Objective

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 4.2

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 PSoC 4 Pioneer Kit
- DMA project: CY8CKIT-044 PSoC 4 M-Series Pioneer Kit

Overview

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

Hardware Setup

This code example requires two kits. A kit listed in Related Hardware runs this slave example; the second 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-42 kit	P3[1]	P3[0]	P0[6]	P0[7]	GND
DMA: CY8CKIT-044 kit	P2[1]	P2[0]	P2[2]	P0[7]	GND

Software Setup

None.

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 into 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.

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.

SPI Slave Polling

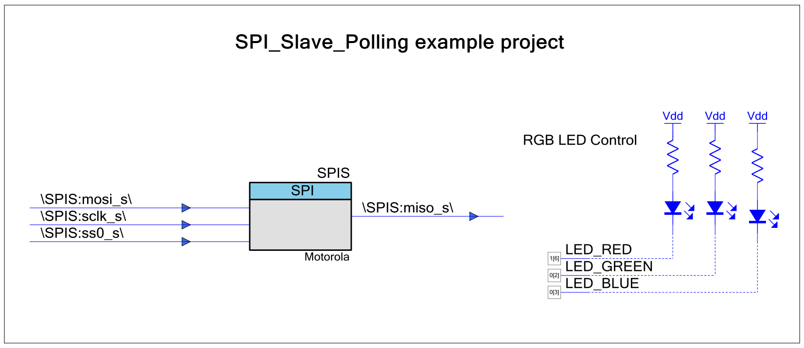
Note that in the SPI protocol, data is sent simultaneously in both directions: master to slave and slave to master.

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

Figure 1. SPI_Slave_Polling Top Design Schematic



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:

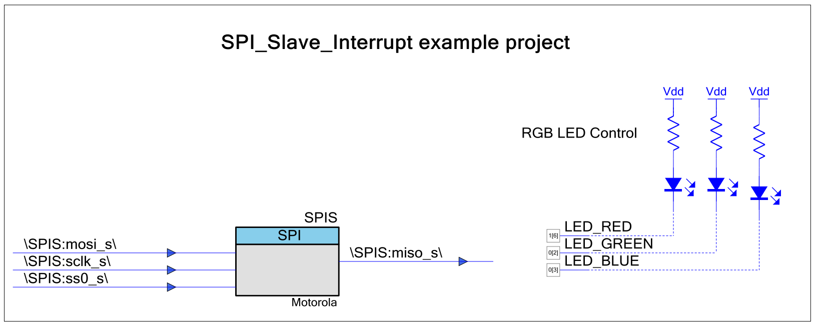
1. The SCB Component is configured as an SPI slave.
2. An interrupt handler function `Slave_ISR()` sets a global variable flag every time it is called. See Slave ISR traits.
3. When the flag is set by the ISR, the flag is reset in main and the SPI slave:
   a. Reads the data from the Rx buffer.
   b. Sets the LED to the color in the command data byte.
   c. Updates the status and loads it into the Tx buffer to be sent to the master.
   d. Waits until it receives dummy data from the master so that it can send what is currently in the Tx buffer.

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 PSoC Creator project:

Figure 2. SPI_Slave_Interrupt Top Design Schematic



SPI Slave DMA

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

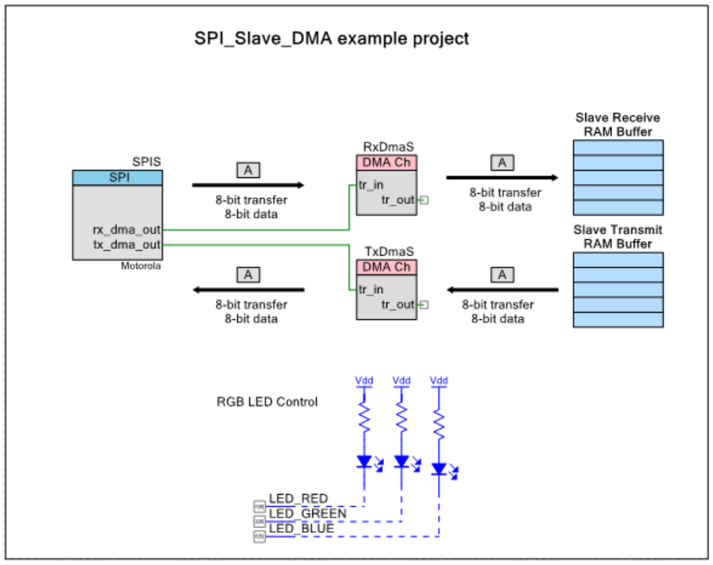
In the SPI_Slave_DMA example, the following functions are performed:

