CE218541 – PSoC 6 MCU Fault-Handling Basics

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 If you do not upgrade, you will see an error like "unable to find CMSIS-DAP device" or "KitProg firmware is out of date".

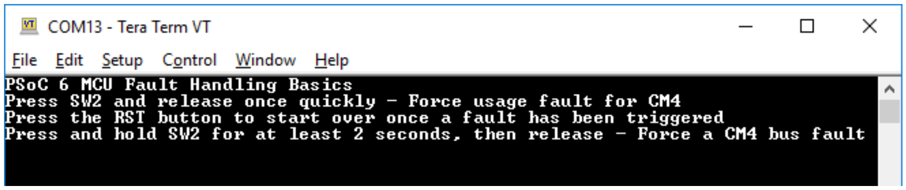
Software Setup

This example uses a terminal emulator. Install a terminal emulator such as Tera Term.

Operation

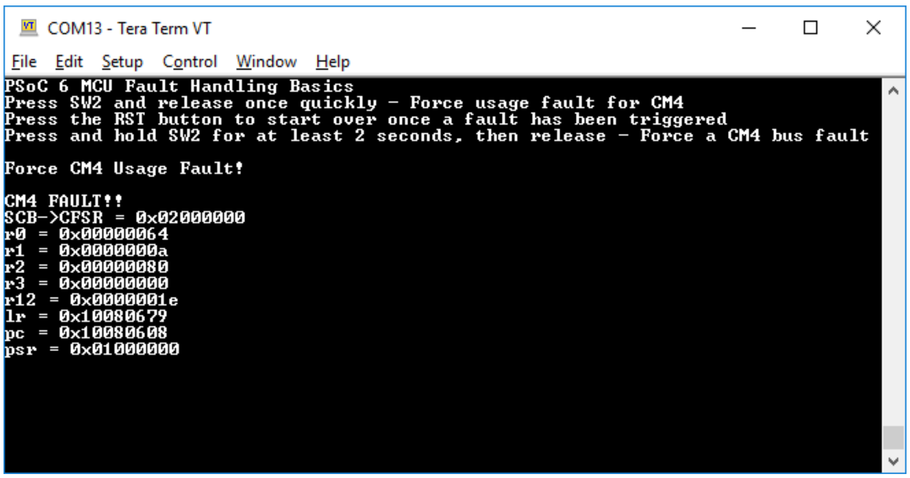
1. Connect the kit to your PC using the provided USB cable.
2. Open your terminal software and select the KitProg COM port. Set the other serial port parameters as follows:
    a. Baud rate: 115200bps
    b. Data: 8-bit
    c. Parity: None
    d. Stop: 1-bit
    e. Flow control: None
3. Import the code 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



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

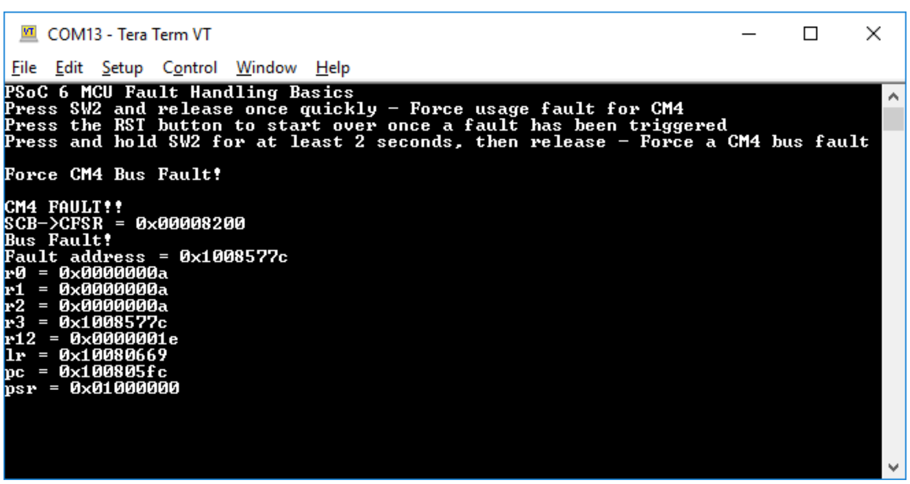
Figure 2. Fault Frame Information for the Usage Fault



7. 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.

