CE221120 - PSoC 6 MCU SPI Master

Objective

This example demonstrates the use of SPI SCB (Serial Communication Block) Component for PSoC® 6 MCU in master mode. Four different subprojects show the use of Peripheral Driver Library (PDL) functions to communicate with an SPI slave.

Overview

The SPI is designed to send command packets to control the RGB LED color on the slave. Four different projects developed in this example are: SPI master using high-level PDL functions, SPI master using low-level PDL functions, SPI master using User-ISR and low-level PDL functions and SPI master using low-level PDL functions and DMA.

Requirements

Tool: PSoC Creator™ 4.2; Peripheral Driver Library (PDL) 3.0.1

Programming Language: C (Arm® GCC 5.4.1 and Arm MDK 5.22)

Associated Parts: All PSoC 6 MCU parts

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

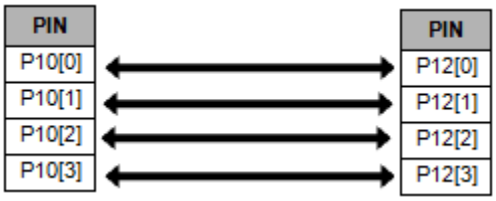
Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure the kit is configured correctly. If the settings are different from the default values, see the 'Selection Switches' table in the kit guide to reset to the default settings.

Table 3 lists the PSoC Creator pin connection settings required on the CY8CKIT-062-BLE Kit. Because the master and slave are on the same device, pins related to both Components are shown in Table 3.

Jumper wires of the same length are used to establish connection between the master and slave on CY8CKIT-062-BLE Kit. P10[0] is connected to P12[0], P10[1] is connected to P12[1], P10[2] is connected to P12[2] and P10[3] is connected to P12[3] as shown in Figure 1.

Figure 1. Pin Connection



Software Setup

None.

Operation

1. Plug the CY8CKIT-062-BLE kit board into your computer's USB port.
2. Connect jumper wires as explained in hardware setup.
3. Build each project and program it into the PSoC 6 MCU device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation.

4. Observe the RGB LED on the board, which changes its color every two seconds. Color changes in the sequence red, green, blue, cyan, purple, yellow, white. After white, the same sequence from red continues.

Design

In all four projects, the Arm Cortex®-M4 (CM4) core acts as a master and the Cortex-M0+ (CM0+) core acts as a slave. Different pins are configured for SPI MOSI, MISO, SCLK, and SS for master and slave. The master sends command packets to control the color of an RGB LED connected to the slave. This document explains the SPI master-related design. CE221121 explains the slave design.

