

## Objective

This example demonstrates the fault handling functionality of PSoC® 6 MCU using Peripheral Driver Library (PDL) System Library (SysLib) and ModusToolbox™ integrated development environment (IDE).

## 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](#), [PSoC 6 WiFi-BT Prototyping Kit](#)

## Overview

This code example demonstrates how to find a fault location using the PDL SysLib API and the Arm® exception handler. The example has two different faults: Cortex® M4 Usage Fault and a Cortex M4 Bus Fault. The example uses a UART to display information for debugging.

## Hardware Setup

This example uses the PSoC 6 BLE Pioneer Kit's default configuration. See the kit guide to make sure the kit is configured correctly.

**Note:** The PSoC 6 BLE Pioneer kit and the PSoC 6 WiFi-BT Pioneer kit ship with KitProg2. ModusToolbox works only 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 If you do not upgrade, you will see an error like "unable to find CMSIS-DAP device" or "KitProg firmware is out of date".

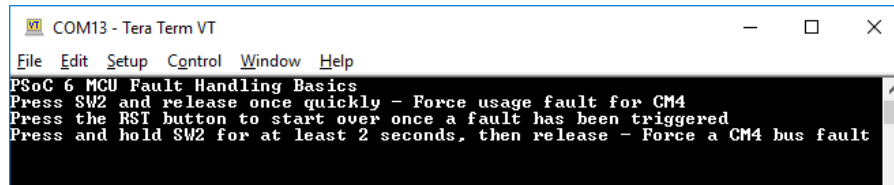
## Software Setup

This example uses a terminal emulator. Install a terminal emulator such as [Tera Term](#).

## Operation

1. Connect the kit to your PC using the provided USB cable.
2. Open your terminal software and select the KitProg COM port. Set the other serial port parameters as follows:
  - a. Baud rate: 115200bps
  - b. Data: 8-bit
  - c. Parity: None
  - d. Stop: 1-bit
  - e. Flow control: None
3. Import the code example into a new workspace. See [KBA225201](#) for more details.
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.
5. Confirm that the terminal program is working. The terminal application should show a message with some operating instructions, like [Figure 1](#).

Figure 1. Operating Instructions in the Terminal Window

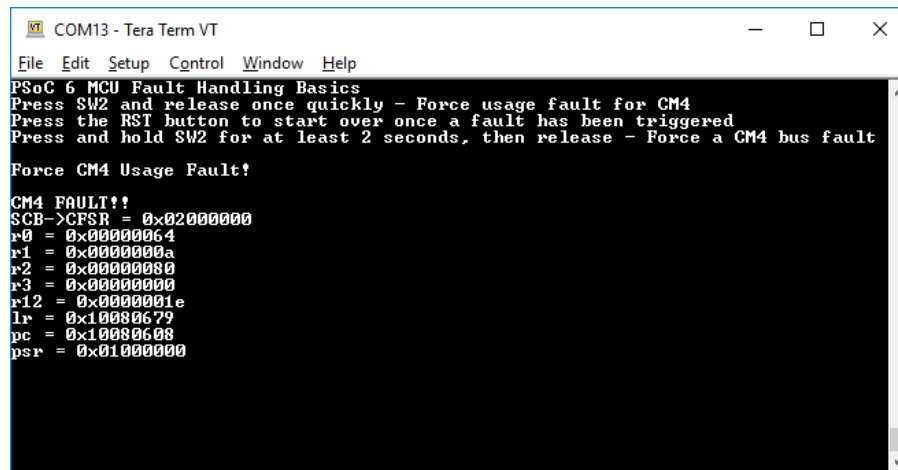


```

COM13 - Tera Term VT
File Edit Setup Control Window Help
PSoC 6 MCU Fault Handling Basics
Press SW2 and release once quickly - Force usage fault for CM4
Press the RST button to start over once a fault has been triggered
Press and hold SW2 for at least 2 seconds, then release - Force a CM4 bus fault
  
```

- Press the SW2 button and release once quickly to cause the CM4 Usage Fault. CM4 Usage Fault message appears in the terminal window.

