PSoC™ 4 timer/counter

About this document

Scope and purpose

This example shows how to use the PSoC™ Creator Timer Counter Pulse Width Modulator (TCPWM) Component configured as a timer/counter in a PSoC™ 4 device.

Requirements

Tool: **PSoC™ Creator** 4.4

Programming language: C (Arm® GCC 5.4.1)

Associated parts: **PSoC™ 4** family

Related hardware: **CY8CKIT-042 PSoC™ 4 pioneer kit**

Table of contents

1 Overview

This example contains three projects that use the Timer Counter Component. The Counter_Count project demonstrates the Timer keeping track of the number of button presses with an LED showing the count changing. The Counter_Frequency_DutyCycle project measures the frequency and duty cycle of an input waveform using counter mode and prints the results over UART. The Counter_Periodic_Interrupt uses the timer mode to create periodic interrupts that blink an LED.

2 Hardware and software setup

2.1 Hardware setup

The code example is set up for CY8CKIT-042. If you are using a different kit, see **Reusing this example**.

The code example uses a UART to send data to a terminal. Connect the UART signals on the PSoC™ 4 device to the USB-UART bridge built into the PSoC™ 5LP device.

For CY8CKIT-042, the USB-UART bridge in KitProg2 module is used:

1. Connect the \UART:rx\ pin P0[4] to P12[7] on header J8.
2. Connect the \UART:tx\ pin P0[5] to P12[6] on header J8.

Other kits use different pins for the UART. Make sure that you select the correct pins for your kit.

2.2 Software setup

This design requires a terminal emulator such as PuTTY or Tera Term running on your computer.

3 Operation

1. Connect the USB cable between the PC and the PSoC™ 4 Pioneer Kit.
2. Build the project and program it into the PSoC™ 4 device. Choose **Debug** > **Program**. For more information on device programming, see PSoC™ Creator Help.
3. Open a terminal emulator on your computer and configure the program to the appropriate COM port. Configure the baud rate to 115200, data bits to 8, no parity bits, stop bit as 1, and no flow control.
4. For Counter_Count project: Press the kit button SW2 and confirm that the count displayed on the terminal increments accordingly.
5. For Counter_Frequency_DutyCycle project: Change the period and compare register settings of the PWM to create different periods and duty cycles. Confirm that the frequency and duty cycle are correctly displayed on the terminal.

   An alternative to using the PWM is to use a digital input pin and feed in your own input waveform. Note that this requires an inverter so that Timer_2 can use the inverse of the input waveform.
6. For Counter_Interrupt project: confirm that the red LED turns on and off every two seconds. No terminal emulator is needed for this project

4 Design and implementation

4.1 Counter_Count

In the Counter_Count example, the following functions are performed:

1. The Counter and UART Components start.
2. When the Timer Component detects a change in count, the LED flashes, and the UART displays a new count value.

