CE95400 – Watchdog Timer Interrupt, Reset, and Wake Up for PSoC® 41xx/42xx Devices

Objective

This example demonstrates how to use the watchdog timer in PSoC 41xx/42xx devices to generate periodic interrupts, reset the system, and wake up from the Deep Sleep low power mode.

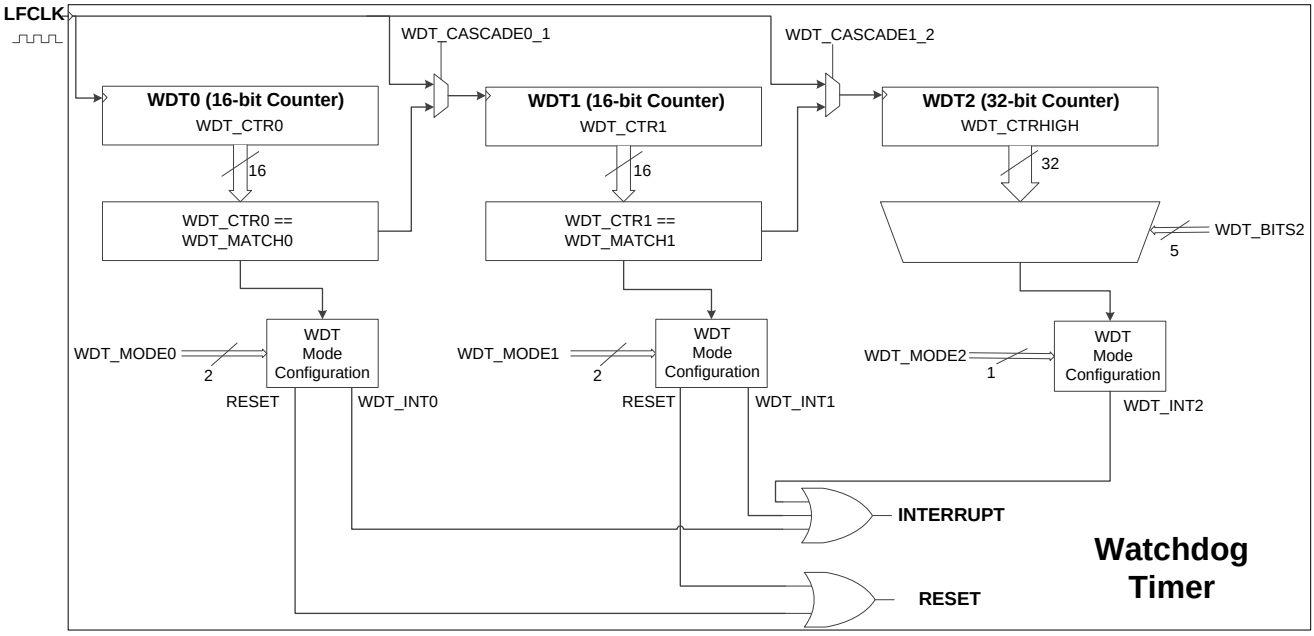
Overview

There are three examples that can run on the CY8CKIT-042 Pioneer kit; however, these can be ported to other boards that contain LEDs and buttons.

- Project #1 shows how to reset the system when the program is out of control due to unexpected inputs or conditions. It also shows how to store log data into flash before reset, which can be read with PSoC Programmer™ for fault analysis.
- Project #2 shows how to wake the system from Deep Sleep, which is a critical function for low-power application design.
- Project #3 combines functions of the above two projects to demonstrate the cascade feature of the watchdog timer.

Figure 1 illustrates the watchdog timer in PSoC 4100/4200 devices.

Figure 1. Watchdog Block Diagram for PSoC 4100/4200 Devices



Note: The watchdog block in the PSoC 4000 family is simpler than in the PSoC 4100/4200 family. It has only one 16-bit counter. See KBA91373 - Watchdog Timer in the PSoC® 4000 Family for a code example for the PSoC 4000 family. See the Watchdog Timer section in the System Reference Guide to learn more about the PSoC 4 watchdog block function and API.

Requirements

Tool: PSoC Creator 3.0 SP2 or later version.

Programming Language: C (GCC 4.7.3), ARM® Cortex®-M0 Assembler

Associated Parts: PSoC 4100/4200 family

Related Hardware: CY8CKIT-042

Design

In PSoC 41xx/42xx devices, once the watchdog timer is enabled, the watchdog timer counts the 32 kHz internal clock “LFCLK” with the internal counter. If a match value is set by writing to the MATCH register, the watchdog timer generates an interrupt when the counter value is equal to the match value. If the counter value is cleared before reaching the match value, the interrupt is never generated. This operation is called “clear the watchdog counter”.

There are some rules for clearing the watchdog counter:

- Clearing must be done before the counter value reaches the match value. For functions that have long execution times, you may have to insert multiple clearing operations during function execution.
- Clearing should not be done in the interrupt service routine (ISR). It is possible for the main loop to be out of control, but the interrupt is still generated periodically. Clearing the watchdog interrupt in the ISR prevents the watchdog from generating a system reset to clear the fault condition.
- Clearing the watchdog counter may require a little time to settle, which can be found in the device datasheet or technical reference menu.

If the clearing operation is halted due to a fault condition such as the firmware entering an infinite loop, the watchdog generates the interrupt. PSoC 41xx/42xx devices can be configured to generate a system reset as soon as the interrupt occurs or on the third consecutive unhandled interrupt (interrupt flag bit is not cleared).

Moreover, you can disable generating the system reset and use the watchdog timer to generate periodic interrupts that can wake the system from the Deep Sleep mode. This feature is extremely useful for low-power designs.

Project #1 – Using the Watchdog Timer to Reset the System

The watchdog timer is configured to generate an interrupt every second. The watchdog counter is cleared periodically in the main loop. If the switch button ‘SW2’ is pressed (simulating a fault condition), the clearing operation stops. After three seconds (the third continuous unhandled interrupt), the system resets.

For the first watchdog interrupt (the counter value equal to the match value), some log data (integers from 0 to 127) is stored in the last row of the internal flash. This log data can be read with the PSoC Programmer for fault analysis.