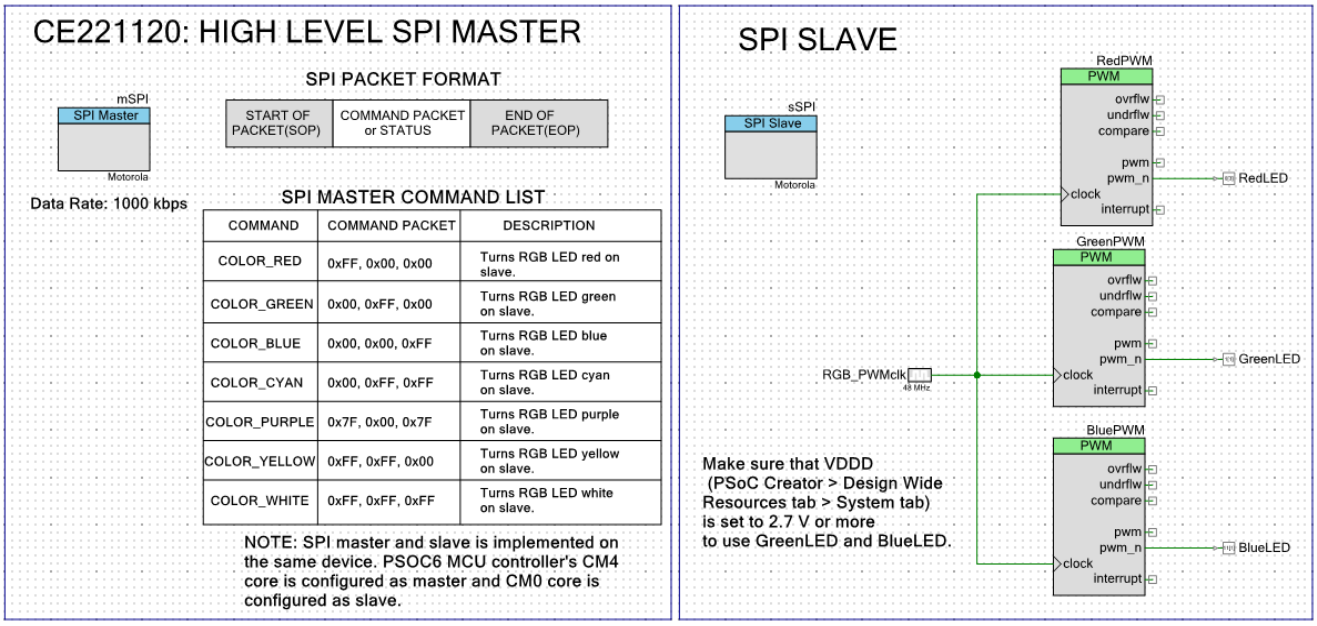
Master APIs are divided into two categories: **High-Level** and **Low-Level**. See the PDL documentation to know more about High-Level and Low-Level functions. To open the PDL documentation, right-click on the SPI Component in PSoC Creator schematics window, and click **Open PDL Documentation.**

The SCB SPI PSoC Creator Component is used in all the example projects. The master sends different command packets to the slave every two seconds. A command packet has the information to set the compare value for three PWM signals that controls the color of the RGB LED connected to the slave. The slave updates its TX buffer with the status packet. Master sends command to read status for the command sent previously. The first byte of the status packet is Start of Packet (SOP), the second byte contains the status where the value 0x00 means success and 0x1F means failure for the command sent by master.

SPI Master Using High-Level Functions

The SPI master shown in Figure 2 has the mSPI (SCB_SPI_PDL) Component configured for master mode and sSPI (SCB_SPI_PDL) Component configured for slave mode at 1000-kbps speed. The SPI master firmware uses high-level PDL functions to communicate with the slave.

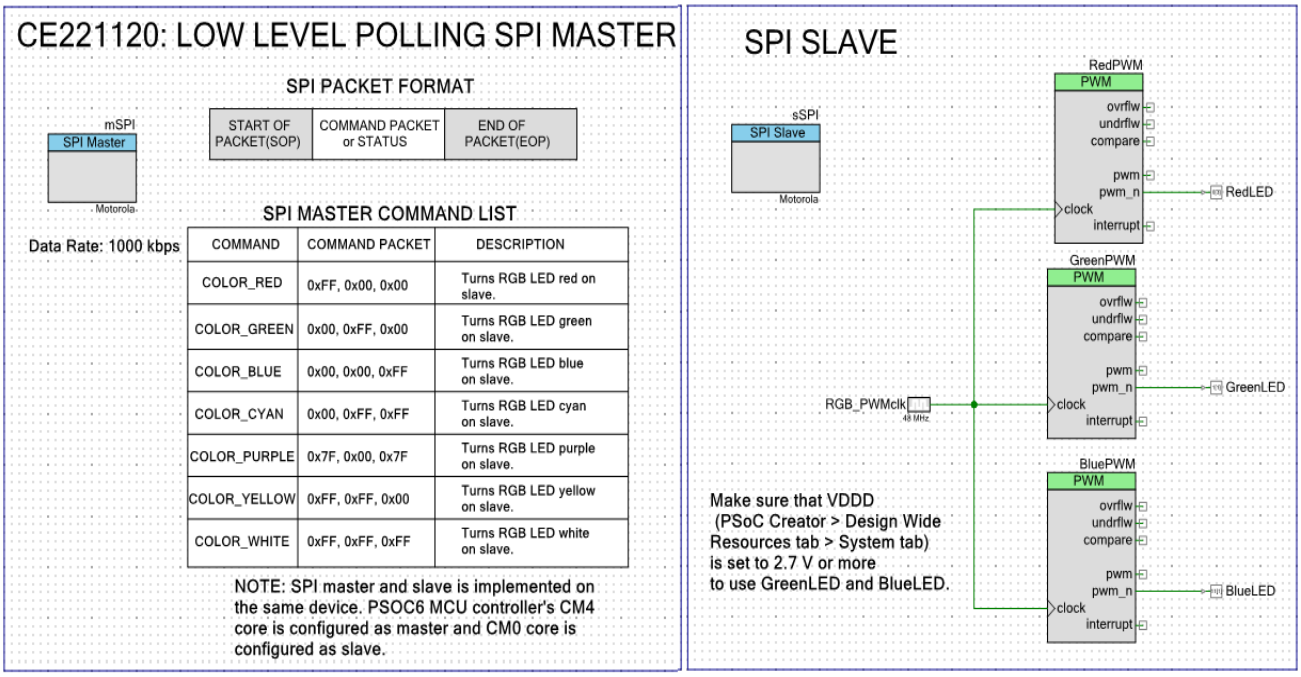
Figure 2. SPI Master and Slave Schematic for High-Level Design



