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.

CE218553 – PSoC 6 MCU: PWM Triggering a DMA Channel

Objective

This code examples demonstrates how to route trigger signals on a PSoC 6 device. The trigger signal, in this code example, is routed from the PWM to the DMA, using PSoC Creator's auto routing feature.

Overview

This code example demonstrates how to route trigger signals in PSoC® 6. In this code example, PSoC Creator is used to configure the trigger multiplexer. This is demonstrated using a PWM trigger routed to a DMA channel. The PWM is connected to an LED to implement a variable intensity. The PWM also triggers the DMA in every cycle. The DMA is used to update the PWM duty cycle to create a breathing effect on the LED.

Requirements

Tool: PSoC Creator 4.2

Programming Language: C (ARM)

Associated Parts: All PSoC 6 MCU parts

Related Hardware: CY8CKIT-062 PSoC 6 BLE Pioneer Kit

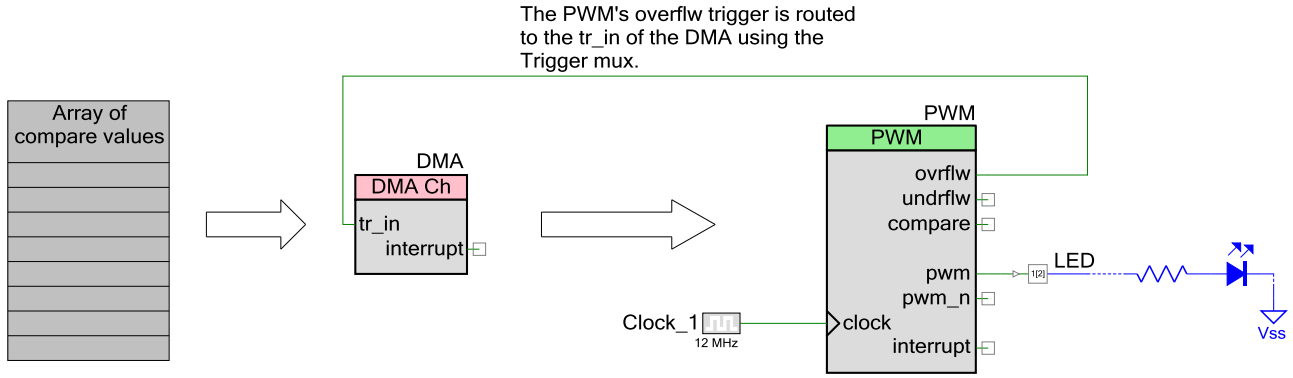
Design

The PSoC 6 device has many digital signals generated from different peripheral blocks. Many of these signals would need to be routed to other peripheral blocks as triggers to some events there. Trigger multiplexers are simple multiplexers that are designed to route these signals from potential source peripherals to destinations. This example demonstrates setting up a trigger route from the PWM to the DMA.

The design implements a PWM that has a duty cycle that is updated on every terminal count of the PWM through a DMA. There is a preset list of compare values in an array. These preset compare values are in a format that, when updated as the compare value of the PWM in every cycle, will generate a breathing pattern. The array with compare values is the source of data transfer for the DMA; the destination is the PWM's compare register. The DMA is triggered using the PWM's terminal count signal. The routing of the terminal count signal to the trigger input of the DMA is accomplished using the trigger multiplexer.

You need to connect the PWM overflow trigger (ovrflw) to the DMA input trigger (tr_in). The PSoC Creator build process takes care of generating the code responsible for routing the trigger signal. See Appendix A: Trigger Multiplexer Routing in PSoC Creator for more details on trigger multiplexer routing. Figure 1 shows the PSoC Creator project schematic.

Figure 1. PSoC Creator Project Schematic



Operation

1. Program the PSoC 6 device on CY8CKIT-062.
2. Observe the breathing pattern on the red LED in the kit.

   Note: In case of using any hardware other than CY8CKIT-062, use wires to connect the pin P0[3] to an LED.