Figure 2. Fault Frame Information for the Usage Fault



```

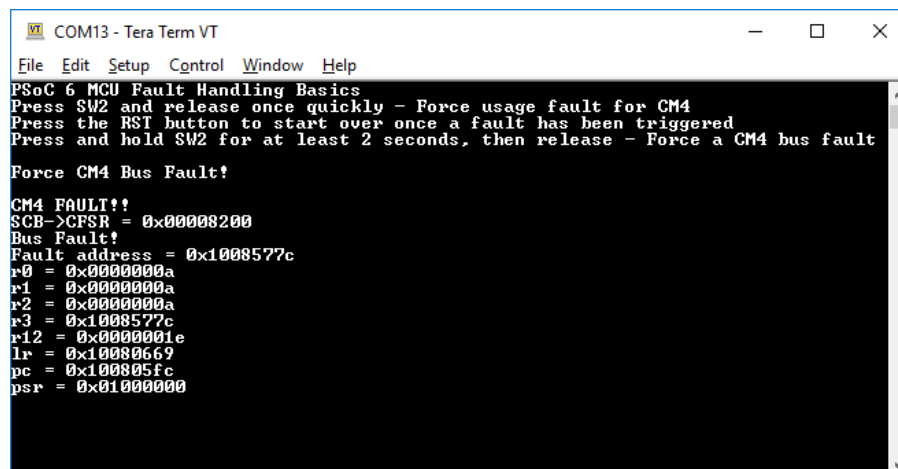
COM13 - Tera Term VT
File Edit Setup Control Window Help
PSoC 6 MCU Fault Handling Basics
Press SW2 and release once quickly - Force usage fault for CM4
Press the RST button to start over once a fault has been triggered
Press and hold SW2 for at least 2 seconds, then release - Force a CM4 bus fault

Force CM4 Usage Fault!

CM4 FAULT!
SCB->CFSR = 0x02000000
r0 = 0x00000064
r1 = 0x0000000a
r2 = 0x00000080
r3 = 0x00000000
r12 = 0x0000001e
lr = 0x10080679
pc = 0x10080608
psr = 0x01000000
  
```

- Press the reset button (SW1). The opening message appears in the terminal again. The board needs to be reset because the CM4 bus fault will not occur after the CM4 usage fault.
- Press and hold SW2 for approximately two seconds and then release the button. The CM4 Bus Fault occurs when you release SW2. The CM4 Bus Fault message appears in the terminal window.

Figure 3. Fault Frame Information for the Bus Fault



```

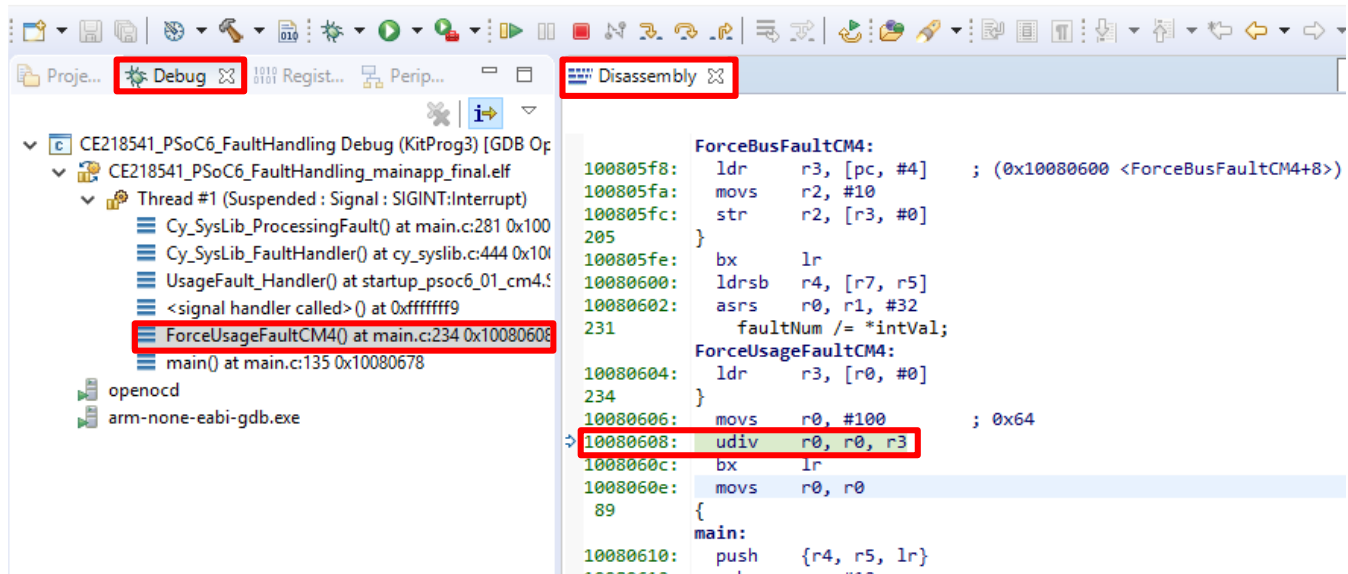
COM13 - Tera Term VT
File Edit Setup Control Window Help
PSoC 6 MCU Fault Handling Basics
Press SW2 and release once quickly - Force usage fault for CM4
Press the RST button to start over once a fault has been triggered
Press and hold SW2 for at least 2 seconds, then release - Force a CM4 bus fault

Force CM4 Bus Fault!

CM4 FAULT!
SCB->CFSR = 0x00008200
Bus Fault!
Fault address = 0x1008577c
r0 = 0x0000000a
r1 = 0x0000000a
r2 = 0x0000000a
r3 = 0x1008577c
r12 = 0x0000001e
lr = 0x10080669
pc = 0x100805fc
psr = 0x01000000
  
