as a UART-USB adapter and power supply.

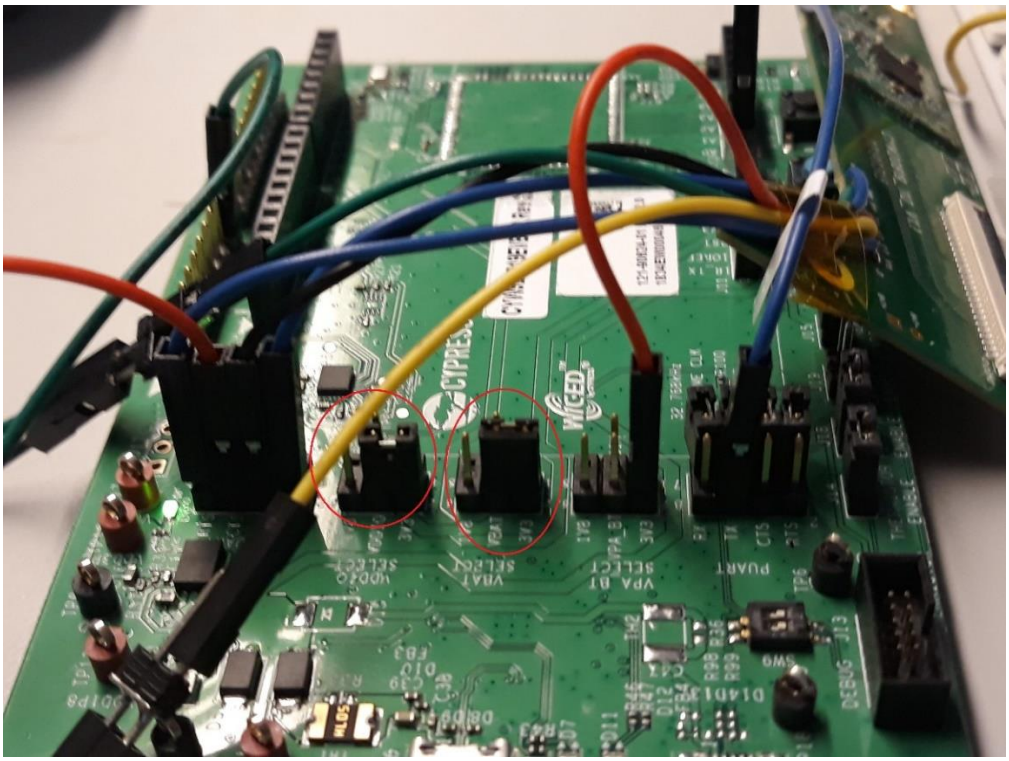


2.1 Power and Ground

1. Connect VDD to EVB J16-P2 where it is labeled as “3V3”.
2. Connect GND to EVB J11-P6, GND.

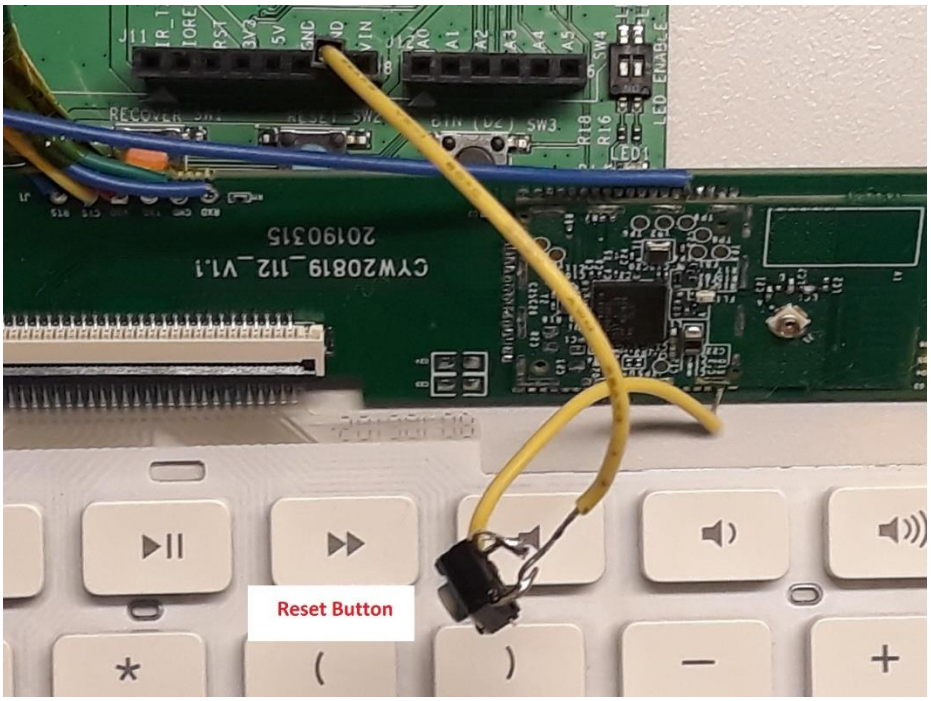


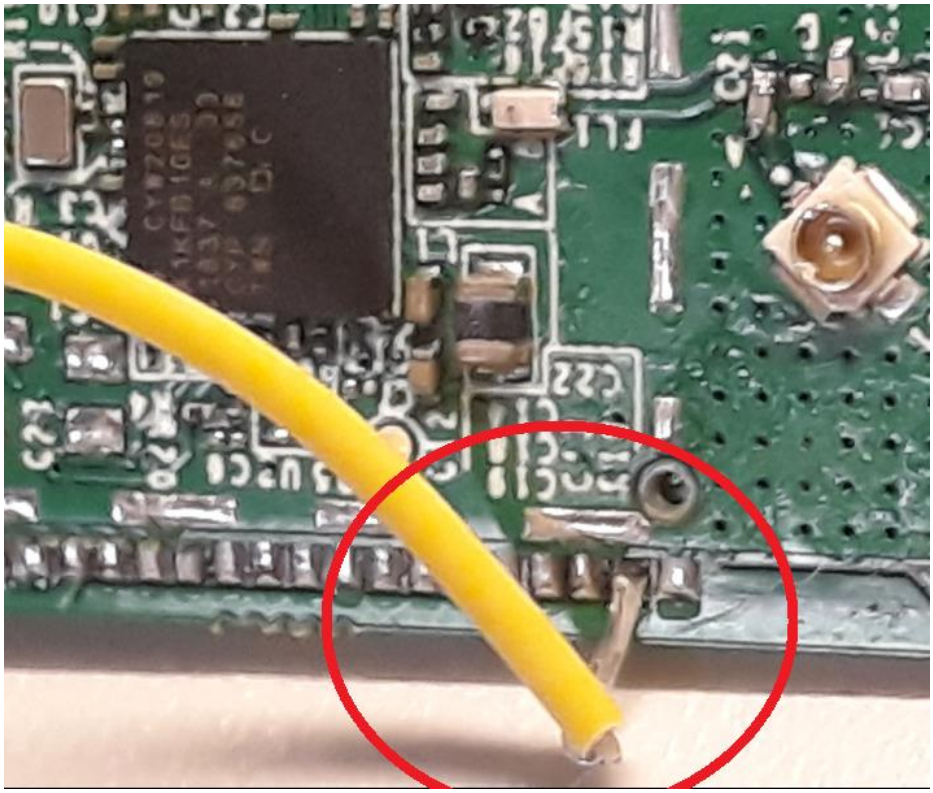
3. To make sure that the board is powered, put jumpers between EVB P2-P4 in J7 and J8 as shown below:



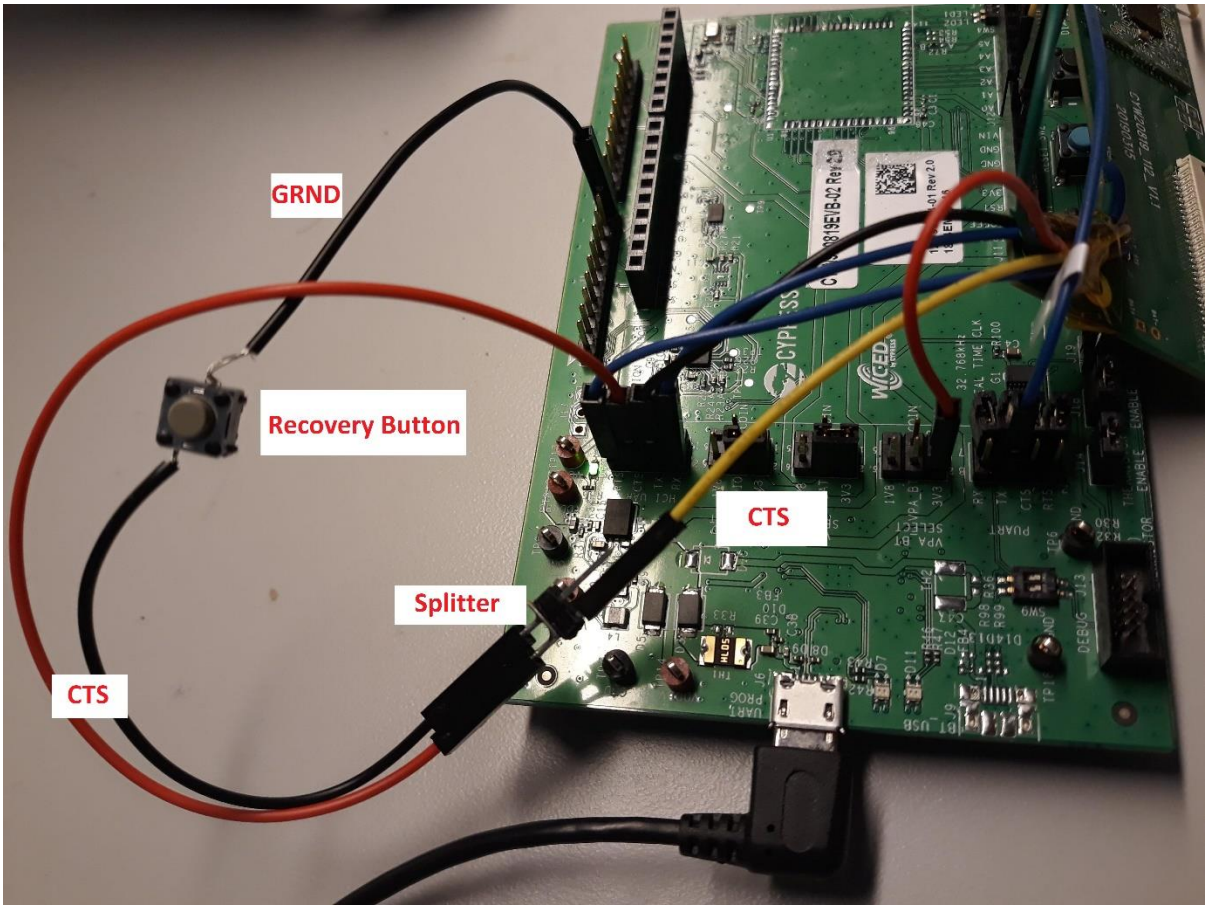
2.2 Reset and Recovery Buttons

Shorting the BT_RST (RST_N) pin in the module to ground will reset CYW20819. Put a push button in between.



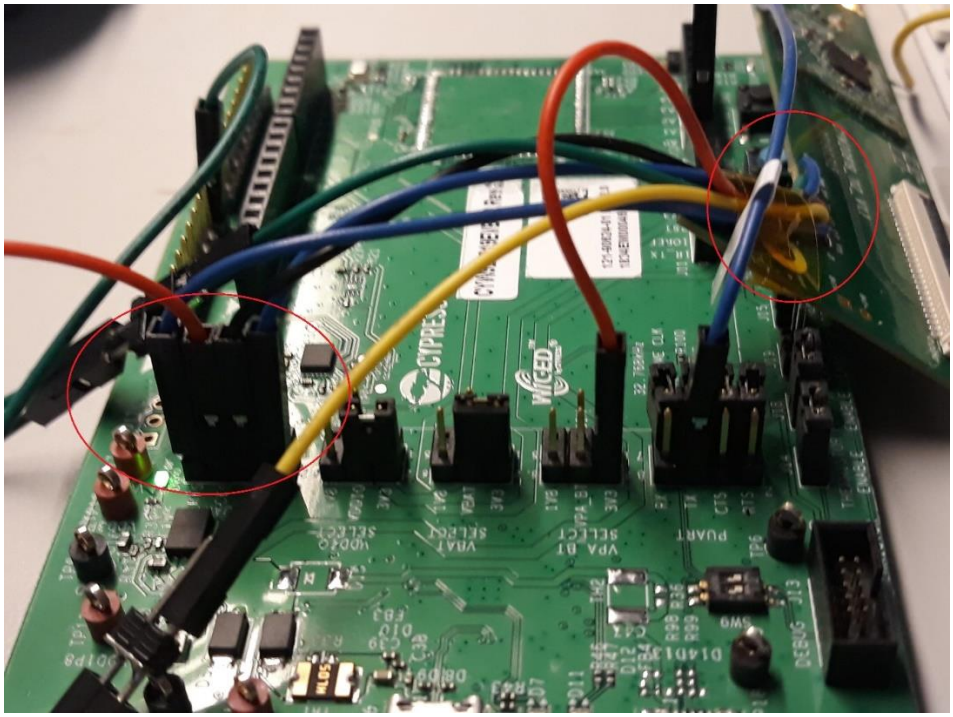


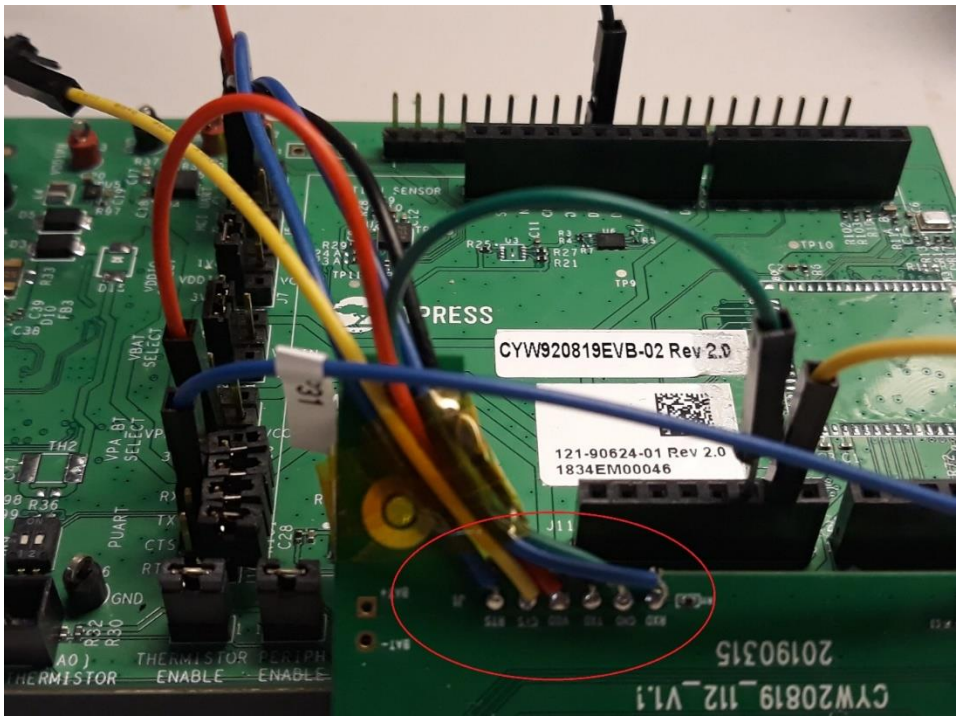
While shorting CTS to ground, resetting the device will allow the device to bypass Flash boot and enter Auto Baud Recovery mode. Because CTS needs to be connected to two places, use a splitter to connect it to a push button and EVB J5-P6.