Components

Table 1 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each.

Table 1. PSoC Creator Components

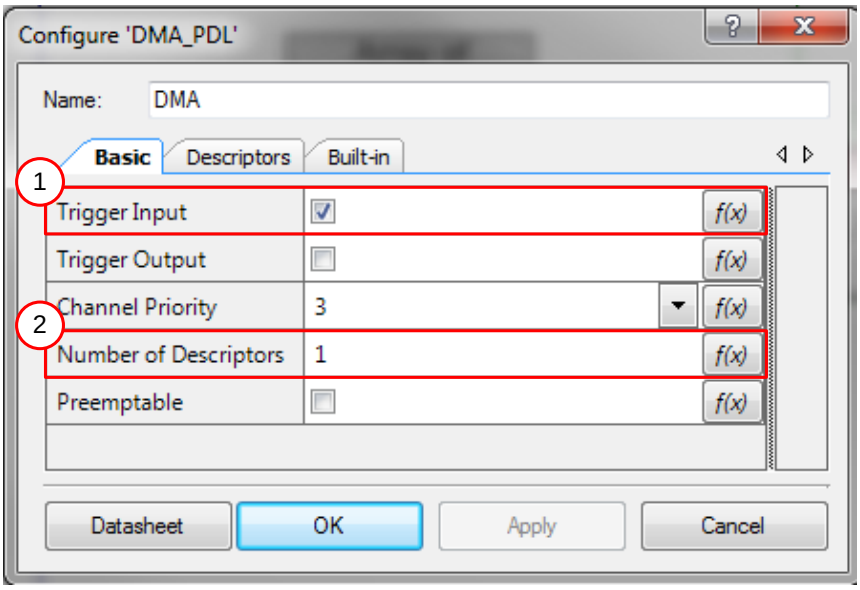
Component	Instance Name	Hardware Resources
DMA	DMA	1 DMA Channel
PWM	PWM	1 TCPWM block

Parameter Settings

DMA Configuration

A DMA channel is used to transfer the compare values from the array to the PWM compare register. Figure 2 shows the DMA Component's basic configuration.

