# CE219490 – PSoC 6 MCU Ramping LED using Smart I/O

## Objective

This example demonstrates the flexibility of Smart I/O in PSoC® 6 MCU, by implementing an LED ramping effect exclusively in hardware with no CPU usage beyond initialization.

## Requirements

**Tool:** ModusToolbox™ IDE 1.1

**Programming Language:** C

**Associated Parts:** All PSoC 6 MCU parts

**Related Hardware:** PSoC 6 BLE Pioneer Kit, PSoC 6 WiFi-BT Pioneer Kit, PSoC6 WiFi-Prototyping Kit

## Overview

This example uses a PWM resource and Smart I/O in PSoC 6 MCU to implement a ramping LED, where an LED gradually cycles through increasing and decreasing brightness levels. There is no CPU usage except for the initialization of PWM and Smart I/O.

## Hardware Setup

This example uses the kit's default configuration. See 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 If you do not upgrade, you will see an error like "unable to find CMSIS-DAP device" or "KitProg firmware is out of date".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

None.

## Operation

Follow the instructions that came with your kit to make sure that your kit is connected to your PC

1. Connect the kit to your PC using the provided USB cable.
2. Import the application into a new workspace. See KBA225201.
3. Build the application. Choose **Project** > **Build All**.
4. 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. Program configurations also build the code.
5. Using a jumper wire, connect **P9[4]** to **P13[7]**.

You can observe the ramping effect on the LED connected to P13[7] on the kit.

## Debugging

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

## Design and Implementation

This design consists of a PWM resource and a Smart I/O resource, both creating square waves of slightly different frequencies. These square waves are routed through an exclusive-OR (XOR) gate within the Smart I/O resource, yielding a signal with a gradually changing duty cycle. The rate of change is proportional to the difference between the output square wave frequencies.

The signal is then output to I/O4 of the Smart I/O port 9. Driving LED with this signal results in a “ramping” effect, where the LED gradually get brighter and dimmer alternately.

The PWM is driven by a 10-kHz clock with a period of 399 counts and a compare value of 200 counts. This gives a 50 percent duty cycle square wave with a 40-ms period. The Smart I/O is clocked at 99 Hz using a divided clock sourced from CLK_PERI. This input clock is divided by 4 using the lookup tables (LUTs) of the Smart I/O resource to produce a square wave with a 40.4-ms period.

To generate a square wave signal with a time period close to 40 ms, a 99-Hz clock is divided by 4 using a synchronous sequential circuit, which is realized using the LUTs of the Smart I/O resource.

To implement a divide-by-4 sequential circuit, consider the state transition values shown in Table 1:

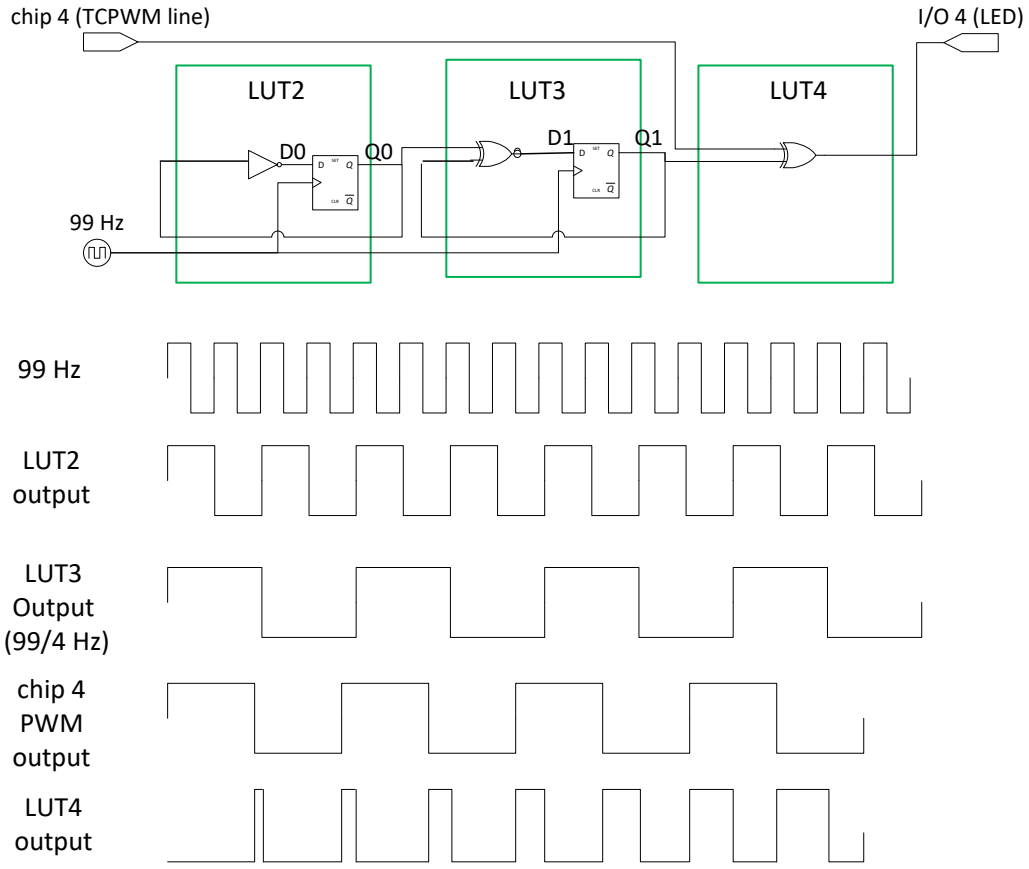
Table 1. State Transition Table for a Divide-by-4 Sequential Circuit