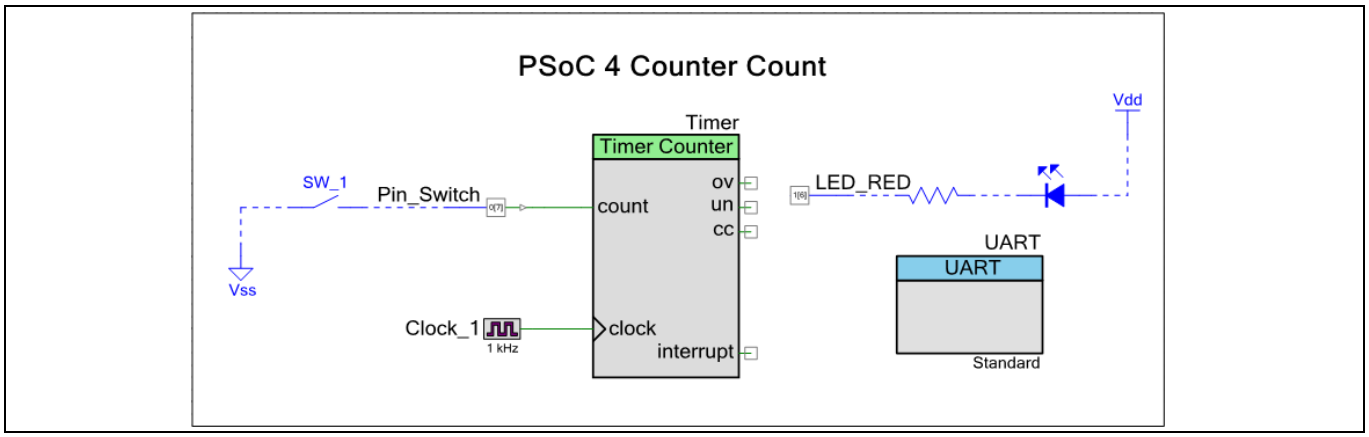

Figure 1 Schematic, Counter_Count

4.2 Counter_Frequency_DutyCycle

In the Counter_Frequency_DutyCycle example, the following functions are performed:

1. The Timer Counter, PWM, and UART Components start.
2. Timer 1 and Timer 2 capture both counts for the HIGH and LOW parts of the waveform.

Using these values, the period, frequency, and duty cycle are calculated and displayed.

- The period is calculated by summing the total count for when the waveform is HIGH (from Timer_1) and when the waveform is LOW (from Timer_2).
- Frequency is calculated by dividing the timer clock speed by the period.
- The duty cycle is calculated by dividing total count for when the waveform is HIGH by the period.