1. The SCB Component is configured as an SPI slave.
2. The DMA transmit and receive channels are initialized and receive 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 PSoC Creator project:

Figure 3. SPI_Slave_DMA Top Design Schematic



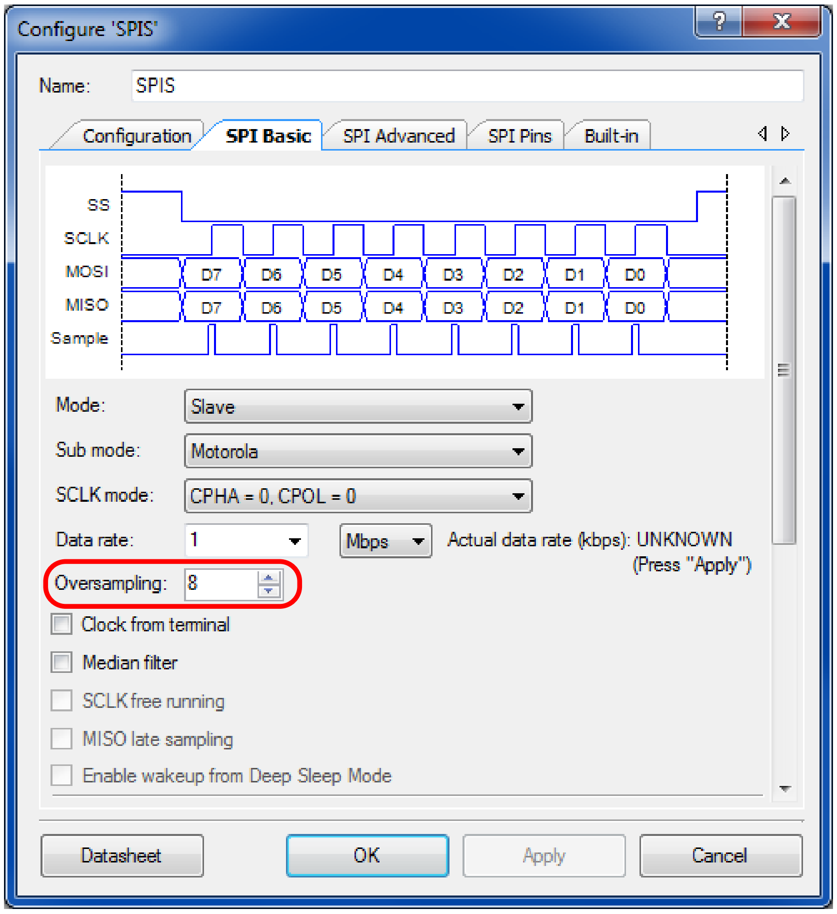
Components and Settings

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	RxDmaM	Handles DMA Rx data transfer – slave to master	See Figure 7
	TxDmaM	Handles DMA Tx data transfer – master to slave	See Figure 8

Figure 4. SPI Basic Tab Configuration for All Projects



Name: SPIS

Mode: Slave

Sub mode: Motorola

SCLK mode: CPHA = 0, CPOL = 0

Data rate: 1 Mbps

Actual data rate (kbps): UNKNOWN (Press "Apply")

Oversampling: 8

Clock from terminal

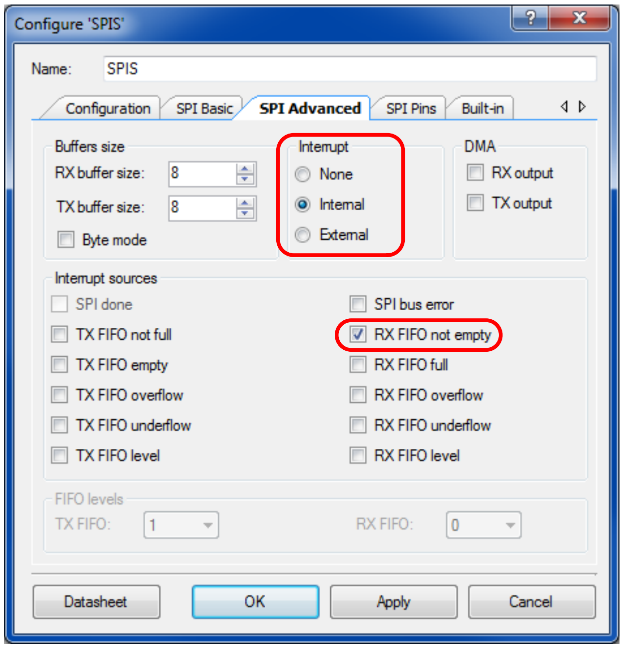
Median filter

SCLK free running

MISO late sampling

Enable wakeup from Deep Sleep Mode

Figure 5. SPI Advanced Tab for Interrupt Project



Configure 'SPIS'