```

9. You can compare the Fault PC with the disassembly code. Start a debug session. For more details, see [KBA224621](#). Figure 4 shows the example for Usage Fault.

Figure 4. Usage Fault Disassembly

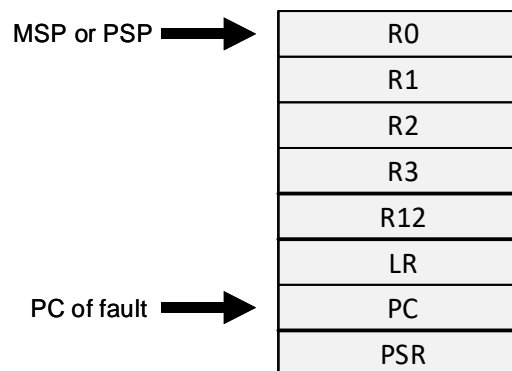


## Design and Implementation

This code example uses architecture and registers provided by Arm for tracking fault expression. The PSoC 6 MCU startup code provides the handling routine, which passes the stack pointer of the exception frame (as shown in Figure 5) into `Cy_SysLib_FaultHandler()`. The handler stores the information using Main Stack Pointer (MSP) or Process Stack Pointer (PSP) so you can debug the fault. This information includes the program counter (PC) value of the fault and the following registers: R0, R1, R2, R3, R12, Link Register (LR), and Program Status Register (PSR) for both Cortex M0+ and CM4.

CM4 has more system control registers that can configure the fault type and read the detailed root cause of a fault. Learn more about the Arm CM4 System Control Block at the [Arm Information center](#)

Figure 5. Arm Cortex M Exception Frame Without Floating-Point Storage



After storing the information, the handler calls `__WEAK void Cy_SysLib_ProcessingFault()`. The default implementation of this function is an infinite loop. You can override this function with a custom function, because of the weak linkage

This code example generates two different faults:

- CM4 Bus Fault exception
- CM4 Usage Fault exception

## Resources and Settings

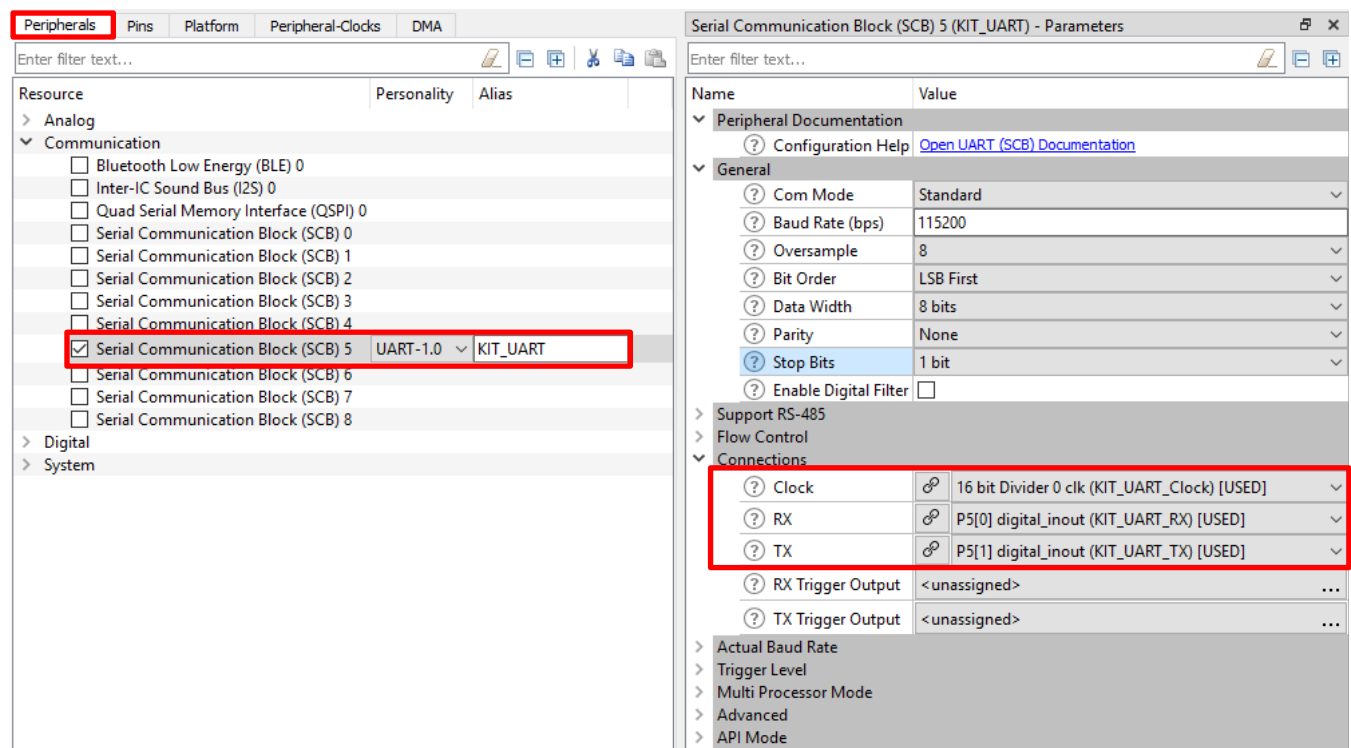
Table 1 lists some of the ModusToolbox resources used in the example, and how they are used in the design. The *design.modus* file contains all the configuration settings. For example, for pin usage and configuration, open the **Pins** tab of the design file.

Table 1. ModusToolbox Resources

Resource	Alias	Purpose	Non-default Settings
SCB	KIT_UART	Provide communication between the host and target	See Figure 6
Digital Output Pin	KIT_UART_TX	Used for UART transmit (Tx)	See Figure 7
Digital Input Pin	KIT_UART_RX	Used for UART receive (Rx)	See Figure 8
	KIT_BTN1	Provide user interface	See Figure 9

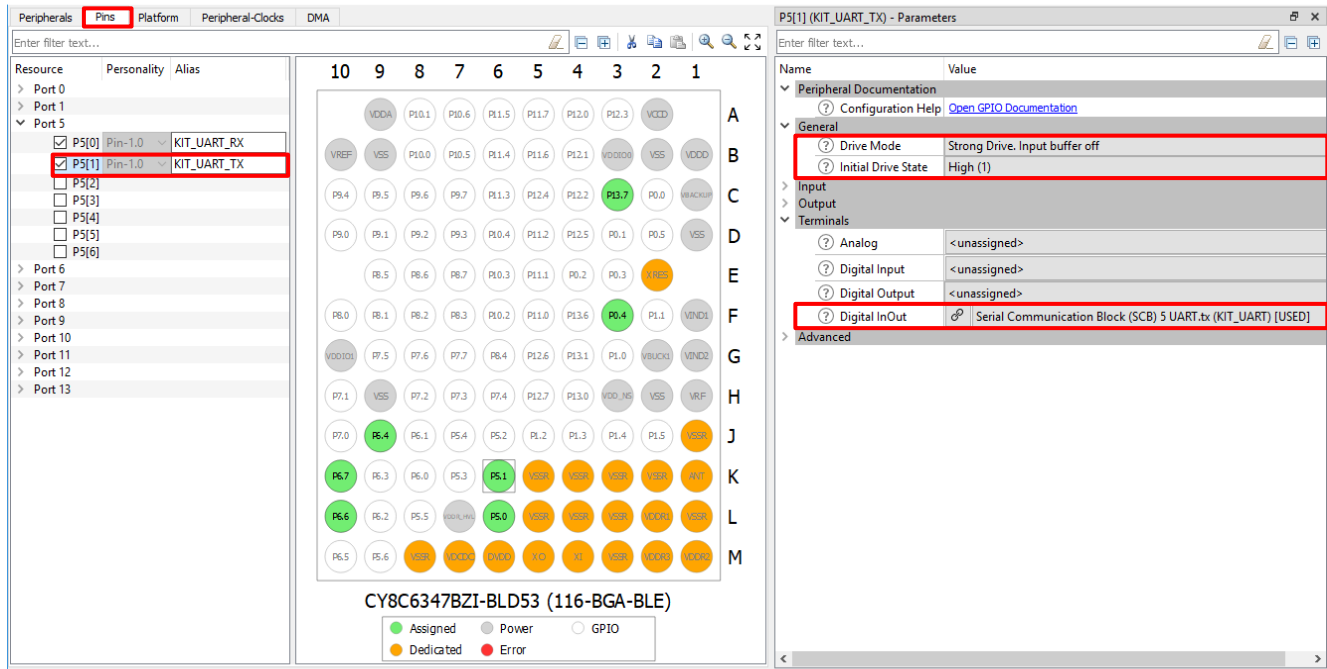
Figure 6 through Figure 9 highlight the non-default settings for each resource in this example.