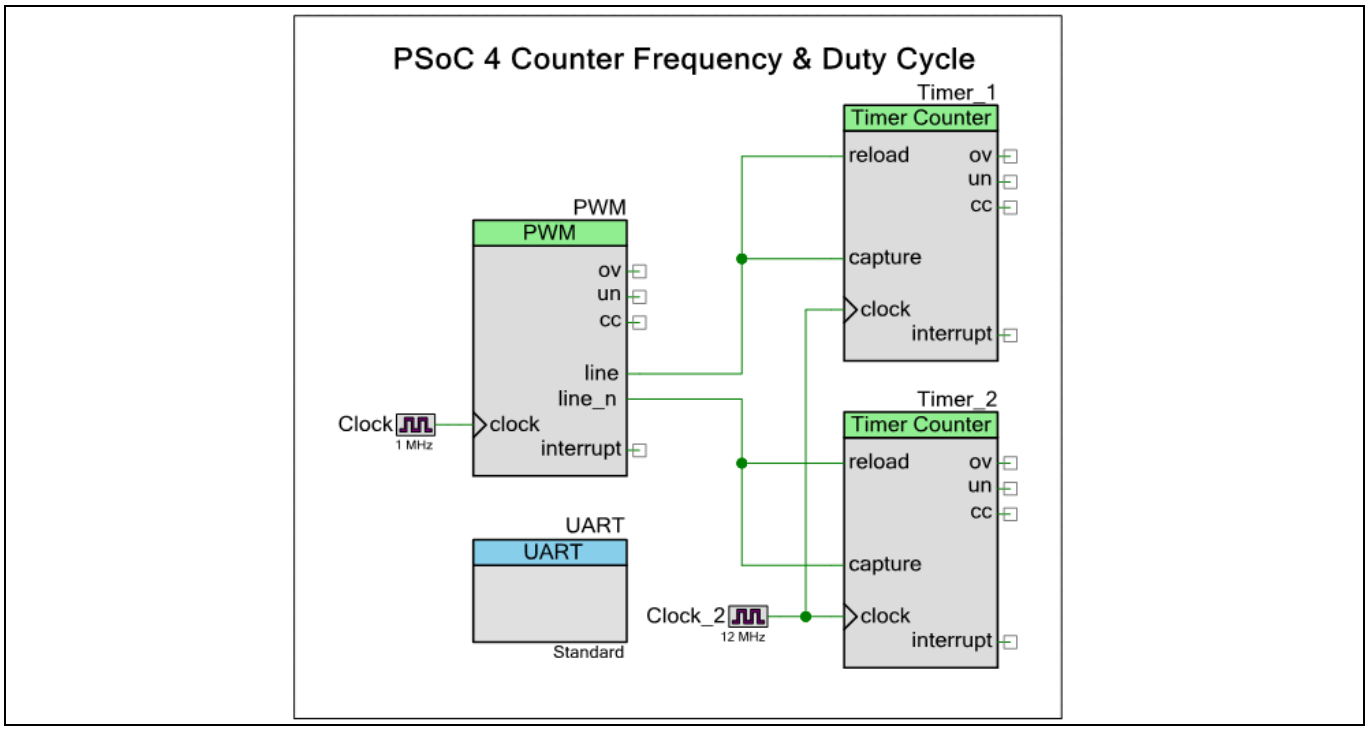

Figure 2 Schematic, Counter_Frequency_DutyCycle

4.3 Counter_Periodic_Interrupt

In the Counter_Periodic_Interrupt example, the following functions are performed:

1. The Timer Counter Component is started
2. The timer counter interrupt handler function TC_InterruptHandler is configured.
3. An interrupt occurs when the timer's count reaches the terminal count, which is determined by the period set for timer.

The ISR_Timer function does the following:

1. Clears the interrupt for terminal count.
2. Toggles the LED ON/OFF state.

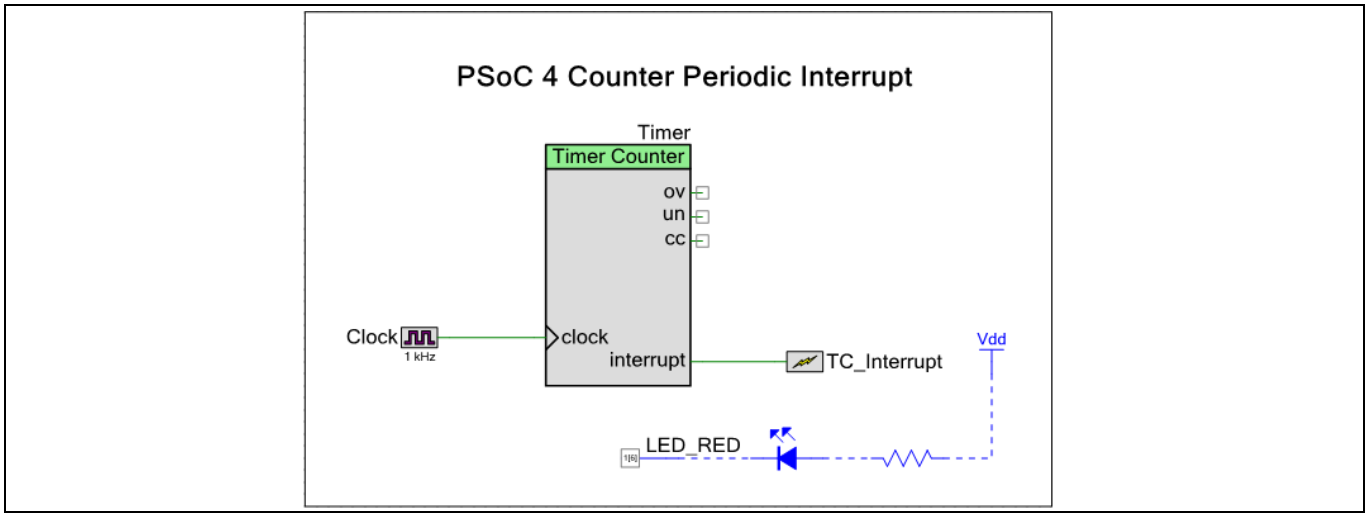

Figure 3 Schematic, Counter_Periodic_Interrupt

4.4 Components and settings

Table 1 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 1 PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
TCPWM (Timer Counter Mode)	Timer	Increases the count by one each time the button is pushed.	See **Figure 4**
TCPWM (Timer Counter Mode)	Timer_1 & _2	Timer_1 captures how long the PWM pulse width is HIGH. Timer_2 captures how long the PWM pulse width is LOW.	See **Figure 5**
TCPWM (Timer Counter Mode)	Timer (for periodic interrupt project)	Generates a periodic interrupt on the terminal count.	Set period register to 2000.
TCPWM (PWM Mode)	PWM	Creates a PWM, the example finds the period of this wave.	Set period register to 4999. Set compare register to 2499.
UART (SCB mode)	UART	Handles communication between the device and the terminal.	None
Digital Input Pin	Pin_Switch	Connects SW2 to the Timer Counter. Pushing SW2 will increase the current count.	See **Figure 6** Used in Counter_Count project only.
Digital Output Pin	LED_RED	Connects to LED on board. In Counter_count project, LED flashes when terminal count is changed. In Counter_Periodic_Interrupt project, LED toggles when terminal count interrupt occurs.	See **Figure 7** Used in Counter_count and Counter_Periodic_Interrupt projects only.

Note: All the projects do not use all the components mentioned in **Table 1**. *See* **Design and implementation** *for more details.*

