PSoC 4 SPI Master

About this document

Scope and purpose

This code example shows how to use a PSoC® Creator™ Serial Communication Block (SCB) Component configured in SPI mode as a master 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 only works 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

1 Overview

This code example contains three projects that utilize an SPI Component configured as a master. The projects demonstrate polling, interrupt, and DMA methods for SPI data communication. Each project sends a command to an SPI slave to change the color of an LED and receives a status indicator from the slave. A second PSoC 4 kit and the code example, **CE224463** – PSoC 4 SPI Slave, 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 master example; the second PSoC 4 kit needs the code example CE224463 – PSoC 4 SPI Slave.

Table 1 shows how to connect the pins to the slave 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 master 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 master project works with any slave 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 slave kit; see code example CE224463 – PSoC 4 SPI Slave.
7. Press the reset buttons on both boards simultaneously. Confirm that the slave kit LED cycles through colors.
8. If a second kit is not available, connect the four pins to an oscilloscope to view the data transfer.

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 Master Polling

Note that in the SPI protocol, data is sent simultaneously in both directions: master to slave and slave to master.

In the SPI_Master_Polling example, the following functions are performed:

1. The SCB Component is configured as an SPI master.
2. The SPI master:
   a) Sends a command to the slave and receives data at the same time. The command sets the LED on the slave kit to a specified color. The received data holds no value because the slave has not yet executed the command.
   b) Waits for a period that allows the slave to execute the command.
   c) Receives data from the slave while sending dummy data.
   d) Checks the received data and sends the next command if the received data is correct. This causes the LED on the slave kit to cycle through a series of colors at a rate controlled by the master.

Figure 1 shows the top-level design of the SPI_Master_Polling example project:

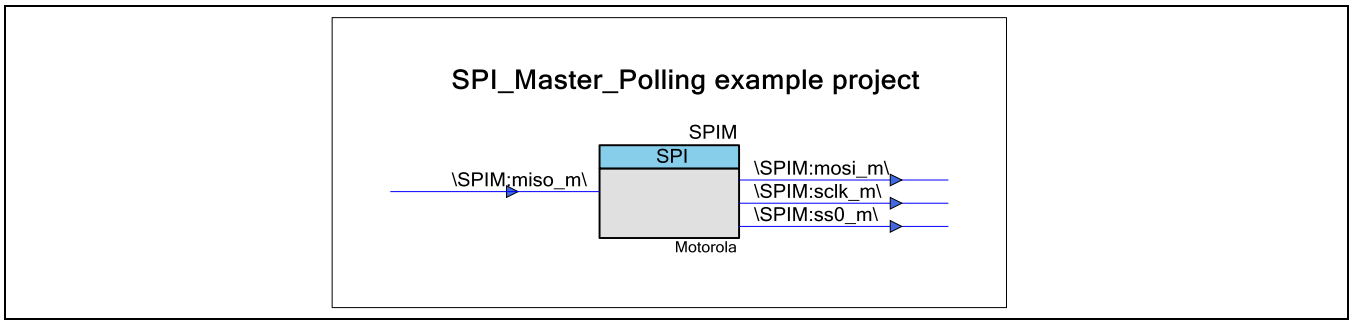

Figure 1 SPI_Master_Polling Top Design Schematic

4.2 SPI Master Interrupt

Note that in the SPI protocol, data is sent simultaneously in both directions: master to slave and slave to master.

In the SPI_Master_Interrupt example, the following functions are performed:

1. The SCB Component is configured as an SPI master.

2. An interrupt handler function `Master_ISR()` sets a global variable flag every time it is called. See **Master ISR Traits**.
3. Enters a state machine:
   - SEND_DATA: Sends a command to the slave and receives data at the same time. The command sets the LED on the slave kit to a specified color. It waits for a period that allows the slave to execute the command. The received data holds no value because the slave has not yet executed the command.
   - DATA_SENT: Changes the flag to SEND_DUMMY on an 'SPI done' interrupt.
   - SEND_DUMMY: Receives data from the slave while sending dummy data.
   - DUMMY_SENT: Changes the flag to READ_STATUS on an 'SPI done' interrupt.
   - READ_STATUS: Reads the status message that was sent by the slave. Checks the received data and sends the next command if the received data is correct. This causes the LED on the slave kit to cycle through a series of colors at a rate controlled by the master. Changes the state to SEND_DATA.

