

Objective

This example demonstrates the write and read operations to the Serial Memory Interface (SMIF) in PSoC® 6 MCU using ModusToolbox™ IDE.

Requirements

Tool: [ModusToolbox™ IDE 1.0](#)

Programming Language: C

Associated Parts: All [PSoC 6 MCU](#) parts

Related Hardware: [PSoC 6 BLE Pioneer Kit](#), [PSoC 6 WiFi-BT Pioneer Kit](#), [PSoC 6 WiFi-Prototyping Kit](#)

Overview

Demonstrates read/write operation to external memory by using Serial memory interface (SMIF) in Quad Serial peripheral interface (QSPI) mode. This example also checks the integrity of the read data against written data.

Hardware Setup

This example uses the PSoC 6 WiFi-BT Pioneer Kit's default configuration. Refer to the kit guide to ensure the kit is configured correctly. You can also use PSoC 6 BLE Pioneer Kit or PSoC 6 WiFi-BT Stamp Board Kit by modifying the application to use the corresponding device on the board.

Note: The PSoC 6 BLE Pioneer kit and the PSoC 6 WiFi-BT Pioneer kit ship with KitProg2. ModusToolbox only works with KitProg3. Before using this code example, make sure that the kit is upgraded to KitProg3. See [ModusToolbox Help > ModusToolbox IDE Documentation > User Guide; section PSoC 6 MCU KitProg Firmware Loader](#). If you do not upgrade, you will see an error like “unable to find CMSIS-DAP device” or “KitProg firmware is out of date”.

Software Setup

This example uses a terminal emulator program. Install one on your PC if you don't have one. The instructions use [Tera Term](#).

Operation

1. Connect the Pioneer board to your PC using the provided USB cable through the USB connector.
2. Open your terminal program and select the KitProg COM port. Set the other serial port parameters as follows:
 - a) Baud Rate: 115200bps
 - b) Data: 8 bits
 - c) Parity: None
 - d) Stop: 1 bit
 - e) Flow Control: None
3. Import the application into a new workspace. See [KBA225201](#).
4. Build the application. Choose **Project > Build All**.
5. Program the PSoC 6 MCU device. Select the **mainapp** project. In the **QuickPanel**, scroll down and click the **Program Kitprog3** item.
6. Observe the KIT_LED2 to determine the status of the SMIF operation.

- Make sure that debug messages display in the terminal window as expected.

Figure 1 is a snapshot of the debug UART terminal output.

Figure 1. Debug UART Terminal Output



You can debug the example to step through the code. Use the **Debug (KitProg3)** configuration. See [KBA224621](#) to learn how to start a debug session with ModusToolbox IDE.

Design and Implementation

The SMIF resource implements a SPI-based communication for interfacing external memory devices with PSoC. SMIF resource is configured with four data lines and single slave select line. This example writes 64 bytes of data to external memory in single Quad SPI mode. The written data is read back to check its integrity. The UART resource outputs debug information to a terminal window. An user LED is also used to indicate the status of read and write operation.

The firmware uses source code (*cycfg_qspi_memslot.c* and *cycfg_qspi_memslot.h* files) generated from the SMIF Configurator. This source code provides declarations for the SMIF driver memory configuration.

Resources

Table 1 lists the resources used in this example, and how they are used in the design.

Table 1. ModusToolbox Resources

Resource	Alias	Purpose	Non-Default Settings
Serial Memory Interface (SMIF)	KIT_QSPI	Transmit data to QSPI NOR Flash	Figure 2 , Figure 3
SCB5	KIT_UART	Transmit debug data	Figure 4
General Purpose Input / Output (GPIO)	KIT_LED2	KIT_LED2	Figure 5

Parameter Settings

Non-default settings for each resource is outlined in red in the following figures.

Figure 2 shows the KIT_QSPI resource parameter settings.

ModusToolbox supports a stand-alone application called QSPI Configurator (star marked section in Figure 2), which enables a user to configure the SMIF through a GUI-based interface. This application is invoked from the SMIF resource. Configure the device as shown in the below image. Save the file in the *GeneratedSource* folder.