| CLK | Present State | | Next State | | D0 | D1 |
|---|---|---|---|---|---|---|
| | Q0 | Q1 | Q0 | Q1 | | |
| ↑ | 0 | 0 | 1 | 1 | 1 | 1 |
| ↑ | 1 | 1 | 0 | 1 | 0 | 1 |
| ↑ | 0 | 1 | 1 | 0 | 1 | 0 |
| ↑ | 1 | 0 | 0 | 0 | 0 | 0 |

From this state transition table, you can observe that Q0 is half the frequency of Clk_SmartIO and Q1 is 1/4$^{th}$ frequency of Clk_SmartIO. This sequential logic can be implemented using the LUTs of the Smart I/O resource.

Figure 1 shows the implementation of this logic using LUT 2 and LUT 3. In addition, the divided clock is XORed with the PWM output using LUT 4 to generate a signal with the duty cycle gradually increasing and decreasing over time. The output of LUT 4 is driven to I/O 4 output.

Figure 1. LUT Configuration and Timing Diagram




## Resources and Settings

Table 2 lists the ModusToolbox resources used in this example, and how they are used in the design. For pin usage and configuration, open the Pins tab of the *design.modus* file.

Table 2. ModusToolbox Resources

| Resource | Alias | Purpose |
|---|---|---|
| Timer Counter PWM (TCPWM) | PWM | Generates 25 Hz, 50% duty cycle square wave |
| Smart I/O | SmartIO | Implements divide-by-4 sequential circuit |
| Digital Output Pin | Pin_LED | Provides visual feedback using the LED |

Figure 2 highlights the non-default settings for the TCPWM.

