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.





ModusToolbox™

WICED Manufacturing Bluetooth Test Tool

Document Number: 002-14799 Rev. *F

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

Contents

1 Introduction 3

2 Setup 4

2.1 Device Configuration 4
2.2 Environment Variables 4

3 Reset Test 5

3.1 reset 5
3.2 reset_highspeed 5

4 LE Receiver Test 6

5 LE Transmitter Test 7

6 LE Test End 8

7 Continuous Transmit Test 9

8 Continuous Receive Test 11

9 Radio TX Test 12

10 Radio RX Test 14

11 BQB RF Test 16

12 Read BD_ADDR Command 17

13 Factory Commit BD_ADDR Command 18

References 19

Document History 20

Worldwide Sales and Design Support 21

1 Introduction

The WICED Manufacturing Bluetooth test tool (WMBT) is used to test and verify the RF performance of the Cypress SoC Bluetooth BR/EDR/LE devices. For LE tests, standard procedures from the Bluetooth Core Specification [1] are utilized. For BR/EDR tests a set of vendor specific commands are introduced and described in this document. Each test sends a Host Controller Interface (HCI) or WICED HCI command to the device and then waits for an HCI Command Complete event from the device.

2 Setup

2.1 Device Configuration

The Cypress Bluetooth device to be tested must expose an HCI UART and that this UART can be connected to a COM port or to a Serial to USB device of a PC. The HCI UART supports HCI Commands and Events described in this document.

The device should be preprogrammed with an application image and should be reset after it has been connected to the PC and the COM port drivers are loaded. The WMBT tool is part of the WICED Bluetooth SDK and can be found under the *wiced_btsdk* folder in the ModusToolbox workspace after an application project has been created and downloaded to the device using ModusToolbox. For example, on Windows, the default location is:

C:\Users\<username>\mtw\wiced_btsdk\tools\btsdk-utils\wmbt

Check the device specific Kit Guide or Quick Start Guide for any DIP switch settings or jumper settings to configure the device to expose the HCI UART interface.

2.2 Environment Variables

2.2.1 MBT_BAUD_RATE

Cypress SoC Bluetooth devices support adjustable baud rates up to 4 Mbps via the `wiced_transport_init()` API included with the WICED Bluetooth SDK. If this API is not utilized in an application to re-configure the baud rate, the default rate of 115.2 Kbps will be used by the device. The `MBT_BAUD_RATE` environment variable must be set to match what the device is using before running WMBT.

As an example, to configure `MBT_BAUD_RATE` for 3 Mbps on a windows command line:

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin>set MBT_BAUD_RATE=3000000
```

2.2.2 TRANSPORT_MODE

The Bluetooth Core Specification [1] defines the HCI, which provides a standardized communication protocol between the BT host stack and BT controller. Cypress SoC Bluetooth devices provide a high level of integration, for example, BT Controller and embedded BT Host Stack in a single chip, to simplify BT product development for customers so that they are not required to be familiar with all HCI commands and events. Typically, when the embedded stack is utilized in the Cypress device and it interfaces to an onboard MCU, the MCU software would likely need to send and receive commands and events to the Cypress device. For such a solution, WICED HCI is defined and provided as an example, see WICED HCI UART Control Protocol [2].

WMBT provides support for both HCI and WICED HCI via the `TRANSPORT_MODE` environment variable. If WICED HCI is desired, your application must implement handlers for `HCI_CONTROL_TEST_COMMAND_ENCAPSULATED_HCI_COMMAND;` see *hci_control_test.c* included with the watch sample application. HCI should be sufficient for most cases since the devices support this by default. The `TRANSPORT_MODE` environment variable must be set to the desired mode before running WMBT.

As an example, to configure `TRANSPORT_MODE` for HCI on a windows command line:

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin>set TRANSPORT_MODE=0
```

3 Reset Test

This test verifies that the device is correctly configured and connected to the PC. If your application re-configures the baud rate, you will want to use the `reset_highspeed` command.

