CE224719 – PSoC 4 Debouncer and Glitch Filter

Objective

This example shows different methods to debounce a switch using both firmware and hardware. The PSoC® Creator™ Debouncer and Glitch Filter Components are used in some of the examples.

Requirements

Tool: PSoC Creator 4.2

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: PSoC 4 family

Related Hardware: CY8CKIT-042 PSoC 4 Pioneer Kit

Overview

This example contains six PSoC Creator projects, two of which use the Debouncer or the Glitch Filter Component.

- The '1_SwitchBounce_Sensitive' project demonstrates a poor design that is sensitive to switch bounce.
- The '2_SwitchDebounce_Polling' project demonstrates switch debouncing using periodic sampling.
- The '3_SwitchDebounce_Interrupt' project demonstrates switch debouncing using interrupt-based sampling.
- The '4_SwitchDebounce_Hardware' project demonstrates hardware switch debouncing using a status register and polling with a clock.
- The '5_SwitchDebounce_Debouncer' project demonstrates switch debouncing using the Debouncer Component.
- The '6_SwitchDebounce_GlitchFilter' project demonstrates switch debouncing using the Glitch Filter Component.

Hardware Setup

This code example is set up for CY8CKIT-042. If you are using a different kit, see Reusing this Example. In this kit, the USB-UART bridge in the 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.

Software Setup

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

Operation

1. Connect the correct pins for your kit, as noted in Hardware Setup.
2. Connect CY8CKIT-042 to your computer using a USB cable.
3. Build each of the projects.

Note: During each project build, one or more static timing analysis (STA) warnings are displayed. The warnings can be ignored; the projects function correctly. To remove the warnings, add Sync Components to the project schematics. For more information, see AN81623, PSoC Digital Design Best Practices.

4. Open a terminal emulator on your computer and configure the program to the appropriate COM port. Configure the baud rate to 115200, 8 data bits, no parity bits, 1 stop bit, and no flow control.

5. For each project:
   a. Program the project into the PSoC 4 device. Choose **Debug** > **Program**. For more information on device programming, see *PSoC Creator Help*.
   b. Press and release the kit SW2 button. Confirm that the count of button presses and releases is displayed on the terminal.
   c. Depending on the project, confirm that as SW2 is pressed and released the displayed raw and debounced counts increase, as Figure 1 and Figure 2 show. Multiple raw count increases indicate that switch bounce has occurred.

Note: Due to the high-quality switches on the CY8CKIT-042, the switch [SW2] does not bounce very often. To create switch bounce, rapidly press and release SW2 by jittering your finger on the button.