Three LEDs indicate the system status:

- Green LED: Turns ON when the system is working normally (the switch SW2 is not pressed)
- Blue LED: Turns ON while the switch SW2 is pressed
- Red LED: Turns ON for one second to indicate that the system was just reset by the watchdog timer

Figure 2 shows the PSoC Creator schematic and Figure 3 illustrates the firmware flow for this project.

Figure 2. Creator Schematic for Project #1

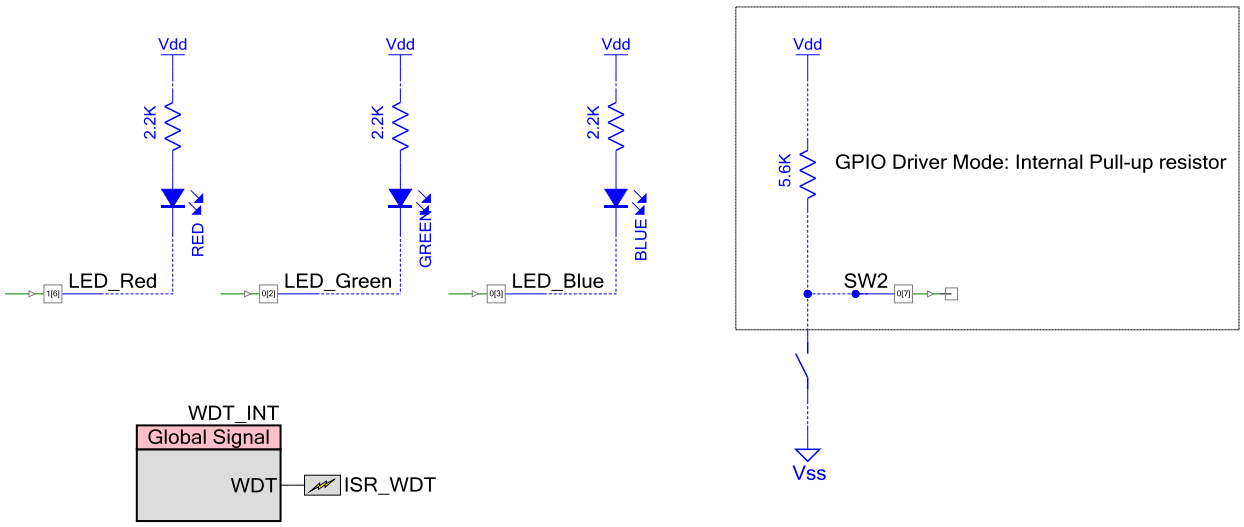
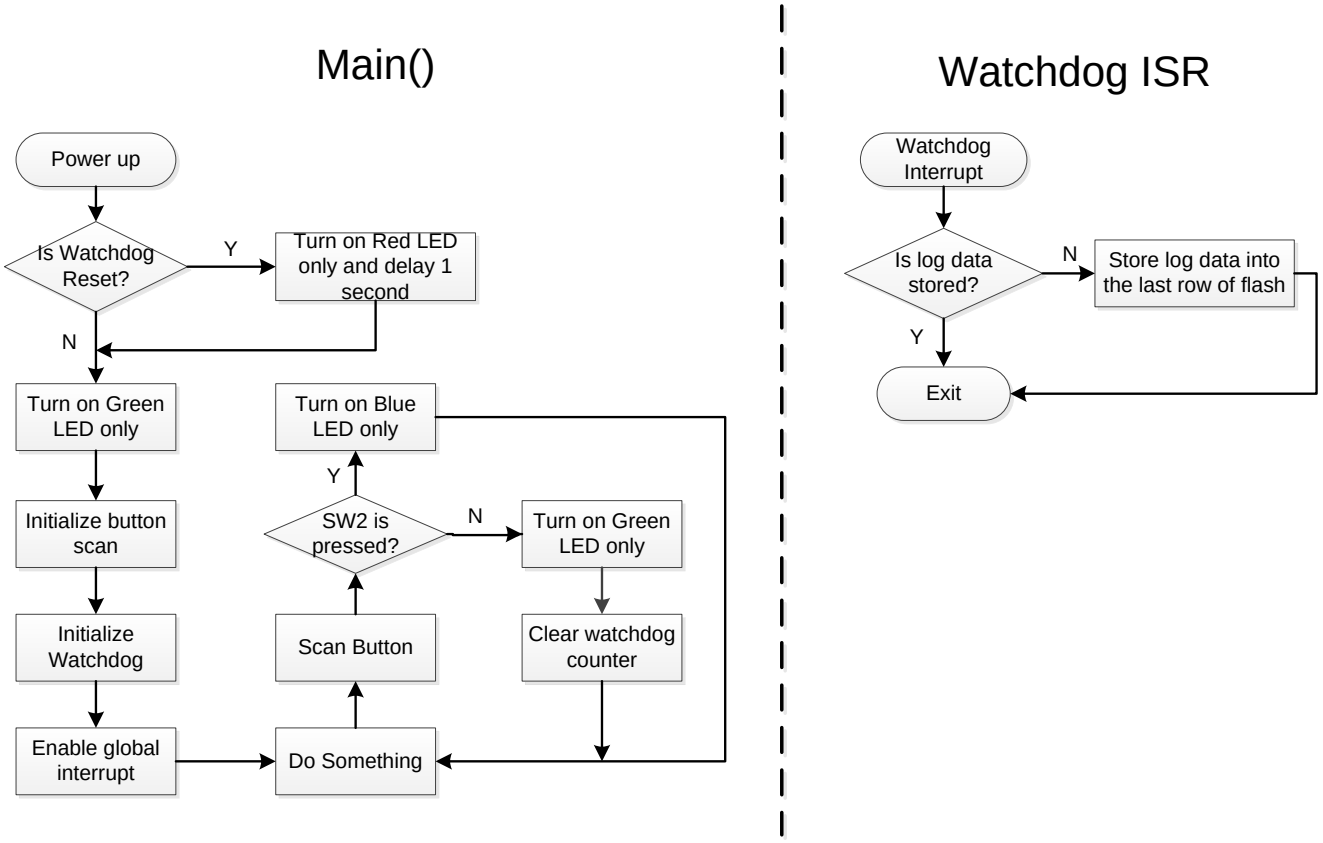


Figure 3. Firmware Flow for Project #1



Project #2 – Wakeup from Deep Sleep

The watchdog timer is configured to generate an interrupt every 250 milliseconds. This interrupt wakes the system from Deep Sleep mode. The system delays 500 milliseconds by calling CyDelay functions and then goes into Deep Sleep again.

