

Objective

This example demonstrates the use of I2C SCB (Serial Control Block) Component for PSoC® 6 MCU in master mode. Three different subprojects show the use of Peripheral Driver Library (PDL) functions to communicate with I2C and EzI2C slave.

Overview

The I2C master for PSoC 6 MCU is designed to send command packets to control the RGB LED color on the slave. Three different projects developed in this example are: I2C master using high-level PDL functions, I2C master using low-level PDL functions, and I2C master communication with EzI2C slave.

Requirements

Tool: PSoC Creator™ 4.2

Programming Language: C (ARM® GCC 5.4-2016-q2-update, ARM MDK 5.22)

Associated Parts: All PSoC 6 MCU parts

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

Design

In all three 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 SCL and SDA for master and slave. Master sends command packets to control the color of an RGB LED connected to the slave. In this document, master-related Components are explained.

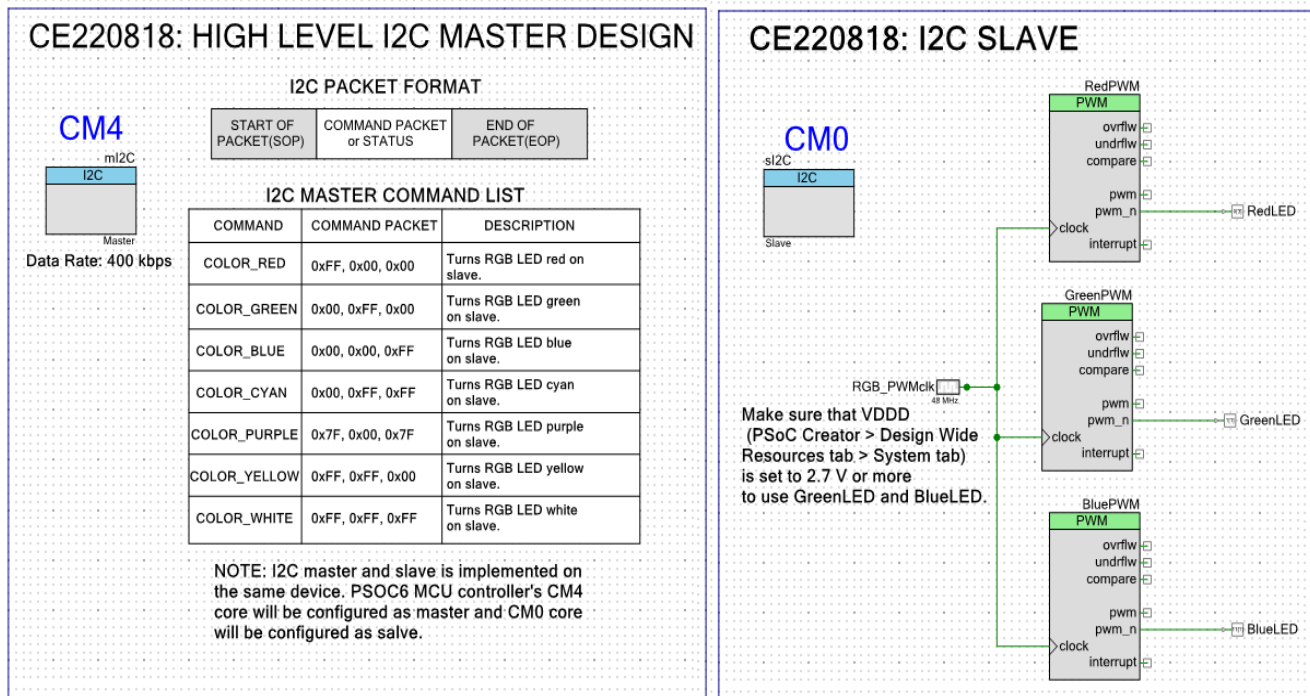
The master APIs are divided into two categories: **Master High-Level** and **Master Low-Level**. Refer PDL documentation to know more about **High-Level** and **Low-Level** functions. To open PDL documentation, right click on the I2C Component in PSoC Creator schematics window and click **Open PDL Documentation**.

The SCB I2C PSoC Creator Component is used in all three I2C master 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.

I2C Master Using High-Level Functions

The I2C master shown in [Figure 1](#) has mI2C (SCB_I2C_PDL) Component configured for master mode and sI2C (SCB_I2C_PDL) Component configured for slave mode at 400-kbps speed. I2C master design uses high-level PDL functions to communicate with the slave.