3.1 reset

Description: Sends an HCI Reset command at 115.2 Kbps to the device and processes the HCI Command Complete event (See Reference [1] [Vol 2, Part E], Section 7.3.2 for details).

Usage: `wmbt reset COMx`

Example:

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt reset COM23
Opened COM23 at speed: 115200
Sending HCI Command:
0000 < 01 03 0C 00 >
Received HCI Event:
0000 < 04 0E 04 01 03 0C 00 >
Success
Close Serial Bus
```

The last byte of the HCI Command Complete event is the operation status, where 0 signifies success.

3.2 reset_highspeed

Description: Sends an HCI Reset command at the configured `MBT_BAUD_RATE` to the device and processes the HCI Command Complete event (See Reference [1] [Vol 2, Part E], Section 7.3.2 for details).

Usage: `wmbt reset_highspeed COM23`

Example:

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt reset_highspeed COM23
Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 03 0C 00 >
Received HCI Event:
0000 < 04 0E 04 01 03 0C 00 >
Success
Close Serial Bus
```

The last byte of the HCI Command Complete event is the operation status, where 0 signifies success.

4 LE Receiver Test

This test configures the chip to receive reference packets at a fixed interval. External test equipment should be used to generate the reference packets.

The frequency on which the device listens for the packets is passed as a parameter. BLE devices use 40 channels, each of which is 2 MHz wide. (See Reference [1] [Vol 2, Part E], Section 7.8.28 for details).

- 2402 MHz maps to Channel 0
- 2480 MHz maps to Channel 39

The following equation can be used to map the channel number to actual center frequency:

$$Frequency = (2 \times Channel) + 2402MHz$$

Usage: `wmbt le_receiver_test COMx <rx_frequency>`

Where:

- rx_frequency = (2402 - 2480) receive frequency, in MHz

The example below starts the LE receiver test on Channel 2 (2406 MHz):

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt le_receiver_test COM23 2406
MBT_BAUD_RATE:  3000000
TRANSPORT_MODE: 0 (HCI)

Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 1D 20 01 02 >
Received HCI Event:
0000 < 04 0E 04 01 1D 20 00 >
Success
Close Serial Bus
```

The last byte of the HCI Command Complete event is the operation status, where 0 signifies success.

Use `wmbt le_test_end COMx` to complete the test and print the number of received packets.

Note: This test will fail if the device is running another test: use `le_test_end` to put the device in an idle state before running this test.

5 LE Transmitter Test

The LE Transmitter Test configures the Cypress SoC BT device to send test packets at a fixed interval. External test equipment may be used to receive and analyze the reference packets.

The frequency on which the device transmits the packets is passed as a parameter. BLE devices use 40 channels, each of which is 2 MHz wide. (See Reference [1] [Vol 2, Part E], Section 7.8.28 for details).

The other two parameters specify the length of the test data and the data pattern to be used (see Reference [1] [Vol 2, Part E], Section 7.8.29 for details).

Usage: `wmbt le_transmitter_test COMx <tx_frequency> <data_length> <data_pattern>`

Where:

- rx_frequency = (2402 - 2480) receive frequency, in MHz
- data_length = 0–37
- data_pattern = 0–7
  0: Pseudo-random bit sequence 9

  1: Pattern of alternating bits: 11110000

  2: Pattern of alternating bits: 10101010

  3: Pseudo-random bit sequence 15

  4: Pattern of all 1s

  5: Pattern of all 0s

  6: Pattern of alternating bits: 00001111

  7: Pattern of alternating bits: 0101

The example below starts the test and instructs the device to transmit packets on Channel 2 (2406 MHz), with a 10-byte payload of all ones (1s):

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin>> wmbt le_transmitter_test COM23 2406 10 4
MBT_BAUD_RATE:  3000000
TRANSPORT_MODE: 0 (HCI)

Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 1E 20 03 02 0A 04 >
Received HCI Event:
0000 < 04 0E 04 01 1E 20 00 >
Success
Close Serial Bus
```

The last byte of the HCI Command Complete event is the status of the operation, where 0 signifies success.

