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.

CE224285 – PSoC 6 MCU External Flash Access in XIP Mode with SMIF

Objective

This example demonstrates how to use the PSoC® Creator™ Serial Memory Interface (SMIF) Component in execute-in-place (XIP) mode with external flash memory on PSoC 6 MCU devices.

Overview

This example uses the SMIF Component in XIP mode to execute a function from external flash memory. A UART Component displays the output and status of the program as it executes.

Requirements

Tool: PSoC Creator 4.3; Peripheral Driver Library (PDL) 3.1.0

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: All PSoC 6 MCU parts

Related Hardware: CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit (default), CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Hardware Setup

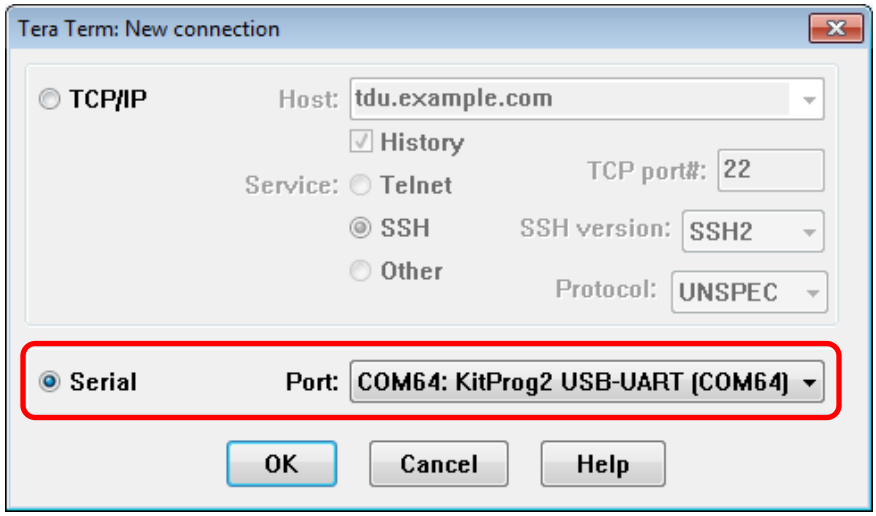
This example uses the kit's default configuration. See the kit guide to ensure the kit is configured correctly.

Software Setup

This section describes the procedure to set up a serial (UART) connection using Tera Term to communicate with the PSoC 6 BLE Pioneer Kit. Tera Term is a free software terminal emulator for Windows, which can be downloaded here. Other terminal emulator programs, such as PuTTY, can also be used.

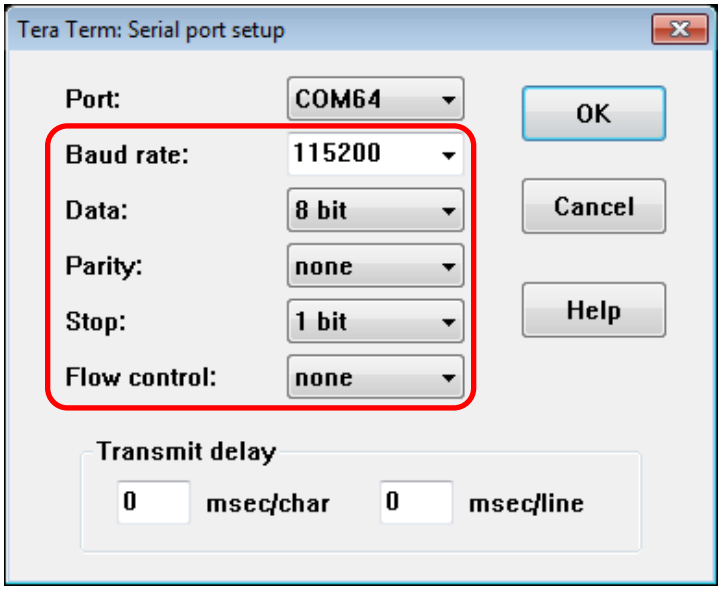
1. Connect the KitProg USB port (J10) on the PSoC 6 MCU Pioneer Kit to the PC using the USB cable.
2. After installing Tera Term, open the program and select the KitProg2 device from the **Port** drop-down list. Click **OK**.

Figure 1. Tera Term Port Selection



3. In Tera Term, select **Setup** > **Serial port** and set **Baud rate: 115200**, **Data: 8 bit**, **Parity: none**, **Stop: 1 bit**, **Flow control: none**. Click **OK**.