Figure 6. UART Configuration



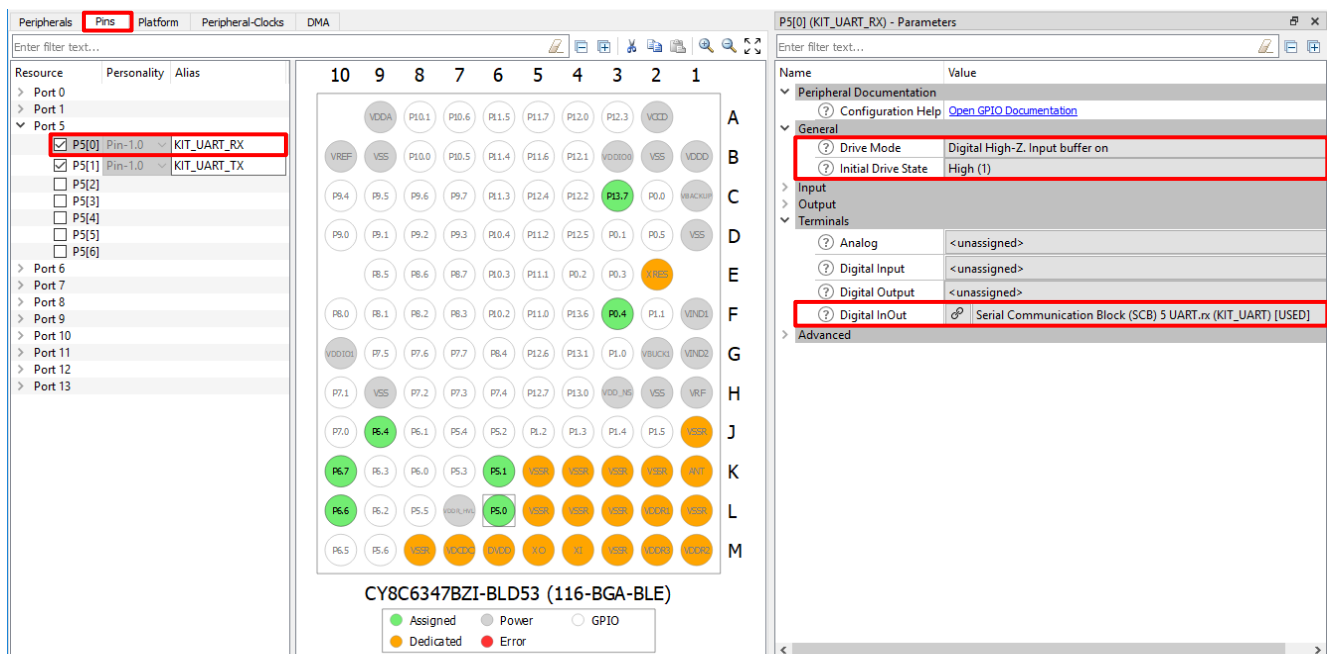
The screenshot shows the ModusToolbox interface for configuring the UART peripheral. On the left, the 'Peripherals' tab is active, and 'Serial Communication Block (SCB) 5' is selected with the alias 'KIT\_UART'. The main window displays the 'Parameters' for this peripheral. The 'General' section is expanded, showing various configuration options. The 'Connections' section is also expanded, showing the clock and data lines. The RX and TX connections are highlighted with a red box, indicating they are configured to use specific pins (P5[0] and P5[1]).