Figure 2. TCPWM Configuration

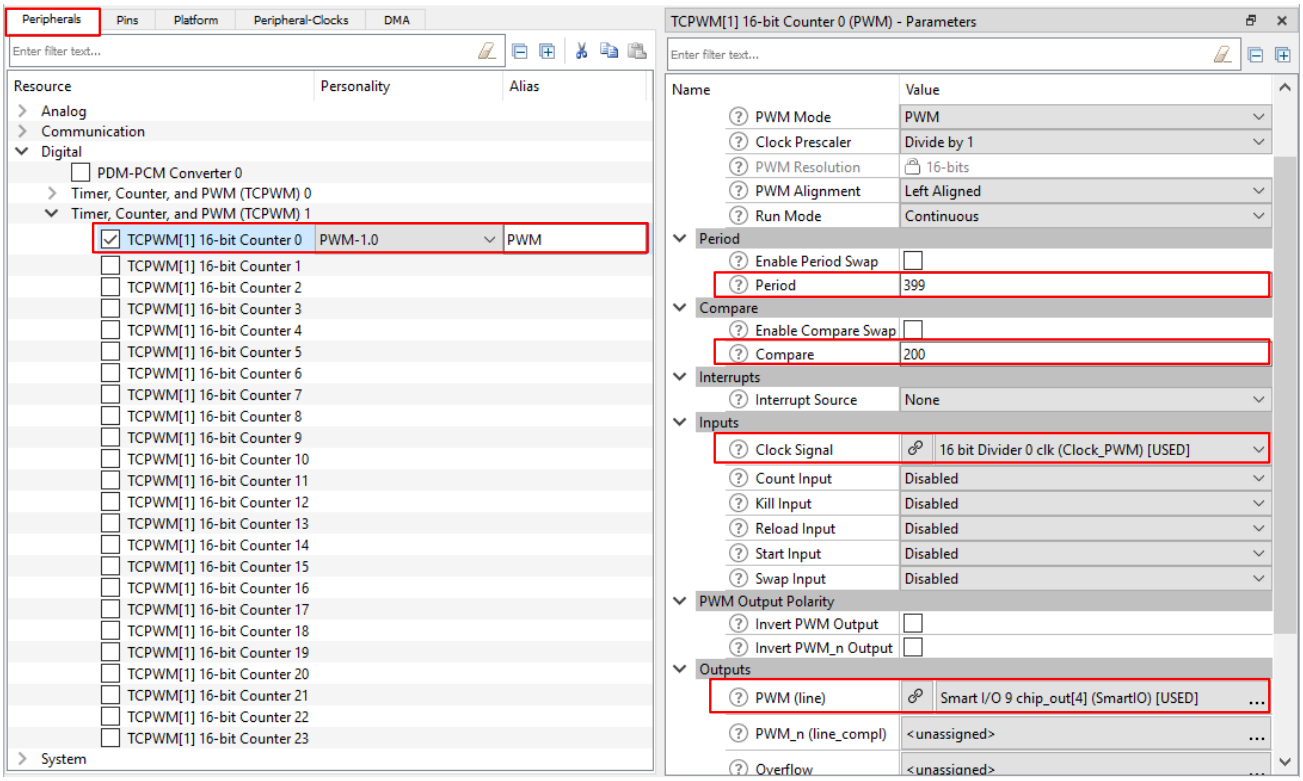



Figure 3 through Figure 5 illustrate the steps for configuring Smart I/O.