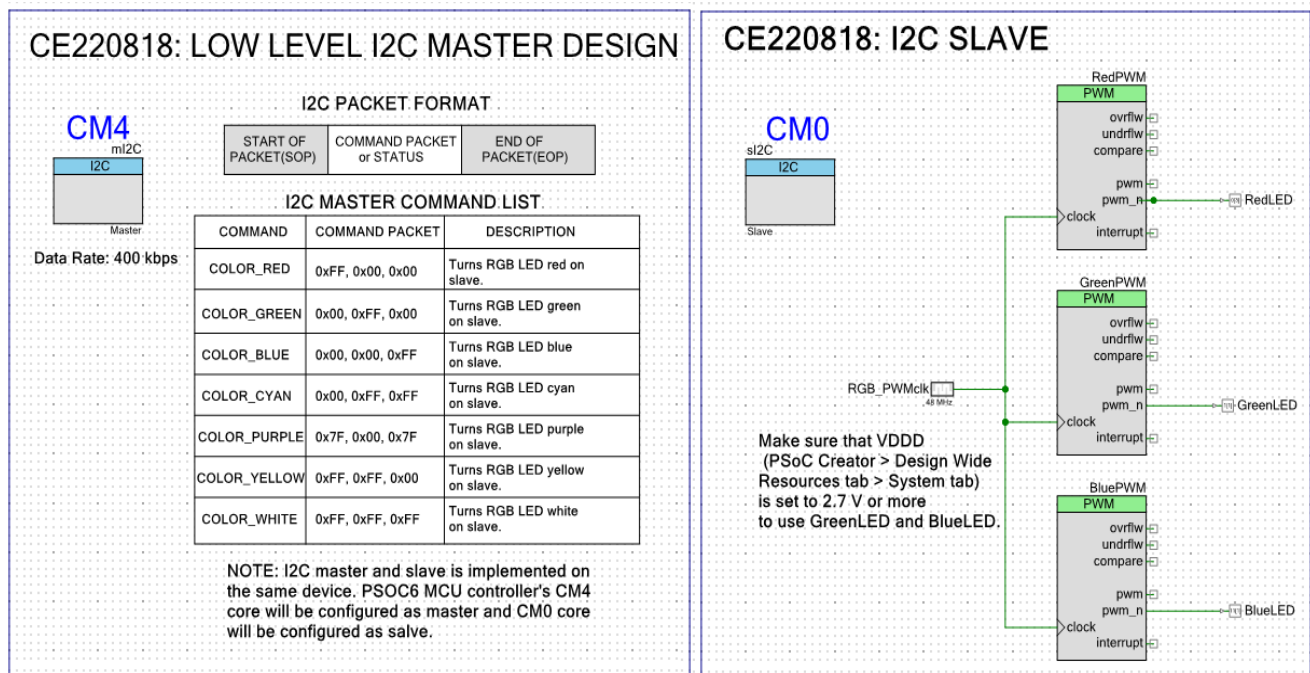
Figure 1. I2C Master and Slave Schematic for High-Level Design



I2C Master Using Low-Level Functions

The I2C master shown in [Figure 2](#) has mI2C (SCB_I2C_PDL) Component configured for master mode and sI2C (SCB_I2C_PDL) Component configured for slave mode at 400-kbps speed. It uses low-level PDL functions to communicate with the slave.

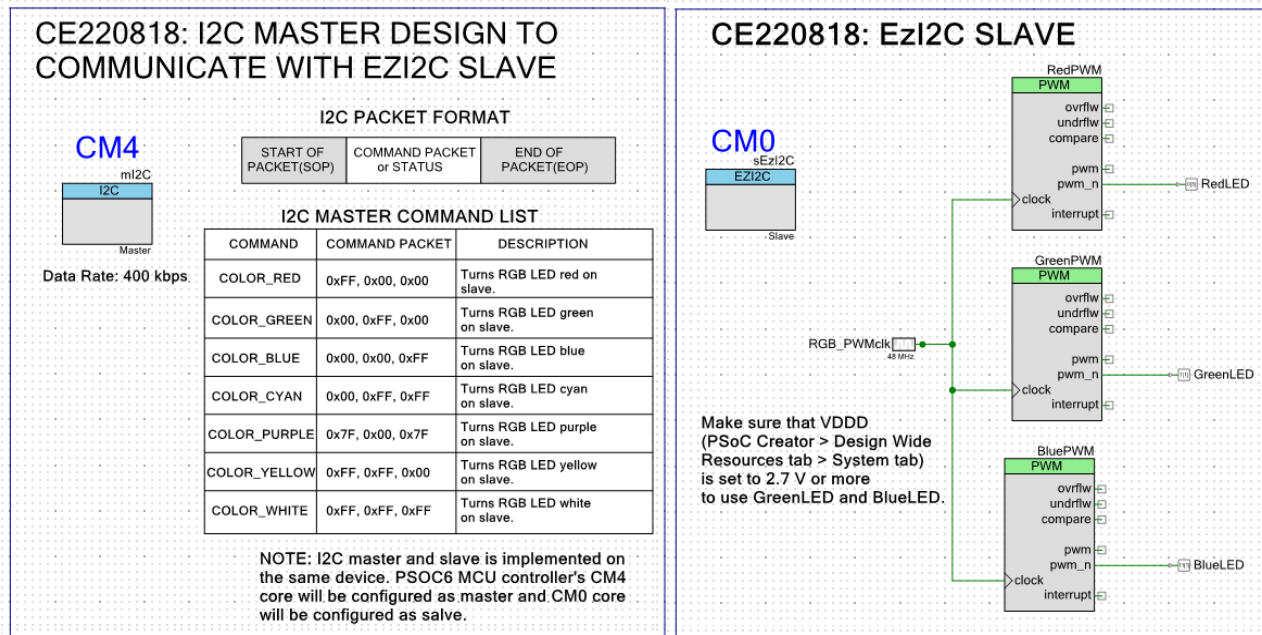
Figure 2. I2C Master and Slave Schematic for Low-Level Design



