

## Objective

This example shows how to use the PSoC® Creator™ Serial Communication Block (SCB) Component configured in SPI mode as a slave in a PSoC 4 device. This example demonstrates polling, interrupt, and DMA methods.

## Requirements

**Tool:** PSoC Creator 4.2

**Programming Language:** C (Arm® GCC 5.4.1)

**Associated Parts:** PSoC 4 family; DMA example works only with PSoC 4 devices with DMA.

**Related Hardware:**

- Polling and Interrupt projects: [CY8CKIT-042 PSoC 4 Pioneer Kit](#)
- DMA project: [CY8CKIT-044 PSoC 4 M-Series Pioneer Kit](#)

## Overview

This example contains three projects that utilize an SPI Component configured as a slave. The projects demonstrate polling, interrupt, and DMA methods for SPI data communication. Each project receives a command from a SPI master to change the color of an LED and sends a status indicator to the master. A second kit and the code example [CE224339 – PSoC 4 SPI Master](#) are recommended.

## Hardware Setup

This code example requires two kits. A kit listed in [Related Hardware](#) runs this slave example; the second kit needs the code example [CE224339 – PSoC 4 SPI Master](#).

[Table 1](#) shows how to connect the pins to the master corresponding pins:

Table 1. Pin Connections to SPI Slave Kit

	MISO	MOSI	SCLK	SS	Ground
<b>Polling / Interrupt: CY8CKIT-42 kit</b>	P3[1]	P3[0]	P0[6]	P0[7]	GND
<b>DMA: CY8CKIT-044 kit</b>	P2[1]	P2[0]	P2[2]	P0[7]	GND

## Software Setup

None.

## Operation

1. Connect two PSoC 4 kits using the instructions in the [Hardware Setup](#) section.
2. Open both the master and slave projects. Multiple instances of PSoC Creator can run at once.
3. Connect the slave PSoC 4 kit to the computer using USB.
4. Make sure that the required project is set as the active project; right-click the project and select **Set As Active Project**. Note that any slave project works with any master project.
5. Build the project that corresponds to that specific board and program it into the PSoC 4 device. Choose **Debug > Program**. For more information on the device programming, see PSoC Creator Help.
6. Program the master kit; see the code example [CE224339 – PSoC 4 SPI Master](#).

7. Press the reset buttons on both boards simultaneously. Confirm that the slave kit LED cycles through colors in the order of red, yellow, green, cyan, blue, and all OFF.

## Design and Implementation

The master sends data to the slave in packets. In this code example, three bytes are sent and received simultaneously. The packet consists of a start byte, a data byte, and an end byte. The start and end bytes are checked, and if the bytes are correct, the data byte is assumed to be correct.

### SPI Slave Polling

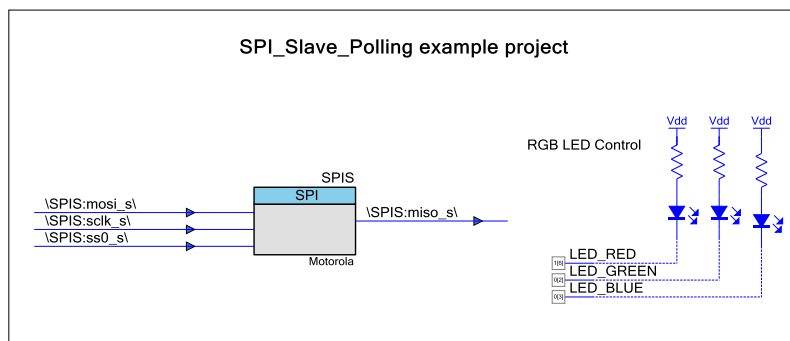
Note that in the SPI protocol, data is sent simultaneously in both directions: master to slave and slave to master.

In the SPI\_Slave\_Polling example, the following functions are performed:

1. The SCB Component is configured as an SPI slave.
2. The SPI slave:
  - a. Waits for a command from the SPI master; it sends a dummy data while master bytes are received.
  - b. Sets the LED to the color in the command data byte.
  - c. Updates the status and loads it into the Tx buffer to be sent to the master.
  - d. Waits until it receives the dummy data from the master so that it can send what is currently in the Tx buffer.

