

## Objective

This code example demonstrates the implementation of an EZI2C Slave using the SCB resource on the PSoC® 6 MCU, using ModusToolbox™ IDE. It also demonstrates how to control the intensity of an LED using a TCPWM.

## Requirements

**Tool:** [ModusToolbox™](#) IDE 1.1; Bridge Control Panel

**Programming Language:** C

**Associated Parts:** All [PSoC 6 MCU](#) parts

**Related Hardware:** [PSoC 6 BLE Pioneer Kit](#), [PSoC 6 WiFi-BT Pioneer Kit](#), [PSoC 6 WiFi-BT Prototyping Kit](#)

## Overview

This code example implements an I<sup>2</sup>C Slave using an SCB resource (configured as EZI2C), which receives the data required to control an LED from an I<sup>2</sup>C Master. In this example, a host PC running Cypress' Bridge Control Panel (BCP) software is used as the I<sup>2</sup>C Master. LED control is implemented using a TCPWM resource (configured as PWM). The intensity of the LED is controlled by changing the duty cycle of the PWM signal.

## Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure the kit is configured correctly.

**Note:** The PSoC 6 BLE Pioneer kit and the PSoC 6 WiFi-BT Pioneer kit ship with KitProg2. ModusToolbox only works with KitProg3. Before using this code example, make sure that the kit is upgraded to KitProg3. See [ModusToolbox Help > ModusToolbox IDE Documentation > User Guide; section PSoC 6 MCU KitProg Firmware Loader](#). If you do not upgrade, you will see an error like "unable to find CMSIS-DAP device" or "KitProg firmware is out of date".

