# Objective

This Module demonstrates accessing the Excelon™-Ultra QSPI F-RAM™ by Renesas Controller (S7G2 group) using Renesas QSPI HAL Module Guide.

|  |
| --- |
|  |

# Overview

This module guide will enable you to effectively use a module in your own design to access Excelon F-RAM. This module guide is updated for F-RAM access using the Renesas QSPI HAL Module Guide as a reference. This module provides a code example that demonstrates accessing different features of the QSPI F-RAM.

The result is displayed by driving the status LED1, which turns green when the result is a pass, and turns red when the result is a fail.

# Requirements

**Tool:** [Renesas e2 studio Integrated Solution Development Environment (ISDE)](https://www.renesas.com/us/en/products/synergy/software/tools/e2-studio.html), [Renesas Synergy™ Software Package (SSP)](https://www.renesas.com/us/en/products/synergy/software/ssp.html)

**Associated Parts:** All S7G2 MCU Group

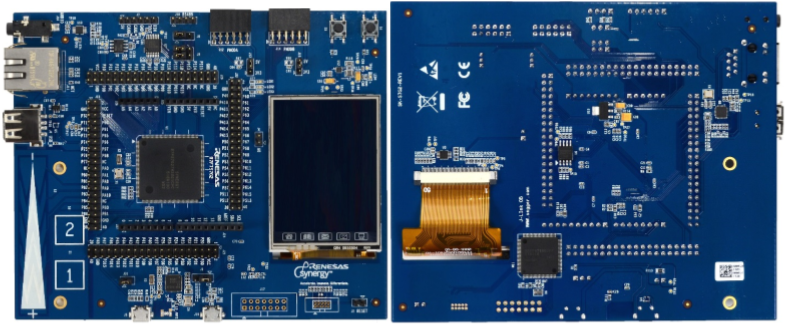
**Related Hardware:** [SK-S7G2 Starter Kit](https://www.renesas.com/us/en/products/synergy/hardware/kits/sk-s7g2.html)

**PC Requirements**: Minimum workstation requirements: Microsoft® Windows® 7 with Intel® Core™ family processor running at 2.0 GHz or higher (or equivalent processor), 8 GB memory, 250 GB hard disk or SSD, USB 2.0, Internet connection

# Hardware Setup

The SK-S7G2 includes one 64Mb (8MB) QSPI flash (W25Q64FVSSIG) connected to the QSPI interface of the microcontroller. The QSPI component (U6) located on the backside of the board is replaced with Cypress Excelon F-RAM CY15B104QSN-SXI) device to get the example code functional explained in next sections. Follow the below instructions, once the F-RAM is soldered on the SK-S7G2 board.

Replace QSPI Flash device with Cypress QSPI F-RAM



To power up the board, follow these steps:

1. Connect the Micro USB end of the supplied USB cable to the SK-S7G2 board J-19 connector (DEBUG\_USB).

NOTE: The kit contains a SEGGER J-Link On-board (OB). The J-Link provides full debug and programming capabilities for the SKS7G2 Kit as described in [quick start guide](https://www.renesas.com/us/en/doc/products/renesas-synergy/doc/r12qs0004eu0101_synergy_sk_s7g2.pdf) of SK-S7G2 Kit available on Renesas website.



1. Connect the other end of the USB cable to the USB port on your PC.

LED4 turns green, indicating a good connection.

# F-RAM QSPI HAL Module for Excelon QSPI F-RAM Access

This module guide will enable you to effectively use a module in your own design. On completion of this guide, you will be able to add this module to your own design, configure it correctly for the target application and write code, using the included application project code as a reference and an efficient starting point. The Quad SPI (QSPI) HAL module is a high-level API for writing the contents of a QSPI F-RAM device connected to the microcontroller and is implemented on the r\_qspi peripheral on the Synergy MCU. Unlike many other modules, there is no callback function for the QSPI.