Figure 2. Serial Port Configuration Settings



Operation

1. Connect the kit and configure the terminal following the instructions in Software Setup.
2. Open the project workspace in PSoC Creator and choose **Build** > **Generate application.**
3. Copy the following and paste in line 367 of the linker descriptor file *cy8c6xx7_cm4_dual.ld*.

```
.cy_xip_code :
{
    KEEP(*(.cy_xip))
} > xip
```

4. Build the project and program it into the PSoC 6 MCU device. Choose **Debug** > **Program**. Flash for both CPUs is programmed in a single program operation. For more information on device programming, see PSoC Creator Help.
5. Observe the output of the program on the UART terminal. Confirm that the output shows the arrays of data read from the external memory.
6. Open PSoC Programmer and connect to the KitProg device associated with your PSoC 6 MCU Kit.
7. In Programmer, navigate to the **Memory Types** window on the right side of the application. Select **Load from device** and select the **SMIF** checkbox. Uncheck all other memory types.
8. Select **File** > **Read To Log** and observe the output. A block of hexadecimal values will appear in the Results window. These are the values stored in the memory-mapped sectors of external memory. Confirm that the first five rows are non-zero values.

Design and Implementation

In XIP mode, the SMIF Component maps bus accesses to external memory device addresses. This allows functions and data programmed into an external memory device to be used as if they were in internal memory. In this example, two functions, `increment()` and `_write()`, are programmed into the external memory using the `CY_SECTION (".cy_xip_code")` attribute. The firmware also uses the pragma `long_calls` to indicate that the functions live far away from the call site. A string is also programmed into external memory using the attribute `CY_SECTION (".cy_xip")`. These sections are defined in the linker descriptor file *cy8c6xx7_cm4_dual.ld*.

On device startup, the SMIF Component is initialized in normal mode and performs reads from and writes to the external memory. Firmware checks the data read from the external memory and compares it with the written data. If the data matches, then the firmware sets the SMIF Component into XIP mode.

Once in XIP mode, an array of data in SRAM is incremented using the external function `increment()` and printed to the UART terminal. After entering XIP mode, the `_write()` function that was placed in external memory is also accessible. As a result, the firmware can now call `printf()` to print data over the UART terminal. For more information on using `printf()` with UART, see CE223001.

SMIF Source Files

The SMIF Configuration Tool and the PDL for the SMIF block provide the functions and structures required to access the external memory device. Each of the files is listed below:

- *cy_smif.c* – PDL provided file that contains the functions needed to set up the SMIF block.
- *cy_smif.h* – PDL provided header file that contains the inline functions, enumerated types, structures, and function declarations for use with the SMIF block configuration functions.
- *cy_smif_memslot.c* – PDL provided file that contains the low-level functions necessary to set up and access the external memory device.
- *cy_smif_memslot.h* – PDL provided file that contains the macros, structures, and function declarations for use with the external memory device configuration functions.
- `cy_smif_memconfig.c` – SMIF Configuration Tool generated file that contains the populated structures that define the operational modes and parameters to be used with the *cy_smif_memslot.c* functions. In this example, this file is placed in the project's *Source* directory so that the project can be used without needing to generate the file. For instructions on how this file can be generated, see Appendix A.
- *cy_smif_memconfig.h* – SMIF Configuration Tool generated file that contains the declarations of the structures that define the operational modes and parameters for the external memory device. In this example, this file is placed in the project's *Source* directory so that the project can be used without needing to generate the file. For instructions on how this file can be generated, see Appendix A.

This example uses additional custom files *smif_mem.c* and *smif_mem.h* to access the external flash memory. The files provide high-level wrappers around the functions in *cy_smif_memslot.c* to simplify external memory accesses.

Components and Settings

Table 1 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.