Name: SPIS

Configuration | SPI Basic | **SPI Advanced** | SPI Pins | Built-in

Buffers size
- RX buffer size: 8
- TX buffer size: 8
- [ ] Byte mode

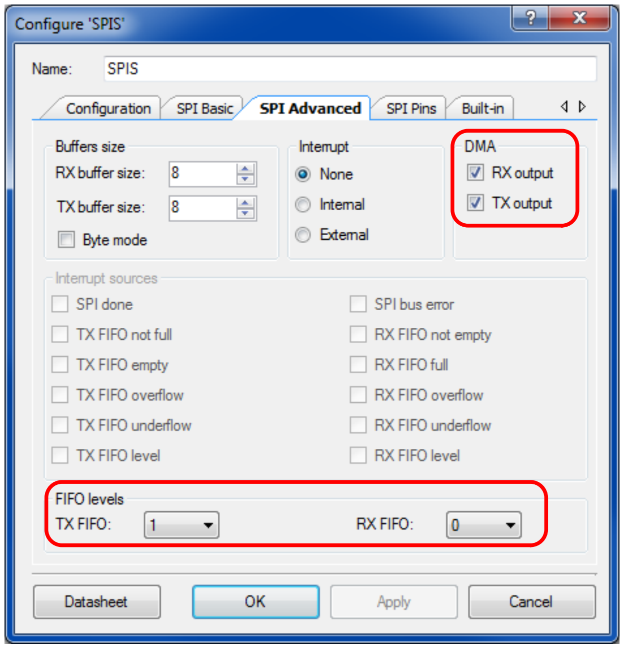
Interrupt
- ( ) None
- [✓ Internal]
- ( ) External

DMA
- [ ] RX output
- [ ] TX output

Interrupt sources
- [ ] SPI done
- [ ] TX FIFO not full
- [ ] TX FIFO empty
- [ ] TX FIFO overflow
- [ ] TX FIFO underflow
- [ ] TX FIFO level
- [ ] SPI bus error
- [✓] RX FIFO not empty
- [ ] RX FIFO full
- [ ] RX FIFO overflow
- [ ] RX FIFO underflow
- [ ] RX FIFO level

FIFO levels
- TX FIFO: 1
- RX FIFO: 0

Figure 6. SPI DMA Settings



Configure 'SPIS'

Name: SPIS

Configuration | SPI Basic | **SPI Advanced** | SPI Pins | Built-in

Buffers size
- RX buffer size: 8
- TX buffer size: 8
- [ ] Byte mode

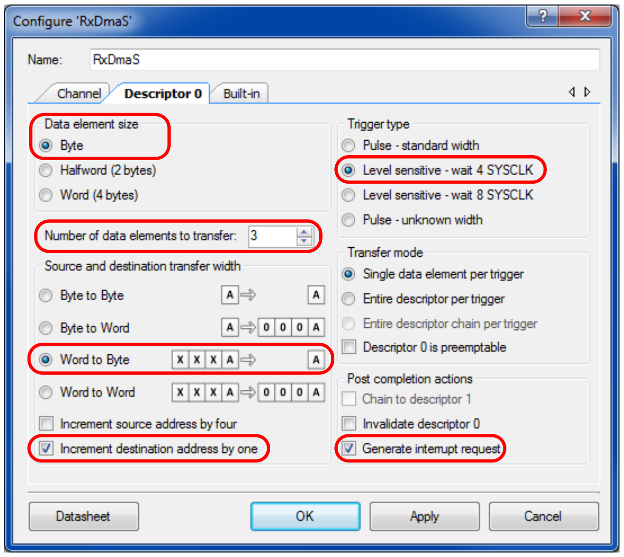
Interrupt
- [✓ None]
- ( ) Internal
- ( ) External

DMA
- [✓] RX output
- [✓] TX output

Interrupt sources
- [ ] SPI done
- [ ] TX FIFO not full
- [ ] TX FIFO empty
- [ ] TX FIFO overflow
- [ ] TX FIFO underflow
- [ ] TX FIFO level
- [ ] SPI bus error
- [ ] RX FIFO not empty
- [ ] RX FIFO full
- [ ] RX FIFO overflow
- [ ] RX FIFO underflow
- [ ] RX FIFO level