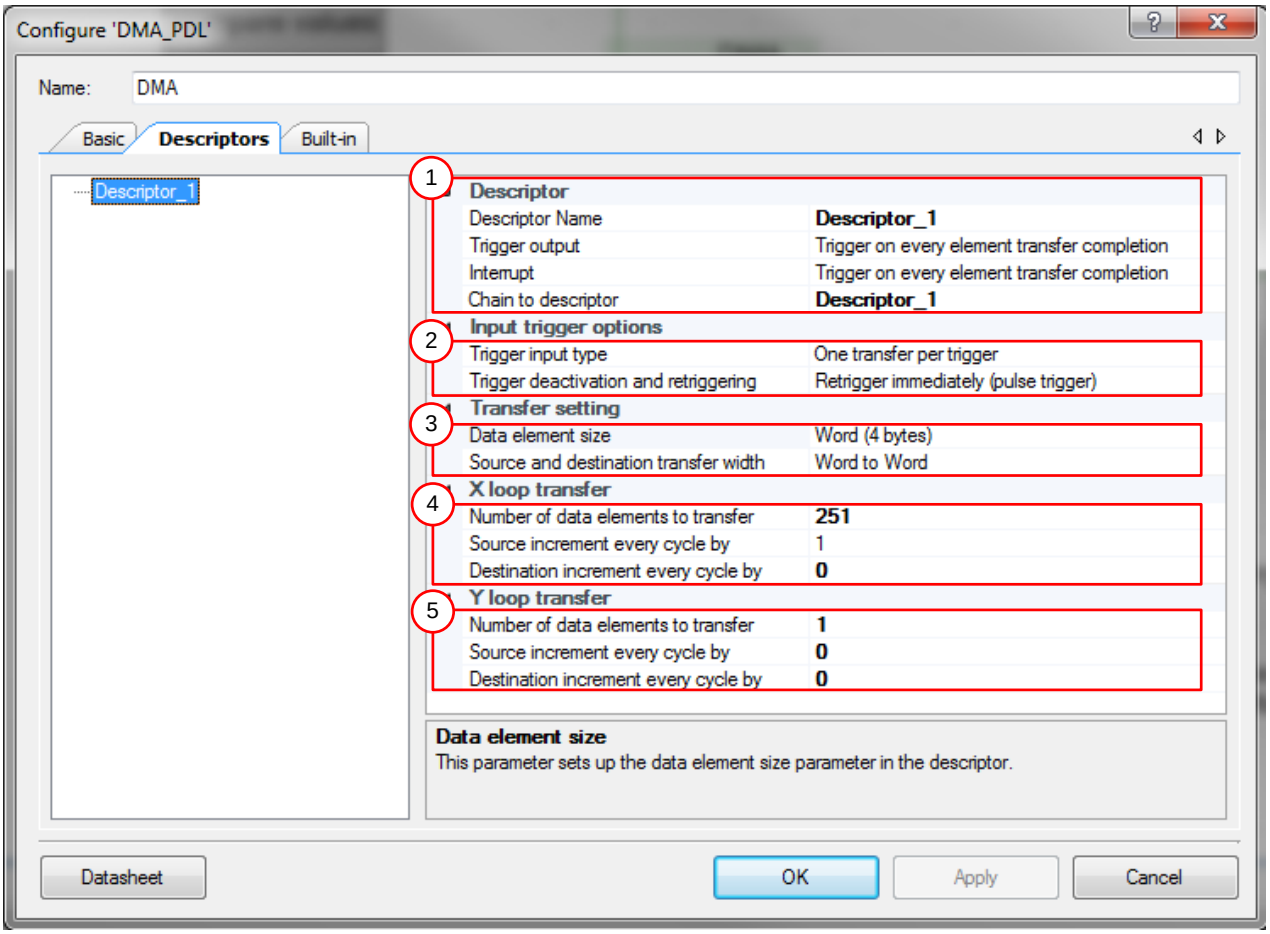
Figure 2. DMA Basic Configuration



1. The **trigger input** to the DMA is enabled so that a trigger signal routed through the trigger multiplexer block could be used to trigger the DMA channel. Enabling the trigger input provides a terminal on the DMA Component with a connection to a signal in the schematic.
2. The DMA is implementing a simple array to single register transfer, which can be achieved by using a single descriptor. The **number of descriptors** is set to 1.

Figure 3 shows the configuration of the DMA descriptor.