GRND
Recovery Button
CTS
Splitter
CTS

2.3 Connecting HCI UART

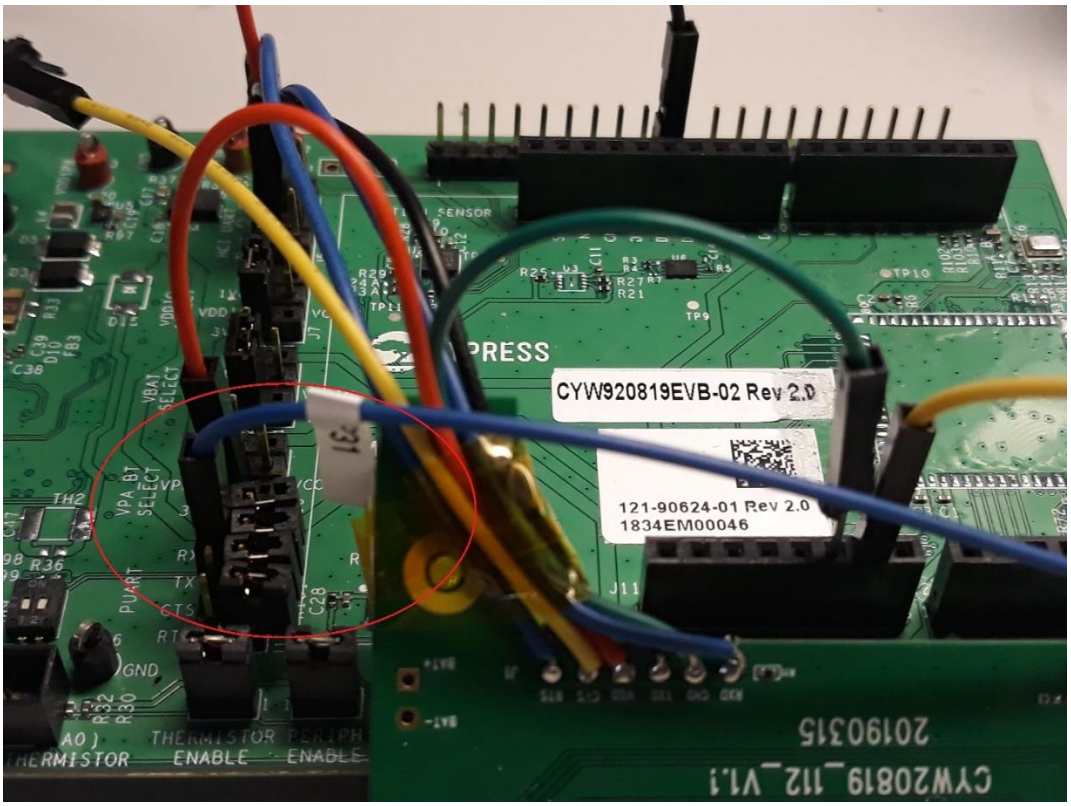
Remove all jumpers from EVB J5 and connect TX, RX, CTS, and RTS signals to the TX, RX, CTS, and RTS pins of the reference keyboard.