Figure 1 shows the top-level design of the PSoC Creator project:

Figure 1. SPI\_Slave\_Polling Top Design Schematic



### SPI Slave Interrupt

Note that in the SPI protocol data is sent simultaneously in both directions: master to slave and slave to master.

In the SPI\_Slave\_Interrupt example, the following functions are performed:

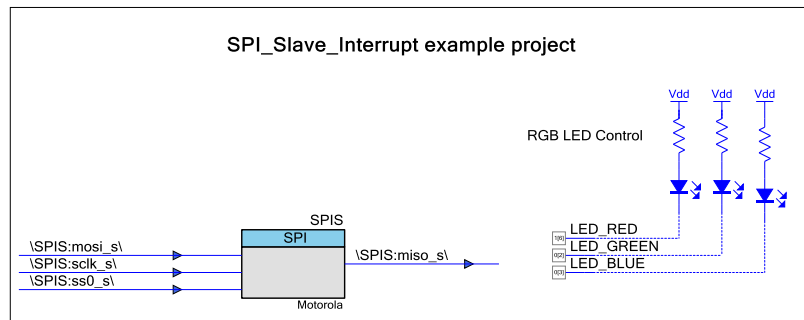
1. The SCB Component is configured as an SPI slave.
2. An interrupt handler function `Slave_ISR()` sets a global variable flag every time it is called. See [Slave ISR traits](#).
3. When the flag is set by the ISR, the flag is reset in main and the SPI slave:
  - a. Reads the data from the Rx buffer.
  - b. Sets the LED to the color in the command data byte.
  - c. Updates the status and loads it into the Tx buffer to be sent to the master.
  - d. Waits until it receives dummy data from the master so that it can send what is currently in the Tx buffer.

The interrupt is triggered by Rx FIFO not empty, that is, when data has been received. `Slave_ISR()` does the following:

1. Sets the flag variable.
2. Clears the interrupt source.
3. Reads all data in the Rx buffer into an array.

Figure 2 shows the top-level design of the PSoC Creator project:

Figure 2. SPI\_Slave\_Interrupt Top Design Schematic



## SPI Slave DMA

Note that in the SPI protocol, data is sent simultaneously in both directions: master to slave and slave to master.

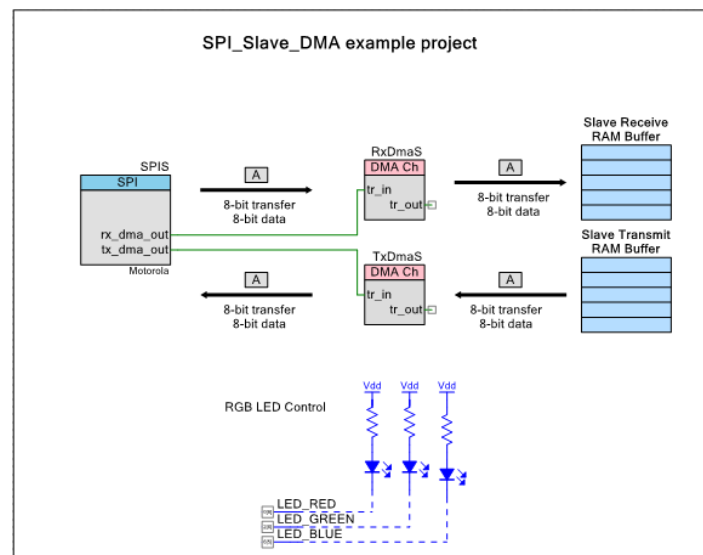
In the SPI\_Slave\_DMA example, the following functions are performed:

1. The SCB Component is configured as an SPI slave.
2. The DMA transmit and receive channels are initialized and receive is enabled.
3. Arrays containing the LED color command are initialized as source buffers for the DMA.
4. A DMA trigger transfers the command array to the SPI Tx buffer. DMA triggers whenever there is data in the SPI Rx buffer or whenever the SPI Tx buffer is empty.
5. Load the first byte of data directly into the Tx buffer, not utilizing the DMA.
6. The DMA transfers all data between an array and the buffers. The received command gets stored in an array connected to the DMA. The Tx buffer data is stored in another array that can be filled to send the command status.
7. Sets the LED to the color in the received command data byte.