## 1 F-RAM QSPI HAL Module Features

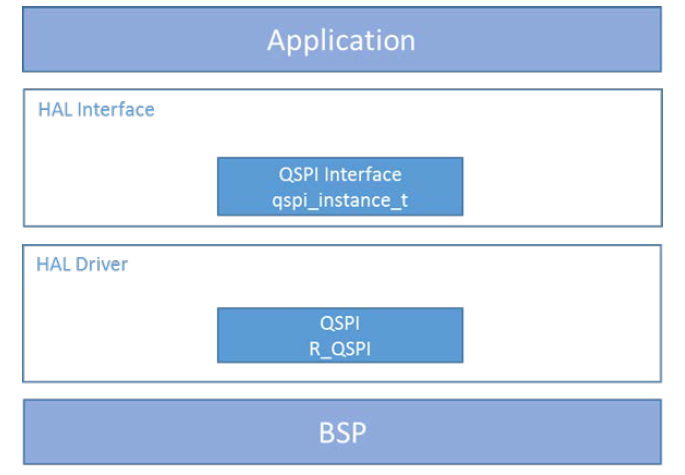
The QSPI HAL Module is used to initialize the QSPI peripheral that allows writing the contents of a QSPI F-RAM device connected to the microcontroller over the Quad SPI interface. Key features include:

• Access Quad SPI F-RAM devices using direct communication mode

• Read data from a QSPI F-RAM device

• Writes the data into QSPI F-RAM device

Figure 1: QSPI HAL Module Block Diagram



## 2 F-RAM QSPI HAL Module APIs Overview

The QSPI interface defines APIs for opening, closing, reading, and writing using the QSPI HAL module. A complete list of the available APIs, an example API call and a short description of each can be found in the following table. A table of status return values follows the API summary table.

Table 1: F-RAM QSPI HAL Module API Summary

|  |  |
| --- | --- |
| **Function Name** | **Example API Call Description** |
| .open | g\_qspi0.p\_api->open(g\_qspi0.p\_ctrl, g\_qspi0.p\_cfg);  Open the QSPI HAL module. |
| .close | g\_qspi0.p\_api->close(g\_qspi0.p\_ctrl);  Close the QSPI HAL module. |
| .read | g\_qspi0.p\_api->read(g\_qspi0.p\_ctrl, (uint8\_t \*)  QSPI\_DEVICE\_ADDRESS, readBuffer, BUFFER\_LENGTH);  Read data from the QSPI F-RAM. |
| .WriteSpi | g\_qspi0.p\_api->WriteSpi(g\_qspi0.p\_ctrl, (uint8\_t \*)  QSPI\_DEVICE\_ADDRESS, writeBuffer, BUFFER\_LENGTH);  Write data to the QSPI F-RAM. |
| .statusGet | g\_qspi0.p\_api->statusGet(g\_qspi0.p\_ctrl, &in\_progress);  Get the write status of the QSPI F-RAM. |
| .infoGet | g\_qspi0.p\_api->infoGet(g\_qspi0.p\_ctrl, &qspi\_info);  Provides information about the underlying QSPI F-RAM, as specified in  bsp\_qspi.c |
| .versionGet | g\_qspi0.p\_api->versionGet(& ssp\_version);  Retrieve the API version with the version pointer. |

**Note**: For more complete descriptions of operation and definitions for the function data structures, typedefs,

defines, API data, API structures and function variables, review the [SSP User’s Manual](C://Renesas/Synergy/e2studio_v7.3.0_ssp_v1.6.0/SSP_Documentation/ssp-user-manual-html-v1.01-sspv1.6.0/html/_page_s_s_p_overview.html) API

References for the associated module.