Two LEDs are used to indicate the system status. Before the system enters Deep Sleep mode, only the green LED is turned ON and is kept ON during the Deep Sleep time. After the system is woken by a watchdog timer interrupt, only the red LED is turned ON. It is kept ON during the firmware delay. Then, only the green LED is turned ON and system goes into Deep Sleep again.

Figure 4 shows the PSoC Creator schematic and Figure 5 illustrates the firmware flow for this project.

Figure 4 Creator Schematic for Project #2

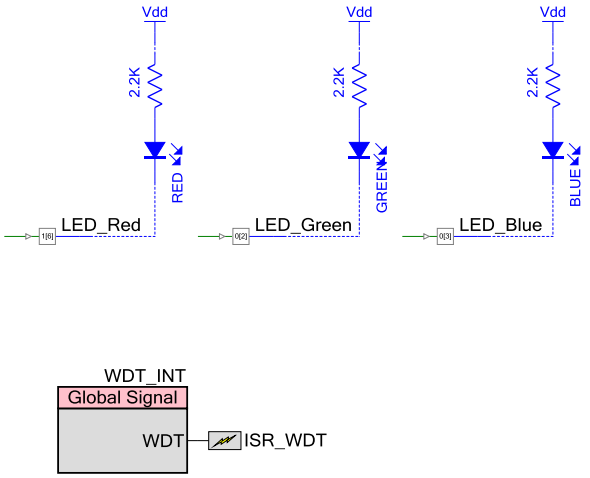
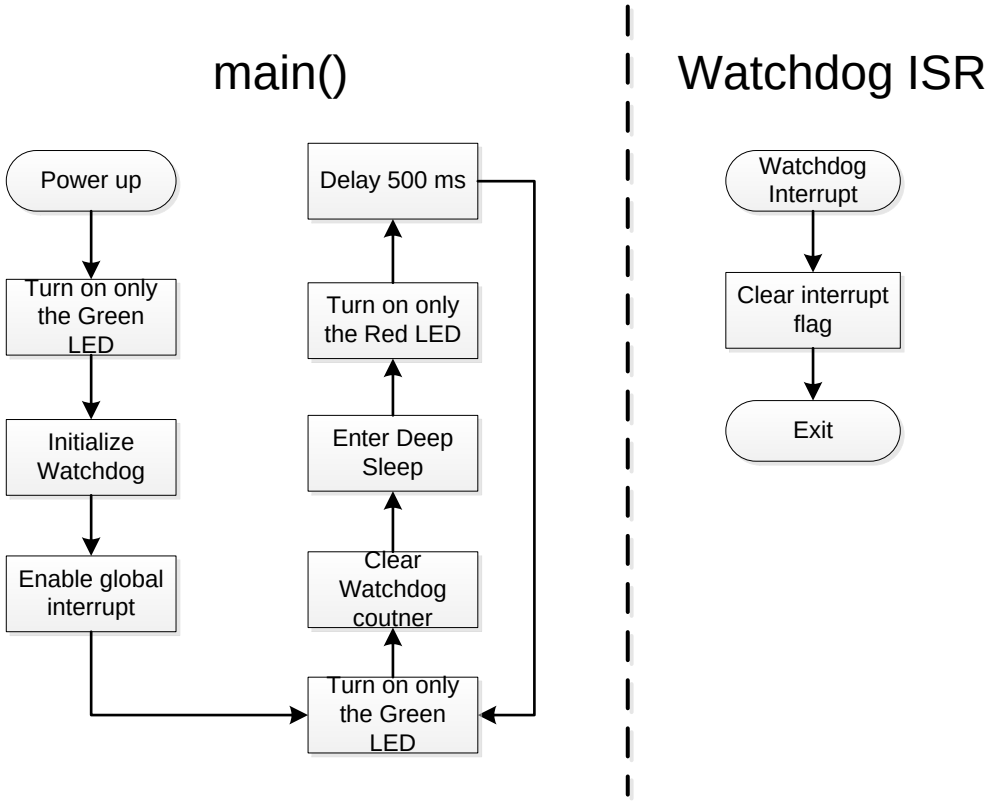


Figure 5 Firmware Flow for Project #2



Project #3 – Wakeup from Sleep with System Reset Enabled

The watchdog timer in PSoC 41xx/42xx devices has two 16-bit counters and one 32-bit counter that can be cascaded. This example demonstrates cascading of Counter 0 and Counter 1. Counter 0 is configured to generate an interrupt only (no system reset) when the counter value is equal to the Counter 0 match value (match event). Counter 1 counts the match event from Counter 0 and compares its value with the Counter 1 match value. When they are equal, it generates an interrupt. If three continuous interrupts are not handled, a system reset occurs. Refer to PSoC 41xx/42xx PSoC 4 Technical Reference Manuals (TRM) for more details.

This code example shows how to use this feature to combine the system reset and sleep wakeup together. These two functions have been shown separately in the earlier two projects.

Counter 0 is configured to generate an interrupt every 250 ms, which wakes the system from Deep Sleep. Counter 1 is configured to generate an interrupt every 1 second. If the interrupt is not handled for three continuous ocurrences, the system is reset and log data is stored into the last row of internal flash as Project #1 shows.

Three LEDs indicate the system status:

- Green LED: Turns ON when the system is working normally (the switch SW2 is not pressed)
- Blue LED: Turns ON when the switch SW2 is pressed
- Red LED: Turns ON to indicate that the system was just reset by the watchdog timer or system has woken up from Deep Sleep

Figure 6 shows the PSoC Creator schematic and Figure 7 illustrates the firmware flow for this project.