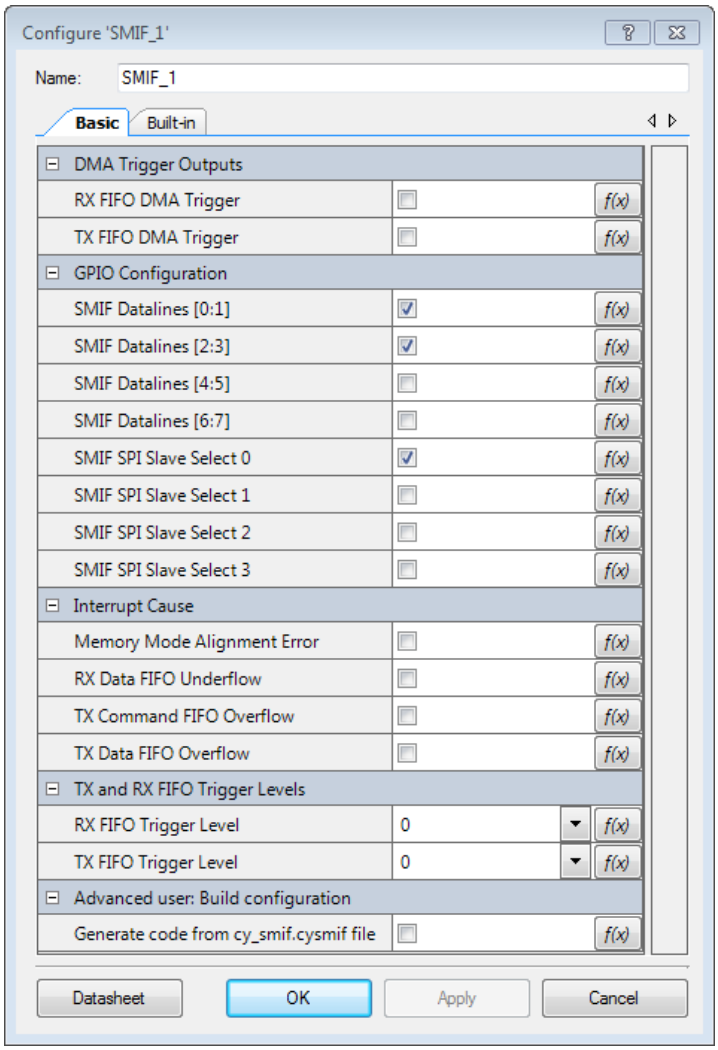
Table 1. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
SMIF	SMIF	Enables communication with external memory	See Figure 3
UART	UART	Enables visual display of information	Default settings only
Digital Output Pin	KIT_LED1	Enables use of LED	Default settings only
Digital Input Pin	KIT_BTN1	Enables use of the Button	Default settings only

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

Figure 3 highlights the non-default settings for the SMIF Component.

Figure 3. The Configuration Settings for the SMIF Component



Reusing This Example

This example is designed for the CY8CKIT-062-WiFi-BT pioneer kit. To port the design to a different PSoC 6 MCU device, kit, or both, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed. For single-core PSoC 6 MCU devices, port the code from *main_cm4.c* to *main.c*.

In some cases, a resource used by a code example is not supported on another device. In that case the example will not work. If you build the code targeted at such a device, you will get errors. See the device datasheet for information on what a particular device supports.

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project
AN215656 – PSoC 6 MCU: Dual CPU System Design	Describes the dual CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-core design
AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project	Describes how to import the code generated by PSoC Creator into your preferred IDE
Code Examples	
CE220823 – PSoC® 6 MCU SMIF Memory Write and Read Operation	This example demonstrates the write and read operations to the Serial Memory Interface (SMIF) in PSoC 6 MCU.
CE220460 – SPI F-RAM Access Using PSoC 6 MCU SMIF	CE222460 provides a code example that implements the SPI host controller on PSoC 6 MCU using the SMIF Component and demonstrates accessing different features of the SPI F-RAM.
CE224073 – SPI F-RAM Access Using PSoC 6 MCU SMIF in Memory Mapped (XIP) Mode	SPI F-RAM Access Using PSoC 6 MCU SMIF in Memory Mapped (XIP) Mode
PSoC Creator Component Datasheets	
Pins	Supports connection of hardware resources to physical pins
SMIF	Supports external memory access
UART	UART communications interface
Device Documentation	
PSoC® 6 MCU Datasheets	PSoC® 6 MCU Technical Reference Manual
Serial NOR Flash (S25FL512S) Datasheet	
Development Kit Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	
CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit	

Appendix A: SMIF Configuration Tool

PSoC Creator supports a stand-alone application, SMIF Configuration Tool, which enables you to configure the SMIF through a GUI-based interface. This application is invoked from the SMIF Component in PSoC Creator. Figure 4 and Figure 5 show how to configure the memory device interfaced with SMIF. This tool generates a **.cysmif* file with these configuration details. Follow these steps to generate SMIF driver memory configuration (**.cysmif*, *cy_smif_memconfig.c,* and *cy_smif_memconfig.h*) files from the SMIF Configuration Tool:

1. Make sure that the `Generate code from cymem file` parameter is selected in the SMIF Component.
2. Remove *cy_smif_memconfig.c* and *cy_smif_memconfig.h* files from the project workspace.
3. Right-click the **SMIF Component** in the PSoC Creator schematic window and click **SMIF Configuration Tool**.
4. Configure the memory part number to match the device on the kit. In this case, S25FL512S.
5. Click **File** and **Save *.cysmif File As…**. Navigate to the project location on your PC. Go to the *CE224285_PSoC_6_ExternalFlashAccess_in_XIP_Mode_with_SMIF\XIP.cydsn* folder, enter the name `cy_smif`, and choose **Save**.
6. If you are using a 3rd party IDE Select **Run** > **Generate source code** and navigate to the folder you wish to save the *cy_smif_memconfig.c* and *cy_smif_memconfig.h* files into. The source files can then be imported into your application in a 3rd party IDE.
7. Close the SMIF Configuration Tool.
8. Build the application. PSoC Creator generates the *cy_smif_memconfig.c* and *cy_smif_memconfig.h* as well as the *cy_smif_memslot.c* and *cy_smif_memslot.h* files. The *cy_smif_memconfig* files provide the external memory parameters to be used with the *cy_smif_memslot* functions.

Figure 4. SMIF Configuration Tool

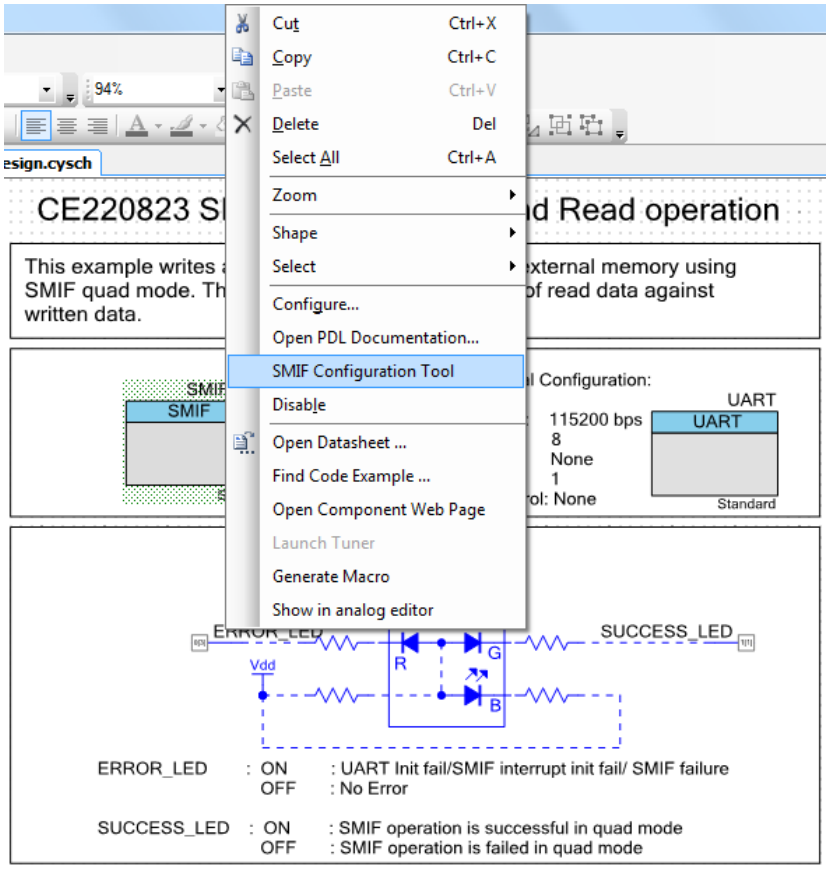
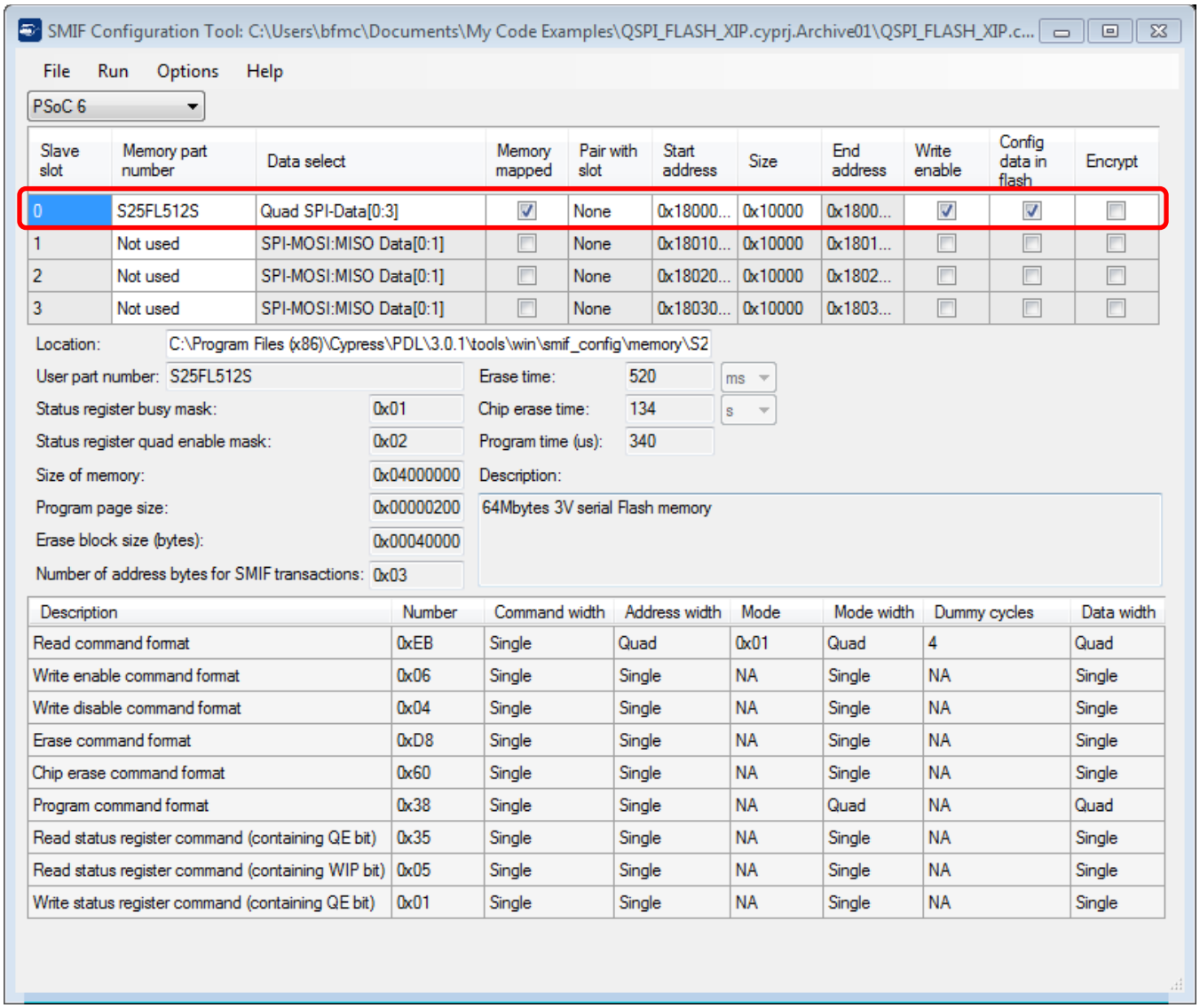