Figure 3. Enabling Smart I/O

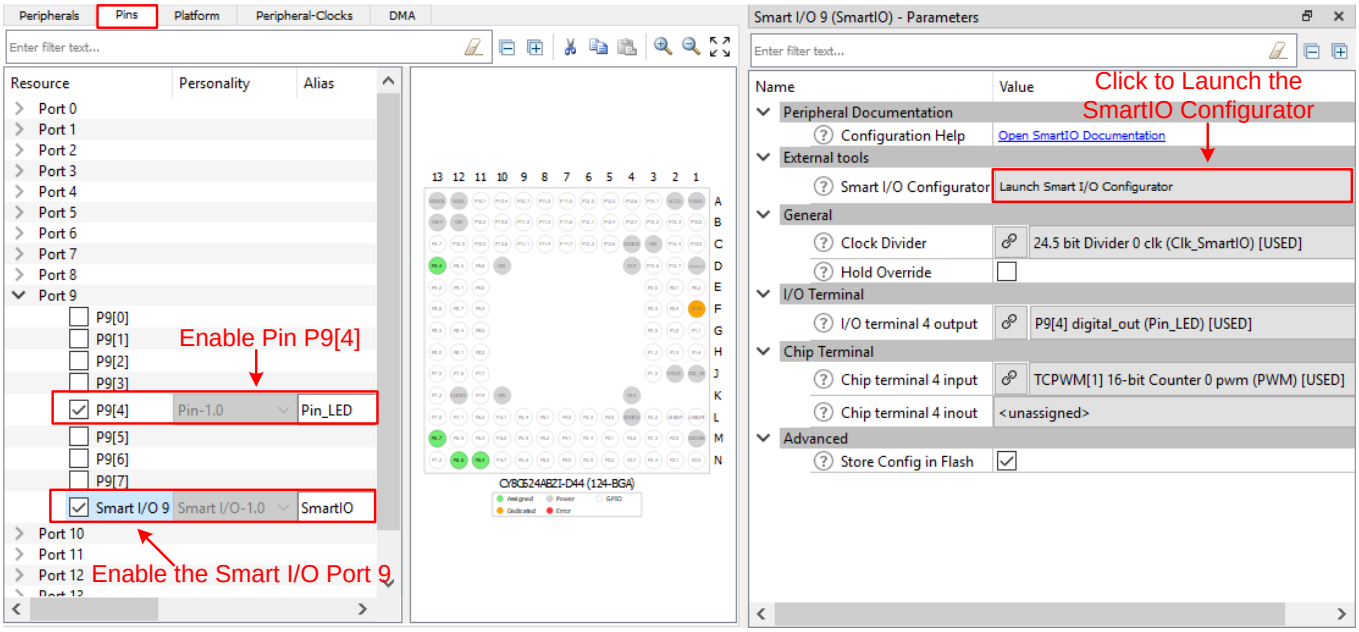


Figure 4. Smart I/O Routing Configuration

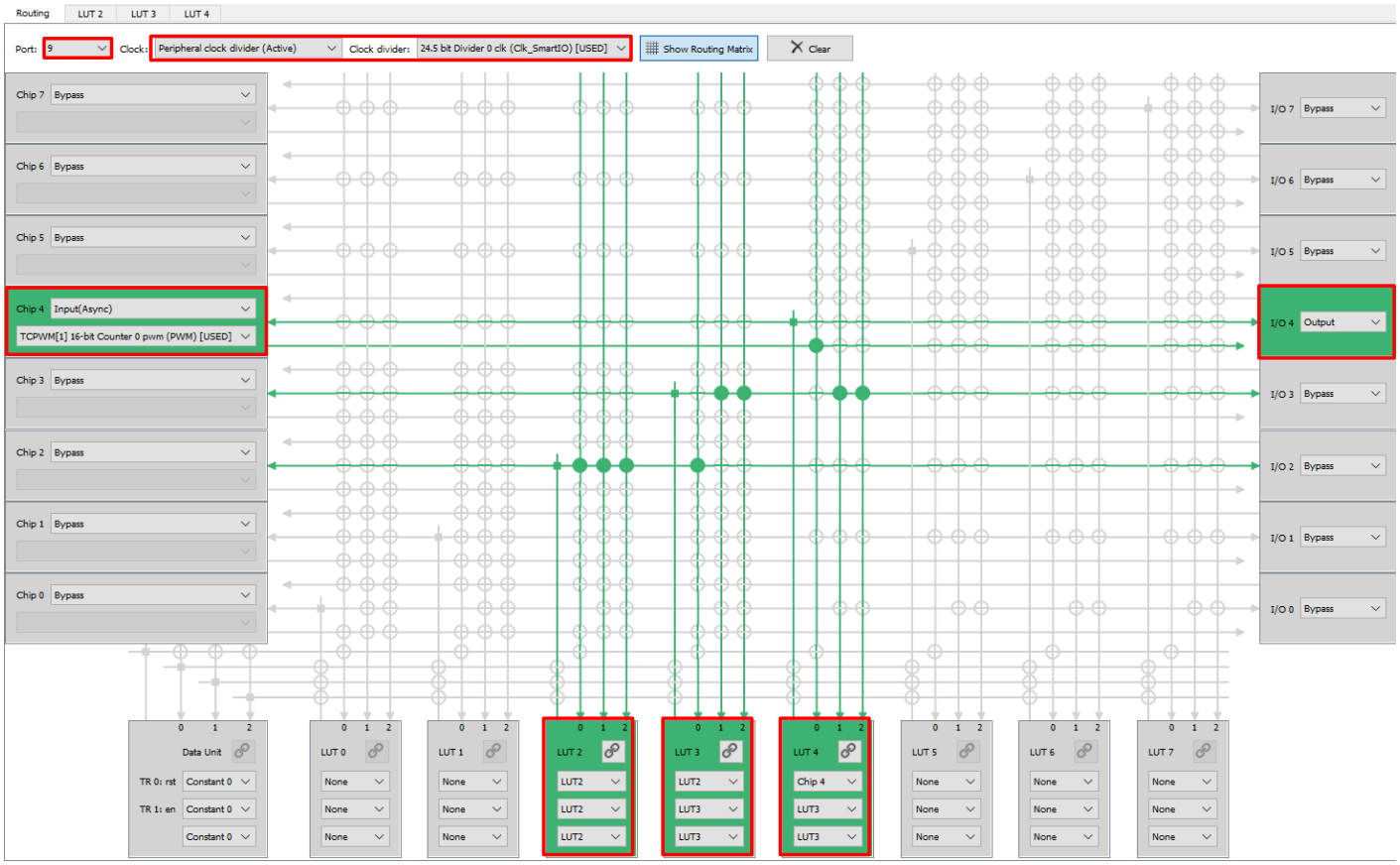



Figure 5. LUT Configuration

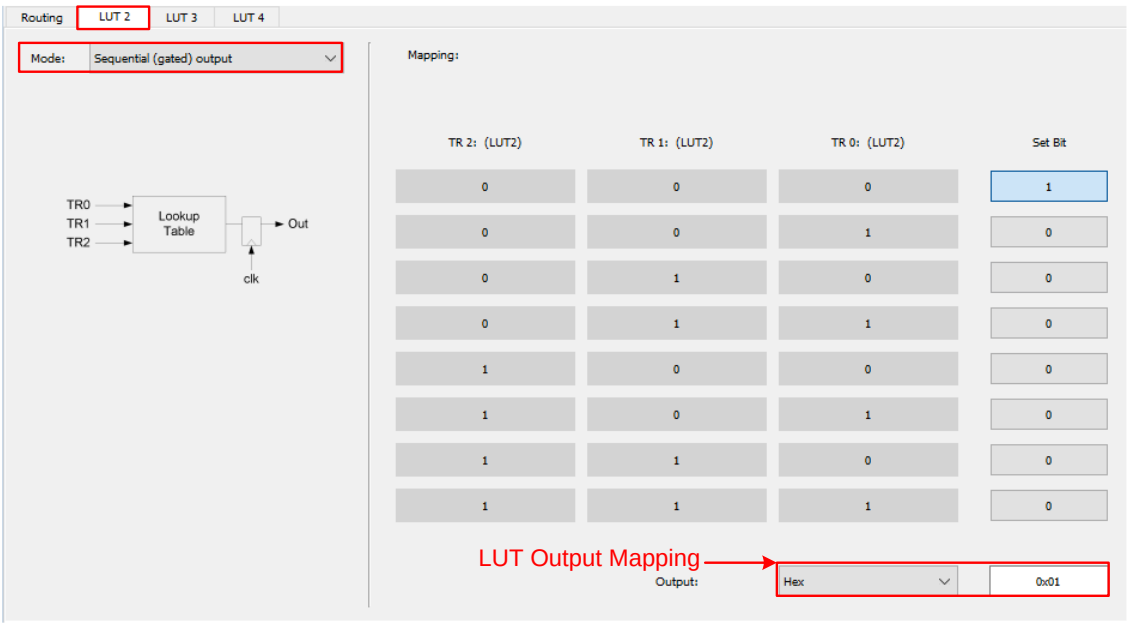



Figure 5 depicts the LUT2 configuration settings. Similarly, configure LUT3 and LUT 4 with the settings shown in Table 3.

Table 3. Smart I/O LUT Configuration

| LUT# | Mode | LUT inputs | | | LUT Output Mapping | Description |
|---|---|---|---|---|---|---|
| | | TR2 | TR1 | TR0 | | |
| LUT2 | Sequential (gated) output | LUT2 | LUT2 | LUT2 | 0x01 | Implements a logic NOT operation |
| LUT3 | Sequential (gated) output | LUT3 | LUT3 | LUT2 | 0x81 | Implements a logic XNOR operation |
| LUT4 | Combinatorial output | LUT3 | LUT3 | Chip 4 | 0x42 | Implements a logic XOR operation |

Figure 6 and Figure 7 show the Peripheral-Clock configuration for Smart I/O and TCPWM resources respectively.