Figure 7. GPIO Pin Configuration for UART Tx



The screenshot shows the PSoC Designer interface for configuring the GPIO pins of a PSoC 6 MCU. The 'Pins' tab is selected, and the pin P5[1] is assigned to the KIT\_UART\_TX peripheral. The configuration parameters for P5[1] are shown in the 'Parameters' window on the right. The 'General' section shows the Drive Mode as 'Strong Drive, Input buffer off' and the Initial Drive State as 'High (1)'. The 'Advanced' section shows the Digital InOut as 'Serial Communication Block (SCB) 5 UART.tx (KIT\_UART) [USED]'.

Figure 8. GPIO Pin configuration for UART Rx



The screenshot shows the PSoC Designer interface for configuring the GPIO pins of a PSoC 6 MCU. The 'Pins' tab is selected, and the pin P5[0] is assigned to the KIT\_UART\_RX peripheral. The configuration parameters for P5[0] are shown in the 'Parameters' window on the right. The 'General' section shows the Drive Mode as 'Digital High-Z, Input buffer on' and the Initial Drive State as 'High (1)'. The 'Advanced' section shows the Digital InOut as 'Serial Communication Block (SCB) 5 UART.rx (KIT\_UART) [USED]'.

Figure 9. GPIO Configuration for Switch

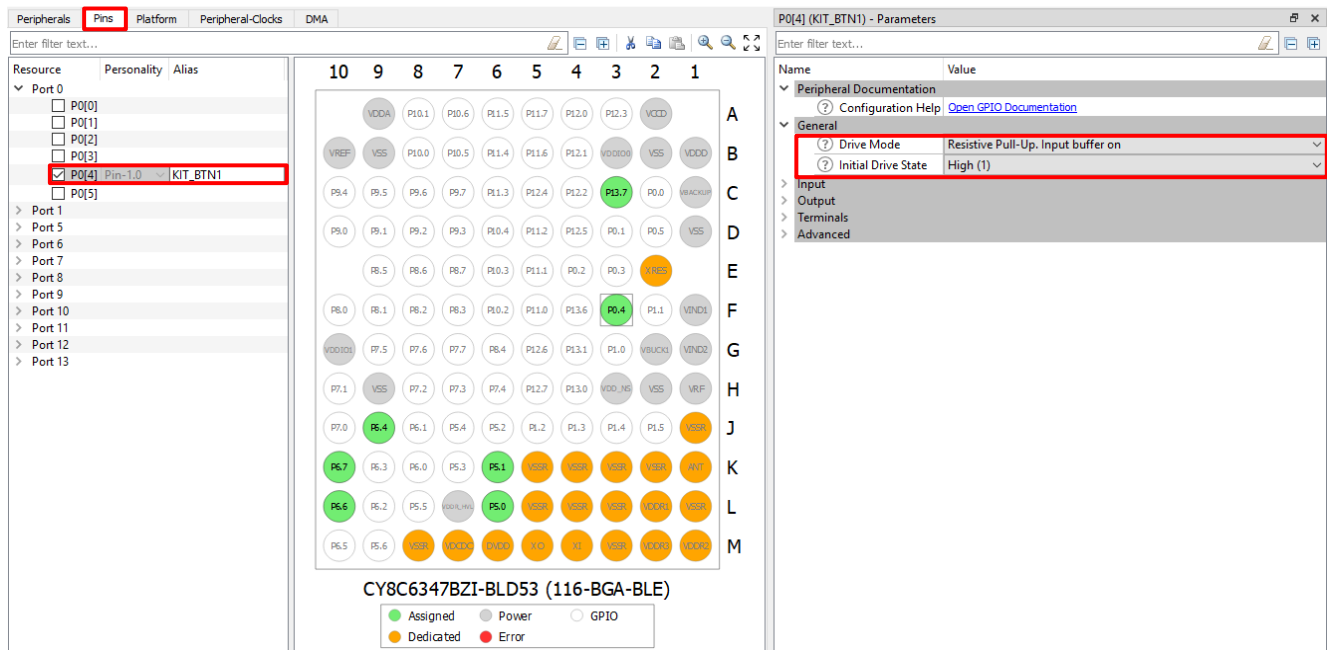
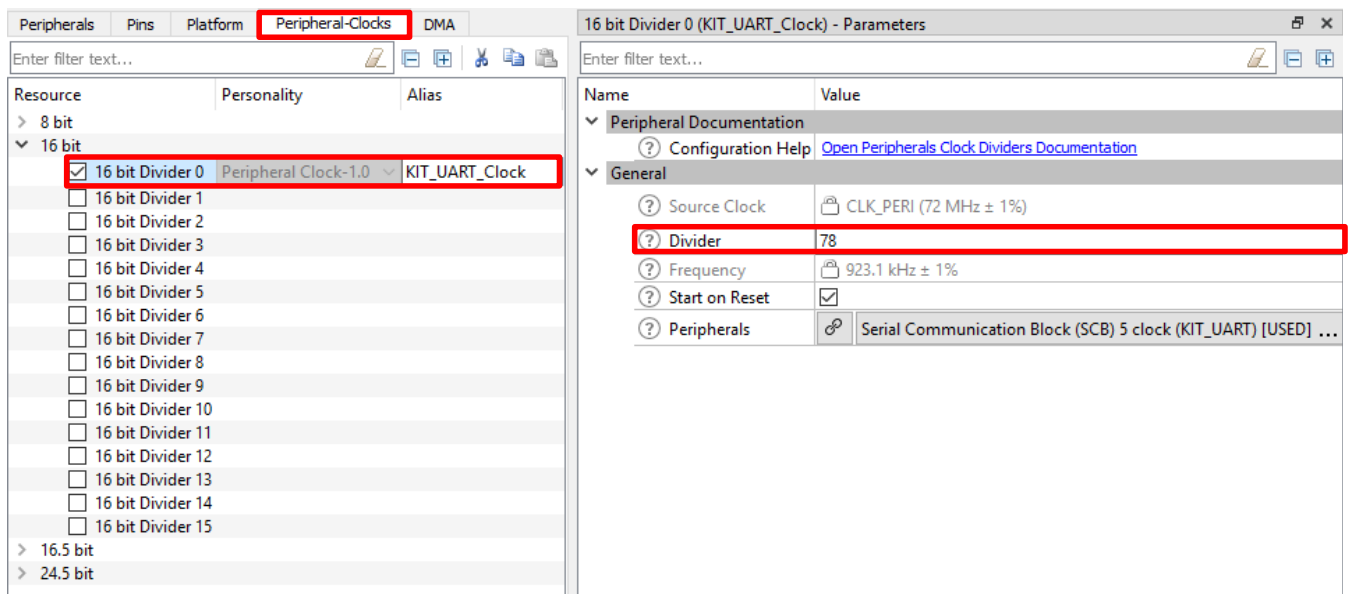


Figure 10 shows the non-default configuration for UART clock.

Figure 10. Peripheral Clock Configuration for UART