SPI Master Using Low-Level Polling

The SPI master shown in Figure 3 has the mSPI (SCB_SPI_PDL) Component configured for master mode and sSPI (SCB_SPI_PDL) Component configured for slave mode at 1000-kbps speed. This project uses low-level PDL functions to communicate with the slave. It polls for status, instead using an interrupt.

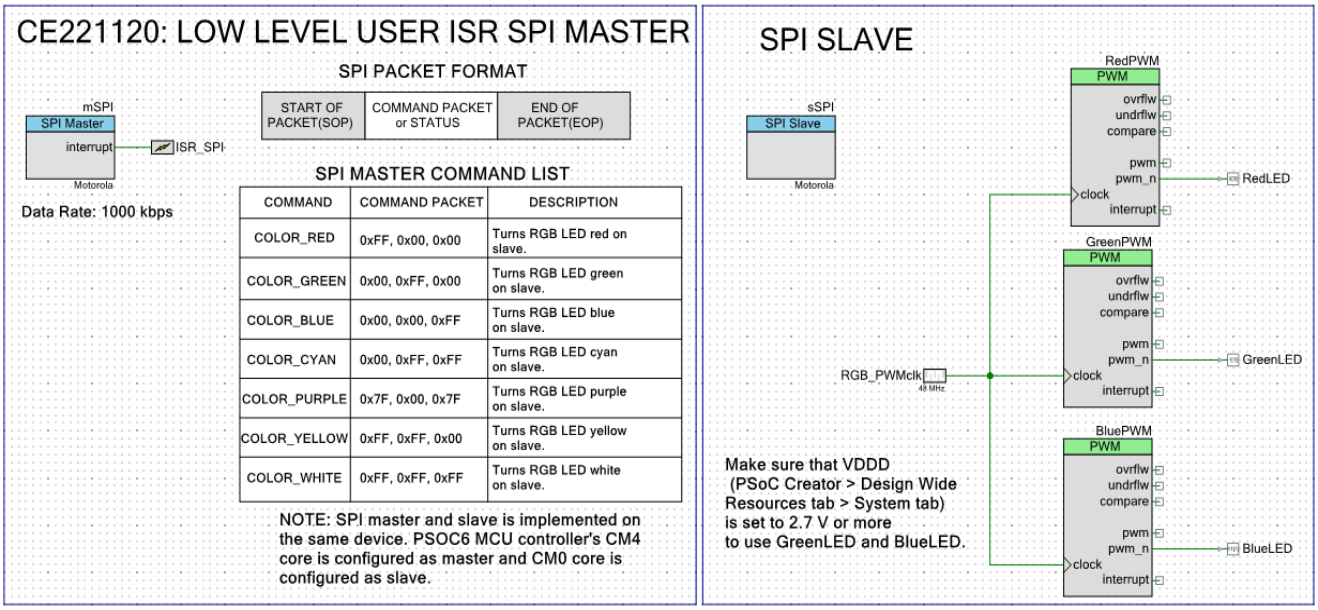
Figure 3. SPI Master and Slave Schematic for Low-Level Design



SPI Master Using Low-Level User ISR

The SPI master shown in Figure 4 has the mSPI (SCB_SPI_PDL) Component configured for master mode and sSPI (SCB_SPI_PDL) Component configured for slave mode at 1000-kbps speed. It uses low-level PDL functions to communicate with the slave. The design uses an interrupt to monitor the master's status to know whether the master has transmitted all data.