Figure 1. Switch Bounce Display for Projects 1 and 2

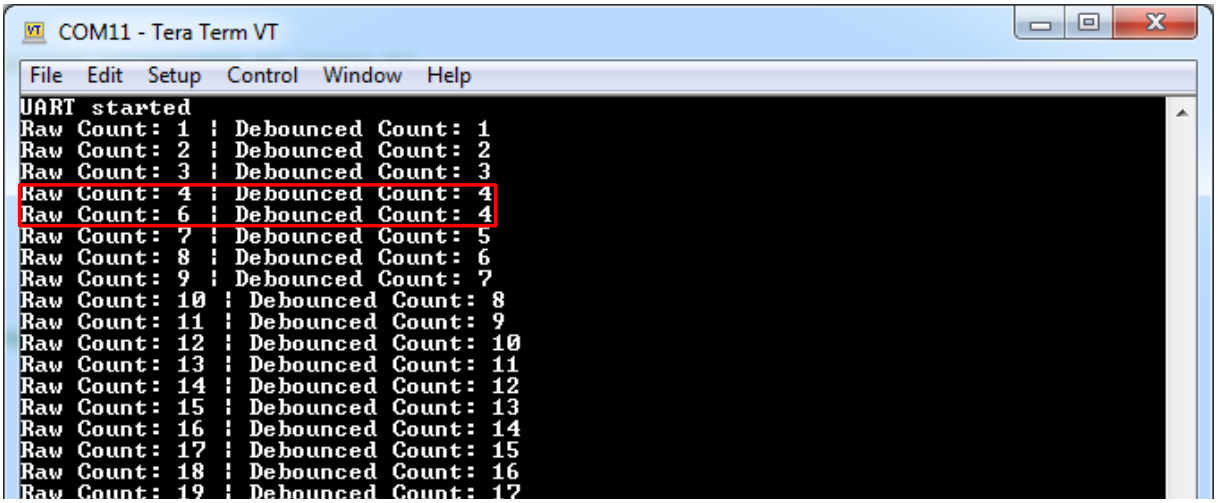
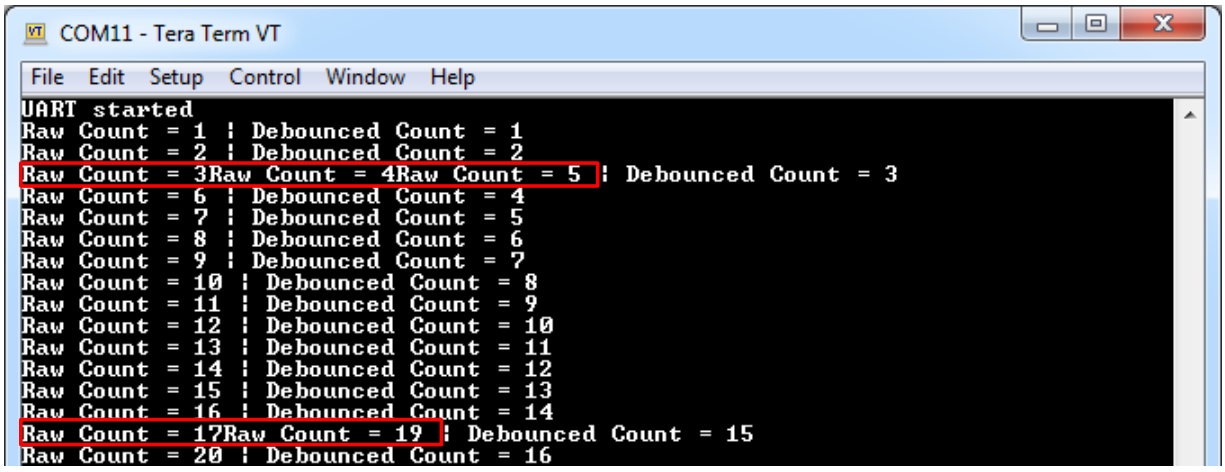


Figure 2. Switch Bounce Display for Projects 3, 4, 5, and 6



Design and Implementation

There are six projects in this example.

All of the projects use two D flip-flops (DFF) and other logic Components, and a Control Register Component. The DFFs detect both the rising and falling edges of the signal from the input pin. (One of the DFFs has an inverter at the clock.) The Control Register is used to generate asynchronous resets (ar) on the DFFs. The reset signal is pulsed HIGH by firmware when an interrupt occurs.

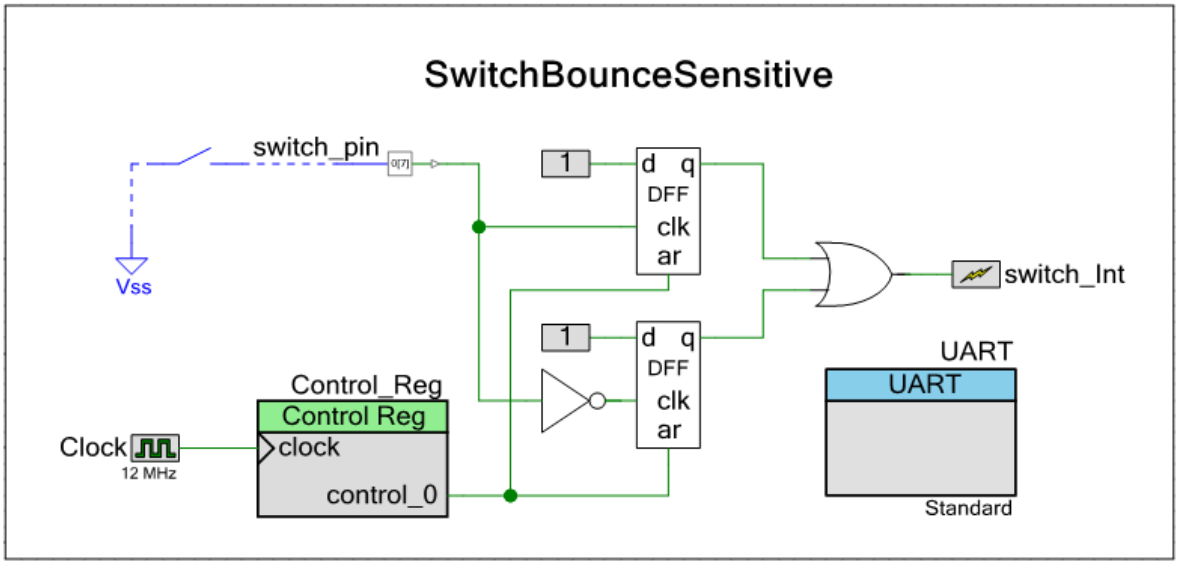
Examples 2, 3, and 4 show different methods of switch debouncing without using the PSoC Creator Debouncer and Glitch Filter Components. Examples 5 and 6 use those Components, respectively.