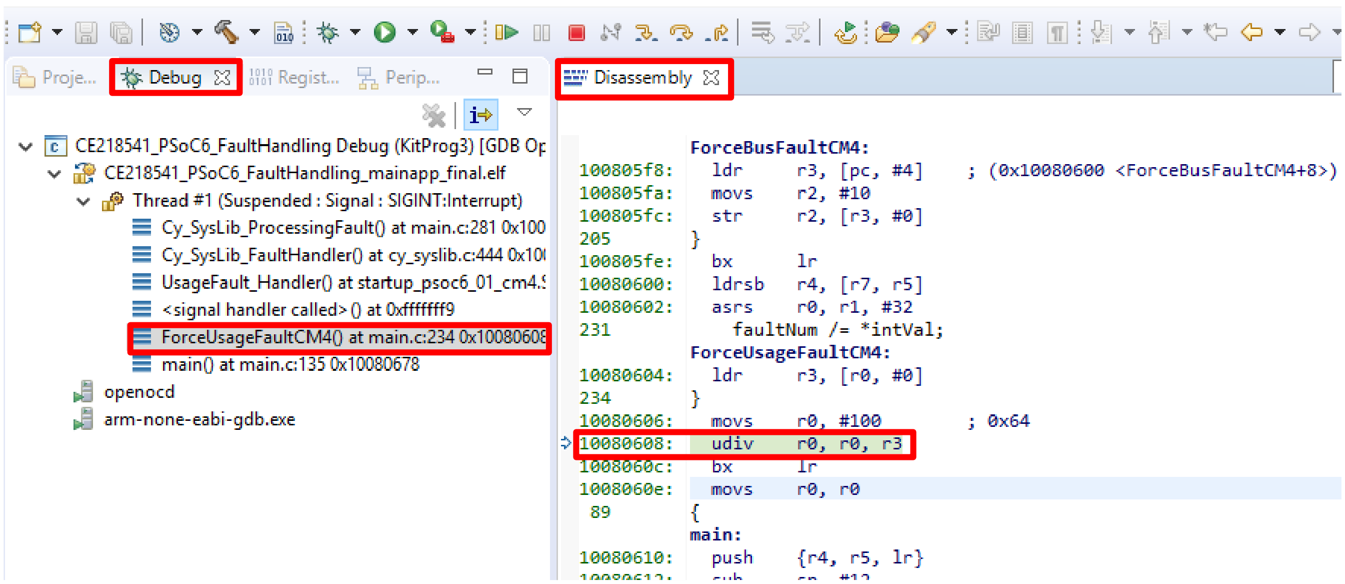
8. Press and hold SW2 for approximately two seconds and then release the button. The CM4 Bus Fault occurs when you release SW2. The CM4 Bus Fault message appears in the terminal window.