Design and implementation

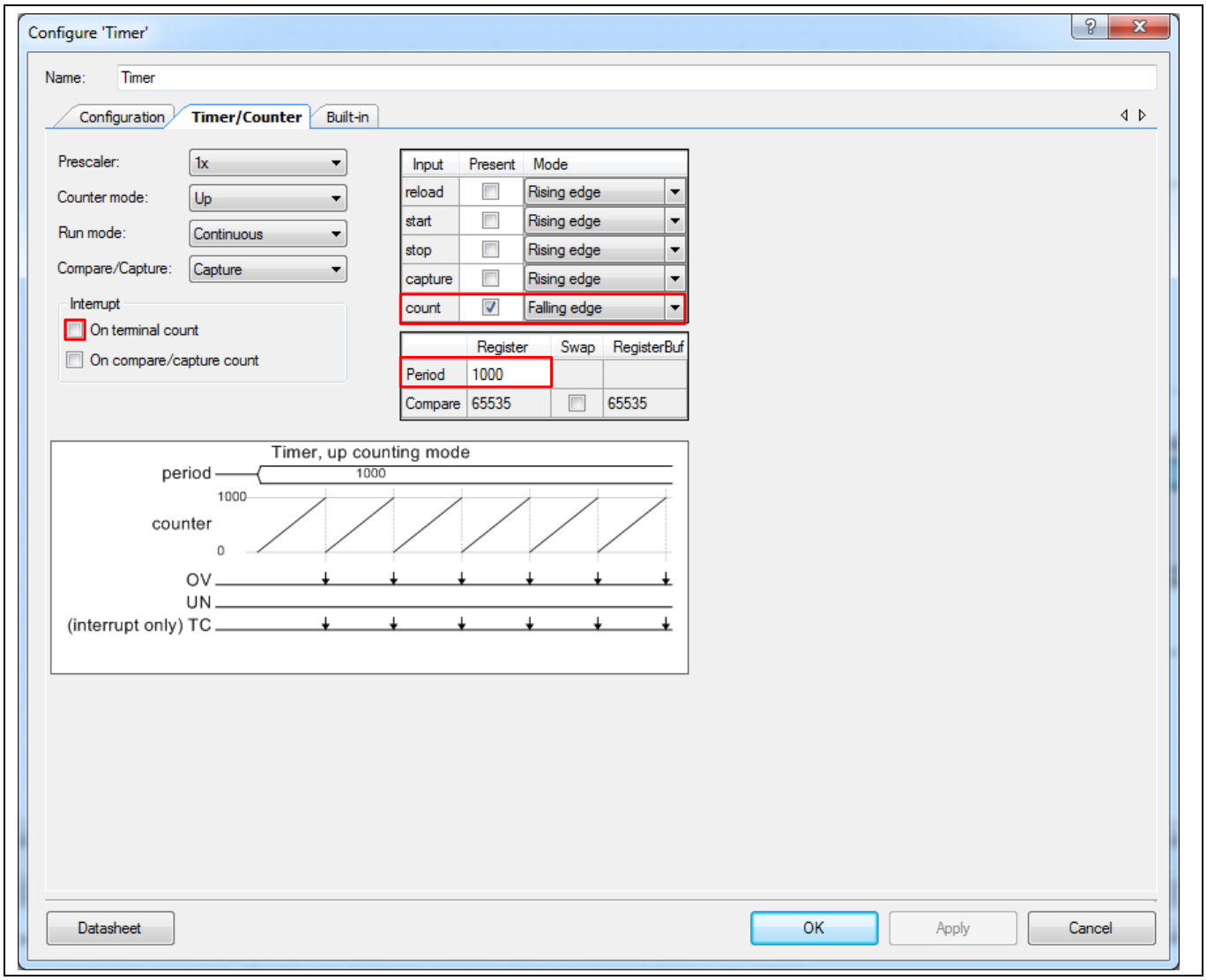

Figure 4 Timer parameter settings

Design and implementation

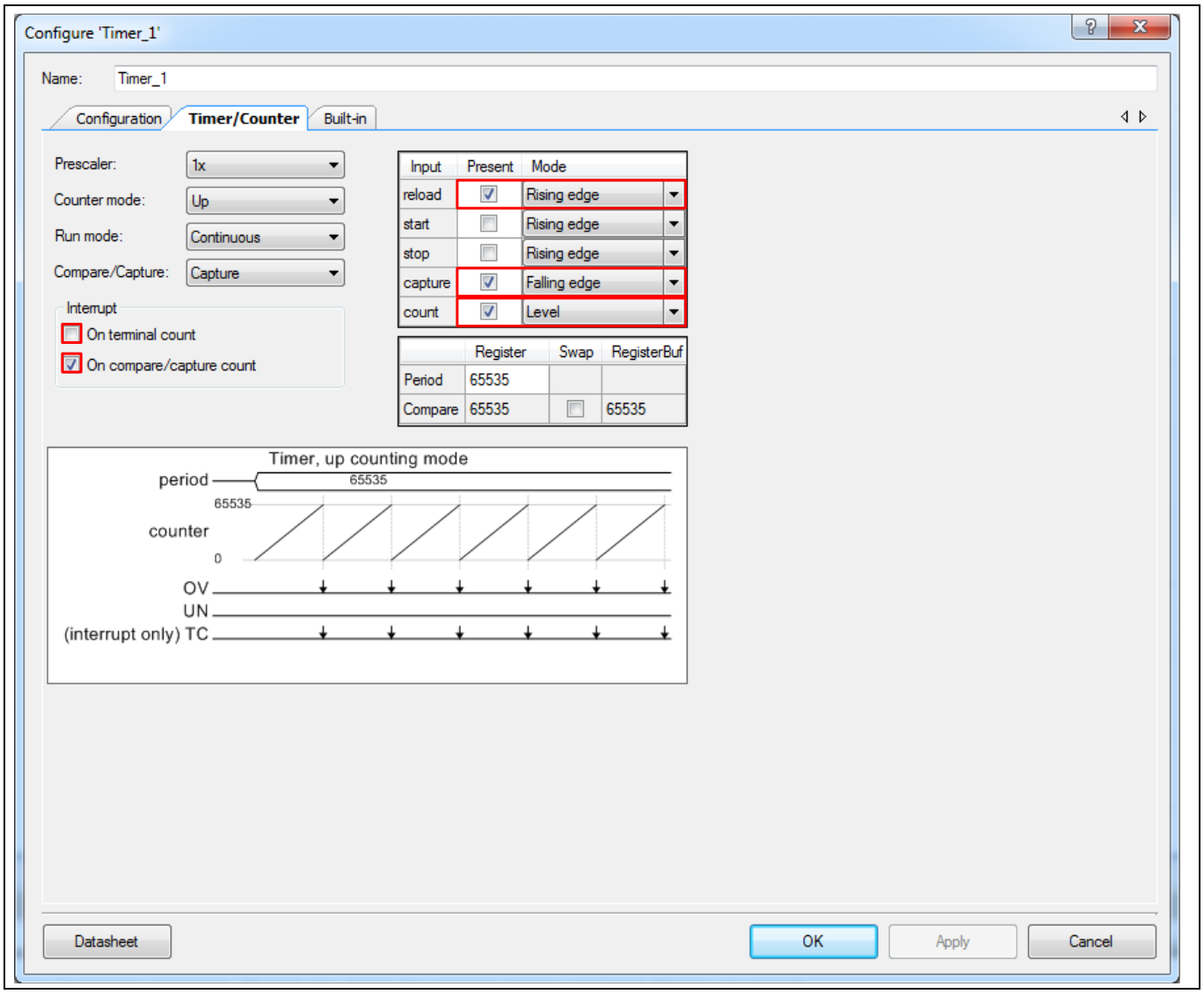

Figure 5 Timer_1 (2) parameter settings

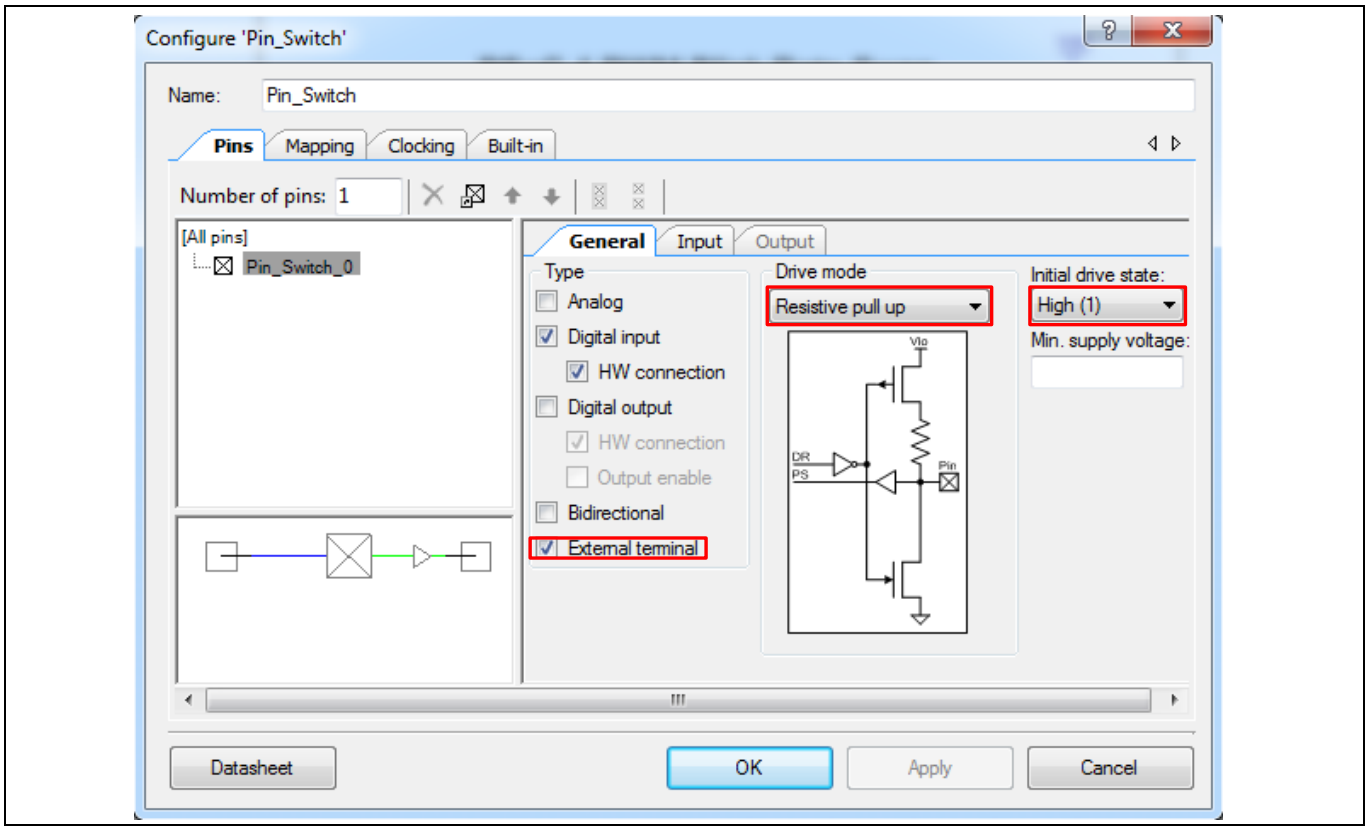

Figure 6 Pin_Switch parameter settings

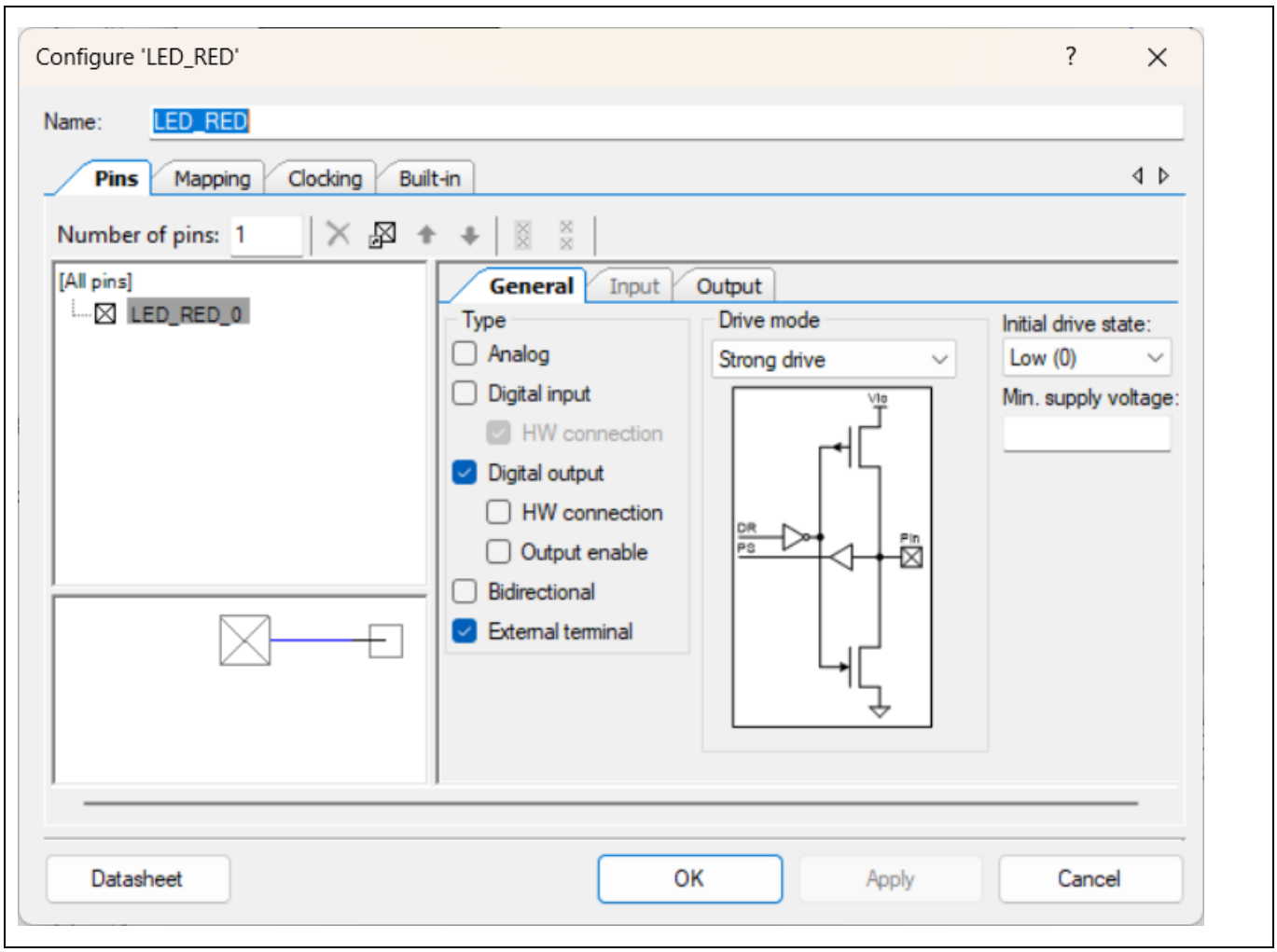

Figure 7 LED_RED parameter settings

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

5 Reusing this example

This example is designed for the CY8CKIT-042 pioneer kit. To port this design to a different PSoC™ 4 device, kit, or, both, do the following:

1. In PSoC™ Creator, select **Project** > **Device Selector** to change the target device. Select your device as listed in **Table 2**.
2. Make sure that the **SysClk Desired frequency** is set to 24 MHz after the device is changed.
3. In the PSoC™ Creator Workspace Explorer, select the **Clocks** interface listed under **Design Wide Resources**.
4. Set the **SysClk Desired Frequency** to 24 MHz, if it is not already.