Table 2: Status Return Values

|  |  |
| --- | --- |
| Name | Description |
| SSP\_SUCCESS | API Call Successful. |
| SSP\_ERR\_INVALID\_ARGUMENT | Invalid parameter is passed. |
| SSP\_ERR\_ASSERTION | p\_cfg was NULL. |
| SSP\_ERR\_NOT\_OPEN | Driver is not opened. |
| SSP\_ERR\_UNSUPPORTED | Driver not able to query the following information from the F-RAM  manufacturer id, memory capacity and memory type. |

**Note**: Lower-level drivers may return common error codes. Refer to the SSP User’s Manual API References for the associated module for a definition of all relevant status return values.

## 3 F-RAM QSPI HAL Module Operational Overview

The F-RAM QSPI HAL module is used to initialize the QSPI peripheral so that the Synergy device can communicate (read, write data) with a QSPI serial F-RAM device.

The driver supports two operation modes: write and read.

* *WriteSpi* writes a bytes (number of bytes specified in code) of data to the QSPI F-RAM device. Use the infoGet

API to get the supported memory size for the underlying QSPI F-RAM device.

* + *Read* operation will read the data from the QSPI F-RAM and store it to the user-provided buffer.

**Note**: After any write operation and before starting the next operation, it is advisable to use the statusGet API to poll the status of operation; not doing this may corrupt user data.

## F-RAM QSPI HAL Module Operational Notes

In the case of using a board supported by the SSP, a BSP-based project (SK-S7G2) and the board having a QSPI memory device pre-installed, the BSP initializes and places the QSPI peripheral in ROM access mode with XIP (execute in place) enabled. This process enables the memory to be read like standard memory, meaning the QSPI HAL module is only be needed when writing the QSPI flash device.

The typical QSPI application writes data on the QSPI F-RAM device. When this driver is not open, the QSPI F-RAM device contents get mapped to 0x60000000 and can be read as if ordinary memory.

This driver has been tested on the S7G2 Synergy microcontroller groups using the QSPI peripheral block and the Cypress QSPI F-RAM device.

## Including F-RAM QSPI HAL Module in an Application

This section describes how to include the F-RAM QSPI HAL module in an application using the SSP configurator.

**Note**: The process assumes you are familiar with creating a project, adding threads adding a stack to a thread and configuring a block within the stack. If you are unfamiliar with any of these items, refer to the first few chapters of the [SSP User’s Manual](C://Renesas/Synergy/e2studio_v7.3.0_ssp_v1.6.0/SSP_Documentation/ssp-user-manual-html-v1.01-sspv1.6.0/html/_i_s_d_e_usage_notes.html) to learn how to manage each of these important steps in creating SSP-based applications.

To add the QSPI Driver to an application, add it to a thread using the stacks selection sequence given in the following

table. (The default name for the QSPI Driver g\_qspi0. This name can be changed in the associated Properties window).

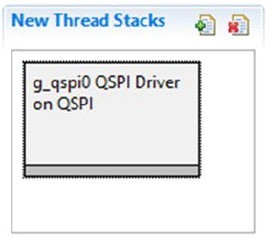
Table 3: QSPI HAL Module Selection Sequence

|  |  |  |
| --- | --- | --- |
| Resource | ISDE Tab | Stacks Selection Sequence |
| g\_qspi0 | Threads | Select HAL/Common and select New > Driver > Storage > QSPI HAL on QSPI |

When the QSPI HAL module on r\_qspi is added to the thread stack as shown in the following figure, the configurator

automatically adds any needed lower-level modules. Modules with a Gray band are individual modules that stand alone.

Figure 2: QSPI HAL Module Stack



## Configuring the QSPI HAL Module

No component options of the QSPI HAL module need to be changed. The driver does not use the QSPI interrupt, so it is not enabled. Consider changing only the name of the driver instance for easier source code development.

The SSP configuration window automatically highlights any selections, such as interrupts or operating modes, requiring configuration for lower-level modules to achieve successful operation. In most applications, the default values for modules in the lower layers of the stack can be accepted. The following tables detail the available options that can be specified in the Properties window.