Figure 5. SMIF Configuration Tool Memory Configuration



SMIF Configuration Tool can also be used for external memories which are not listed in the memory part number dropdown list. To create a custom memory number, follow these steps:

1. Make sure that the `Generate code from cymem file` parameter is selected in the SMIF Component.
2. Remove *cy_smif_memconfig.c* and *cy_smif_memconfig.h* files from the project workspace.
3. Right-click the **SMIF Component** in PSoC Creator schematics window and click **SMIF Configuration Tool**.
4. Select **File** > **Create *.cymem File**. A new window, Edit Memory, opens.
5. In the Edit Memory window, select the location to store the new *.cymem* file. Navigate to *C:\Program Files (x86)\Cypress\PDL\3.1.0\tools\win\smif_config\memory* Enter the filename and select **Save**.

Other directories can be used; however the device will not appear in the memory part number drop-down menu. Instead, the custom *.cymem* file must be selected using the **<browse…>** option at the bottom of the memory part number drop-down list.

6. Enter the custom name for the part in the **User part number** field.
7. Fill in the remaining fields following information from the device datasheet.
8. Select **Save** and follow steps 4-9 at the beginning of Appendix A.

Document History

Document Title: CE224285 – PSoC 6 MCU External Flash Access in XIP Mode with SMIF

Document Number: 002-24285

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6292324	BFMC	4/24/2019	New code example
*A	6974465	BFMC	9/29/2020	Updated Hyperlinks, PSoC Creator Version

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 | 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
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2019-2020. 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.