PSoC 4 SPI Slave

About this document

Scope and purpose

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 4.4

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, CY8CKIT-044, CY8CKIT-045S
- **DMA project:** CY8CKIT-044, CY8CKIT-045S

Table of contents

About this document **1**
Requirements **1**
Table of contents **1**
1 Overview **2**
2 Hardware and Software Setup **3**
2.1 Hardware Setup 3
2.2 Software Setup 3
3 Operation **4**
4 Design and Implementation **5**
4.1 SPI Slave Polling 5
4.2 SPI Slave Interrupt 5
4.3 SPI Slave DMA 6
4.4 Components and Settings 7
4.5 Reusing This Example 10
References **11**
Revision history **12**

1 Overview

This code 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 PSoC 4 kit and the code example **CE224339** – PSoC 4 SPI Master are recommended.

2 Hardware and Software Setup

2.1 Hardware Setup

This code example requires two PSoC 4 kits. A kit listed in **Related Hardware** runs this slave example; the second PSoC 4 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
Polling / Interrupt: CY8CKIT-042 Kit	P3[1]	P3[0]	P0[6]	P0[7]	GND
Polling / Interrupt / DMA: CY8CKIT-045S Kit	P5[1]	P5[0]	P5[2]	P5[3]	GND
Polling / Interrupt / DMA: CY8CKIT-044 Kit	P2[1]	P2[0]	P2[2]	P0[7]	GND

2.2 Software Setup

None.

3 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 onto 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.

4 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.

4.1 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:
   - Waits for a command from the SPI master; it sends a dummy data while master bytes are received.
   - Sets the LED to the color in the command data byte.
   - Updates the status and loads it into the Tx buffer to be sent to the master.
   - 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 SPI_Slave_Polling project:

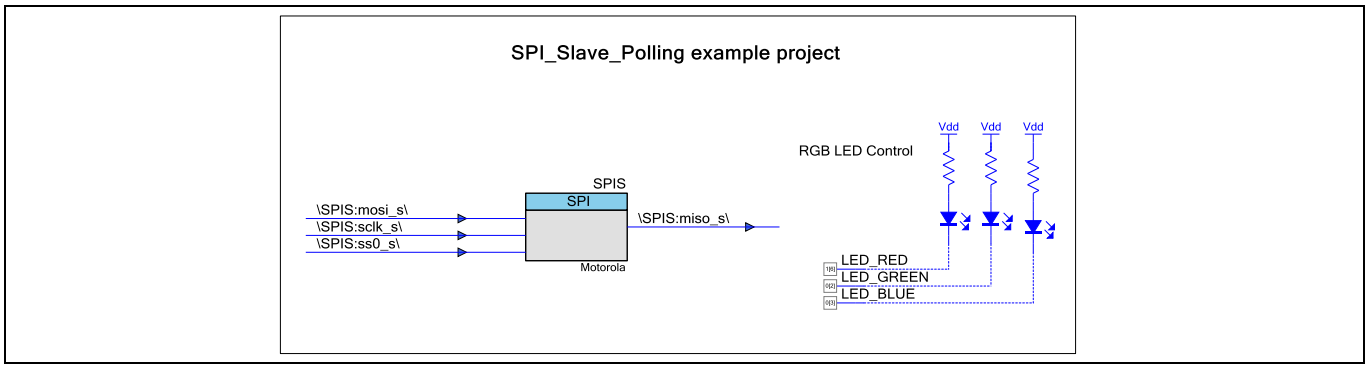


Figure 1 SPI_Slave_Polling Top Design Schematic

4.2 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:
   - Reads the data from the Rx buffer.
   - Sets the LED to the color in the command data byte.
   - Updates the status and loads it into the Tx buffer to be sent to the master.
   - Waits until it receives dummy data from the master so that it can send what is currently in the Tx buffer.