Figure 6 Creator Schematic for Project #3

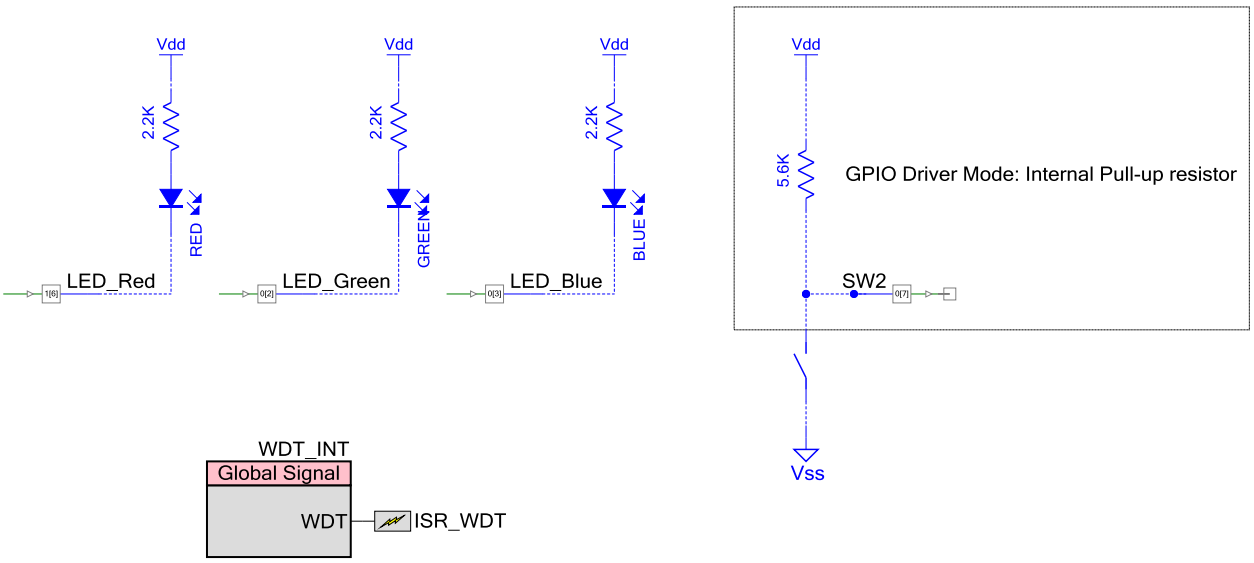
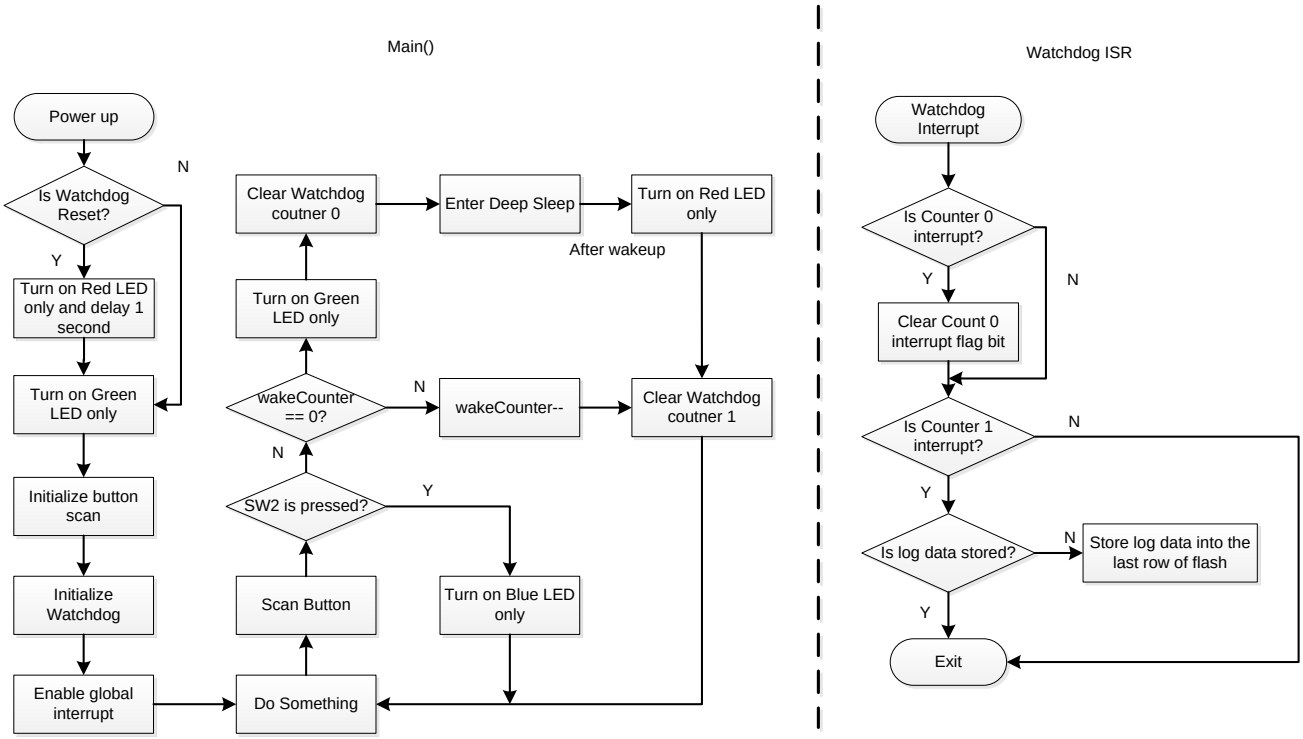


Figure 7 Firmware Flow for Project #3



Design Considerations

The pull-up resistor for the SW2 button is inside the chip GPIO block. No external resistor is used.

CY8CKIT-042 uses a three-in-one LED package that contains red, green, and blue LEDs. These may be separated in other kits, or even replaced by other display methods, such as a segment LCD.