Note that all the examples disable interrupts before obtaining a copy of the count; and reenable interrupts afterwards. This allows for an accurate representation of the count without interrupts that may occur interfering with operations that use the count.

1_SwitchBounce_Sensitive

There is no switch debouncing implemented in this project, as Figure 3 shows. This means that bouncing from switch presses and releases will occur.

Figure 3. PSoC Creator Schematic for Project 1_SwitchBounce_Sensitive



In this example, the following firmware functions are performed:

1. Interrupt handlers and UART are configured. Interrupt sources are cleared. The count is initialized to zero.
2. When the count value changes, the UART displays the new value.

When the switch_Int interrupt occurs, SwInt_Handler is called, and the following functions are performed:

1. The interrupt source is cleared.
2. The count is incremented.

2_SwitchDebounce_Polling

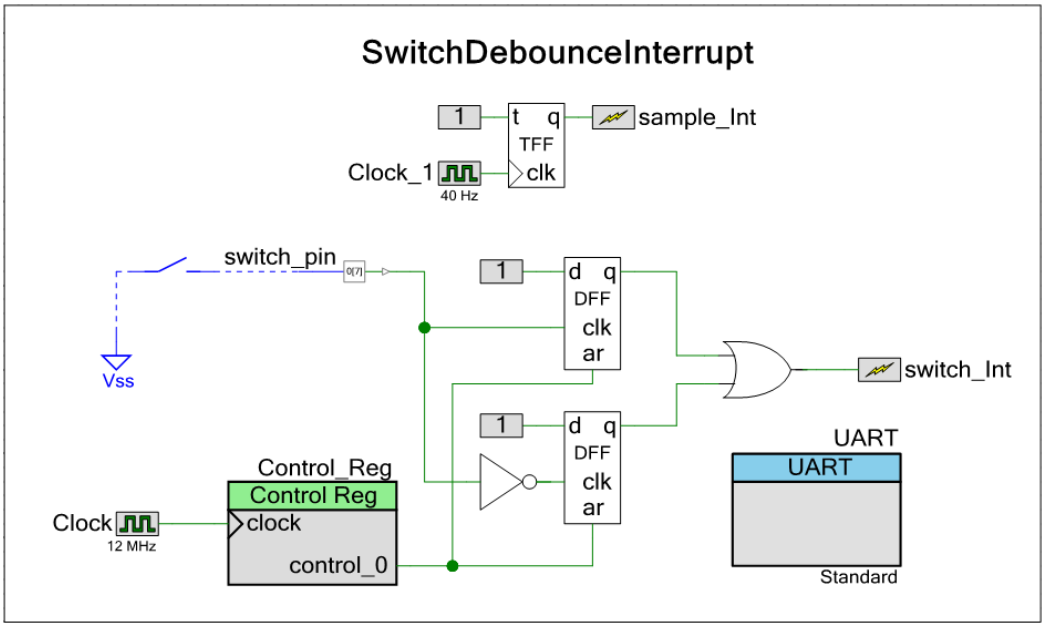
This project is similar to 1_SwitchBounce_Sensitive. It uses the same schematic as in Figure 3 and adds debouncing in firmware by directly sampling 'switch_pin' at periodic intervals, implemented by calling `CyDelay()`. Raw and filtered counts are displayed.

Note: Sampling rates for switch debouncing typically range from 10 Hz to 200 Hz.

3_SwitchDebounce_Interrupt

This project is similar to 1_SwitchBounce_Sensitive. The schematic shown in Figure 4 adds Components to generate periodic interrupts to sample the pin. The interrupt rate, and thus the pin sampling rate, is one half the frequency of Clock_1. When the 'sample_Int' interrupt occurs, if a switch transition is detected the filtered count is incremented.