Design and Implementation

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 SPI_Slave_Interrupt project:

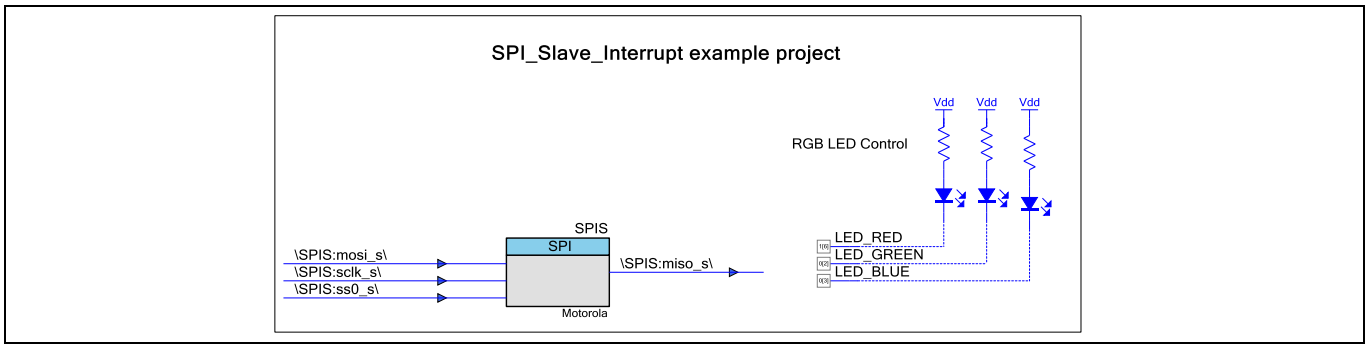


Figure 2 SPI_Slave_Interrupt Top Design Schematic

4.3 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 channel 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 SPI_Slave_DMA project:

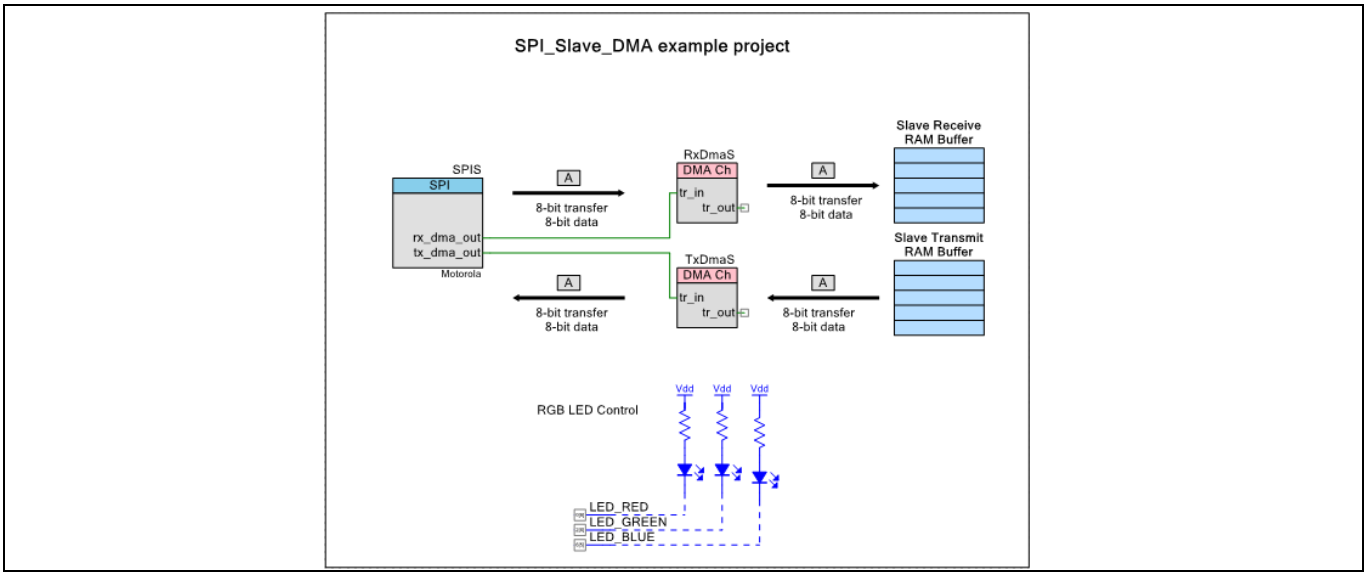


Figure 3 SPI_Slave_DMA Top Design Schematic

4.4 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	RxDmaS	Handles DMA Rx data transfer – slave to master	See **Figure 7**
	TxDmaS	Handles DMA Tx data transfer – master to slave	See **Figure 8**