This design can be extended to run on other kits, such as the CY8CKIT-001 with CY8CKIT-038.

Hardware Setup

For basic kit board setup, see the corresponding Kit Guide. For setup with a specific kit, do the following:

1. CY8CKIT-042: No special settings are required for this kit.
2. CY8CKIT-038: This kit works with CY8CKIT-001 to evaluate PSoC 4 functions. You can choose three of the four LEDs (LED1 through 4) on CY8CKIT-001 to replace the RGB LED on CY8CKIT-042. Use three wires to connect LEDs to pin sockets near the breadboard on CY8CKIT-001. You can also choose SW1 or SW2 on CY8CKIT-001 to replace the SW2 button on CY8CKIT-042. In addition, use one wire to connect SW1 or SW2 to the pin socket near the breadboard on CY8CKIT-001. See CY8CKIT-001 user guide for more details.

Software Setup

No special software setup is required. All supported compilers can be used with any optimization.

Components

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

Table 1. List of PSoC Creator Components

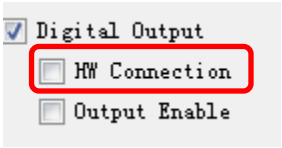
Component	Hardware Resources
Digital Output Pin	Three PSoC 4 GPIO configured as digital output without pull-up or pull-down resistors
Digital Input Pin	One PSoC 4 GPIO configured as digital input with a pull-up resistor
Interrupt	One interrupt service routine with the highest priority
Global Signal	Connect the WDT interrupt to the Interrupt Component
Pin	Three pins for LED indicators; one pin for switch button detection

Parameter Settings

Digital Output Pins:

Clear the “HW Connection” option to prevent build errors.

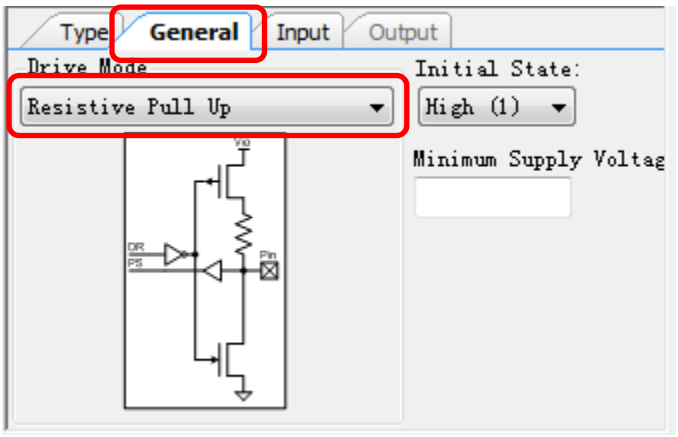
Figure 8. Uncheck HW Connection



Digital Input Pins:

1. Clear the “HW Connection” option to prevent build errors.
2. Set the drive mode as “Resistive Pull Up” in the General Tab:

Figure 9. Set Pull-up Resistor Drive Mode



Design-Wide Resources

Figure 10 and Figure 11 show the pin assignments for each example. No other design-wide resource needs to be changed from its default setting.

Figure 10. Pin Assignment for Project #1 and Project #3

Name	Port	Pin
LED_Blue	P0[3]	27
LED_Green	P0[2] SCB0:SPI:SS3	26
LED_Red	P1[6]	43
SW2	P0[7] SCB1:SPI:SS0, WAKEUP	31

Figure 11 Pin Assignment for Project #2

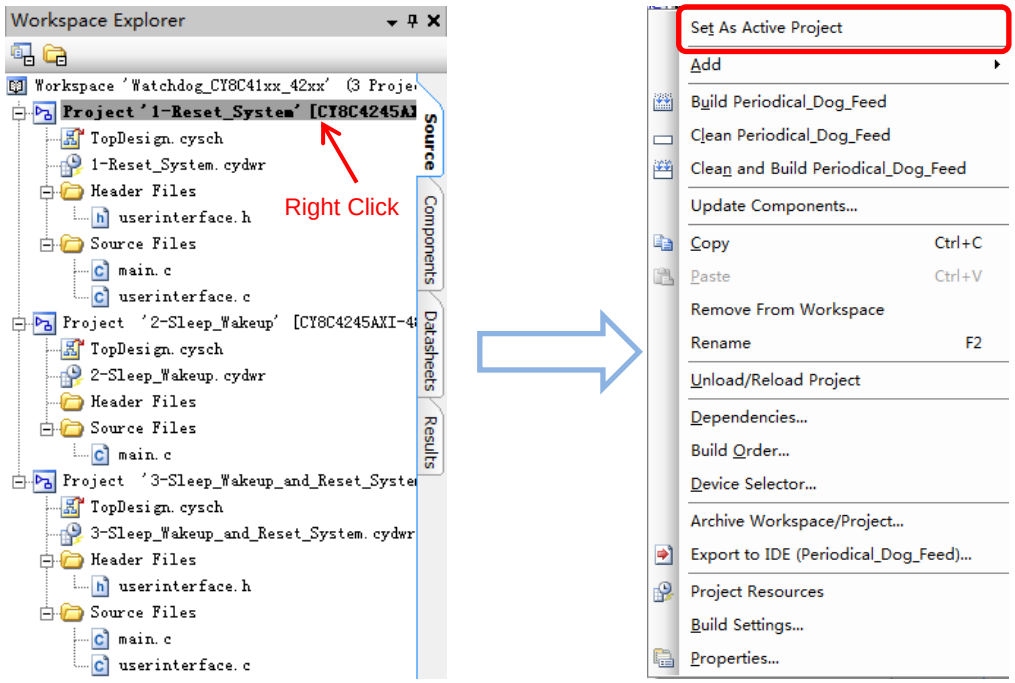
Name	Port	Pin
LED_Blue	P0[3]	27
LED_Green	P0[2] SCB0:SPI:SS3	26
LED_Red	P1[6]	43

Operation

Project #1 – Using the Watchdog Timer to Reset the System

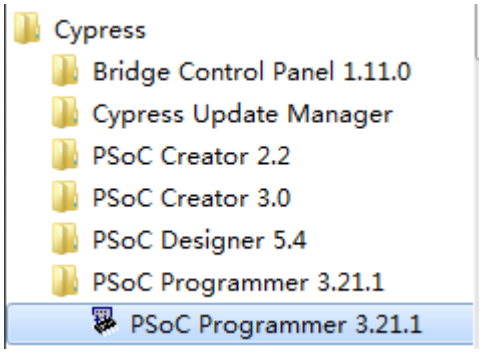
1. In the Workspace Explorer, right-click the “Project ‘1-Reset_System’” and select “Set As Active Project”. See Figure 12.