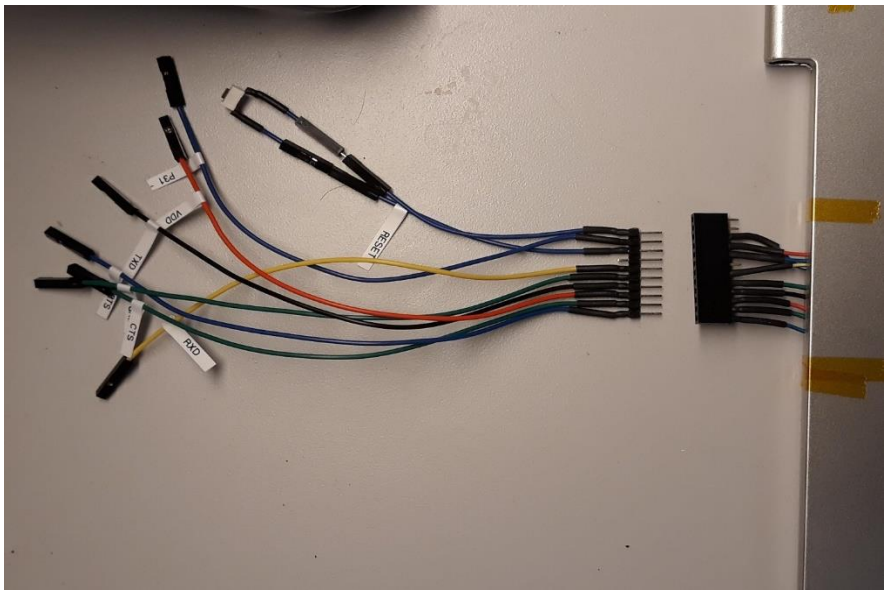
2.4 Connecting PUART

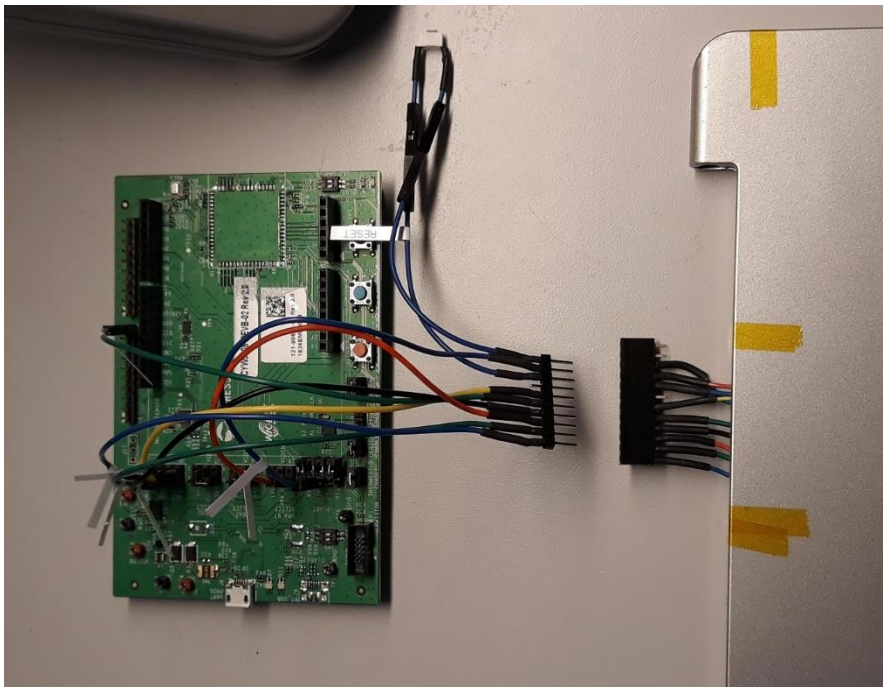
Remove all jumpers from EVB J10 and connect EVB P31 J10-P6 (TX).



When the EVB board is connected to a USB port on a host PC, the system will enumerate two serial ports: one for HCI UART and one for PUART. The HCI UART port is used for programing from ModusToolbox. Use a serial port terminal application to open the PUART port so that the firmware debug output can be shown. Use 115200 Baud, 8 bits, no parity. Use 'LF' as the 'return' character for both TX and RX lines.

The wiring can be arranged as shown:





With this arrangement, after programming, the device can be detached easily and used as a stand-alone device.

3 Programming

3.1 Auto Baud Recovery Mode for Programming

When upgrading firmware through the HCI UART, the device must be put into Auto Baud Recovery Mode for programming. Do the following to put the device into this mode:

1. Press and hold the **Recovery** button.
2. While the **Recovery** button is held down, press and release the Reset button.
3. Release the **Recovery** button.

If it is done properly, the PUART port should not show any output after the **Reset** button is pressed in Step 2.

If the Recovery button is not available, do the following to put the device into Auto Baud Recovery mode.

1. Use the Client Control application supplied with BTSDK, or any other serial terminal application to open the HCI UART port.

   When the port is opened, it is equivalent to the Recovery button being pressed because the CTS line is driven low for hardware flow control.

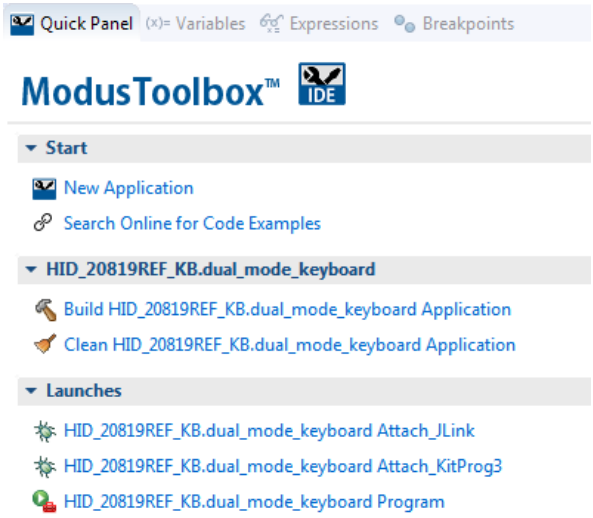
2. Press the **Reset** button while the port is open, and then close the port to be ready for programming.