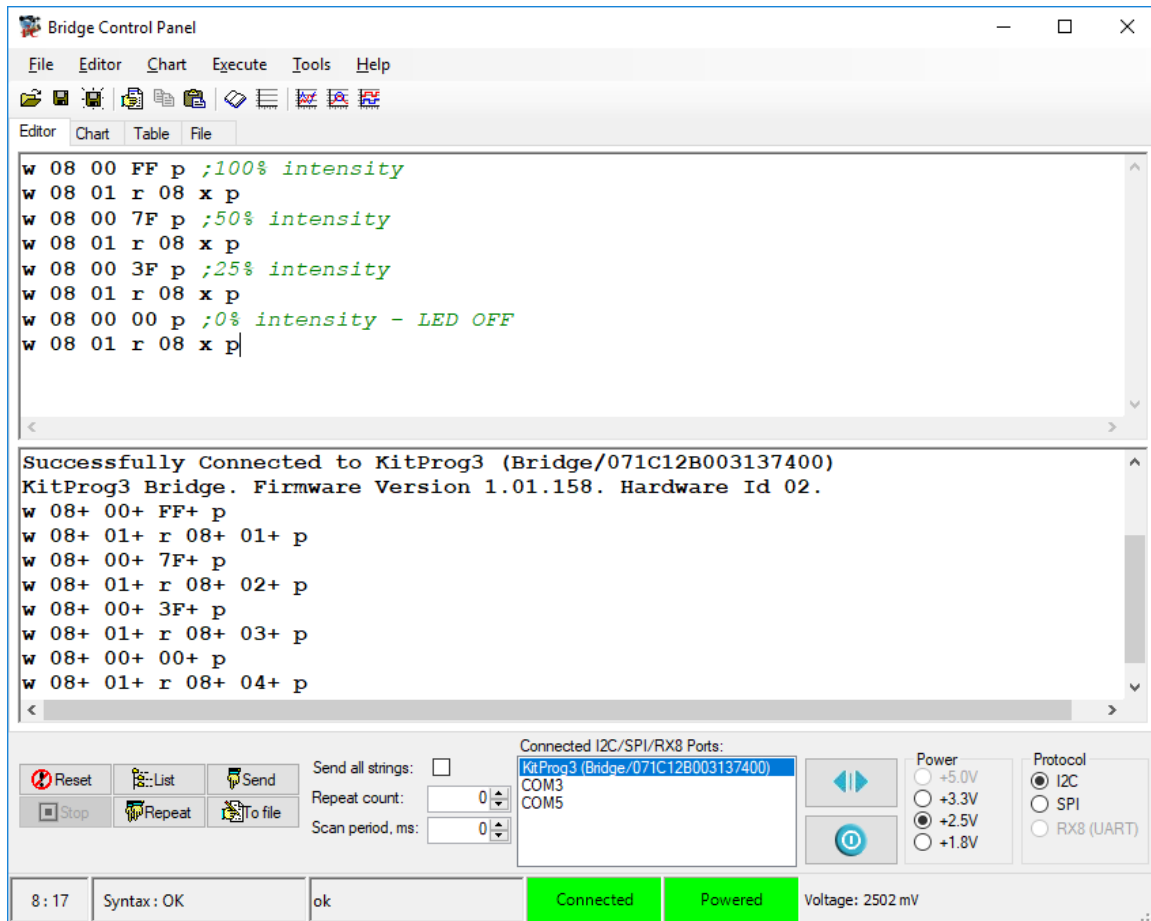
## Software Setup

This section describes how to set up the BCP software for transmitting and receiving data over I<sup>2</sup>C.

The BCP software is installed automatically as part of the [PSoC Programmer](#) installation. Follow these steps to configure BCP:

1. Open the BCP from **Start > All Programs > Cypress > Bridge Control Panel <version> > Bridge Control Panel <version>**.
2. Select **KitProg3/<serial\_number>** under **Connected I2C/SPI/RX8 Ports** (see [Figure 1](#)). Note that the PSoC 6 Kit must be connected to the USB port of your computer.

Figure 1. Bridge Control Panel



3. Select **Tools > Protocol Configuration**, navigate to the **I2C** tab, and set the **I2C speed** to '100 kHz'. Click **OK**. BCP is now ready for transmitting and receiving data.

## Operation

1. Connect the kit to your PC using the provided USB cable.
2. Import the code example into a new workspace. If you aren't familiar with this process, see [KBA225201](#).
3. Program the PSoC 6 MCU device. In the project explorer, select the **mainapp** project. In the Quick Panel, scroll to the **Launches** section and click the **Program (KitProg3)** configuration.
4. Configure the BCP software as described in the section [Software Setup](#).
5. In the **Editor** tab of BCP, type the command to send LED intensity data and then click **Send**. Observe that the LED turns ON with the specified intensity.

For example, sending the command 'w 08 00 FF p' will turn the LED ON with full intensity; sending the command 'w 08 00 00 p' will turn the LED OFF.

6. Type the command 'w 08 01 r 08 x p' to read the number of Write operations performed after device program/reset.

## Debugging

You can debug the example to step through the code. Use **Debug (KitProg3)** configuration. If you are unfamiliar with how to start a debug session with ModusToolbox IDE, see [KBA224621](#).

## Design and Implementation

In this example, EZI2C is configured with a 2-byte buffer (memory), which can be accessed by the I<sup>2</sup>C Master. The first byte is writable and holds the LED intensity and the second byte, which is read-only, holds the number of Write operations performed after the device reset.

EZI2C allows an I<sup>2</sup>C Master to either access the individual byte from the Slave memory (by specifying the memory address in the Write command) or all memory bytes at once.

To control the intensity of the LED, a PWM resource with a period value of 255 is used. The duty cycle of the PWM resource is controlled in firmware and specified by the I<sup>2</sup>C Master. Changing the duty cycle of the PWM signal will result in change in the LED intensity. The intensity range in this example is 0x01 to 0xFF; 0x00 turns the LED OFF.

## Resources and Settings

[Table 1](#) lists some of the ModusToolbox resources used in the example, and how they are used in the design. The *design.modus* file contains all the configuration settings. For example, for pin usage and configuration, open the **Pins** tab of the design file.

To see all the settings, review the *design.modus* file in the project.