Figure 12. Choose Project #1 as Active Project



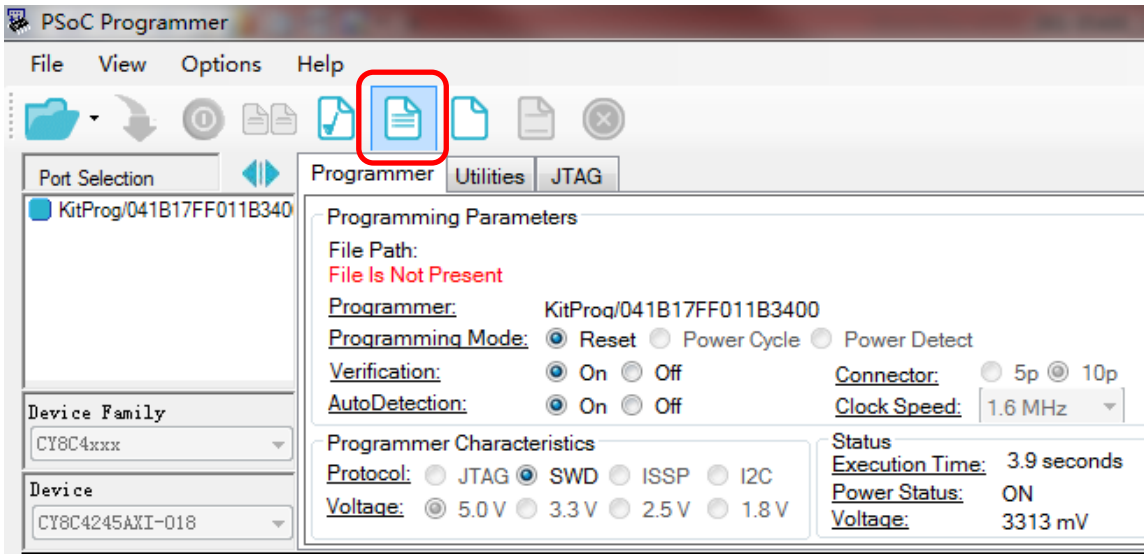
2. Build the project to generate the hex file.
3. Connect the PC to CY8CKIT-042 J10 with the USB cable.
4. Download the hex file to the PSoC 4 chip.
5. Keep the USB cable connection for the power supply.
   a) Press the Reset button on the kit and observe the LED status on the top right corner of CY8CKIT-042. The green LED first blinks and then turns ON, which means that the system has reset and entered the normal working stage. Until a further input, the green LED remains in the ON state.
   b) Press the SW2 button and observe the status LED. The green LED is turned OFF and the blue LED is turned ON. After the blue LED is turned ON, release the SW2 button. The green LED is turned ON again and blue LED is turned OFF. The blue LED indicates that the button SW2 is pressed.
   c) Press and hold SW2 again for more than three seconds. Observe the status LED during the operation. The blue LED is turned on at first; after approximately three seconds, the red LED is turned ON for approximately 1 second, and then the blue LED is turned ON again. The red LED indicates that the system was just reset by the watchdog because the watchdog counter was not cleared.
   d) When the SW2 button is released, the green LED is turned ON.
   e) Press and hold SW2 again. Observe the status LED during the operation. The red LED is turned ON periodically, which means that the system continues to reset.
6. Launch PSoC Programmer from the Start menu as shown in Figure 13. The operation of PSoC Programmer can be found in the User Guide.

Figure 13. Launch PSoC Programmer



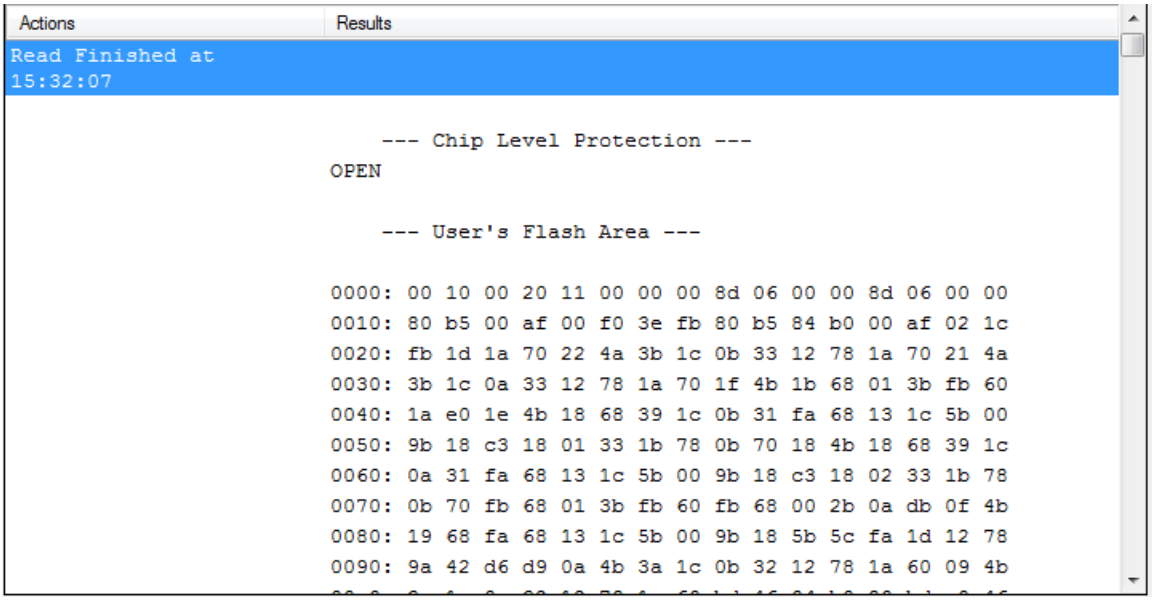
7. In PSoC Programmer, click the “Read” button to read the flash data (red box in Figure 14). Pay attention to the PSoC Programmer setting shown in Figure 14.