Figure 3. Fault Frame Information for the Bus Fault



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

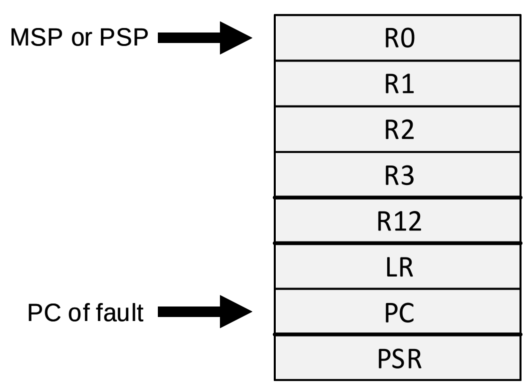


Design and Implementation

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

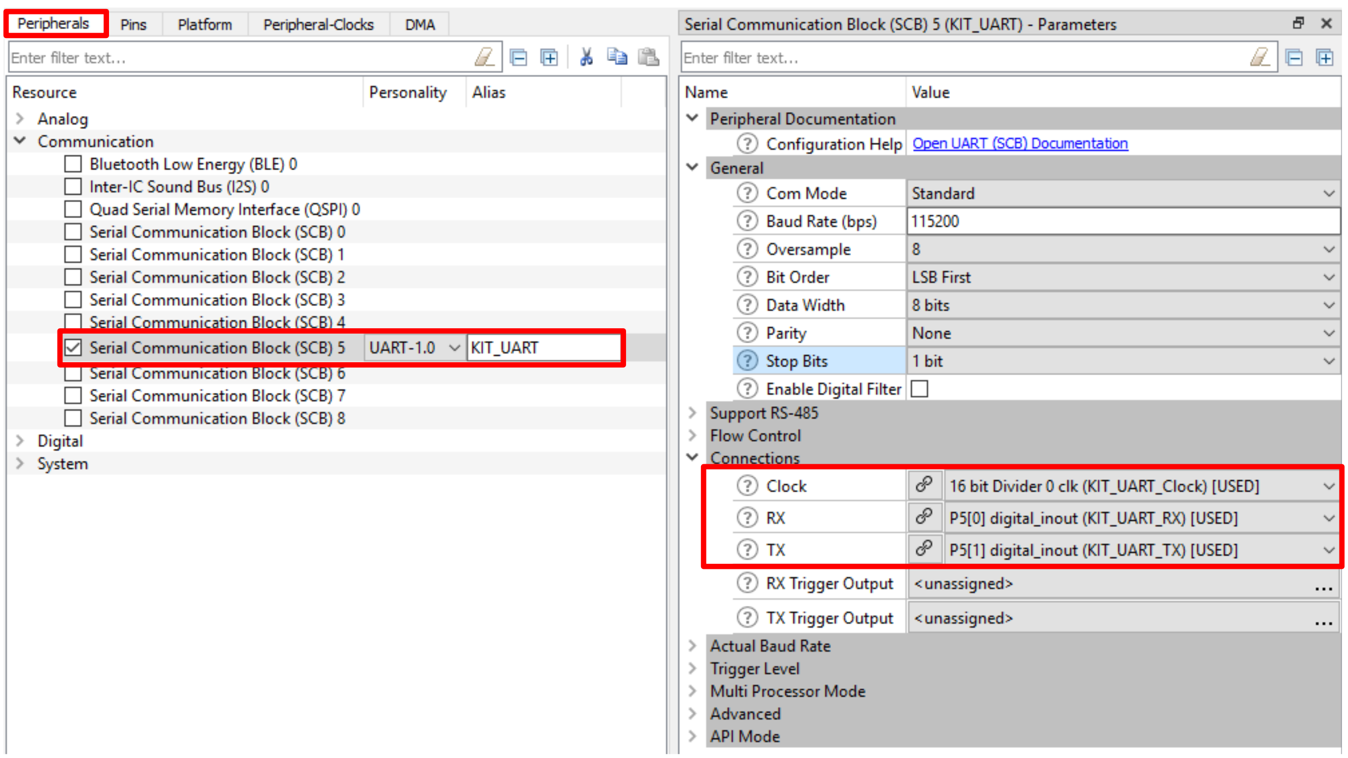