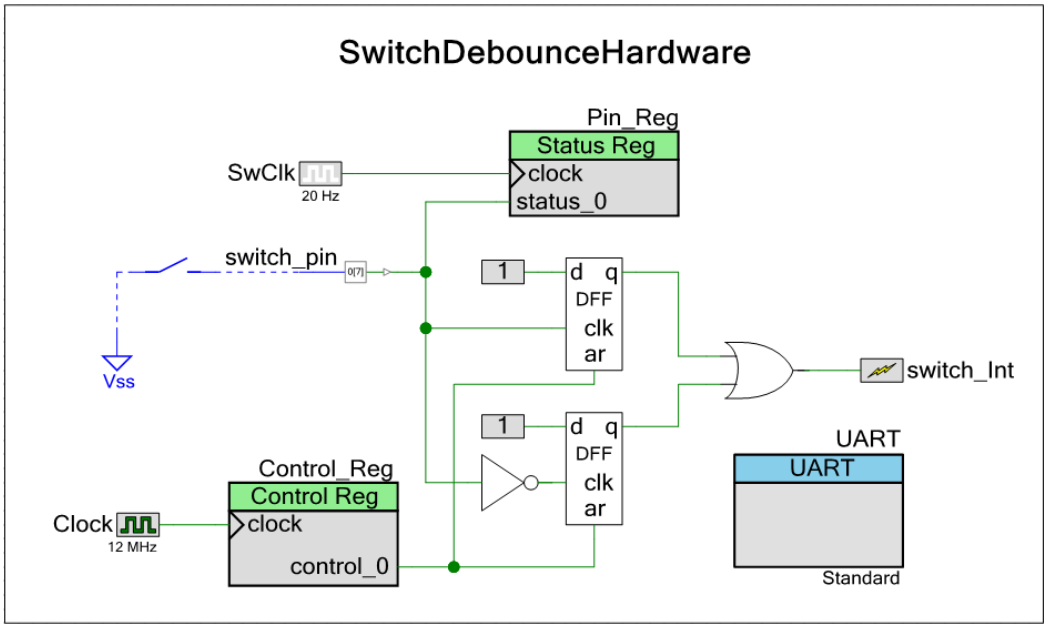
Figure 4. PSoC Creator Schematic for Project 3_SwitchDebounce_Interrupt



4_SwitchDebounce_Hardware

This project adds a Status Register Component to the schematic in Figure 3, as Figure 5 shows. The Status Register samples the pin in hardware instead of firmware. The sample rate is the frequency of SwClk. Firmware polls the Status Register to determine whether a transition has occurred.

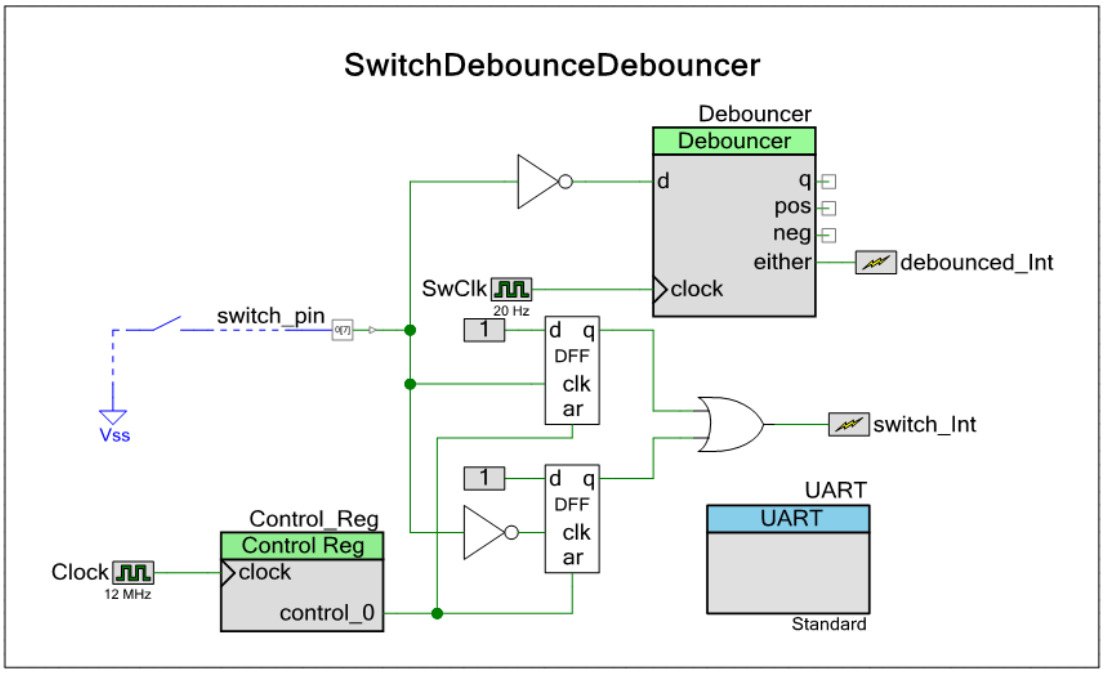
Figure 5. PSoC Creator Schematic for Project 4_SwitchDebounce_Hardware