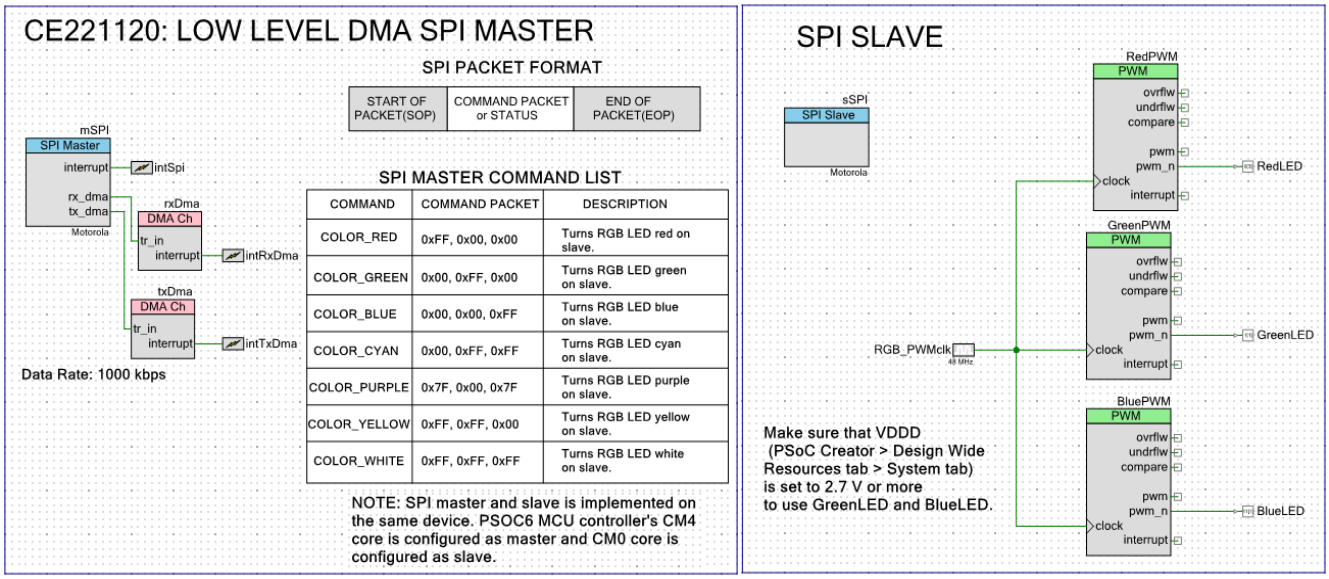
Figure 4. SPI Master and Slave Schematic for Low-Level User ISR Design



SPI Master Using Low-Level DMA

The SPI master shown in Figure 5 has the mSPI (SCB_SPI_PDL) Component configured for master mode and sSPI (SCB_SPI_PDL) Component configured for slave mode at 1000-kbps speed. This example shows how to use DMA to transfer data from a RAM array to the SPI TX buffer, and shows how to use DMA to transfer data from the SPI RX buffer to a RAM array. DMA components txDma and rxDma are configured to handle the data from SPI Tx and Rx FIFO.

Figure 5. SPI Master and Slave Schematic for Low-Level DMA



Components and Settings

Table 1 lists the PSoC Creator Components used in four sub-examples, how they are used in the design and the hardware resources used by each Component. Non-default settings for each Component are listed in Table 2. Interrupts to be enabled are shown in Table 4, Table 5 and Table 6.