Figure 14. Read Operation in PSoC Programmer



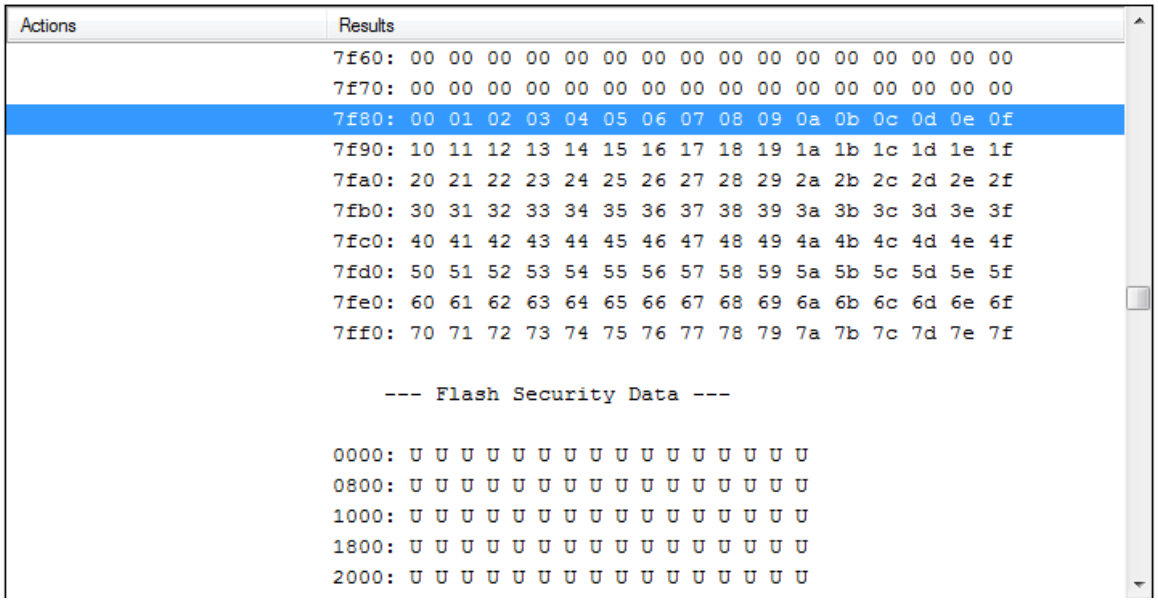
The flash data is read and displayed in the result window of PSoC Programmer. Figure 15 shows the data at the beginning rows of flash.

Figure 15. Data in Beginning Rows of Flash



8. Scroll down the flash data to the rows at the end of flash. You can see the log data that was stored in flash when the watchdog reset occurred. In this code example, they are the incremental integers from 0 to 128 (Figure 16).

Figure 16. Log Data after Watchdog Reset Happens

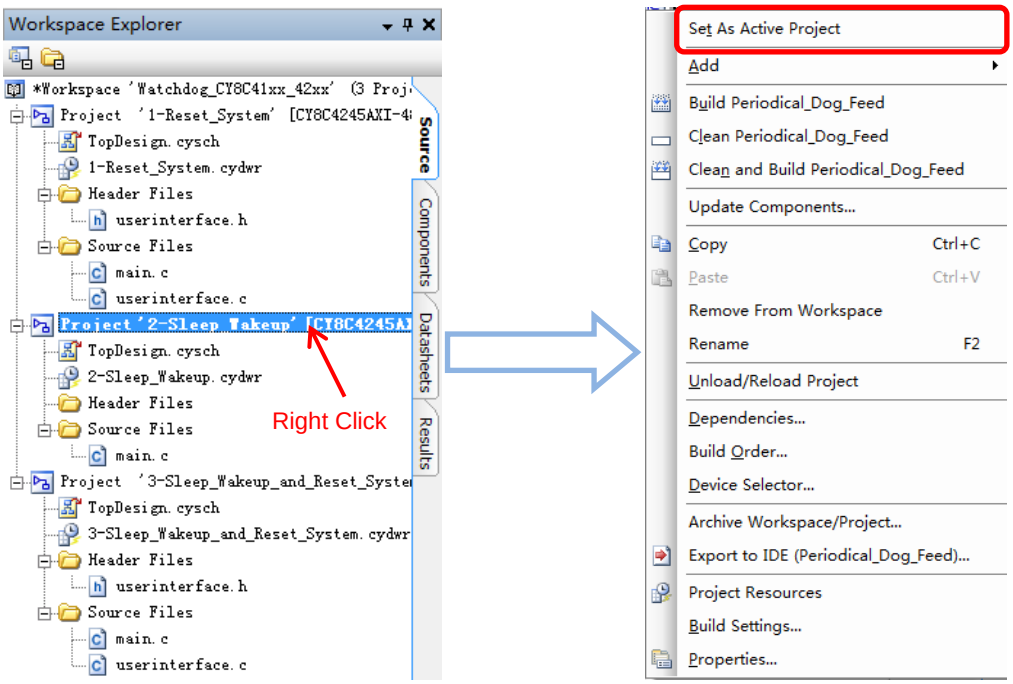


Note: This log can be seen only after a watchdog reset has happened (after the red LED is turned ON and no new hex file is programmed in to flash). Pressing the reset button has no effect on this log data.

Project #2 – Wakeup from Deep Sleep

1. In the Workspace Explorer, right-click the “Project ‘2-Sleep_Wakeup’” and select “Set As Active Project” (see Figure 17).