5_SwitchDebounce_Debouncer

The Debouncer Component handles both debouncing and edge detection. In this example, the 'either' output activates for both press and release events, as Figure 6 shows. This output is connected to an interrupt; the handler simply increments the filtered count.

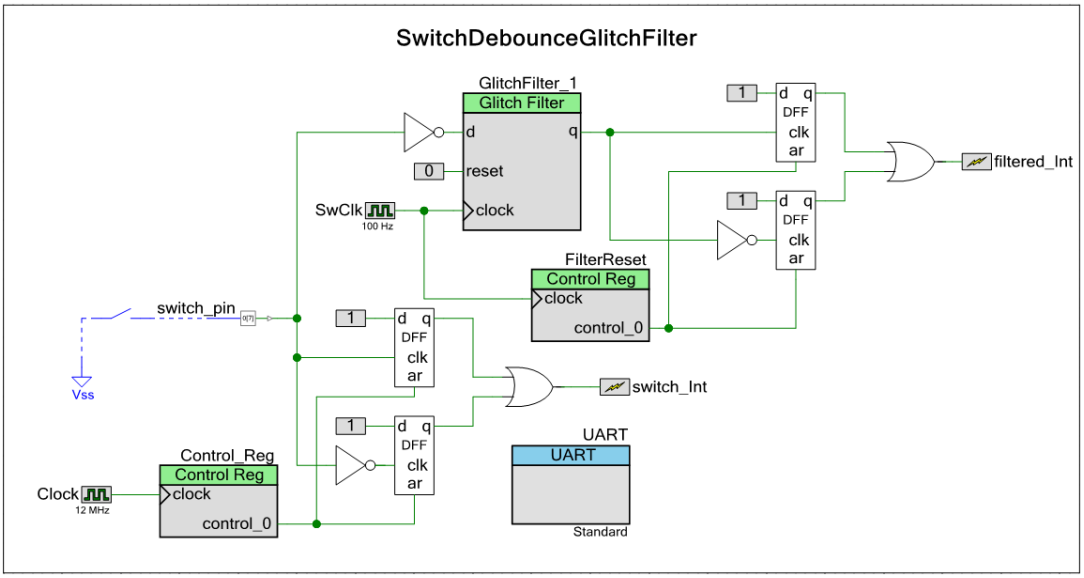
Figure 6. PSoC Creator Schematic for Project 5_SwitchDebounce_Debouncer



6_SwitchDebounce_GlitchFilter

The Glitch Filter Component is typically used to filter out unwanted pulses, for example due to noisy communication channels. But it can also be used for switch debouncing. The Glitch Filter Component does not include edge detection, so its output 'q' is routed to two DFFs, as Figure 7 shows. The inverter on the 'd' input removes a false initial edge detection.