Table 1. PSoC Creator Components

Component	Instance Name	Purpose	Hardware Resources
SPI (SCB_SPI_PDL)	mSPI, sSPI	Provides SPI master and slave connection	Two SCB peripheral blocks
DMA(DMA_PDL)	txDma, rxDma	Provides direct memory access for SPI Master.	Two DMA peripheral blocks
System Interrupt(SysInt)	intSpi, intTxDma, intRxDma, ISR_SPI	Configure the interrupt	Four Interrupt Component
TCPWM (TCPWM_PWM_PDL)	RedPWM, GreenPwm, BluePWM	Generate PWM signals	Three TCPWM peripheral blocks
Clock(SysClk_PDL)	RGB_PWMclk	Generate clock for TCPWM	One Clock peripheral
GPIO(GPIO_PDL)	RedLED, GreenLED, BlueLED	Provides connection for LED's	Three GPIO peripherals

Table 2. Parameter Settings

Component	Instance Name	Non-default Parameter Settings
SPI (SCB_SPI_PDL)	mSPI (For High Level and Low Level Polling)	Tab Basic- Mode: Master, RX Data Width: 8, TX Data Width: 8.
SPI (SCB_SPI_PDL)	mSPI (For Low Level User ISR)	Tab Basic- Mode: Master, RX Data Width: 8, TX Data Width: 8. Tab Advanced- Interrupt: External, Master Interrupt Source (Checked boxes): SPI Done.
SPI (SCB_SPI_PDL)	mSPI (For Low Level DMA)	See Figure 6.
SPI (SCB_SPI_PDL)	sSPI	RX Data Width: 8, TX Data Width: 8.
TCPWM (TCPWM_PWM_PDL)	RedPWM, GreenPWM, BluePWM	Period 0: 255u, Compare 0 : 0u
Clock(SysClk_PDL)	RGB_PWMclk	Frequency: 48 MHz
DMA(DMA_PDL)	txDma	See Figure 7.
DMA(DMA_PDL)	rxDma	See Figure 8.