FIFO levels
- TX FIFO: 1
- RX FIFO: 0

Figure 7. RxDmaS Configuration Settings



Configure 'RxDmaS'

Name: RxDmaS

Channel | **Descriptor 0** | Built-in

Data element size
- [✓ Byte]
- ( ) Halfword (2 bytes)
- ( ) Word (4 bytes)

Number of data elements to transfer: 3

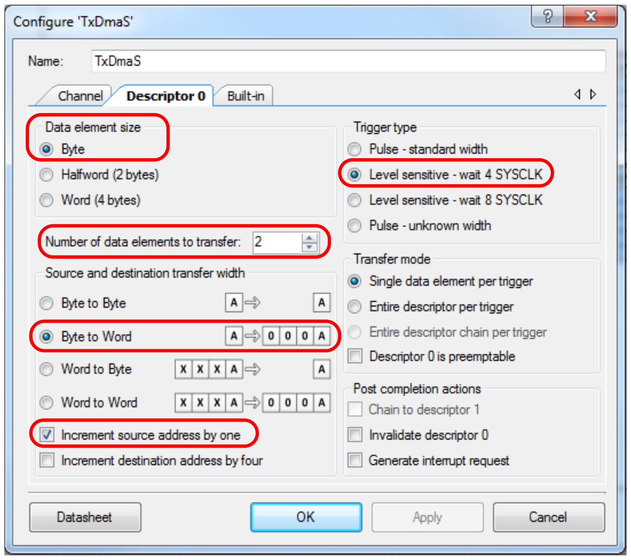
Source and destination transfer width
- ( ) Byte to Byte
- ( ) Byte to Word
- [✓ Word to Byte]
- ( ) Word to Word
- [ ] Increment source address by four
- [✓] Increment destination address by one

Trigger type
- ( ) Pulse - standard width
- [✓ Level sensitive - wait 4 SYSCLK]
- ( ) Level sensitive - wait 8 SYSCLK
- ( ) Pulse - unknown width

Transfer mode
- [✓ Single data element per trigger]
- ( ) Entire descriptor per trigger
- ( ) Entire descriptor chain per trigger
- [ ] Descriptor 0 is preemptable

Post completion actions
- [ ] Chain to descriptor 1
- [ ] Invalidate descriptor 0
- [✓] Generate interrupt request

Figure 8. TxDmaS Configuration Settings



Configure 'TxDmaS'

Name: TxDmaS

Channel | **Descriptor 0** | Built-in

Data element size
- [✓ Byte]
- ( ) Halfword (2 bytes)
- ( ) Word (4 bytes)

Number of data elements to transfer: 2

Source and destination transfer width
- ( ) Byte to Byte
- [✓ Byte to Word]
- ( ) Word to Byte
- ( ) Word to Word
- [✓] Increment source address by one
- [ ] Increment destination address by four

Trigger type
- ( ) Pulse - standard width
- [✓ Level sensitive - wait 4 SYSCLK]
- ( ) Level sensitive - wait 8 SYSCLK
- ( ) Pulse - unknown width

Transfer mode
- [✓ Single data element per trigger]
- ( ) Entire descriptor per trigger
- ( ) Entire descriptor chain per trigger
- [ ] Descriptor 0 is preemptable

Post completion actions
- [ ] Chain to descriptor 1
- [ ] Invalidate descriptor 0
- [ ] Generate interrupt request

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

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 pin layout tab in PSoC Creator.
- 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.

Related Documents

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	
AN79953 – Getting Started with PSoC® 4	Describes PSoC 4 devices and how to build your first PSoC Creator project
Code Examples	
CE224339 – PSoC 4 SPI Master	Companion code example for this code example.
PSoC Creator Component Datasheets	
SCB	Supports I^{2}C, SPI, UART and EZI2C communication protocols using the serial control block (SCB)
DMA	Supports direct memory access (DMA) controllers
Device Documentation	
PSoC 4 Datasheets	PSoC 4 Technical Reference Manuals
Development Kit Documentation	
CY8CKIT-042 PSoC 4 Pioneer Kit	
CY8CKIT-044 PSoC 4 M-Series Pioneer Kit	
PSoC 4 Kits	
Tool Documentation	
PSoC Creator	Look in the **Downloads** tab for Quick Start and User Guides

Document History

Document Title: CE224463 – PSoC 4 SPI Slave

Document Number: 002-24463

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6260029	NRSH	07/24/2018	New code example
*A	6671851	NRSH	09/13/2019	Update to new code example template and updated Figure 3.

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