Figure 7. GPIO Pin Configuration for UART Tx

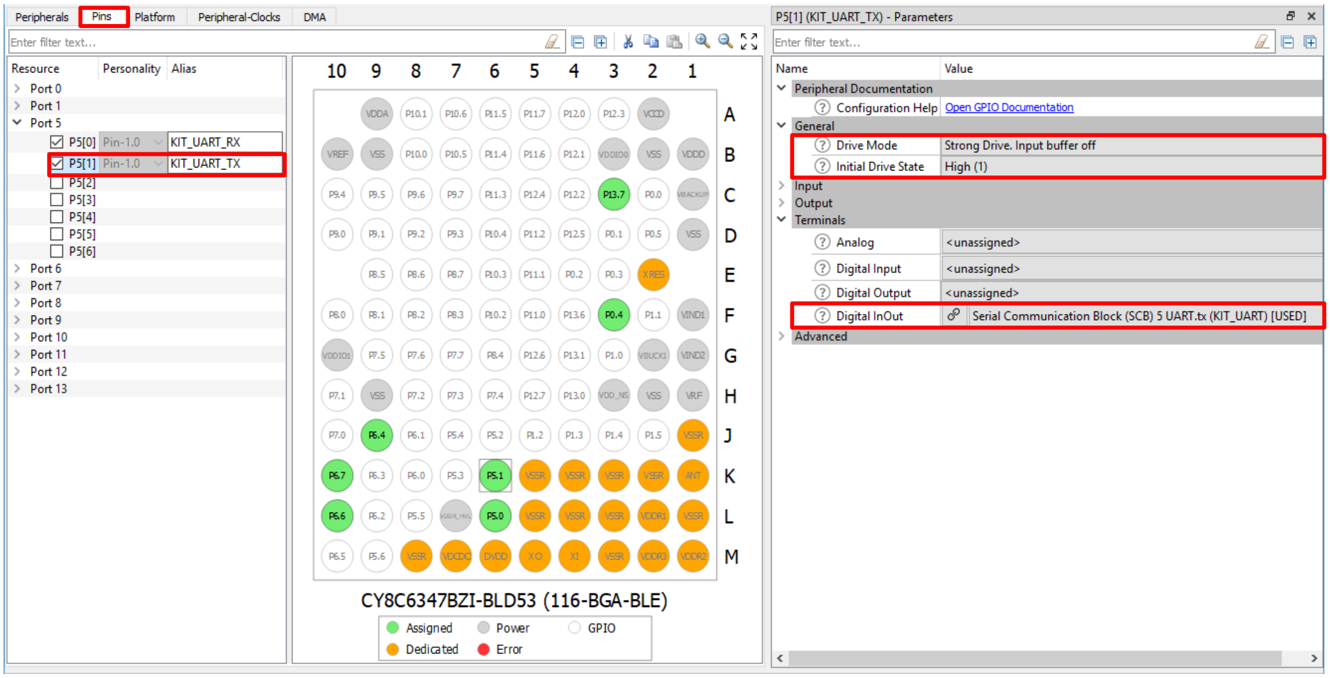


Figure 8. GPIO Pin configuration for UART Rx

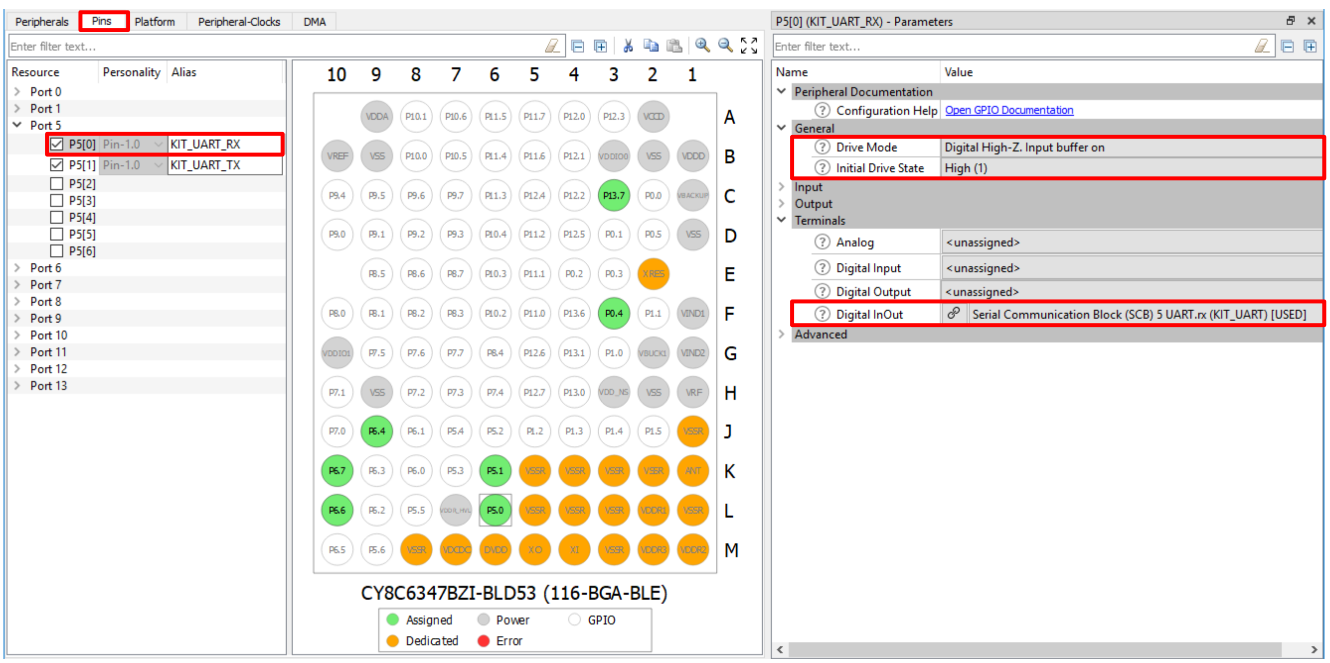


Figure 9. GPIO Configuration for Switch

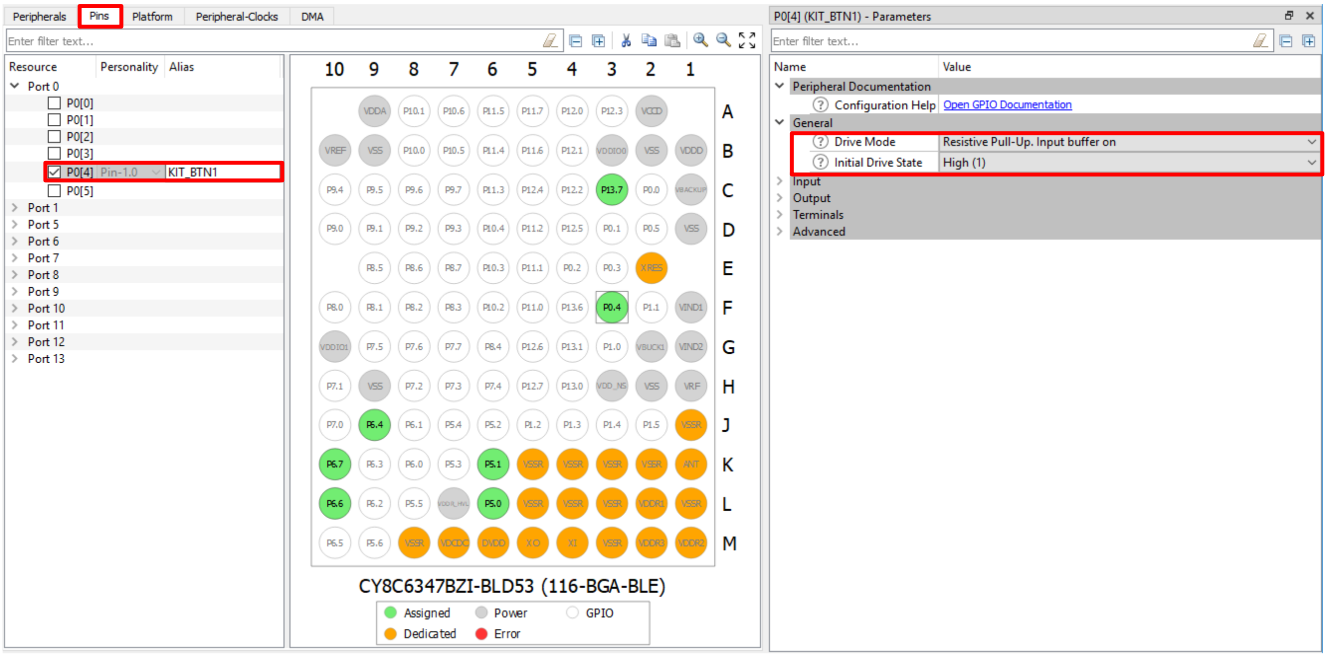
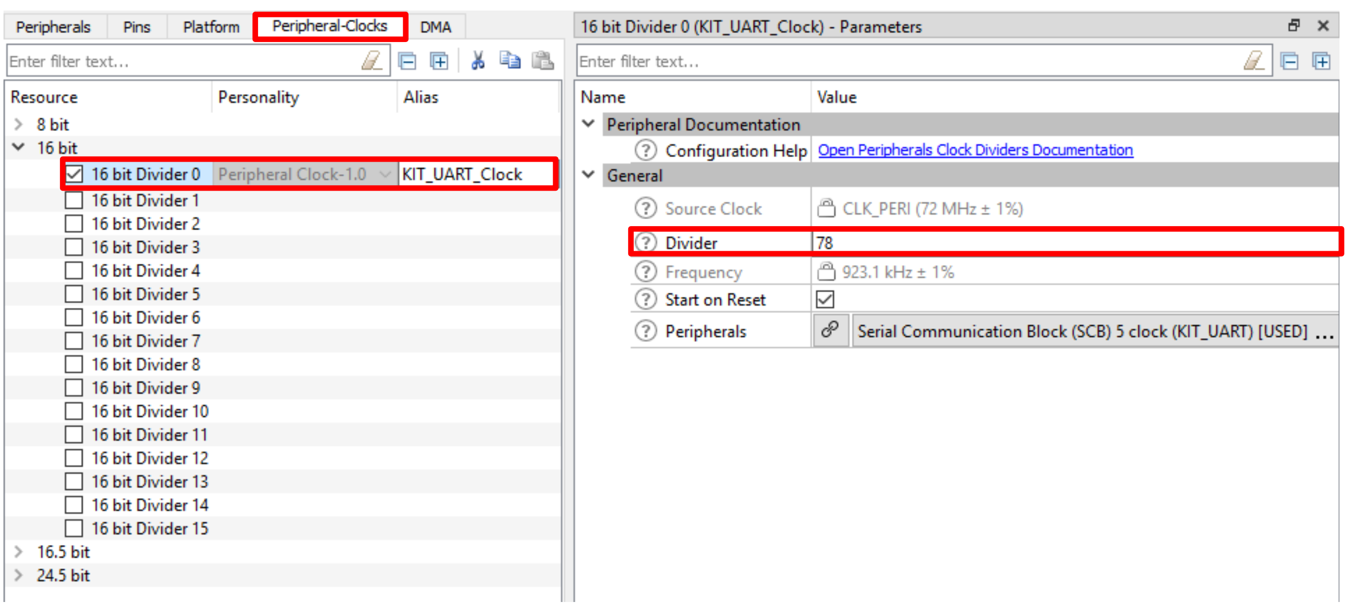


Figure 10 shows the non-default configuration for UART clock.

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	
AN210781 – 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
AN221774 – Getting Started with PSoC 6 MCU	Describes PSoC 6 MCU devices and how to build your first PSoC Creator project
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	
CY8CPROTO-063 BLE PSoC 6 BLE Prototyping Kit	
Tool Documentation	
ModusToolbox IDE	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 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	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