Figure 6. Low -Level DMA SPI Master Parameter Settings

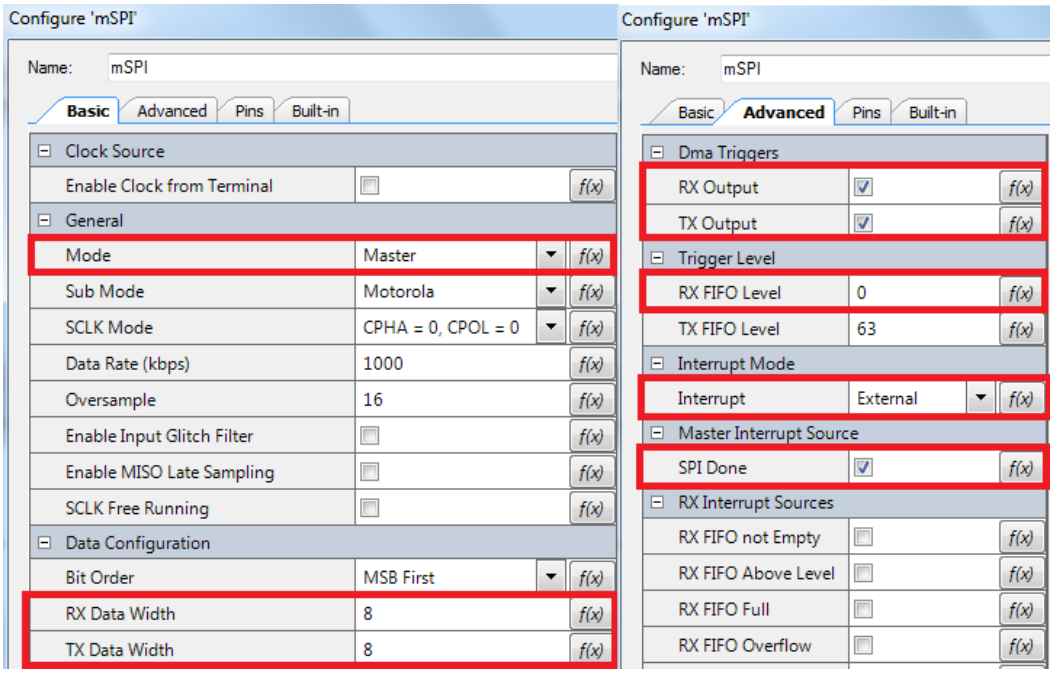


Figure 7. txDMA Parameter Settings

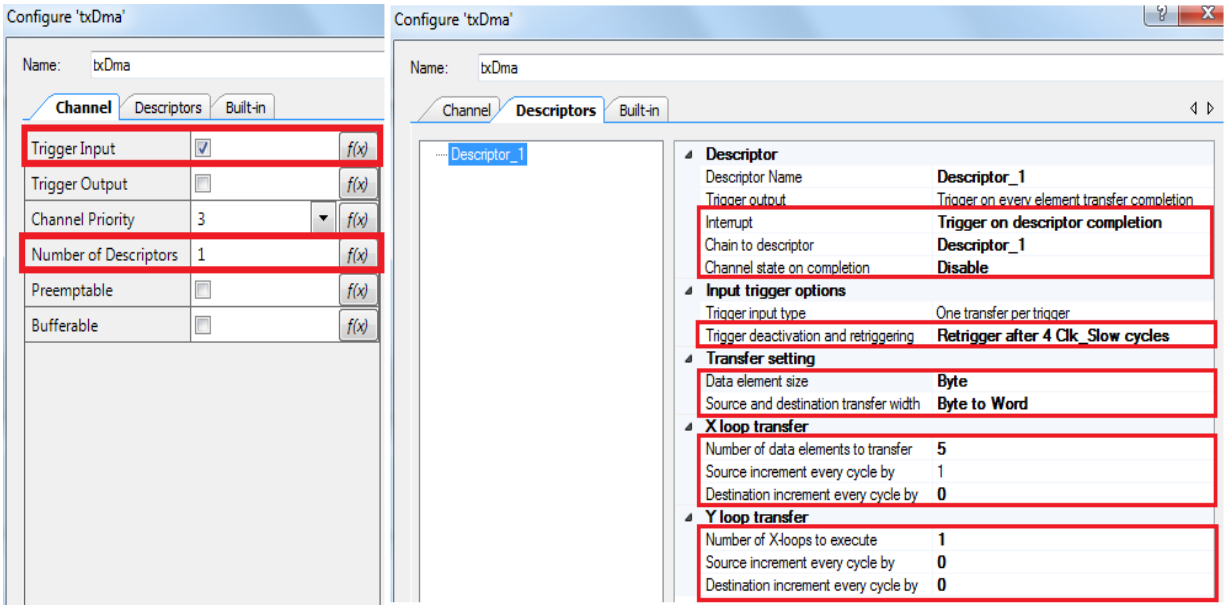
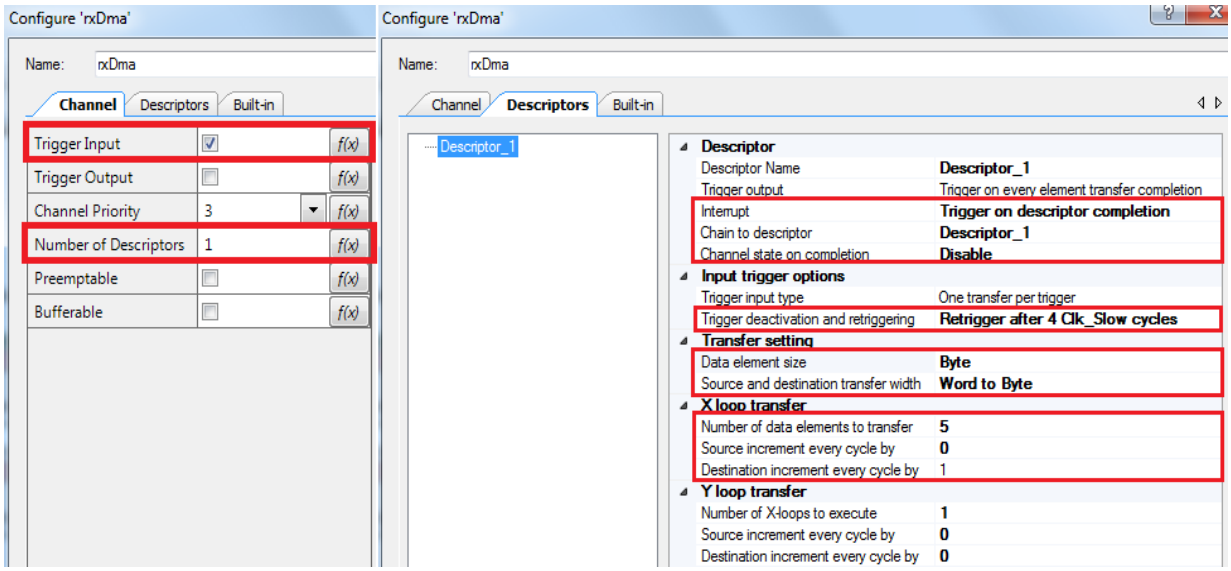