Figure 3. DMA Descriptor Configuration



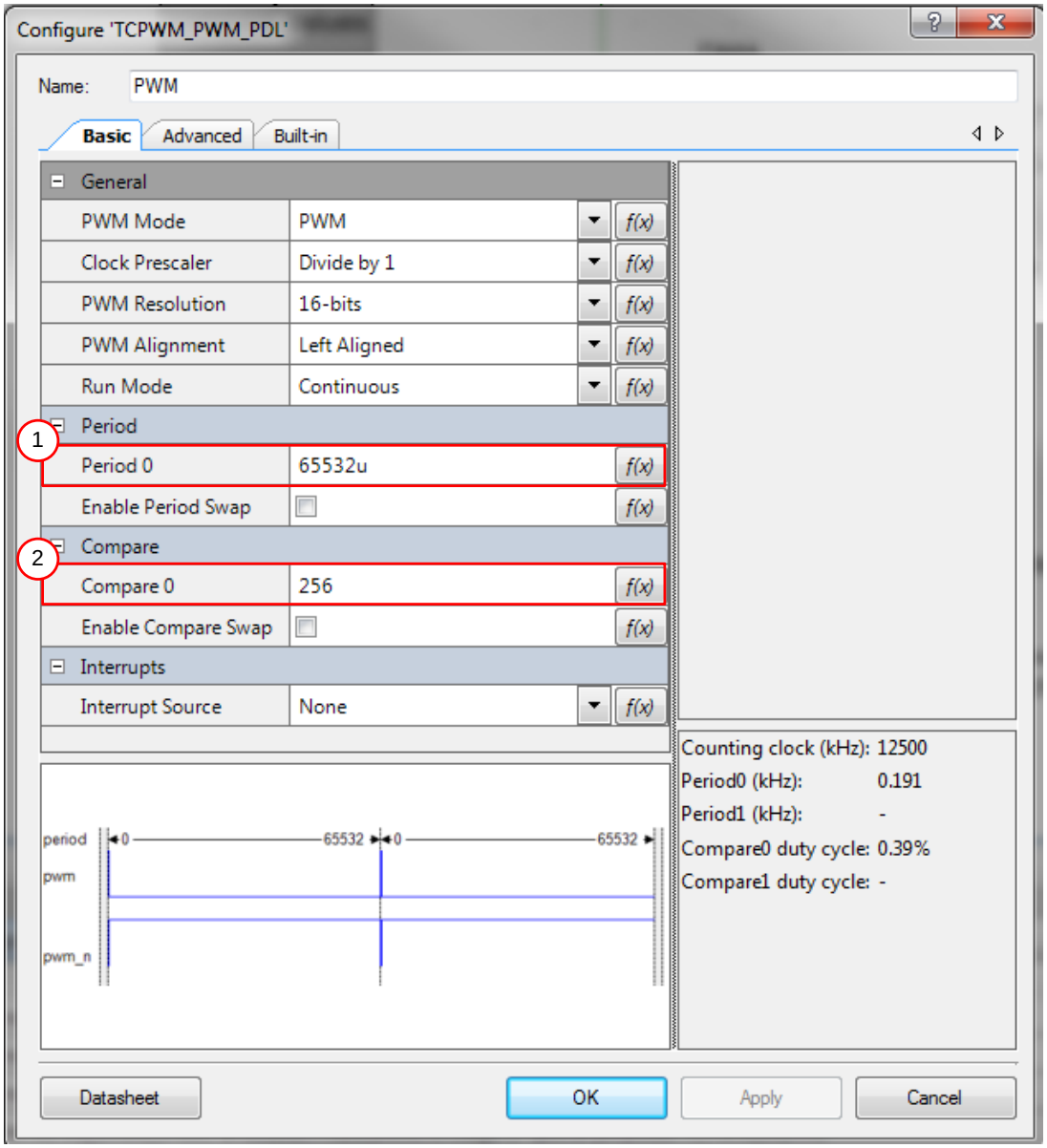
There is only one descriptor used for this DMA transfer. The descriptor configuration determines the characteristics of the DMA transfer.

1. The descriptor settings set up the name for the descriptor. These configurations also set the nature of the **trigger output** and **interrupt** outputs. These settings are left at their default settings because the trigger output or interrupt output are not used in this code example. The **chain to descriptor** is set as Descriptor_1, which will make the DMA execute the same descriptor in a loop.
2. The input trigger options determine how the descriptor handles trigger inputs. In this code example, the **trigger input type** is set to 'one transfer per trigger' because a single transfer from the array to compare register is required on every trigger generated by the terminal count.
3. Transfer settings are left at their default values. The **data size** is word (4 bytes) and the **source and destination transfer widths** are word-to-word because both the array and the compare register are 32-bit wide.
4. The X loop transfer setting sets up the x loop for the descriptor. The DMA can set up two nested loops of transfer; x loop is the inner loop of transfer. See the DMA Component datasheet for details. This descriptor transfers a 251-element array to a single compare register over 251 triggers. Therefore, the **number of data elements to transfer** is set as 251. Because the data source is an array, the address needs to be incremented after each transfer. Therefore, the **source increment every cycle by** 1. Because the destination is a constant compare register, **destination increment every cycle by** 0.
5. The Y loop is not utilized in this code example. Therefore, the number of data elements to transfer is set as 1. Both source and destination increments are set to 0.