Note: You may want to open your ISDE, create the module and explore the property settings in parallel with looking over the configuration settings in the following table. This helps to orient you and provides a useful ‘hands-on’ approach to learning the ins and outs of developing with SSP.

Table 4: Configuration Settings for the QSPI HAL Module on r\_qspi

|  |  |  |
| --- | --- | --- |
| **ISDE Property** | **Value** | **Description** |
| Parameter Checking | Default (BSP) | Parameter checking level. |
| Name | g\_qspi0 | Name of the QSPI HAL instance.  This may be edited to an application specific name if required. |

**Note**: The example values and defaults given for settings come from a project using the Synergy S7G2 MCU. Other MCUs may have different default values and available configuration settings.

### 5.1 QSPI HAL Module Pin Configuration

The QSPI peripheral module uses pins on the MCU to communicate to external devices. I/O pins must be selected and

configured as required by the external device. The following table indicates the method for selecting pins within the SSP

configuration window. The subsequent table lists an example selection sequence for the QSPI pins.

Table 5: Pin Selection Module Pin Configuration

|  |  |  |
| --- | --- | --- |
| **Resource** | **ISDE Tab** | **Pin selection Module** |
| QSPI | Pins | Select Peripherals > Storage: QSPI > QSPI0 |

**Note**: The selection sequence assumes the QSPI0 is the desired hardware target for the driver.

Table 6: Pin Configuration Settings for the QSPI Hal Module

|  |  |  |
| --- | --- | --- |
| **Property** | **Value** | **Description** |
| Pin Group Selection | * Mixed * \_A only | Pin group selection |
| Operation Mode | * Disabled * Custom * Single or Dual * Quad | Operation Mode |
| QSPCLK | None, P500 | QSPI clock output pin |
| QSSL | None, P501 | QSPI slave select pin |
| QIO0 | None, P502 | Data 0 input/output |
| QIO1 | None, P503 | Data 1 input/output |
| QIO2 | None, P504 | Data 2 input/output |
| QIO3 | None, P505 | Data 3 input/output |

**Note**: The example values come from a project using the Synergy S7G2 and the SK-S7G2 Kit. Other Synergy Kits and other Synergy MCUs may have different available pin configuration settings.

## Using the QSPI HAL Module in an Application

Once the module has been configured and the files generated, the QSPI is ready to be used in an application.

The typical steps in using the QSPI HAL module in an application are:

1. Initialize the QSPI HAL module using the open API call.
2. Read a block of data using the read API call.
3. write data using the WriteSpi API call.
   * 1. Use infoGet API to get the memory size supported by the underlying F-RAM.
     2. statusGet API can be used to poll the status of write operation.
4. Close the QSPI HAL module using the close API call.

## The QSPI HAL Module Application Project

The application project associated with this module guide demonstrates the steps used in a full design. The project can be found using the link in the References section at the end of this document. You can read over the code in hal\_entry.c file. The application project demonstrates the typical use of the QSPI HAL module APIs. The application project includes the following steps:

1. Turns off the three user LEDs of SK-S7G2: LED1, LED2 & LED3
2. Initializes and opens the QSPI HAL module.
3. Erases the first sector of QSPI F-RAM device.
4. Waits until the erase process completes.
5. Writes a test pattern to the QSPI F-RAM device.
6. Waits until the write process completes.
7. Reads data from the F-RAM device using the QSPI HAL module.
8. Closes the QSPI HAL module.
9. Compares the write and read buffers.
10. If write and read buffers are equal LED1 (green LED) toggles. If the buffers are unequal, indicating an error

condition, LED2 (red LED) toggles.

1. Waits for the push button (SW4) to be pressed by the user.
2. LED3 illuminates to indicate the next stage of the application.
3. Reads data from the flash device, via ROM access mode.
4. Compares write and read buffers.
5. If buffers are equal LED1 (green LED) toggles. If the buffers are unequal, indicating an error condition, LED2 (red

LED) toggles.