Figure 2. SMIF Resource Parameter Settings

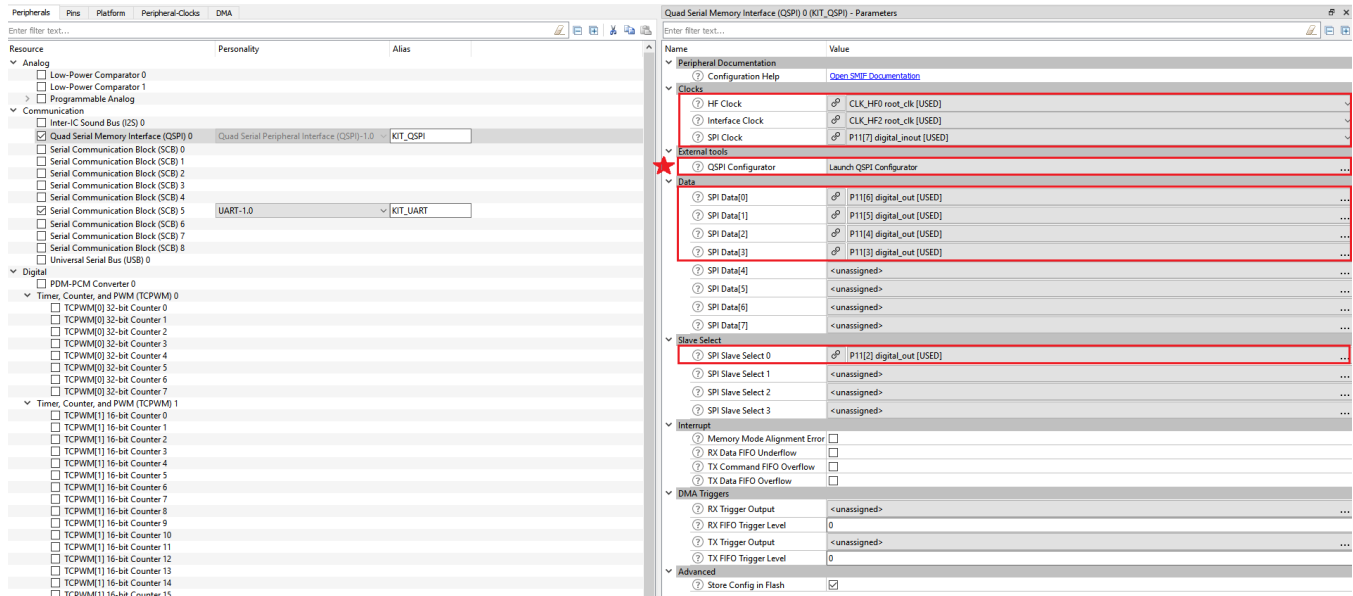


Figure 3 shows the QSPI Configurator.

Figure 3. QSPI Configurator

File Options Help

PSoC 6 ▾

Slave Slot	Memory Part Number	Data Select	Memory Mapped	Pair With Slot	Start Address	Size	End Address	Write Enable	Config Data In Flash	Encrypt
0	S25FL512S ▾	Quad SPI-Data[0:3] ▾	<input checked="" type="checkbox"/>	None ▾	0x18000000	0x10000	0x1800FFFF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
1	Not used ▾	SPI-MOSI:MISO Data[0:1] ▾	<input type="checkbox"/>	None ▾	0x18010000	0x10000	0x1801FFFF	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Not used ▾	SPI-MOSI:MISO Data[0:1] ▾	<input type="checkbox"/>	None ▾	0x18020000	0x10000	0x1802FFFF	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Not used ▾	SPI-MOSI:MISO Data[0:1] ▾	<input type="checkbox"/>	None ▾	0x18030000	0x10000	0x1803FFFF	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Location: C:/Users/yejt/ModusToolbox_1.0/tools/qspi-configurator-1.0/data/memory/S25FL512S.cymem