Use `wmbt le_test_end COMx` to complete the test.

Note: This test will fail if the device is running another test: use `le_test_end` to put the device in an idle state before running this test.

6 LE Test End

This command stops the LE Transmitter or LE Receiver Test that is in progress.

The number of packets received during the test is reported by the device and printed out. The value will always be zero if the LE Transmitter Test was active (See Reference [1] [Vol 2, Part E], Section 7.8.30 for details).

Usage: `wmbt le_test_end COMx`

The example below stops the active test:

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt le_test_end COM23
MBT_BAUD_RATE:  3000000
TRANSPORT_MODE: 0 (HCI)

Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 1F 20 00 >
Received HCI Event:
0000 < 04 0E 06 01 1F 20 00 00 00 >

Success num_packets_received = 0

Close Serial Bus
```

7 Continuous Transmit Test

Note: Unlike the LE tests, this test uses 79 frequencies, each 1 MHz wide.

This test configures the Cypress SoC BT device to turn the carrier ON or OFF. When the carrier is ON the device transmits according to the specified transmit mode, modulation type, frequency, and power level.

Usage: `wmbt tx_frequency_arm COMx <carrier on/off> <tx_frequency> <mode> <modulation_type> <tx_power>`

Where:

- carrier on/off:
  1: carrier ON

  0: carrier OFF
- tx_frequency: (2402 – 2480) transmit frequency, in MHz
- tx_mode: selects unmodulated or modulated with pattern
  0: Unmodulated

  1: PRBS9

  2: PRBS15

  3: All Zeros 4: All Ones

  5: Incrementing Symbols
- tx_modulation_type: selects 1 Mbps, 2Mbps, or 3 Mbps modulation. Ignored if mode is unmodulated.
  0: GFSK

  1: QPSK

  2: 8PSK

  3: LE
- tx_power = (–25 to +3) transmit power, in dBm

The example below turns the carrier ON and instructs the device to transmit an unmodulated pattern on 2402 MHz at 3 dBm.

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt tx_frequency_arm COM23 1 2402 1 2 3
MBT_BAUD_RATE:  3000000
TRANSPORT_MODE: 0 (HCI)

Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 14 FC 07 00 00 01 02 08 03 00 >
Received HCI Event:
0000 < 04 0E 04 01 14 FC 00 >
Success
Close Serial Bus
```

To stop the test, send the command a second time to the same COM port with the carrier on/off parameter set to zero (0).

```
<workspace>\wiced_btsdk\tools\btsdk-utils \wmbt\bin> wmbt tx_frequency_arm COM23 0 2402 1 2 3
MBT_BAUD_RATE:  3000000
TRANSPORT_MODE: 0 (HCI)

Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 14 FC 07 01 02 00 00 00 00 00 >
Received HCI Event:
0000 < 04 0E 04 01 14 FC 00 >
Success
Close Serial Bus
```

8 Continuous Receive Test

This test configures the Cypress SoC BT device to turn ON the receiver in a non-hopping continuous mode. The frequency to be used by the device is passed as a parameter.

Usage: `wmbt receive_only COMx <rx_frequency>`

Where:

- rx_frequency = (2402 – 2480) receiver frequency, in MHz

The example below instructs the Cypress SoC BT device to set the receiver to frequency of 2046 MHz.

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt receive_only COM23 2406
MBT_BAUD_RATE:  3000000
TRANSPORT_MODE: 0 (HCI)

Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 2B FC 01 04 >
Received HCI Event:
0000 < 04 0E 04 01 2B FC 00 >
Success
Close Serial Bus
```

9 Radio TX Test

This test is the connectionless transmit test that sends Bluetooth packets. The test configures the Cypress SoC BT device to transmit the selected data pattern which is governed by a specified frequency and a specified logical channel at a specified power level.

Usage: `wmbt radio_tx_test COMx <bd_addr> <frequency> <modulation_type> <logical_channel> <bb_packet_type> <packet_length> <tx_power>`

Where:

- bd_addr: BD_ADDR of Tx device (6 bytes)
- frequency: Set to 0 to use a normal Bluetooth hopping sequence, or 2402 MHz to 2480 MHz to transmit on a specified frequency without hopping.
- modulation_type: Sets the data pattern
  0: 0x00 8-bit Pattern
  1: 0xFF 8-bit Pattern
  2: 0xAA 8-bit Pattern
  3: 0xF0 8-bit Pattern
  4: PRBS9 Pattern
- logical_channel: Sets logical channel to Basic Rate (BR) or Enhanced Data Rate (EDR) for ACL packets.
  0: EDR
  1: BR
- bb_packet_type: Baseband packet type to use
  3: DM1
  4: DH1/2-DH1
  8: 3-DH1
  10: DM3/2-DH3
  11: DH3/3-DH3
  14: DM5/2-DH5
  15: DH5/3-DH5
- packet_length: 0 to 65535. The device will limit the maximum packet length based on the baseband packet type. For example, if DM1 packets are sent, the maximum packet size is 17 bytes.
- tx_power: –25 dBm to +3 dBm

The example below instructs the Cypress SoC BT device to transmit 0xAA pattern on 2402 MHz frequency using an ACL connection with Basic Rate DM1 packets at –3 dBm.

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt radio_tx_test COM23 112233445566
2402 2 1 3 17 -3
MBT_BAUD_RATE:  3000000
TRANSPORT_MODE: 0 (HCI)
Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 51 FC 10 66 55 44 33 22 11 01 00 03 01 03 11 >
0010 < 00 08 FD 00 >
```

```
Received HCI Event:
0000 < 04 0E 04 01 51 FC 00 >
Success
Close Serial Bus
```

The last byte of the HCI Command Complete event is the operation status, where 0 signifies that the operation was successful and the test started to run. The test continues to run until the board is reset.

10 Radio RX Test

This test issues a command to the Cypress SoC BT device to set the radio to camp on a specified frequency. While the test is running, the BT device periodically sends reports about received packets.

Usage: `wmbt radio_rx_test COMx <bd_addr> <frequency> <modulation_type> <logical_channel> <bb_packet_type> < packet_length>`

Where:

- bd_addr: `BD_ADDR` of Tx device (6 bytes)
- frequency: Frequency to listen to from 2402 MHz to 2480 MHz
- modulation_type: Sets the data pattern to compare received data
  0: 0x00 8-bit pattern

  1: 0xFF 8-bit pattern

  2: 0xAA 8-bit pattern

  3: 0xF0 8-bit pattern

  4: PRBS9 pattern
- logical_channel: Sets the logical channel to BR or EDR for ACL packets
  0: EDR

  1: BR
- bb_packet_type: Sets the packet type of the expected packets
  3: DM1

  4: DH1/2-DH1

  8: 3-DH1

  10: DM3/ 2-DH3

  11: DH3/3-DH3

  14: DM5/2-DH5

  15: DH5/3-DH5
- packet_length: 0 to 65535. The device compares the length of the received packets with the specified packet_length.

The Cypress SoC BT device generates a statistics report of the RX Test every 1 second when testing is performed.

The example below instructs the device to tune the receiver on 2402 MHz frequency. The test verifies that the 0xAA pattern is received using DM1 packet types (Basic Rate).