The first section of hal\_entry.c includes the auto-generated header file that references the QSPI instance structure. The next section has the definitions for LEDs and the pushbutton switch I/O ports. This section also includes the default buffer size for the read and write data buffers and the physical address location where the QSPI is mapped. Within the function hal\_entry.c, the variables are declared. Once the LEDs and data buffers initialize, the QSPI HAL module initializes with the open API call. The application code then makes API calls WriteSpi, status get and read and close. Once the instance is closed, read data is compared to the write data. If the data matches, then the Green LED toggles while checking the state of the pushbutton switch, SW4. If there is a data mismatch, the Red LED toggles. When pressed, the orange LED illuminates and the application again read the flash device contents. As the QPSI driver is closed, the QSPI peripheral accesses the QSPI flash via ROM access mode. The read data is compared with write data. If the data matches, then the Green LED toggles. If there is a data mismatch, the Red LED toggles.

## 8. Customizing the QSPI HAL Module for a Target Application

This module guide detailed a QSPI HAL module that is generic in its support of QSPI flash devices. A given board’s BSP is written to support a specific QSPI device. For example, SK-S7G2 boards use a Winbond W25Q64FVSSIG QSPI flash device.

No customization of the QPSI is possible using the Synergy Configuration IDSE. To target a different QSPI device, a new BSP and a new bsp\_qspi.c file must be created. The bsp\_qspi.c file contains code for device-specific QSPI commands; for example, taking the QSPI device in and out of ROM access mode. During operation, the QSPI HAL module uses generic QSPI commands to do QSPI operations.

Following files are updated for QSPI F-RAM Access.

1. hal\_entry.c: changed the module project name
2. qspi\_hal.c: Removed code related to flash device and updated for F-RAM device
3. qspi\_hal.h: changed the module project name
4. bsp\_qspi.h: Removed Flash aceess and added F-RAM access related variables
5. bsp\_qspi.c: Removed code related to flash device and updated for F-RAM device
6. r\_qspi.c: Removed code related to flash device and updated for F-RAM device
7. r\_qspi.h: Defined F-RAM device ID
8. r\_qspi\_api.h: Removed Flash aceess and added F-RAM access related API
9. r\_qspi\_private\_api.h: Removed Flash aceess and added F-RAM access related API

## Running the QSPI HAL Module Application Project

To run the QSPI HAL module application project and to see it executed on a target kit, simply import it into your ISDE, compile and run debug.

To implement the QSPI HAL module application in a new project, follow the steps given in this section to define, configure, auto-generate files, add code, compile and debug the application on the target kit. Following these steps gives you a hand-on approach that helps make the development process with SSP more practical, while just reading over this guide tends to be more theoretical.

Note: The steps are in sufficient detail for someone experienced with the basic flow through the Synergy development

process. If these steps are not familiar, refer to the first few chapters of the [SSP User’s Manual](C://Renesas/Synergy/e2studio_v7.3.0_ssp_v1.6.0/SSP_Documentation/ssp-user-manual-html-v1.01-sspv1.6.0/html/_i_s_d_e_usage_notes.html) for a description of how to accomplish these steps.

1. Create a new Renesas Synergy C project for the SK-S7G2 board called “S7\_SK\_QSPI”.

2. Select the S7G2-SK BSP project template and create project.

3. Open Configuration.xml from generated project and select the Thread tab.

4. Add the QSPI HAL Module to HAL/Common thread from New Stack > Driver > Storage > QSPI HAL on QSPI

5. Click the Generate Project Content button.

6. Add code from the supplied below project file or copy the file over the generated files and save them

1. hal\_entry.c
2. qspi\_hal.c
3. qspi\_hal.h
4. bsp\_qspi.h
5. bsp\_qspi.c
6. r\_qspi.c
7. r\_qspi.h
8. r\_qspi\_api.h
9. r\_qspi\_private\_api.h

7. **Once files are saved, make sure that these files are made read only.** If not, code will be replaced by default supported BSP code which is written for QSPI Flash Device.