## Reusing This Example

This example is configured for the supported kit(s). 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 (see [Table 2](#)).

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

Kit Name	Device Used	KIT_UART_RX	KIT_UART_TX	KIT_BTN1
CY8CKIT-062-WiFi-BT	CY8C6247BZI-D54	P5[0]	P5[1]	P0[4]
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	P5[0]	P5[1]	P0[4]
CY8CPROTO-062-4343W	CY8C624ABZI-D44	P5[0]	P5[1]	P0[4]

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 which resources a device supports.

## Related Documents

Application Notes	
<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 PSoC Creator project
<a href="#">AN221774</a> – Getting Started with PSoC 6 MCU	Describes PSoC 6 MCU devices and how to build your first 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 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>	
<a href="#">CY8CPROTO-063 BLE PSoC 6 BLE Prototyping Kit</a>	
Tool Documentation	
<a href="#">ModusToolbox IDE</a>	ModusToolbox simplifies development for IoT designers. It delivers easy-to-use tools and a familiar microcontroller (MCU) integrated development environment (IDE) for Windows, macOS, and Linux.

## 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: CE218541 – PSoC 6 MCU Fault-Handling Basics

Document Number: 002-25737

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6393573	AJYA	11/26/2018	New code example
*A	6489136	AJYA	02/19/2018	Updated code example to 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.