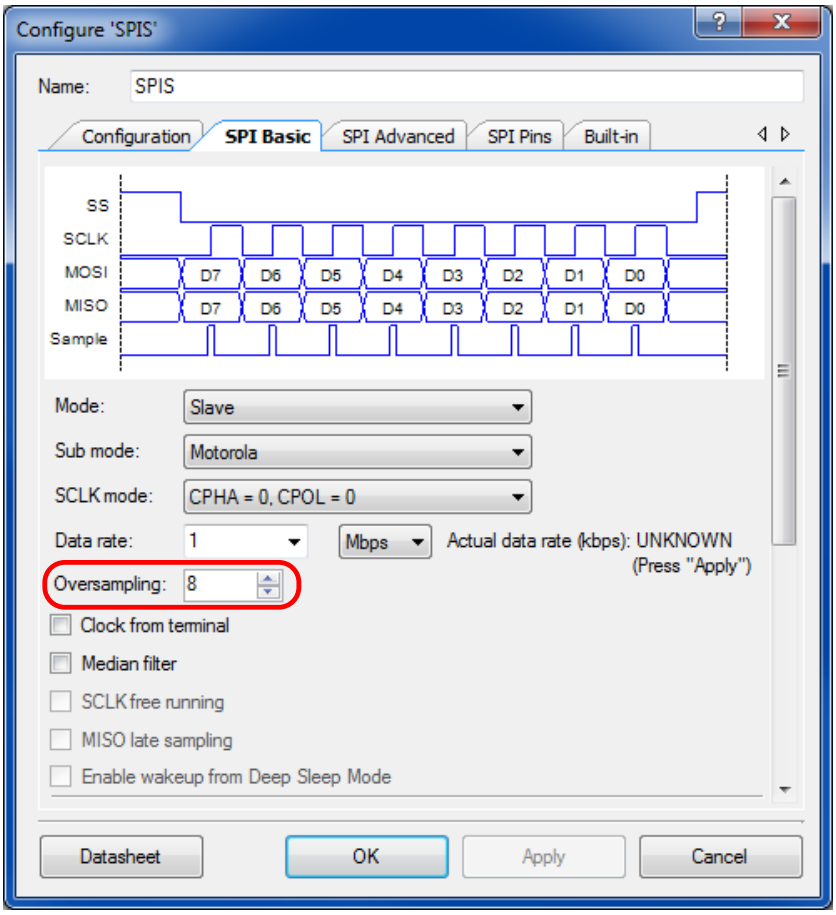


Figure 4 SPI Basic Tab Configuration for All Projects

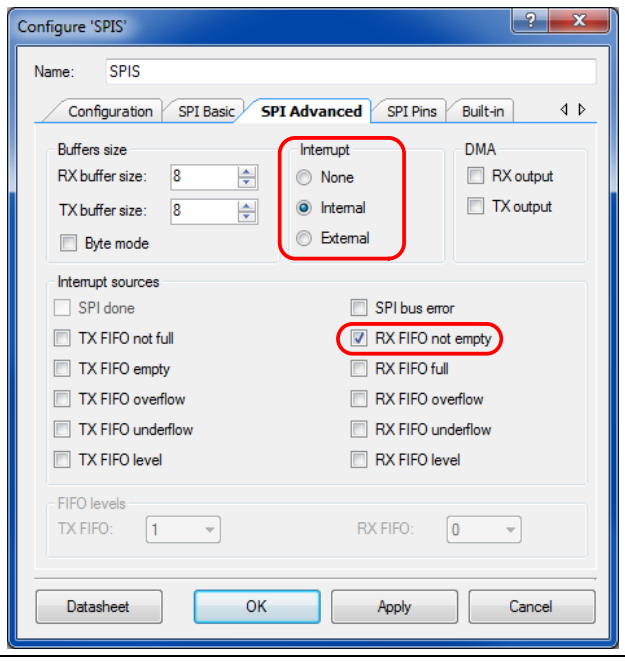


Figure 5 SPI Advanced Tab for Interrupt Project

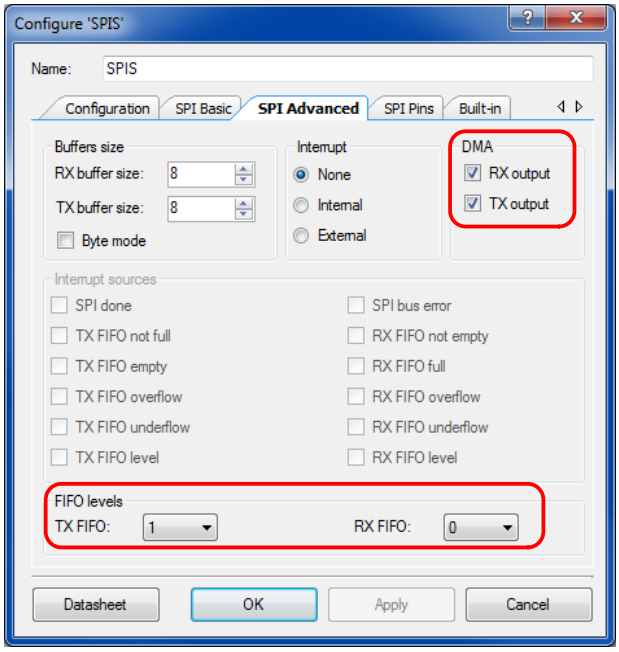


Figure 6 SPI DMA Settings

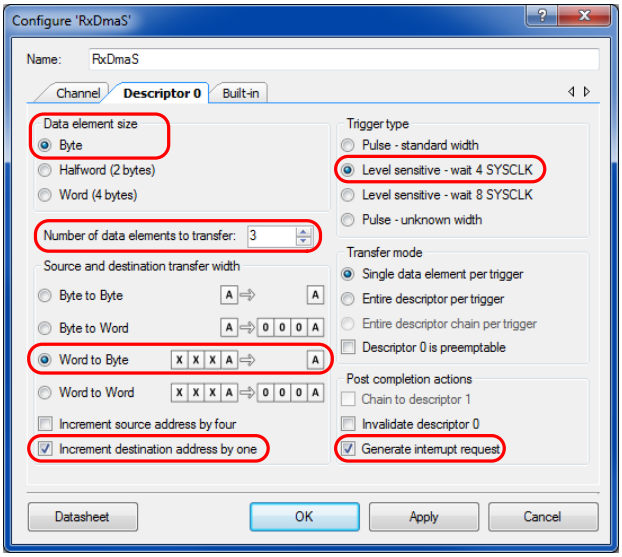


Figure 7 RxDmaS Configuration Settings

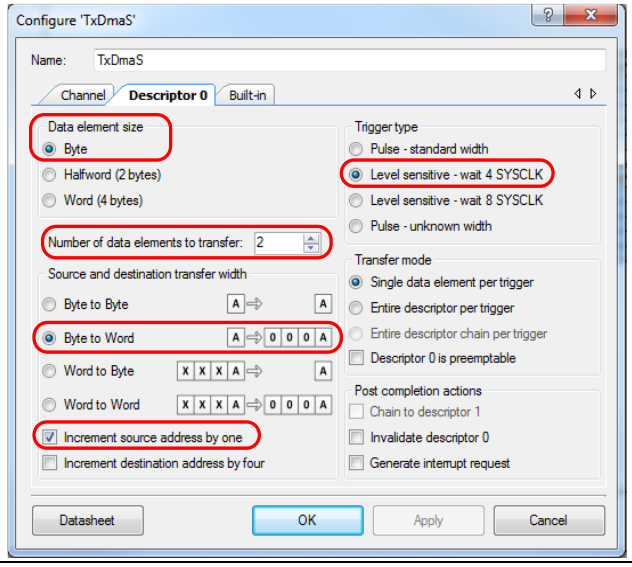


Figure 8 TxDmaS Configuration Settings

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

4.5 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 Pins tab in the Design Wide Resources.
- 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.

References

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

[1] **AN79953** – Getting Started with PSoC® 4: Describes PSoC 4 devices and how to build your first PSoC Creator project

Code Examples

[2] **CE224339** – PSoC 4 SPI Master: Companion code example for this code example.

PSoC Creator Component Datasheets

[3] **SCB**: Supports I²C, SPI, UART and EZI2C communication protocols using the serial control block (SCB)

[4] **DMA**: Supports direct memory access (DMA) controllers

Device Documentation

[5] **PSoC 4 Datasheets**

[6] **PSoC 4 Technical Reference Manuals**

Development Kit Documentation

[7] **CY8CKIT-042 PSoC 4 Pioneer Kit**

[8] **CY8CKIT-044 PSoC 4 M-Series Pioneer Kit**

[9] **CY8CKIT-045S PSoC 4500S Pioneer Kit**

[10] **PSoC 4 Kits**

Tool Documentation

[11] **PSoC Creator** : Look in the **Downloads** tab for Quick Start and User Guides

Revision history

Document version	Date of release	Description of changes
**	2018-07-24	New code example
*A	2019-09-13	Update to new code example template and updated Figure 3.
*B	2020-11-23	Update the code example to support CY8CKIT-045S.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-11-23

Published by

Infineon Technologies AG

81726 München, Germany

© 2020 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Go to www.cypress.com/support

Document reference

002-24463 Rev. *B

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”) .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.