User part number: S25FL512S Erase time: 520 ms ▾

Status register busy mask: 0x01 Chip erase time: 134 s ▾

Status register quad enable mask: 0x02 Program time (μs): 340

Size of memory: 0x04000000 Description: 64Mbytes 3V serial Flash memory

Program page size: 0x00000200

Erase block size (bytes): 0x00040000

Number of address bytes for SPI transactions: 0x03

SPI modes supported: ☐ Single ☐ Dual ☒ Quad ☐ Octal

Single SPI Commands								Dual SPI Commands								Quad SPI Commands								Octal SPI Commands							
Description	Number	Command Width	Address Width	Mode	Mode Width	Dummy Cycles	Data Width																								
Read command format	0xEB	Single ▾	Quad ▾	0x01	Quad ▾	4	Quad ▾																								
Write enable command format	0x06	Single ▾	Single ▾	NA	Single ▾	NA	Single ▾																								
Write disable command format	0x04	Single ▾	Single ▾	NA	Single ▾	NA	Single ▾																								
Erase command format	0xD8	Single ▾	Single ▾	NA	Single ▾	NA	Single ▾																								
Chip erase command format	0x60	Single ▾	Single ▾	NA	Single ▾	NA	Single ▾																								
Program command format	0x38	Single ▾	Single ▾	NA	Quad ▾	NA	Quad ▾																								
Read status register command (containing QE bit)	0x35	Single ▾	Single ▾	NA	Single ▾	NA	Single ▾																								
Read status register command (containing WIP bit)	0x05	Single ▾	Single ▾	NA	Single ▾	NA	Single ▾																								
Write status register command (containing QE bit)	0x01	Single ▾	Single ▾	NA	Single ▾	NA	Single ▾																								