Figure 7. PSoC Creator Schematic for Project 6_SwitchDebounce_GlitchFilter



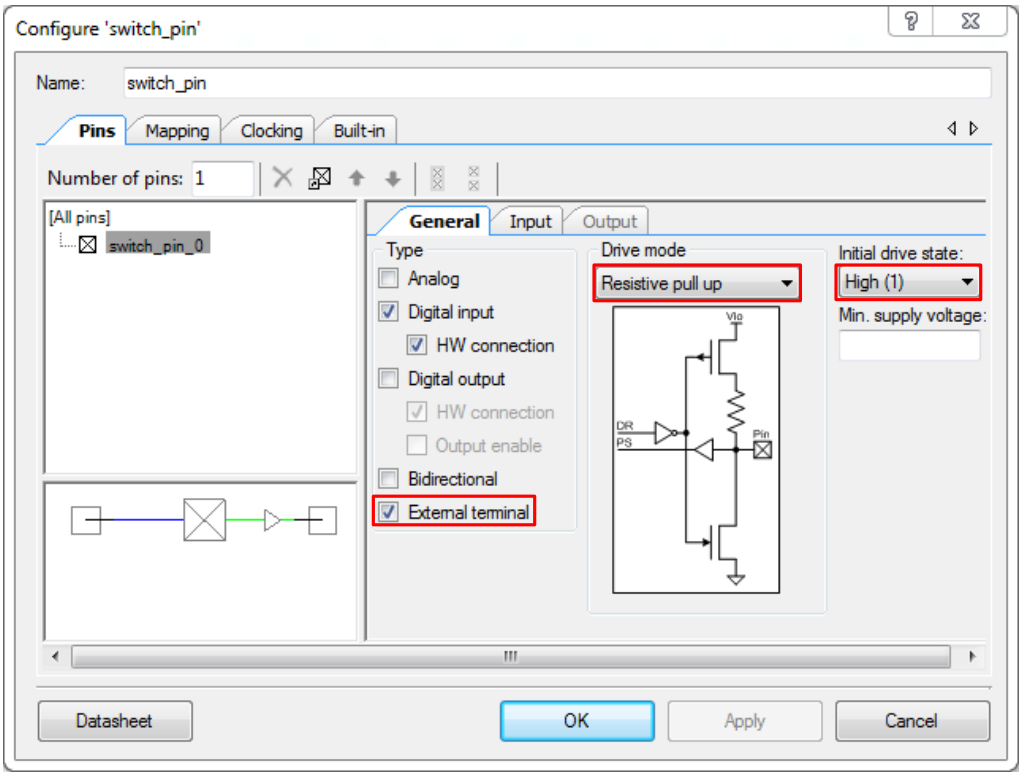
Components and Settings

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 and Settings

Component	Instance Name	Purpose	Non-default Settings
Digital Input Pin	switch_pin	Handles the SW2 connection on device	See Figure 8
Clock	Clock	Clock source for Components	None
D Flip Flop	cydff	Allows for digital value storage	Set PresetOrReset to Asynchronous Reset
Toggle Flip Flop	cy_tff	Allows digital values to be toggled	None
Control Reg	Control_Reg	CPU writes digital signal outputs	Set number of outputs to 1
Status Reg	Pin_Reg	CPU reads digital signal inputs	Set number of inputs to 1
Debouncer	Debouncer	Filters out unwanted transitions on digital inputs	None
Glitch Filter	GlitchFilter	Filters out unwanted “glitch” pulses on digital inputs	Set Glitch length (samples) to 1 (10 ms)
UART	UART	Handles UART communication	None

Figure 8. switch_pin Parameter Settings



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

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 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	CY8C4247LQI-BL483
CY8CKIT-044	CY8C4247AZI-M485
CY8CKIT-046	CY8C4248BZI-L489

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 device supports.

Related Documents

Application Notes	
AN79953 – Getting Started with PSoC® 4	Describes PSoC 4 and shows how to build the attached code example
AN60024 – PSoC® 3, PSoC 4, PSoC 5LP Switch Debouncer and Glitch Filter	Describes how switch debouncing can be done using hardware, software, or both on a PSoC device.
Code Examples	
CE224406 – Low Level API (Polling, ISR, DMA) using UART with PSoC 4	This example has three project files that use method include polling, ISR, and DMA. The polling example polls repeatedly. The ISR example uses user interrupts from terminal. The DMA example uses DMA functions.
PSoC Creator Component Datasheets	
SCB	A multifunction hardware block that implements the following communication Components: I2C, SPI, UART, and EZI2C.
Debouncer	This Component takes a signal and generates a clean output by eliminating unwanted oscillations on digital input lines. A common use is to debounce mechanical switches that tend to make multiple transitions.
Glitch Filter	This Component removes unwanted “glitch” pulses from a digital input signal that is usually high or low.
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 IO pin.
D Flip Flop	The D Flip Flop is used to store digital values. It includes features such as Synchronous / Asynchronous reset or preset. The width of array for input and output can be configured.
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.
Control Register	The Control Register allows the firmware to output digital signals.
Status Register	The Status Register allows the firmware to read digital signals.
Device Documentation	
PSoC 4 Datasheets	PSoC 4 Technical Reference Manuals
Development Kit (DVK) Documentation	
CY8CKIT-042 PSoC® 4 Pioneer Kit	
PSoC 4 Kits	
Tool Documentation	
PSoC Creator	Go to the **Downloads** tab for Quick Start and User Guides

Document History

Document Title: CE224719 – PSoC 4 Debouncer and Glitch Filter

Document Number: 002-24719

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6297188	SYAO	09/07/2018	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

Community 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.