PWM Configuration

The PWM configuration is shown in Figure 4.

Figure 4. PWM Configuration



Most of the PWM configurations except the period and compare values are left at their default values.

1. The period value of the PWM is set to 65532. The PWM is clocked by a 12-MHz clock and the terminal count is used to trigger the DMA. This period value (65532) makes sure that the terminal count and consequently the DMA trigger occurs at a frequency of 12/65532 MHz. The breathing rate of the LED will be determined by this rate of DMA trigger multiplied by the number of elements in the array of compare values.
2. The Compare value is set to 256 here. However, the compare value is updated later using this DMA. Therefore, this value starts the PWM at a very small duty cycle.

Related Documents

Table 2 lists the relevant application notes, code examples, Component datasheets, and device and DVK documentation.

Table 2. Related Documents

Application Notes	
AN210781: Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	This application note introduces the PSoC 6 BLE device
Code Examples	
CE218553- PSoC 6 MCU- PWM triggering a DMA channel	This code example is a simple case of a PWM triggering a DMA channel.
PSoC Creator Component Datasheets	
TCPWM	PULSE WIDTH MODULATOR (TCPWM_PWM_PDL)
DMA	DIRECT MEMORY ACCESS (DMA_PDL)
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual (TRM)	PSoC 6 MCU with BLE Architecture Technical Reference Manual
PSoC 6 MCU: PSoC 63 with BLE Register Technical Reference Manual	PSoC 6 MCU with BLE Register Technical Reference Manual
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 63 with BLE Datasheet
PSoC 6 MCU: PSoC 62 Datasheet	PSoC 62 Datasheet
Development Kit (DVK) Documentation	
CY8CKIT-062-BLE: PSoC 6 BLE Pioneer Kit	

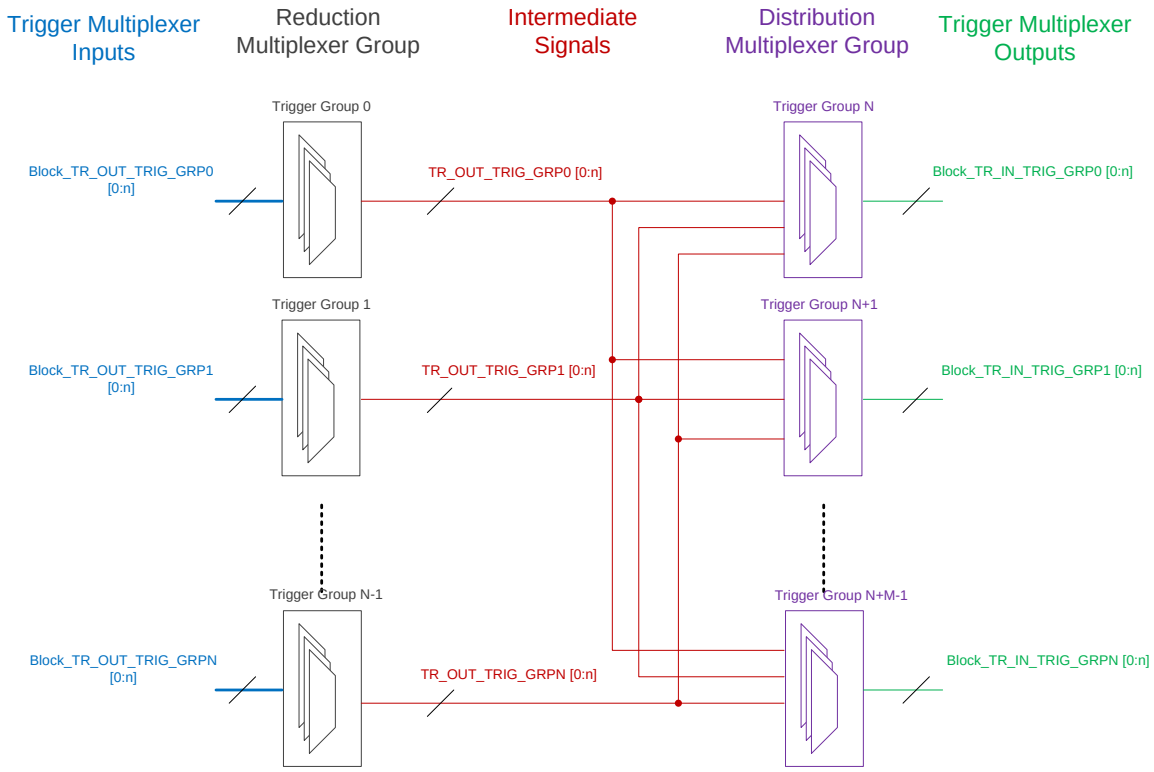
Appendix A: Trigger Multiplexer Routing in PSoC Creator