Figure 4. KIT_UART Configuration

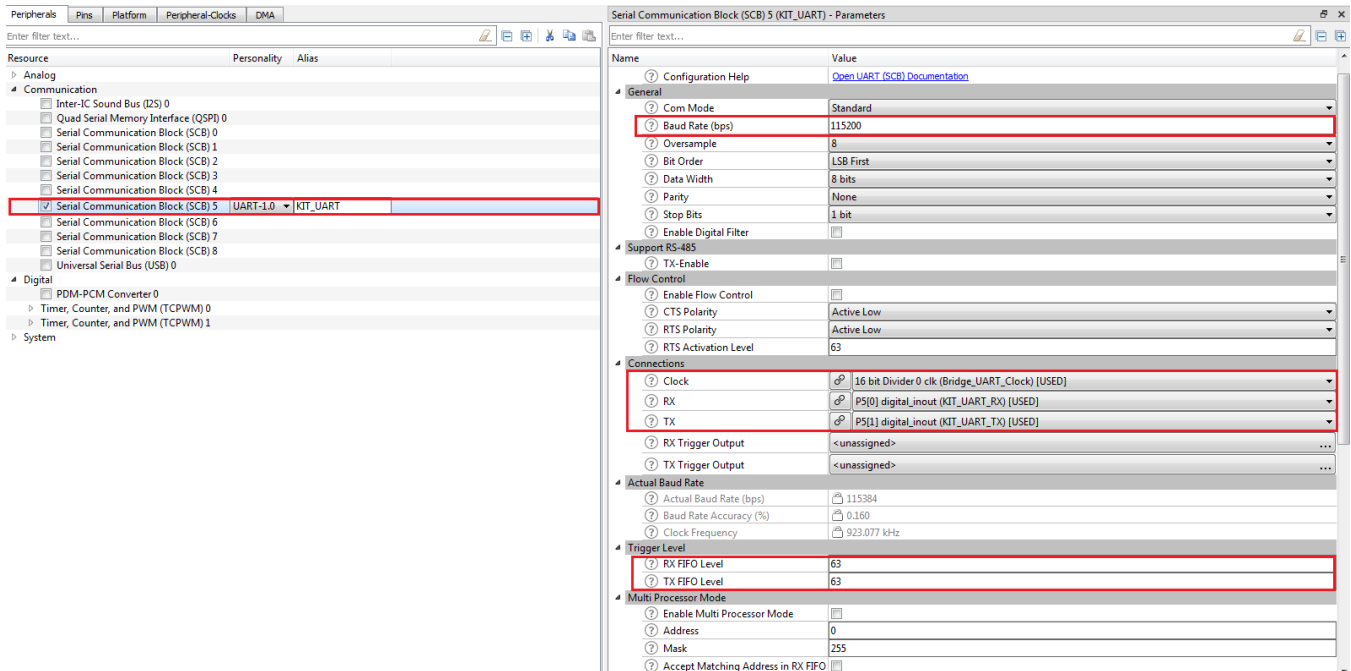
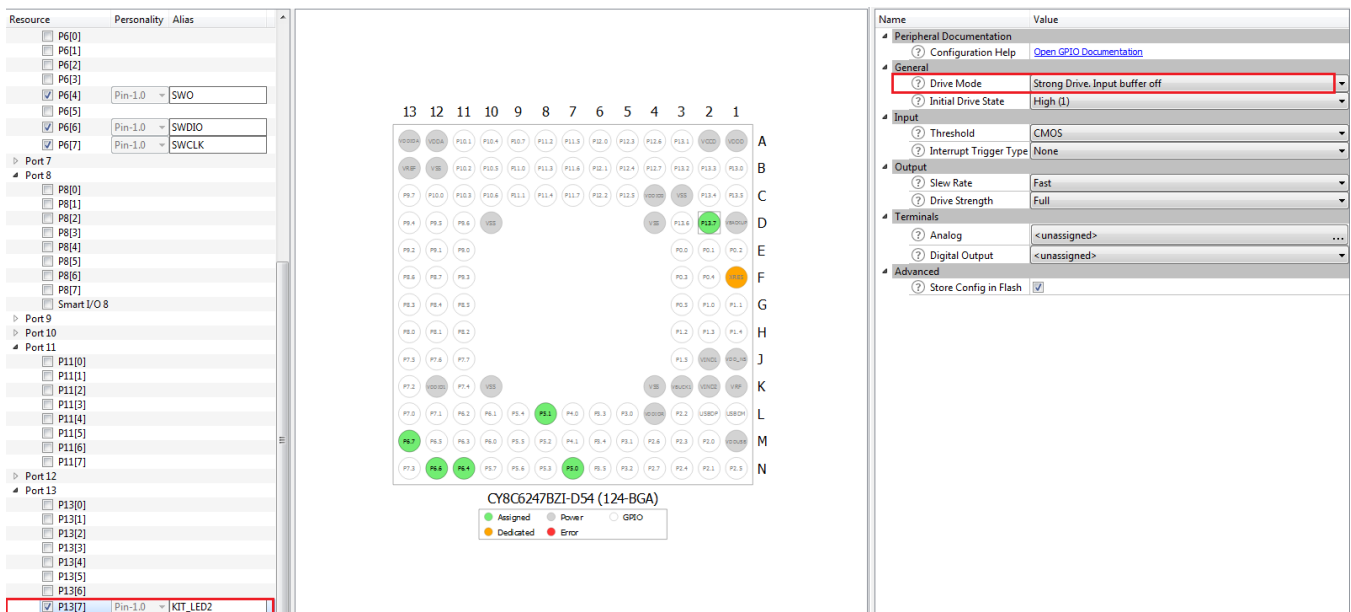


Figure 5. KIT_LED2 Configuration



Related Documents

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

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices.
AN221774 – Getting Started with PSoC 6 MCU	Describes PSoC 6 MCU devices and how to build your first ModusToolbox application and 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-CPU design.
Code Examples	
CE218472 - PSoC 6 MCU Comparing External Voltages Using a Low-Power Comparator	
Visit the Cypress GitHub site for a comprehensive collection of code examples using ModusToolbox IDE	
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kit Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	
CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit	
CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit	
Tool Documentation	
ModusToolbox	The Cypress IDE for IoT designers

Cypress Resources

Cypress provides a wealth of data at www.cypress.com to help you to select the right device, and quickly and effectively integrate the device into your design.

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

Document History

Document Title: CE220823 - PSoC 6 MCU SMIF Memory Write and Read Operation

Document Number: 002-25536

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6369719	YEKT	11/02/2018	Initial public release

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
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2018. 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.