Figure 6. Peripheral-Clock Configuration for Smart I/O

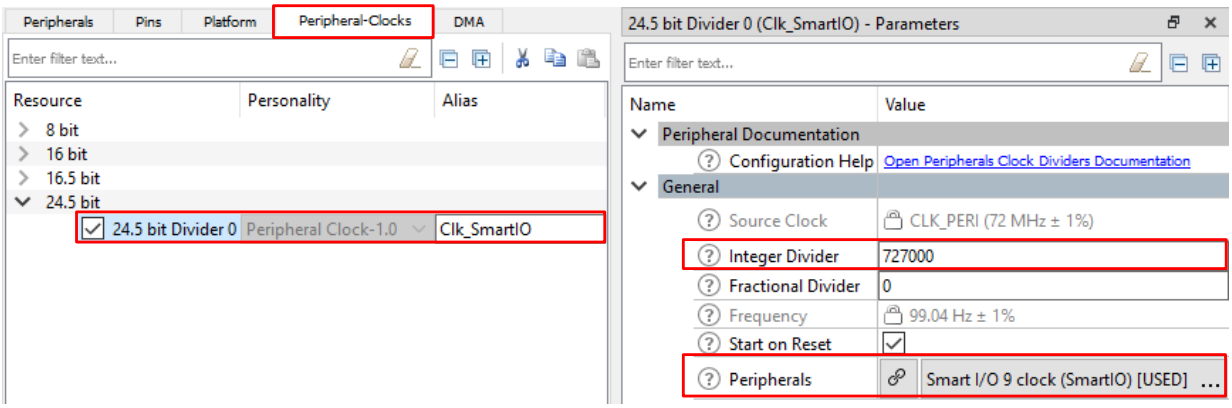
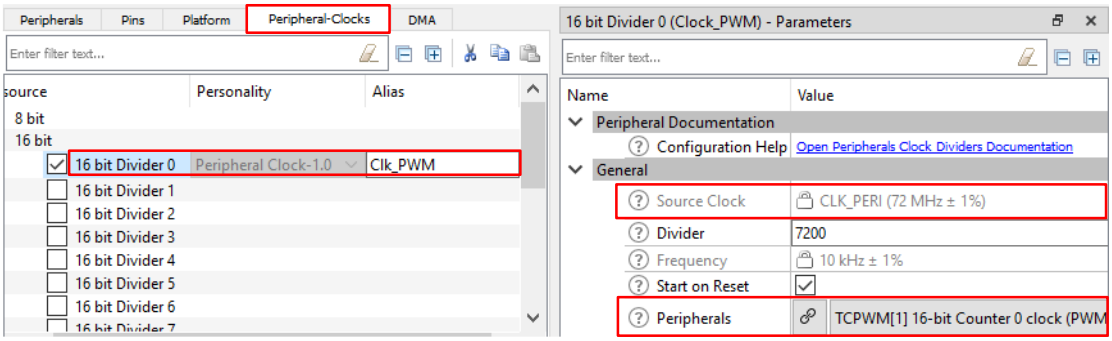



Figure 7. Peripheral-Clock Configuration for TCPWM




## Reusing This Example

This example is designed for the supported kits. To port the design to a different PSoC 6 MCU device, right-click an application project and choose **Change Device**. If changing to a different kit, you may need to reassign pins.

Table 4. Device and Pin Mapping Table across PSoC 6 MCU Kits

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

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.

## Related Documents

For a comprehensive list of PSoC 6 MCU resources, see KBA223067 in the Cypress community.

| | |
|---|---|
| **Application Notes** | |
| AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity | Describes PSoC 6 MCU with BLE Connectivity devices. |
| AN215656 – 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 Cypress GitHub site for a comprehensive collection of code examples using ModusToolbox IDE | |
| **Device Documentation** | |
| PSoC 6 MCU: PSoC 63 with BLE Datasheet | PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual |
| **Development Kits** | |
| CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit | |
| CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit | |
| CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit | |
| **Tool Documentation** | |
| ModusToolbox IDE | The Cypress IDE for IoT designers |

## Cypress Resources

Cypress provides a wealth of data at 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: CE219490 – PSoC 6 MCU Ramping LED using Smart I/O

Document Number: 002-25568

| Revision | ECN | Orig. of Change | Submission Date | Description of Change |
|---|---|---|---|---|
| ** | 6373851 | VKVK | 11/02/2018 | New code example |
| *A | 6482040 | VKVK | 02/11/2019 | Updated the project with 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 | 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 | Code Examples | Projects | Videos | Blogs | Training| Components

## Technical Support

cypress.com/support



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709