   This will be the same effect as the Recovery button steps above. Again, after the Reset button is pressed, if the device is in Auto Baud Recovery Mode, it should not have any output on the PUART port.

With this method, you must ensure that no character or command is sent to the port after reset because when sending any data, the CTS line can be toggled for flow control and the device will exit Auto Baud Recovery Mode.

3.2 Building and Downloading Firmware

1. In ModusToolbox IDE, click on the **New Application** link in the Quick Panel to create a new project.
2. Select the **CYW920819REF-KB-01** platform and then select **wiced_btsdk**. This installs the prerequisite SDK support needed for the keyboard application. This needs to be performed only once.
3. After **wiced_btsdk** has been created, click on the **New Application** link again in the Quick Panel, and select the **CYW920819REF-KB-01** platform and then the **HID-20819REF-KB** application. This installs the **HID_20819REF_KB.dual_mode_keyboard** application.
4. Use the **Program** Launch link in the Quick Panel to build the application and program it to the board.



Alternatively, after the application and **wiced_btsdk** have been created in the ModusToolbox IDE, the firmware can be built and downloaded using the command line. Do the following:

Open a command prompt (CMD, xterm, etc.) and change directory to the ModusToolbox workspace folder (the *mtw* folder in the user home directory by default), and execute the following commands:

```
$ cd HID_20819REF_KB/hid/dual_mode_keyboard
$ make clean
$ make program
```

After download, the application executes; the PUART output should show as follows:

```
<<CY DUAL MODE KB start>>
OTA_FW_UPGRADE
SKIP_PARAM_UPDATE
…
```

The application uses the following default values:

HCI_UART (TESTING_USING_HCI=0) transport is disabled

PUART=P31 (Baud 115200, 8 bits, no parity)

Device will sleep (power off for HIDOFF), need to use Recovery for reprogramming.

The key matrix is enabled.

3.3 TESTING_USING_HCI Option

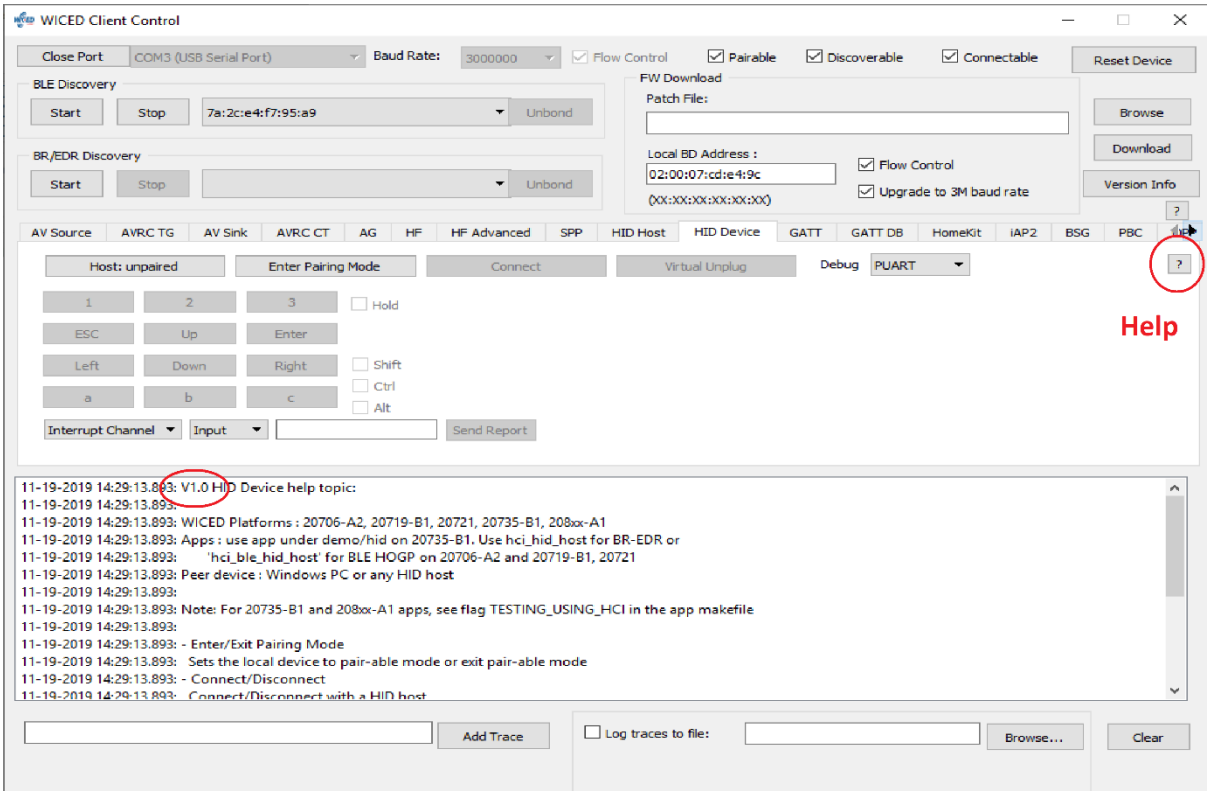
With the reference keyboard platform, HCI transport is disabled by default. To use BTSDK host utilities (Client Control or BtSpy), the TESTING_USING_HCI option must be set to '1'. This can be configured in the application makefile, or can be supplied as a command line override as follows:

```
$ make clean
$ make program TESTING_USING_HCI=1
```