Figure 8. rxDMA Parameter Settings



Design-Wide Resources

Make sure that V_{DDD} (**PSoC Creator** > **Design Wide Resources** tab > **System** tab) is set to 2.7 V or more to use greenLED and blueLED.

Table 3 shows the pin assignment for the code example.

Table 3. Pin Names and Location

Pin Name	Location
mSPI:miso_m	P10[1]
mSPI:mosi_m	P10[0]
mSPI:sclk_m	P10[2]
mSPI:sso_m	P10[3]
sSPI:miso_s	P12[1]
sSPI:mosi_s	P12[0]
sSPI:sclk_s	P12[2]
sSPI:sso_s	P12[3]
RedLED	P0[3]
GreenLED	P1[1]
BlueLED	P11[1]

Table 4, Table 5 and Table 6 show the interrupts to be enabled and priority to be set for different SPI Master sub- projects.

Table 4. Interrupt Settings for High- and Low-Level Polling Master Design

Instance Name	Interrupt Number	CM0Enable	CM0Priority(1-3)	CM0Vector(3-29)	CM4Enable	CM4Priority(0-7)
mSPI_SCB_IRQ	42	▫	–	–	✓	7
sSPI_SCB_IRQ	47	✓	3	9	▫	–

Table 5. Interrupt Settings for Low-Level User ISR Master Design

Instance Name	Interrupt Number	CM0Enable	CM0Priority(1-3)	CM0Vector(3-29)	CM4Enable	CM4Priority(0-7)
ISR_SPI	42	▫	–	–	✓	7
sSPI_SCB_IRQ	47	✓	3	9	▫	–

Table 6. Interrupt Settings for Low- Level DMA Master Design

Instance Name	Interrupt Number	CM0Enable	CM0Priority(1-3)	CM0Vector(3-29)	CM4Enable	CM4Priority(0-7)
intRxDma	51	▫	–	–	✓	7
intSpi	42	▫	–	–	✓	7
intTxDma	50	▫	–	–	✓	7
sSPI_SCB_IRQ	47	✓	3	9	▫	–

Reusing This Example

This example is designed for the CY8CKIT-062-BLE pioneer kit. To port the design to a different PSoC 6 MCU device and/or kit, 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 (for example, an IP block) 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.

SPI master projects designed in this example can be used to communicate with other slave devices not located on the same board.

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 63 with Bluetooth Low Energy (BLE) Connectivity and how to build your first PSoC Creator project
PSoC Creator Component Datasheets	
SPI	Supports SPI communication
TCPWM	Supports PWM Signal Generation
Clock	Supports clock signal Generation
GPIO	Supports Analog, Digital I/O and Bidirectional signal types
DMA	Supports up to 16 DMA channels
SysInt	Interrupt vectoring and control
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kit (DVK) Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	

Document History

Document Title: CE221120 – PSoC 6 MCU SPI Master

Document Number: 002-21120

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5894789	VJYA	01/15/2018	New Code Example

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

Forums | WICED IOT Forums | 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.