The interrupt `Master_ISR()` is triggered when the data transfer is complete. `Master_ISR()` does the following:

- Sets the flag variable depending on the current stage.
- Clears the interrupt source.

Figure 2 shows the top-level design of the SPI_Master_Interrupt example project:

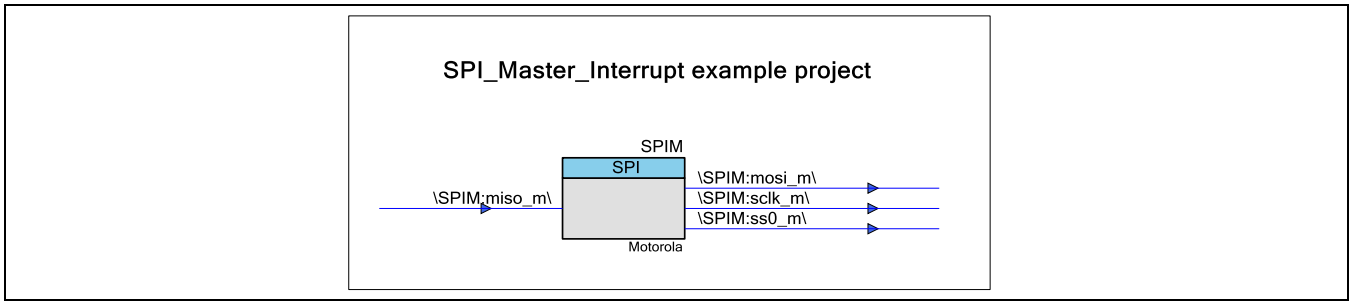

Figure 2 SPI_Master_Interrupt Top Design Schematic

4.3 SPI Master DMA

Note that in the SPI protocol, data is sent simultaneously in both directions: master to slave and slave to master.

In the SPI_Master_DMA example, the following functions are performed:

1. The SCB Component is configured as an SPI master.
2. The DMA transmit and receive channels are initialized and enabled.
3. Two arrays are initialized as the source buffers for DMA. One is the master transmit buffer while the other is the master receive buffer.
4. The SPI Master Component triggers the DMA transfer. The trigger is active when the receive (Rx) FIFO is not empty, or when the transmit (Tx) FIFO has less than seven elements.

5. The command is stored in the DMA Tx buffer and when there is room in the Tx FIFO, DMA transfers the data from the buffer into the FIFO.
6. Once data is in the Tx FIFO, the SPI master can send the data to the slave.
7. When data is sent to the slave, the slave must send the same amount of data back. The data is stored in the Rx FIFO and then moved by DMA into the Rx buffer.
8. Dummy data is then sent to the slave so that the slave can send back the command status.
9. If the command status is correct, the command is updated, and the new command is sent to the slave. This causes the LED on the slave kit to cycle through a series of colors at a rate controlled by the master. If the command is status is not correct, the command is not updated. The process is repeated from Step 6.

Figure 3 shows the top-level design of the SPI_Master_DMA example project:

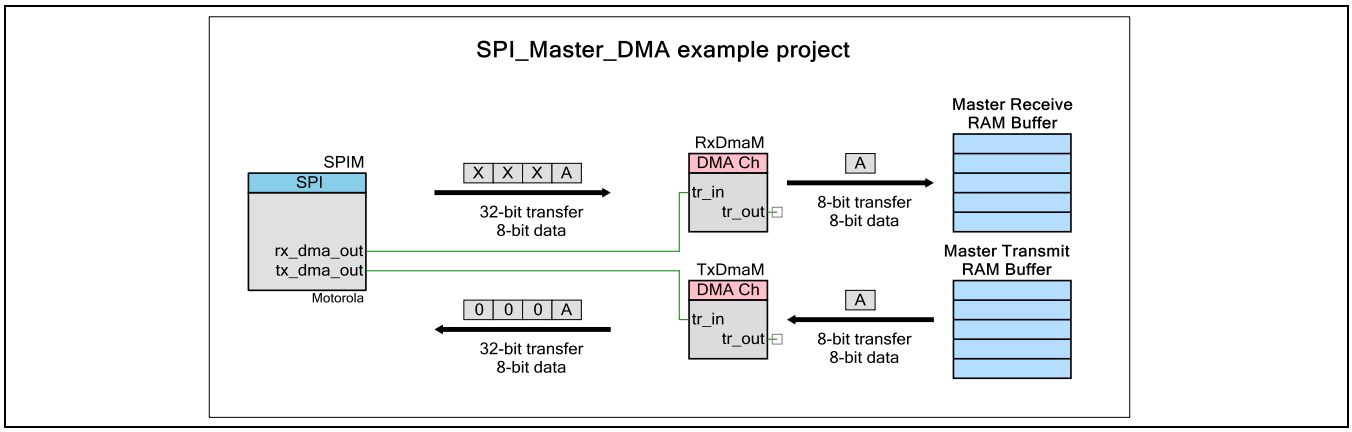

Figure 3 SPI_Master_DMA Top Design Schematic

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)	SPIM	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	TxDmaM	Handles DMA Tx data transfer – master to slave	See **Figure 7**