PUART output:

```
<<CY DUAL MODE KB start>>
TESTING_USING_HCI
SLEEP_ALLOWED=2
OTA_FW_UPGRADE
SKIP_PARAM_UPDATE
…
```

Use ModusToolbox 2.0 with BTSDK 2.1 or higher to ensure the correct version of Client Control which supports BR/EDR HID. Click **Help**; the output should have v1.0 HID as the following:



3.4 SLEEP_ALLOWED Option

There are four levels of Sleep options, which are configurable in the makefile or via command line override.

In makefile, change SLEEP_ALLOWED_DEFAULT=n

In Cygwin command line, use SLEEP_ALLOWED=n

Where n is:

0 No sleep allowed

1 Sleep is allowed without Deep Sleep:
Radio powered down and digital core is mostly powered down except for RAM, registers, and some core logic. The device can wake up either after a programmed period or upon receiving an external event. Processor is paused during Sleep and does not require boot upon wake up. The device consumes approximately 50 uA.

2 Deep Sleep (shutdown Sleep) is allowed of type ePDS (Extended Power Down Sleep):
Only the main RAM and ePDS control circuitry retains power. All other components are powered OFF. The device can wake up either after a programmed period or upon receiving an external event. The processor is paused during Sleep and does not require boot upon wake up. The device consumes about 8 uA floor current.

3 Deep Sleep (shutdown Sleep) is allowed of type HIDOFF:
The device is powered OFF with minimum wake up control circuit. The device can wake up either after a programmed period or upon receiving an external event. In this Deep Sleep mode, the firmware always requires full-boot when waking up from HIDOFF. The device consumes about 1 uA.

By default, the application is set to `SLEEP_ALLOWED=2`. If TESTING_USING_HCI is enabled and the HCI UART port is opened by Client Control, Deep Sleep will be disabled.

PUART output:

```
<<CY DUAL MODE KB start>>
SLEEP_ALLOWED=2
OTA_FW_UPGRADE
SKIP_PARAM_UPDATE
…
```

3.5 LED Options

The LED functionality is enabled by default. It can be disabled by changing the makefile to use `LED_SUPPORT_DEFAULT=0`, or via a command line override using the `LED=0` parameter. The LED function should be turned off for power measurement.

There are four LEDs available on the keyboard reference board: WHITE, BLUE, YELLOW, and RED.

- WHITE: Used for the Caps Lock indicator. This LED is set or cleared by the host after a Bluetooth HID link is connected. When you press the **Caps Lock** key, the host receives the key state, and it changes the Caps Lock state at the host, which will send the report back to the reference board to set or clear this LED accordingly.
- BLUE: Used for LE link status. This LED blinks while in the pairing state, is solid on when the LE link is up, and off when the link is down.
- YELLOW: Used for BR/EDR link status. This LED blinks while in the pairing state, is solid on the BR/EDR link is up, and off when the link is down.
- RED: Error indicator. Not used.