Table 2 Development kits and associated devices

Development kit	Device
CY8CKIT-042	CY8C4245AXI-483
CY8CKIT-042-BLE-A	CY8C4248LQI-BL583
CY8CKIT-044	CY8C4247AZI-M485
CY8CKIT-046	CY8C4248BZI-L489

5. Change the UART RX and TX pins in the CYDWR file as per the selected kit. See **Table 3** for pin details. Additionally, for CY8CKIT-042, the PSoC™ 4 UART signals need to be physically connected to KitProg2 (PSoC™ 5 LP) UART signals using jumper wires. Connect P12[7] and P12[6] on header J8 to P0[4] and P0[5] on header J4. For other kits, the pins are already connected.

Table 3 UART Pin assignments

Pin name	Development kit			
	CY8CKIT-042	CY8CKIT-042-BLE-A	CY8CKIT-044	CY8CKIT-046
\UART:rx\	P0[4]	P1[4]	P7[0]	P3[0]
\UART:tx\	P0[5]	P1[5]	P7[1]	P3[1]

6. Change the Switch and LED pins in the CYDWR file as per the selected kit. See **Table 4** for pin details.

Table 4 Switch and LED Pin assignments

Pin name	Development kit			
	CY8CKIT-042	CY8CKIT-042-BLE-A	CY8CKIT-044	CY8CKIT-046
Pin_Switch	P0[7]	P2[7]	P0[7]	P0[7]
LED_RED	P1[6]	P2[6]	P0[6]	P5[2]

As explained in **Design and implementation section**, there are three projects in the workspace. All the three projects use UART component. LED and Switch components are used as per project requirement. See **Table 1** for more details.

In some cases, a resource used by a code example (for example, a Universal Digital Block (UDB)) 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 device supports.

6 References

Application notes

[1] **AN79953** – Getting started with PSoC™ 4: Describes the PSoC™ 4 device and shows how to build the attached code example

Code examples

[2] **CE224593** – PSoC™ 4 TCPWM PWM: Demonstrates PWM driving an LED with changing blink rates and also three PWM 120 degrees out of phase from each other driving LEDs.

PSoC™ Creator Component datasheets

[3] **TCPWM:** Multifunctional component that can implement the following functionalities: PWM, Timer/Counter, and Quadrature Decoder.

[4] **SCB:** A multifunction hardware block that implements the following communication Components: I2C, SPI, UART, and EZI2C

[5] **General Purpose Input/Output (GPIO):** A multifunctional component that allows hardware resources to connect to a physical port-pin and provides access to external signals through an appropriately configured physical I/O pin.

[6] **Interrupt:** The interrupt component defines hardware triggered interrupts. There are three types of system interrupt waveforms that can be processed by the interrupt controller: Level, Pulse, and Edge.

Device documentation

[7] **PSoC™ 4 datasheets**

[8] **PSoC™ 4 technical reference manual**

Development kit (DVK) documentation

[9] **CY8CKIT-042 PSoC™ 4 pioneer kit**

[10] **PSoC™ 4 Kits**

Tool documentation

[11] **PSoC™ Creator user guide**

[12] **PSoC™ Creator quick start guide**

Revision history

Document version	Date	Description of changes
**	2018-12-20	Initial version
*A	2024-07-30	Update document to Infineon template. Minor updates to code and document. Removed references of obsolete kits.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2024-07-30

Published by

Infineon Technologies AG

81726 München, Germany

Do you have a question about this document?

Go to erratum@infineon.com

Document reference

002-24594 Rev.*A

IMPORTANT NOTICE

The information contained in this document is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this document must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this document.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.