I2C Master for Communication with EzI2C Slave

The I2C master shown in [Figure 3](#) has the mI2C (SCB_I2C_PDL) Component configured for master mode and sEzI2C (SCB_EZ_I2C_PDL) Component configured for slave mode at 400-kbps speed. PDL functions are used to communicate with the EzI2C slave.

Figure 3. I2C Master and EzI2C Slave Schematic



Design Considerations

This code example is designed to run on CY8CKIT-062-BLE with PSoC 6 MCU. To port the design to other devices and kits, you must change the target device in Device Selector, and change the pin assignments in the *cydwr* settings.

I2C master projects designed in this example can be used to communicate with other slave devices not located on the same board. Interrupts to be enabled are shown in [Table 3](#).

Hardware Setup

The code example works with the default settings on the CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit. 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 2](#) lists the PSoC Creator pin connection settings required on the CY8CKIT-062-BLE Kit. Since the master and slave are on the same device, pins related to both Components are shown in [Table 2](#).

Jumper wires are used to establish connection between the master and slave on CY8CKIT-062-BLE Kit. P6[0] is connected to P9[0] and P6[1] is connected to P9[1].

Operation

1. Connect CY8CKIT-062-BLE to a USB port on your PC.
2. Connect jumper wires as explained in hardware setup.
3. Build and program each I2C master project into CY8CKIT-062-BLE. For more information on building a project or programming a device, see PSoC Creator Help.
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.

Components

Table 1 lists the PSoC Creator Components used in all three sub-examples and the hardware resources used by each Component.

Table 1. PSoC Creator Components.

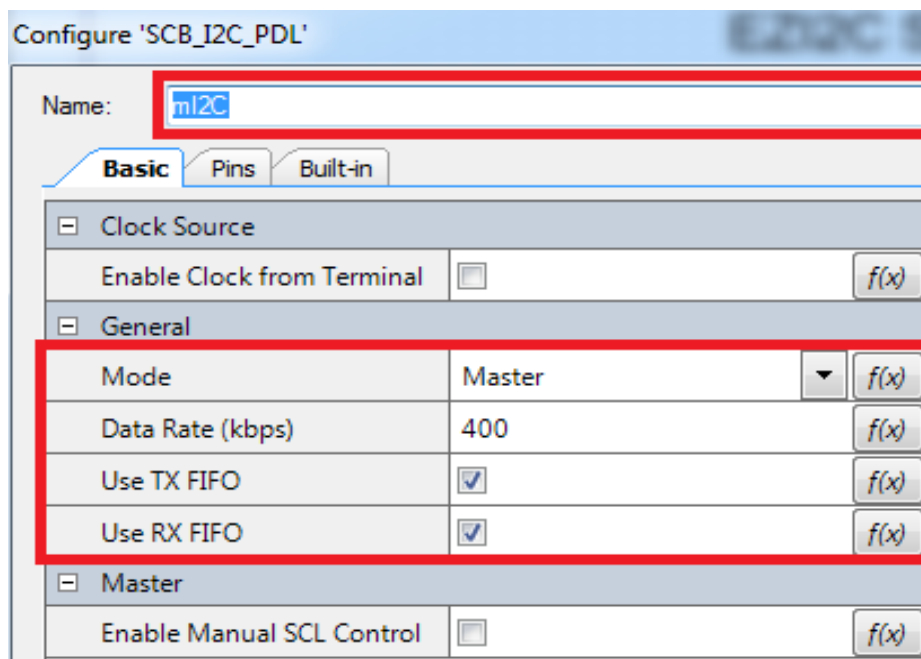
| Component | Instance Name | Hardware Resources |
|----------------------|---------------|-----------------------------|
| I2C (SCB_I2C_PDL) | mI2C, sI2C | Two SCB peripheral blocks |
| EzI2C(SCB_EzI2C_PDL) | sEzI2C | Single SCB peripheral block |

Parameter Settings

Non-default settings for each Component are outlined in red in the following figures.