**Note:** The first byte transmitted from the slave must be pre-loaded directly into the Tx buffer, typically by firmware. This is because the slave never knows when it needs to send data, and by the time the DMA can load bytes into the buffer, the first bit has already been sent. Therefore, when looking at Figure 8, the number of data elements to transfer is one less than the received data elements.

Figure 3 shows the top-level design of the PSoC Creator project:

Figure 3. SPI\_Slave\_DMA Top Design Schematic



## Components and Settings

Table 2 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 2. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
SPI (SCB)	SPIS	Handles SPI communication	See Figure 4 for all projects. Interrupt project: See Figure 5 for additional configuration. DMA project: See Figure 6 for additional configuration.
DMA	RxDmaM	Handles DMA Rx data transfer – slave to master	See Figure 7
	TxDmaM	Handles DMA Tx data transfer – master to slave	See Figure 8

Figure 4. SPI Basic Tab Configuration for All Projects

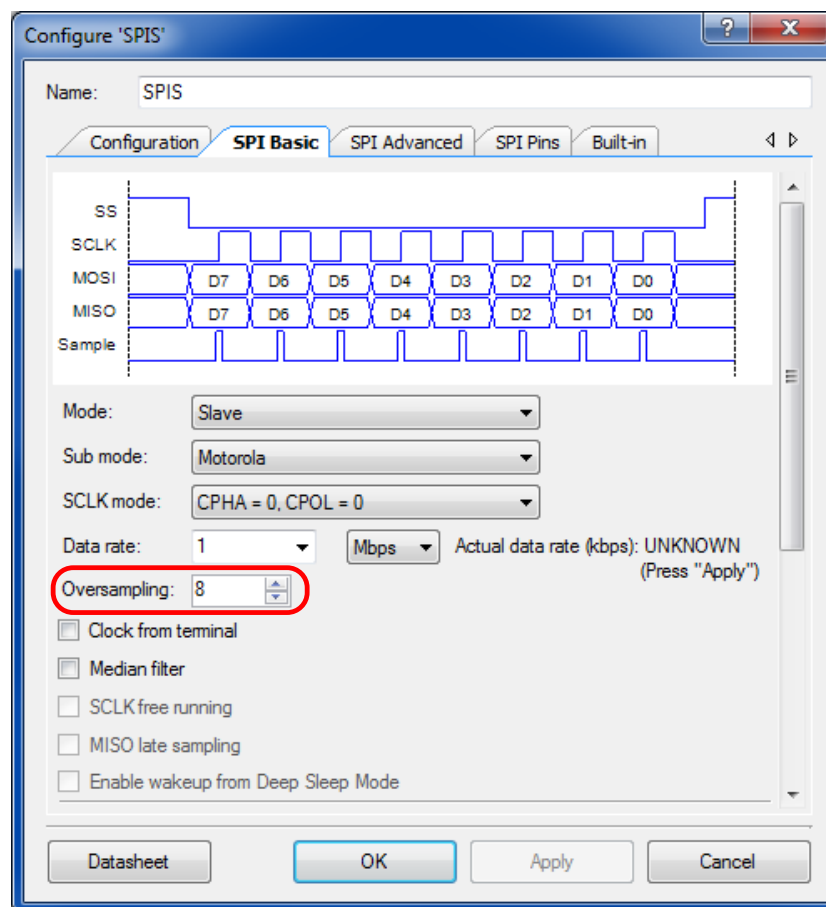


Figure 5. SPI Advanced Tab for Interrupt Project

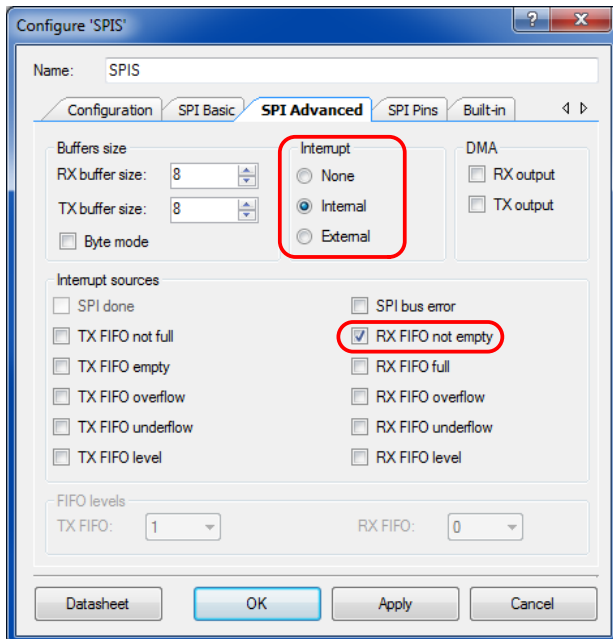


Figure 6. SPI DMA Settings

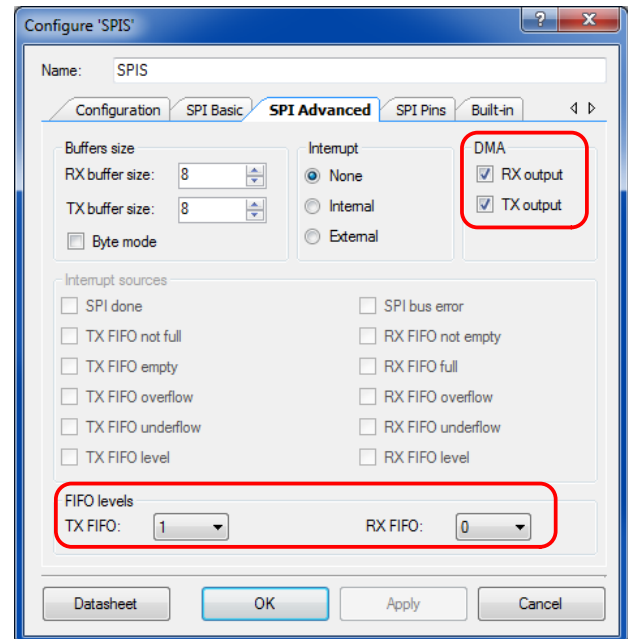


Figure 7. RxDmaS Configuration Settings

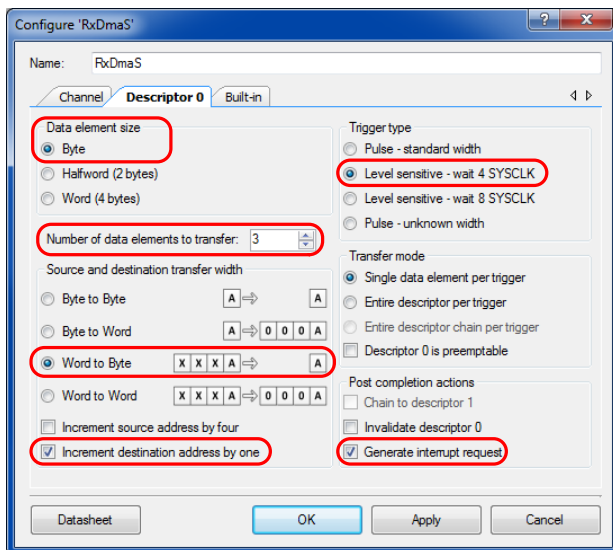
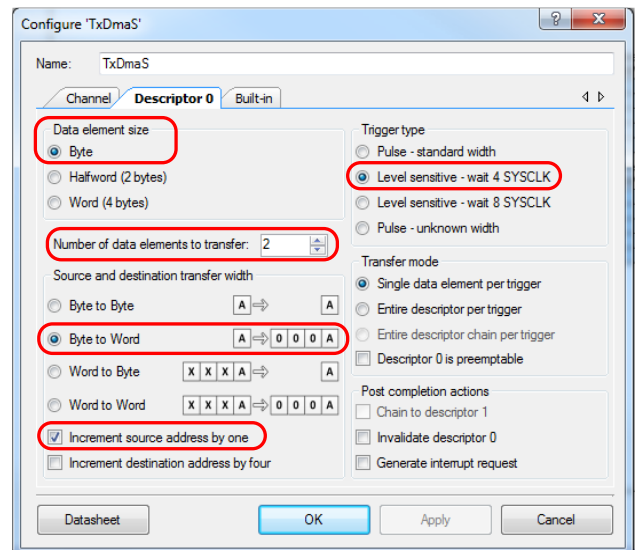


Figure 8. TxDmaS Configuration Settings



For information on the hardware resources used by a Component, see the Component datasheet.