Table 1. ModusToolbox Resources

Resource	Alias	Purpose	Non-default Settings
SCB3	KIT_I2C	Provide I2C communication between Master and Slave	<b>Clock:</b> 8 bit Divider 0 clk <b>SCL:</b> P6[0] <b>SDA:</b> P6[1]
TCPWM[0] Counter 3	KIT_LED2_PWM	Generate the PWM signal	<b>Clock:</b> 8 bit Divider 1 clk <b>Period:</b> 255u <b>Compare:</b> 0u
GPIO P13[7]	KIT_LED2	Drive the PWM signal to LED	<b>Drive Mode:</b> Strong Drive, Input buffer off <b>Digital Output:</b> TCPWM[0] 32-bit Counter 3 pwm_n

## Reusing This Example

This example is configured for the supported kit(s). To use the design on a different PSoC 6 MCU kit, import the application for that kit. If you are unsure how to import an application, see [KBA225201](#). If changing to a different kit, you may need to reassign pins.

Table 2. Device and Pin Mapping Across PSoC 6 MCU Kits

Kit Name	Device Used	LED	I2C_SCL	I2C_SDA
CY8CKIT-062-WiFi-BT	CY8C6247BZI-D54	P13[7]	P6[0]	P6[1]
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	P13[7]	P6[0]	P6[1]
CY8CPROTO-062-4343W	CY8C624ABZI-D44	P13[7]	P6[0]	P6[1]

In some cases, a resource used by a code example (for example, a peripheral) 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 which resources a device supports.

## Related Documents

Application Notes	
<a href="#">AN221774</a> – Getting Started with PSoC 6 MCU	Describes PSoC 6 MCU devices and how to build your first ModusToolbox application and PSoC Creator project
<a href="#">AN210781</a> – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first ModusToolbox application and PSoC Creator project
<a href="#">AN215656</a> – PSoC 6 MCU: Dual-CPU System Design	Describes the dual-CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-CPU design
Code Examples	
Visit the <a href="#">Cypress GitHub site</a> for a comprehensive collection of code examples using ModusToolbox IDE	
Device Documentation	
<a href="#">PSoC 6 MCU: PSoC 62 Datasheet</a>	<a href="#">PSoC 6 MCU: PSoC 62 Architecture Technical Reference Manual (TRM)</a>
<a href="#">PSoC 6 MCU: PSoC 63 with BLE Datasheet</a>	<a href="#">PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual</a>
Development Kits	
<a href="#">CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit</a>	
<a href="#">CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit</a>	
<a href="#">CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit</a>	
Tool Documentation	
<a href="#">ModusToolbox IDE</a>	The Cypress IDE for IoT designers

## Cypress Resources

Cypress provides a wealth of data at [www.cypress.com](http://www.cypress.com) to help you to select the right device, and quickly and effectively integrate the device into your design.

For the PSoC 6 MCU devices, see [KBA223067](#) in the Cypress community for a comprehensive list of PSoC 6 MCU resources.

## Document History

Document Title: CE220541 – PSoC 6 MCU SCB EZI2C

Document Number: 002-25530

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6369410	SNVN	11/21/2018	New code example
*A	6481994	SNVN	02/06/2019	Code example updated for ModusToolbox 1.1

## 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	<a href="http://cypress.com/arm">cypress.com/arm</a>
Automotive	<a href="http://cypress.com/automotive">cypress.com/automotive</a>
Clocks & Buffers	<a href="http://cypress.com/clocks">cypress.com/clocks</a>
Interface	<a href="http://cypress.com/interface">cypress.com/interface</a>
Internet of Things	<a href="http://cypress.com/iot">cypress.com/iot</a>
Memory	<a href="http://cypress.com/memory">cypress.com/memory</a>
Microcontrollers	<a href="http://cypress.com/mcu">cypress.com/mcu</a>
PSoC	<a href="http://cypress.com/psoc">cypress.com/psoc</a>
Power Management ICs	<a href="http://cypress.com/pmhc">cypress.com/pmhc</a>
Touch Sensing	<a href="http://cypress.com/touch">cypress.com/touch</a>
USB Controllers	<a href="http://cypress.com/usb">cypress.com/usb</a>
Wireless Connectivity	<a href="http://cypress.com/wireless">cypress.com/wireless</a>

### PSoC® Solutions

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6 MCU](#)

### Cypress Developer Community

[Community](#) | [Code Examples](#) | [Projects](#) | [Videos](#) | [Blogs](#)  
| [Training](#) | [Components](#)

### Technical Support

[cypress.com/support](http://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 - 2019. 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](http://cypress.com). Other names and brands may be claimed as property of their respective owners.