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt radio_rx_test COM23 112233445566
2402 2 1 3 17

MBT_BAUD_RATE:  3000000

TRANSPORT_MODE: 0 (HCI)

Opened COM23 at speed: 3000000

Sending HCI Command:

0000 < 01 52 FC 0E 66 55 44 33 22 11 E8 03 00 03 01 03 >

0010 < 11 00 >

Received HCI Event:

0000 < 04 0E 04 01 52 FC 00 >
```

```
Success
Radio RX Test is running. Press the Enter key to stop the test.
```

WMBT reports connectionless Rx Test statistics every second.

The example below shows the Rx Test statistics report:

```
Statistics Report received:
  [Rx Test statistics]
    Sync_Timeout_Count:     0x0
    HEC_Error_Count:        0x0
    Total_Received_Packets: 0x31f
    Good_Packets:           0x31f
    CRC_Error_Packets:      0x0
    Total_Received_Bits:    0x1a878
    Good_Bits:              0x1a878
    Error_Bits:             0x0
```

Press **Enter** to stop the test.

11 BQB RF Test

This test issues the commands necessary to configure the Cypress SoC BT device into a test mode for BQB RF testing using a Bluetooth tester, see BQB RF Test Setup [3].

Usage: `wmbt enable_bqb_test_mode COMx`

Before executing this command to configure the device for test mode, you must ensure you application does not have any timers running or any over the air BT activity enabled. For example, if advertisements are enabled or a periodic application timer is enabled, it may be possible to interfere with the BQB test results.

Example:

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt enable_bqb_test_mode COM23
MBT_BAUD_RATE:  3000000
TRANSPORT_MODE: 0 (HCI)

Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 05 0C 03 02 00 02 >
Received HCI Event:
0000 < 04 0E 04 01 05 0C 00 >
Success
Close Serial Bus
Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 1A 0C 01 03 >
Received HCI Event:
0000 < 04 0E 04 01 1A 0C 00 >
Success
Close Serial Bus
Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 03 18 00 >
Received HCI Event:
0000 < 04 0E 04 01 03 18 00 >
Success
Close Serial Bus
```

12 Read BD_ADDR Command

This command reads the BD_ADDR that is currently programmed in the DUT.

Usage: `wmbt read_bd_addr COMx`

Example:

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt read_bd_addr COM23

MBT_BAUD_RATE:  3000000
TRANSPORT_MODE: 0 (HCI)

Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 09 10 00 >
Received HCI Event:
0000 < 04 0E 0A 01 09 10 00 66 55 44 33 22 11 >

Success BD_ADDR = 112233445566

Close Serial Bus
```

13 Factory Commit BD_ADDR Command

This command writes the BD_ADDR to the Static Section (SS) area of flash.

To utilize this command, the BD_ADDR must initially be set to all FFs.

To set the initial BD_ADDR to all FFs, build and download an example application into the device with ModusToolbox command line make, including the BT_DEVICE_ADDRESS directive in your make command; for example:

```
make -f modus.mk BT_DEVICE_ADDRESS=FFFFFFFFFFFF program
```

Usage: `wmbt factory_commit_bd_addr COMx <bd_addr>`

Example:

```
<workspace>\wiced_btsdk\tools\btsdk-utils\wmbt\bin> wmbt factory_commit_bd_addr COM23
112233445566
MBT_BAUD_RATE:  3000000
TRANSPORT_MODE: 0 (HCI)

Opened COM23 at speed: 3000000
Sending HCI Command:
0000 < 01 10 FC 07 66 55 44 33 22 11 00 >
Received HCI Event:
0000 < 04 0E 04 01 10 FC 00 >
Success
Close Serial Bus
```

References

[1] Bluetooth Core Specification, Version 4.2 (see Bluetooth Core Specification 4.2)

[2] WICED HCI UART Control Protocol (002-16618)

[3] BQB RF Test Setup (002-15369)

Document History

Document Title: WICED Manufacturing Bluetooth Test Tool

Document Number: 002-14799

Revision	ECN	Submission Date	Description of Change
**	–	02/19/2016	Initial version
*A	5450962	09/27/2016	Updated in Cypress template
*B	5834940	07/27/2017	Updated logo and copyright
*C	5862775	08/23/2017	Updated template
*D	6306618	08/30/2018	Updated to reflect ModusToolbox Updated Sales and Copyright
*E	6486042	02/20/2019	Updated Factory Commit BD_ADDR Command Updated template
*F	6701132	10/15/2019	Updated for ModusToolbox 2.0 paths

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, 2016-2019. 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