Figure 17. Choose Project #2 as Active Project



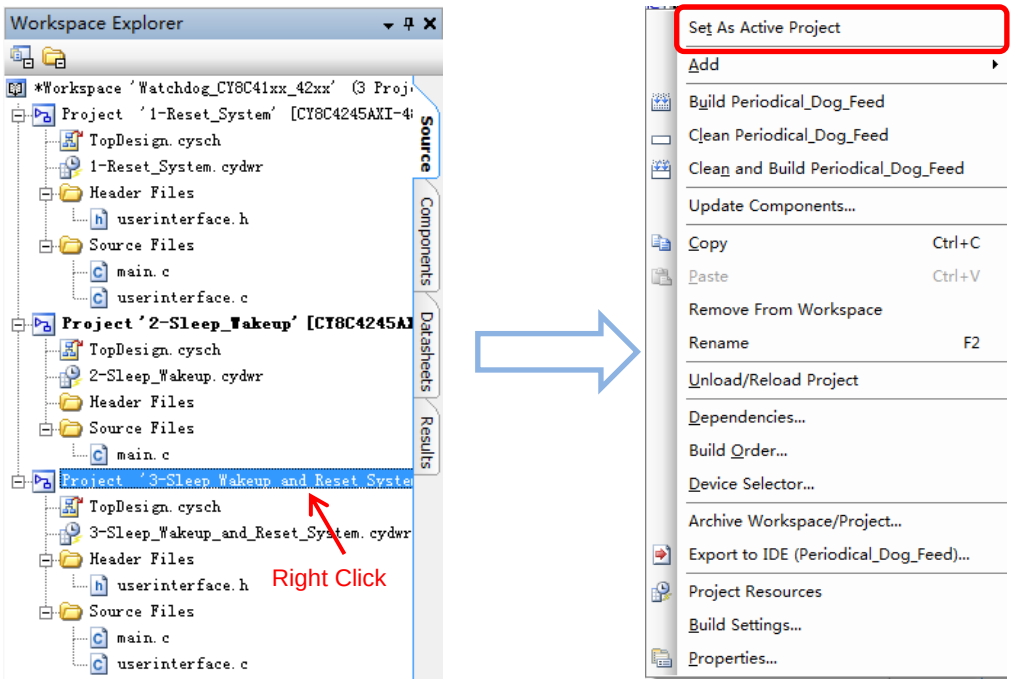
2. Build the project to generate the hex file.

3. Connect the PC to CY8CKIT-042 J10 with the USB cable.
4. Download the hex file to the PSoC 4 chip.
5. Keep the USB cable connection for the power supply.
   a) Press the Reset button on the kit and observe the LED status on the top right of the CY8CKIT-042. The green LED first blinks and then turns ON, which means that the system has reset and entered the normal working stage.
   b) Later, the green LED is turned OFF and the red LED is turned ON. This sequence is repeated indefinitely. The red LED indicates that the system has just woken up from deep sleep by a watchdog interrupt.

Project #3 – Wakeup from Sleep with System Reset Enabled

1. In Workspace Explorer, right-click the "Project '3-Sleep_Wakeup_and_Reset_System'" and select "Set As Active Project" (see Figure 18).
2. Build the project to generate the hex file.
3. Connect the PC to the CY8CKIT-042 J10 with the USB cable.
4. Download the hex file to the PSoC 4 chip.
5. Keep the USB cable connection for the power supply.
   a) Press the Reset button on the kit and observe the LED status on the top right corner of CY8CKIT-042. The green LED first blinks and then turns ON, which means that the system has reset and entered the normal working stage.
   b) If SW2 is not pressed, the system behaves as described in Project #2 – Wakeup from Deep Sleep.
   c) If SW2 is pressed, the system behaves as described in Project #1 – Using the Watchdog Timer to Reset the System.

Figure 18. Choose Project #3 as Active Project



Upgrade Information

N/A

Related Documents

Table 2 lists relevant application notes, code examples, knowledge base articles, device datasheets, and Component datasheets.

Table 2. Related Documents

Document	Title	Comment
AN79953	Getting Started with PSoC® 4	Introduces you to PSoC® 4, an ARM® Cortex™-M0 MCU based programmable system-on-chip. It helps you explore the architecture and Creator development tools.
AN86439	PSoC® 4 - Using GPIO Pins	Shows how to use PSoC® 4 GPIO pins effectively and take full advantage of their features. Major topics include GPIO basics, configuration options, mixed-signal use, registers, interrupts, and low-power behavior.
AN90114	PSoC® 4000 Family Low-Power System Design Techniques	Introduces the low-power modes offered by the PSoC® 4000 family and teaches the methods to design low-power systems.
AN90799	PSoC® 4 Interrupts	Explains the interrupt architecture in PSoC 4 and its configuration in PSoC Creator™ IDE with the help of three example projects.
AN89610	PSoC® 4 and PSoC 5LP ARM Cortex Code Optimization	Shows how to optimize C and assembler code for the ARM Cortex CPUs in PSoC® 4 and PSoC 5LP. Gcc and Keil Microcontroller Development Kit (MDK) C compilers are supported.
001-94480	System Reference Guide	Describes functions supplied by the PSoC Creator cy_boot Component. The cy_boot Component provides the system functionality for a project to give better access to chip resources. The functions are not part of the Component libraries but may be used by them. You can use the function calls to reliably perform needed chip functions.
Knowledge Base Articles		
KBA91373	Watchdog Timer in the PSoC® 4000 Family	How is the watchdog timer (WDT) in the PSoC® 4000 family different from the one in the PSoC 4100/4200 family? How do you use the PSoC 4000 WDT to generate a periodic ISR?
Device Documentation		
PSoC 4 Datasheets		PSoC 4 Technical Reference Manuals
Development Kit (DVK) Documentation		
PSoC 4 Kits		

Document History

Document Title: CE95400 – Watchdog Timer Interrupt, Reset, and Wake Up for PSoC® 41xx/42xx Devices

Document Number: 001-95400

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4644987	BOBH	05/12/2015	New spec
*A	5599248	MBGR	02/07/2017	Minor changes to grammar for readability Updated template

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

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, 2015-2017. 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.