## Reusing This Example

This example can be ported to various PSoC 4 devices, kits, or both. Before porting note that:

- The master and slave data rates must be the same.
- The SCB blocks may not have the same pinouts on every device; this means that some wires may need to be moved. See the pin layout tab in PSoC Creator.
- The DMA project works only with PSoC 4 devices with DMA. To see a list of available devices with DMA, in PSoC Creator, select **File > New > Project > Target Device > PSoC 4**, and then choose **Launch device selector** in the second drop-down menu. Find **DMA channels** in the filter and deselect **none**. This will give a list of all PSoC 4 devices with DMA.

To port the code to a new device, in PSoC Creator, select **Project > Device Selector** and change to the target device.

## Related Documents

For a comprehensive list of PSoC 6 MCU resources, see [KBA223067](#) in the Cypress community.

For a comprehensive list of PSoC 3, PSoC 4, and PSoC 5LP resources, see [KBA86521](#) in the Cypress community.

Application Notes	
<a href="#">AN79953 – Getting Started with PSoC® 4</a>	Describes PSoC 4 devices and how to build your first PSoC Creator project
Code Examples	
<a href="#">CE224339 – PSoC 4 SPI Master</a>	Companion code example for this code example.
PSoC Creator Component Datasheets	
<a href="#">SCB</a>	Supports I <sup>2</sup> C, SPI, UART and EZI2C communication protocols using the serial control block (SCB)
<a href="#">DMA</a>	Supports direct memory access (DMA) controllers
Device Documentation	
<a href="#">PSoC 4 Datasheets</a>	<a href="#">PSoC 4 Technical Reference Manuals</a>
Development Kit Documentation	
<a href="#">CY8CKIT-042 PSoC 4 Pioneer Kit</a>	
<a href="#">CY8CKIT-044 PSoC 4 M-Series Pioneer Kit</a>	
<a href="#">PSoC 4 Kits</a>	
Tool Documentation	
<a href="#">PSoC Creator</a>	Look in the <b>Downloads</b> tab for Quick Start and User Guides

## Document History

Document Title: CE224463 – PSoC 4 SPI Slave

Document Number: 002-24463

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6260029	NRSH	07/24/2018	New code example
*A	6671851	NRSH	09/13/2019	Update to new code example template and updated Figure 3.

## 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	<a href="http://cypress.com/arm">cypress.com/arm</a>
Automotive	<a href="http://cypress.com/automotive">cypress.com/automotive</a>
Clocks & Buffers	<a href="http://cypress.com/clocks">cypress.com/clocks</a>
Interface	<a href="http://cypress.com/interface">cypress.com/interface</a>
Internet of Things	<a href="http://cypress.com/iot">cypress.com/iot</a>
Memory	<a href="http://cypress.com/memory">cypress.com/memory</a>
Microcontrollers	<a href="http://cypress.com/mcu">cypress.com/mcu</a>
PSoC	<a href="http://cypress.com/psoc">cypress.com/psoc</a>
Power Management ICs	<a href="http://cypress.com/pmic">cypress.com/pmic</a>
Touch Sensing	<a href="http://cypress.com/touch">cypress.com/touch</a>
USB Controllers	<a href="http://cypress.com/usb">cypress.com/usb</a>
Wireless Connectivity	<a href="http://cypress.com/wireless">cypress.com/wireless</a>

### PSoC® Solutions

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6 MCU](#)

### Cypress Developer Community

[Community](#) | [Projects](#) | [Videos](#) | [Blogs](#) | [Training](#) | [Components](#)

### Technical Support

[cypress.com/support](http://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, 2018-2019. 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](http://cypress.com). Other names and brands may be claimed as property of their respective owners.