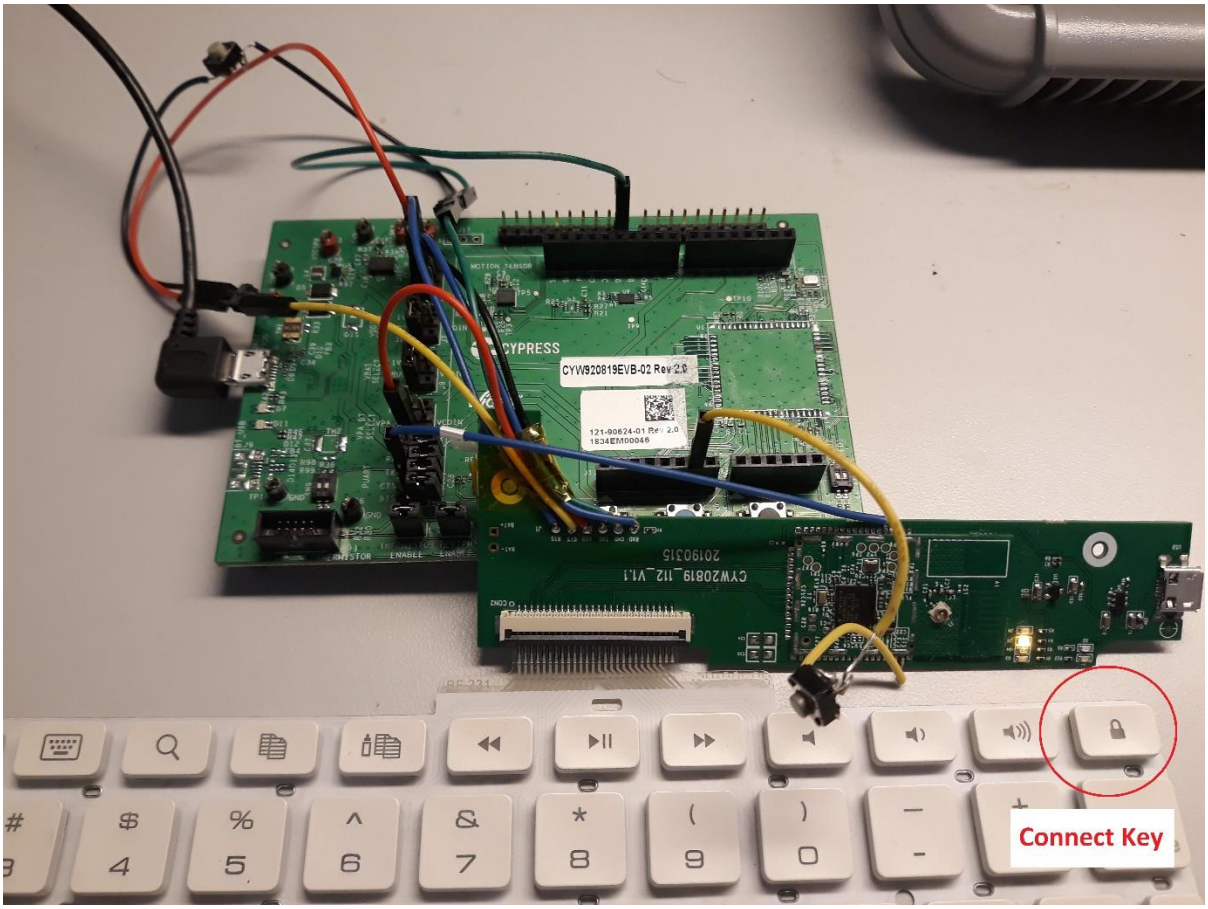
4 Testing the Reference Keyboard CYW920819REF-KB-01 Platform

Do the following:

1. Connect the CYW920819EVB-02 USB port to the host PC.
2. Connect a terminal emulation program to the PUART serial port to allow the application debug output to be shown.
3. Build and program the image to device.
4. Press the Reset button to reset the device. The Blue LED should blink five times to indicate that the firmware is running.

Because there is no “Connect” button designed for the reference keyboard to initiate Bluetooth connections, the right-most top ‘Lock’ key is used as the **Connect** button. Pressing this key once will start BR/EDR pairing. While the EVB is in BR/EDR pairing, pressing this key one more time will switch to the EVB to LE pairing. Pressing it one more time while the EVB is in LE pairing will stop the pairing process.

Pressing the **Connect** key will perform a ‘virtual cable unplug’ and erase any previously paired host information. Thus, in that state, pairing would need to be performed again to reconnect back to any previously paired host. When the pairing information is erased in the reference platform, the host also must ‘unpair’, ‘forget’, or otherwise remove the paired device from the host’s Bluetooth configuration so that it may be paired again to reconnect.



4.1 BR/EDR Link Test

1. Press the **Connect** key once for BR/EDR pairing mode. The YELLOW LED should blink.
2. Pair to a BR/EDR host.
3. After successful pairing, the YELLOW LED should glow to indicate that the link is connected.
4. From the host, open a text editor or notepad application, and then start typing on the reference keyboard to make sure that keystrokes are sent to the host.
5. Press the **Caps Lock** key several times to make sure that the WHITE LED is toggled to show the Caps Lock status. When it is locked, pressing any letter key should result in upper-case letters appearing at the host.
6. Disconnect the link from the host side (if the host is capable) and press a key from the reference keyboard to make sure that it can reconnect back to the host. After reconnecting, verify that the keystrokes can still be received by the host.
7. Press the **Reset** button to disconnect the keyboard. Press a key on the keyboard to reconnect. After reconnecting, verify that keystrokes can still be received by the host.
8. Power cycle the keyboard and press a key from the keyboard to reconnect. After reconnecting, verify that keystrokes can still be received by the host.

4.2 LE Link Test

1. Press the **Connect** button twice for LE pairing mode. The BLUE LED should blink.
2. Pair to an LE host. Note that it is better to use a different host than the one used for BR/EDR testing to avoid connection problems. This is because the keyboard has the same Bluetooth address for both BR/EDR and LE, the host may not be able to differentiate between the two Bluetooth links from the same device address.
3. After successful pairing, the BLUE LED should be solid to indicate that the link is connected.
4. Repeat steps 4 through 8 from the BR/EDR link test.

Document Revision History

Document Title: HID Reference Keyboard User Guide

Document Number: 002-29203

Revision	ECN	Issue Date	Description of Change
**	6745617	12/09/2019	Initial release
*A	6817269	02/26/2020	Updated SLEEP_ALLOWED Option.

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

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, 2019-2020. 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.