The PSoC 6 device has many digital signals generated from different peripheral blocks. Many of these signals would need to be routed to other peripheral blocks as triggers to some events there. Trigger multiplexers are simple multiplexers that are designed to route these signals from potential source peripherals to destinations.

The trigger multiplexer block is responsible for the entire trigger routing in the device. The trigger multiplexer block is implemented by using multiple trigger multiplexers. Trigger multiplexers are grouped into trigger multiplexer groups. There are multiple trigger multiplexer groups in a device, which combine to form the trigger multiplexer block. The trigger multiplexer block is architected into two layers. Each layer is formed by separate set of multiple trigger groups.

On the input side are the reduction trigger multiplexers and the output side are the distribution trigger multiplexers. A trigger signal route involves two multiplexer connections. First, the trigger multiplexer inputs are routed to intermediate signals using reduction multiplexers and then the intermediate signals are routed to relevant trigger multiplexer outputs using the distribution multiplexers.

Figure 5. Trigger Multiplexer Block Architecture



The routing of trigger multiplexers is achieved by configuring two trigger multiplexers per path. PSoC Creator's build process automatically generates the code required for trigger multiplexer routing. This trigger multiplexer routing code is in cyfitter_cfg.c, as a part of the CySystemInit (). The following code snippet is from the Cy_SystemInit() function of this code example and it shows the two Cy_TrigMux_Connect() calls to connect the two trigger multiplexers needed for the route.

```
/* Perform Trigger Mux configuration */
Cy_TrigMux_Connect(TRIG11_IN_TCPWM0_TR_OVERFLOW4, TRIG11_OUT_TR_GROUP0_INPUT15,
CY_TR_MUX_TR_INV_DISABLE, TRIGGER_TYPE_TCPWM_TR_OVERFLOW);

Cy_TrigMux_Connect(TRIG0_IN_TR_GROUP11_OUTPUT7, TRIG0_OUT_CPUSS_DW0_TR_IN1,
CY_TR_MUX_TR_INV_DISABLE, TRIGGER_TYPE_TR_GROUP_OUTPUT__LEVEL);
```

The first `Cy_TrigMux_Connect()` connects the TCPWM's overflow signal through a reduction multiplexer to an intermediate signal called "trigger 11" of trigger group 0. The second `Cy_TrigMux_Connect()` connects this intermediate trigger signal through a distribution multiplexer to the DMA's trigger input line.

For more information on the trigger multiplexer routing and architecture, see the Technical Reference Manual.

Document History

Document Title: CE218553 - PSoC 6 MCU: PWM Triggering a DMA Channel

Document Number: 002-18553

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5780847	QVS	06/21/2017	New code example
*A	5857219	QVS	8/17/2017	Updated for PSoC Creator 4.2

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

Cypress Developer Community

Forums | WICED IOT Forums | 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, 2017. 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. 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.