	RxDmaM	Handles DMA Rx data transfer – slave to master	See **Figure 8**

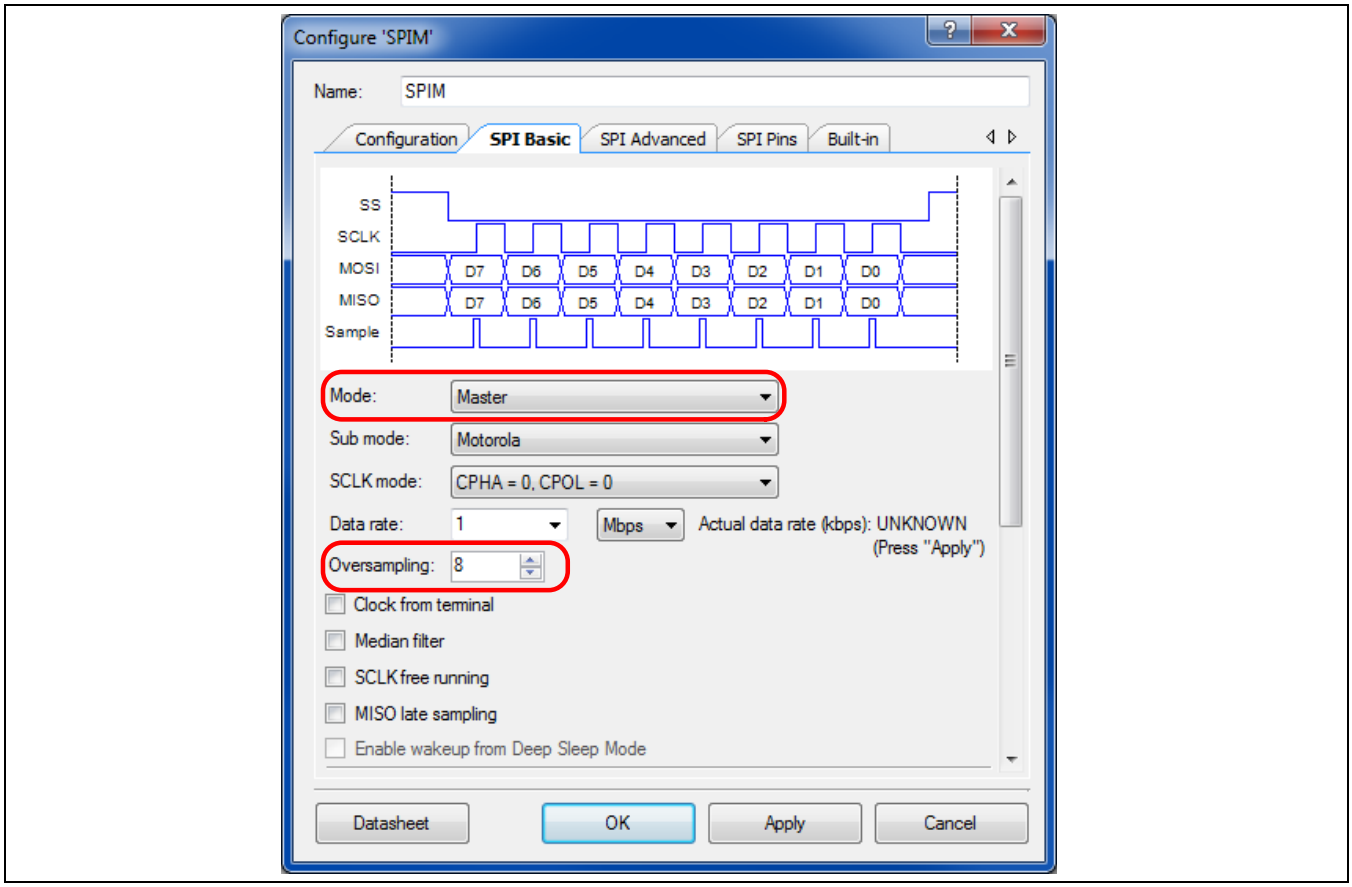

Figure 4 SPI Basic Tab Configuration for All Projects

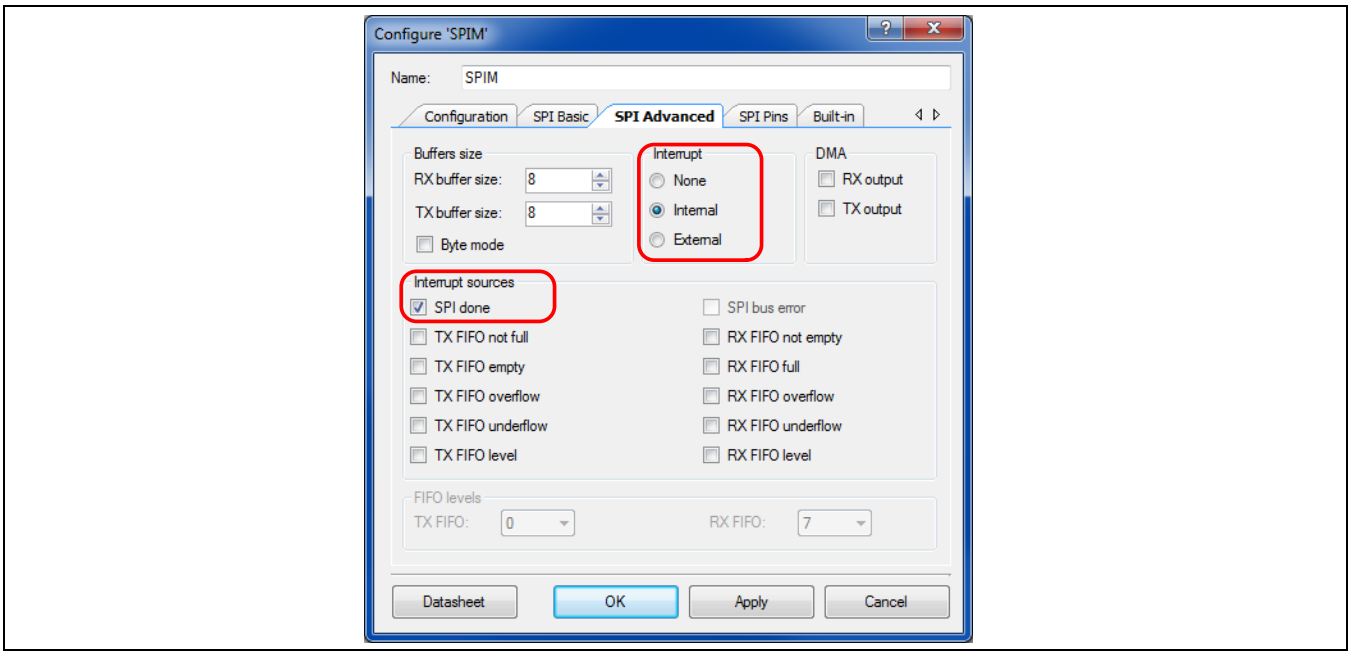

Figure 5 SPI Advanced Tab for Interrupt Project

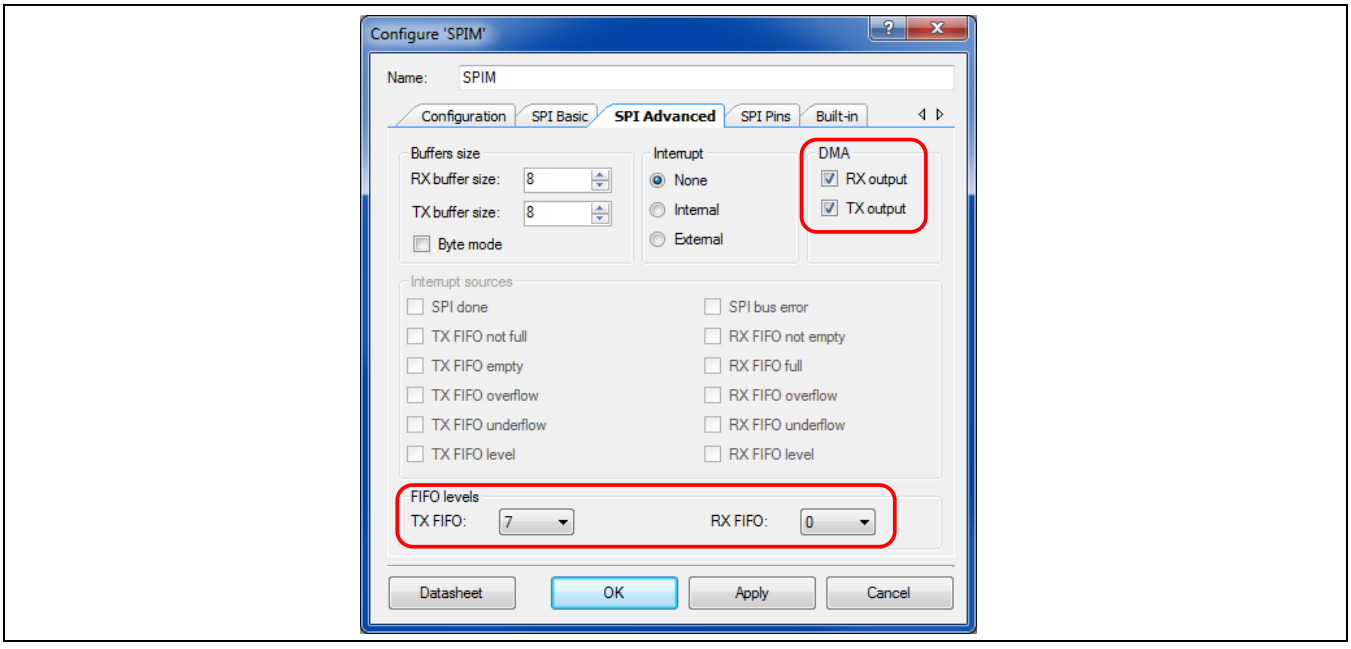

Figure 6 SPI Advanced Tab for DMA Project

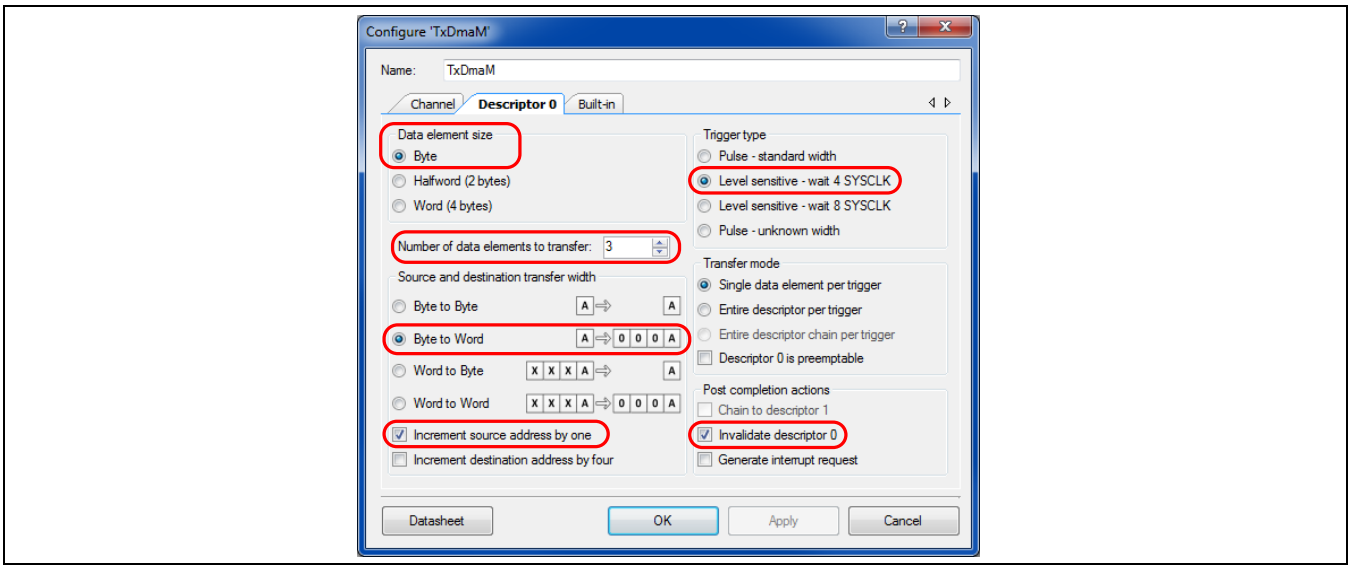

Figure 7 TxDmaM configuration Settings

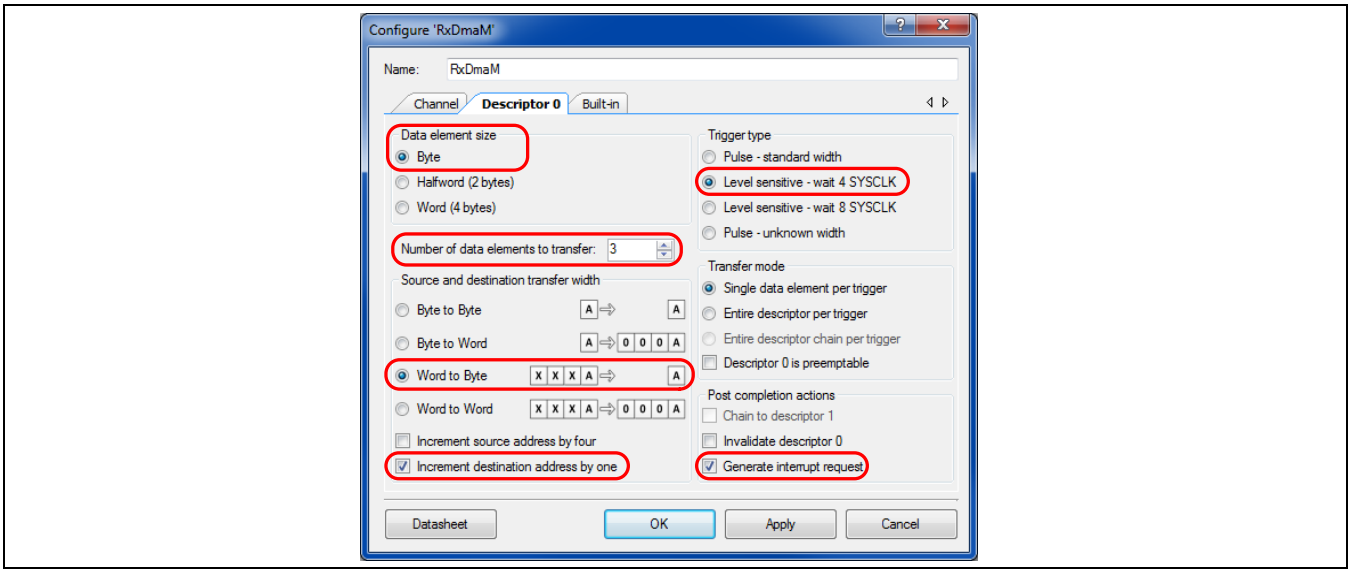

Figure 8 RxDmaM Configuration Settings

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.

Design and Implementation

- 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. 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 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] **CE224463** – PSoC 4 SPI Slave: 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	Updated code example template, updated description of DMA design and implementation.
*B	2020-11-19	Updated the code example to support CY8CKIT-045S.

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

Edition 2020-11-19

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to cypress.com/support

Document reference
002-24339 Rev. *B

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. 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) with respect to any and all information given in this application note.

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.