Figure 4 shows the master I2C Component parameter settings. Same settings are used in all the three projects.

Figure 4. I2C Master Component Parameter Settings



Configure 'SCB_I2C_PDL'

Name: **mI2C**

Basic Pins Built-in

☐ Clock Source

Enable Clock from Terminal ☐ f(x)

☐ General

Mode Master ▼ f(x)

Data Rate (kbps) 400 f(x)

Use TX FIFO ☒ f(x)

Use RX FIFO ☒ f(x)

☐ Master

Enable Manual SCL Control ☐ f(x)

Figure 5 shows the I2C slave Component parameter settings. Same settings are used in for the projects: I2C master using high-level functions, I2C master using low-level functions.

Figure 6 shows the EzI2C slave Component parameter settings for the project I2C master communication with EzI2C slave.

Figure 5. I2C Slave Component Parameter Settings

Configure 'SCB_I2C_PDL'

Name: **sI2C**

Basic Pins Built-in

☐ Clock Source

Enable Clock from Terminal ☐ f(x)

☐ General

Mode Slave ▼ f(x)

Data Rate (kbps) 400 f(x)

Use TX FIFO ☒ f(x)

Use RX FIFO ☒ f(x)

☐ Slave

Slave Address (7-bit) 0x24 f(x)

Slave Address Mask (8-bit) 0xFE f(x)

Accept Matching Address in RX FIFO ☐ f(x)

Accept General Call Address ☐ f(x)

Enable Wakeup from Deep Sleep Mode ☐ f(x)

Figure 6. EzI2C Slave Component Parameter Settings

Configure 'SCB_EZI2C_PDL'

Name: **sEzI2C**

Basic Pins Built-in

☐ Clock Source

Enable Clock from Terminal ☐ f(x)

☐ General

Data Rate (kbps) 400 f(x)

Number of Addresses 1 ▼ f(x)

Primary Slave Address (7-bit) 0x24 f(x)

Sub-Address Size 8 bits ▼ f(x)

Enable Wakeup from Deep Sleep Mode ☐ f(x)

Design-Wide Resources

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

Table 2 shows the pin assignment for the code example.

Table 2. Pin Names and Location

| Pin Name | Location |
|----------|----------|
| ml2C:sda | P6[1] |
| ml2C:scl | P6[0] |
| sl2C:sda | P9[1] |
| sl2C:scl | P9[0] |
| RedLED | P0[3] |
| GreenLED | P1[1] |
| BlueLED | P11[1] |

Table 3 and Table 4 show the interrupts to be enabled and priority to be set.

Table 3. Interrupt Settings for High- and Low-Level Master Design

| Instance Name | Interrupt Number | CM0Enable | CM0Priority(1-3) | CM0Vector(3-29) | CM4Enable | CM4Priority(0-7) |
|---------------|------------------|-----------|------------------|-----------------|-----------|------------------|
| ml2C_SCB_IRQ | 44 | □ | – | – | ✓ | 7 |
| sl2C_SCB_IRQ | 43 | ✓ | 3 | 9 | □ | – |

Table 4. Interrupt Settings for CE220818_I2C_Master_EzI2C_Slave.

| Instance Name | Interrupt Number | CM0Enable | CM0Priority(1-3) | CM0Vector(3-29) | CM4Enable | CM4Priority(0-7) |
|----------------|------------------|-----------|------------------|-----------------|-----------|------------------|
| ml2C_SCB_IRQ | 44 | □ | – | – | ✓ | 7 |
| sEzI2C_SCB_IRQ | 43 | ✓ | 3 | 9 | □ | – |

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 | |
| I2C | Supports I ² C communication |
| EzI2C | Supports EzI2C slave communication |
| 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: CE220818 – PSoC 6 MCU I2C Master

Document Number: 002-20818

| Revision | ECN | Orig. of Change | Submission Date | Description of Change |
|----------|---------|-----------------|-----------------|-----------------------|
| ** | 5880339 | VJYA | 09/18/2017 | 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](#)

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, 2017. 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. 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.