8. Build the project after making files mentioned in the step 6 as read only.

7. Connect to the host PC via a micro USB cable to J19 on SK-S7G2.

8. Start to debug the application.

9. The green LED should start blinking.

# 10 Reference Documents

1. QSPI F-RAM Datasheet: [CY15B104QSN/CY15V104QSN, Excelon™-Ultra 4-Mbit (512K × 8) Quad SPI F-RAM](https://www.cypress.com/file/400726/download)
2. [Renesas QSPI HAL Module Guide – Application Project](https://www.renesas.com/us/en/software/D6001438.html)
3. [Renesas SSP User Manual](C://Renesas/Synergy/e2studio_v7.3.0_ssp_v1.6.0/SSP_Documentation/ssp-user-manual-html-v1.01-sspv1.6.0/html/_i_s_d_e_usage_notes.html)
4. [SK-S7G2 Starter Kit Documentation](https://www.renesas.com/us/en/products/synergy/hardware/kits/sk-s7g2.html)

Document History

Document Title: Excelon-Ultra QSPI F-RAM Access Using Renesas QSPI HAL Driver Module

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Orig. of Change | Submission Date | Description of Change |
| \*\* | GVCH | 07/16/2019 | New Code Example |

# 

# 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](http://www.cypress.com/?id=1062).

[Products](http://www.cypress.com/products)

|  |  |
| --- | --- |
| Arm® Cortex® Microcontrollers | [cypress.com/arm](http://www.cypress.com/products/32-bit-arm-cortex-mcus) |
| Automotive | [cypress.com/automotive](http://www.cypress.com/applications/automotive-solutions) |
| Clocks & Buffers | [cypress.com/clocks](http://www.cypress.com/products/clocks-buffers) |
| Interface | [cypress.com/interface](http://www.cypress.com/products/interface) |
| Internet of Things | [cypress.com/iot](http://www.cypress.com/internet-things-iot) |
| Memory | [cypress.com/memory](http://www.cypress.com/products/memory-products) |
| Microcontrollers | [cypress.com/mcu](http://www.cypress.com/mcu) |
| PSoC | [cypress.com/psoc](http://www.cypress.com/psoc/) |
| Power Management ICs | [cypress.com/pmic](http://www.cypress.com/products/power-management) |
| Touch Sensing | [cypress.com/touch](http://www.cypress.com/products/touch-sensing) |
| USB Controllers | [cypress.com/usb](http://www.cypress.com/products/usb-controllers) |
| Wireless Connectivity | [cypress.com/wireless](http://www.cypress.com/products/wireless-connectivity) |

[PSoC](http://www.cypress.com/psoc)[®](http://www.cypress.com/psoc) [Solutions](http://www.cypress.com/psoc)

[PSoC 1](http://www.cypress.com/products/psoc-1) | [PSoC 3](http://www.cypress.com/products/psoc-3) | [PSoC 4](http://www.cypress.com/products/psoc-4) | [PSoC 5LP](http://www.cypress.com/products/32-bit-arm-cortex-m3-psoc-5lp) | [PSoC 6 MCU](http://cypress.com/psoc6)

# [Cypress Developer Community](http://www.cypress.com/cdc)

[Community](https://community.cypress.com/welcome) | [Projects](http://www.cypress.com/projects) | [Video](http://www.cypress.com/video-library)s | [Blogs](http://www.cypress.com/blog) | [Training](http://www.cypress.com/training) | [Components](http://www.cypress.com/cdc/community-components)

# [Technical Support](http://www.cypress.com/support)

[cypress.com/support](http://www.cypress.com/support)

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

|  |  |
| --- | --- |
| C:\Work\Templates\Cypress Logo Files\Cypress Logo_New_White (Reverse)_PNG.png | 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, including Spansion LLC (“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 does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. 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. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, 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 or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

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.