Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE218541 - PSoC 6 MCU Fault-Handling Basics

Objective

This example demonstrates the fault handling functionality of PSoC® 6 MCU, using Peripheral Driver Library (PDL) Sysyem Library (SysLib).

Overview

This code example demonstrates how to find a fault location using the PDL SysLib and the Arm® exception handler. The example has three different faults: Arm Cortex® M0+ Hard Fault, Cortex M4 Usage Fault, and a Cortex M4 (CM4) Bus Fault. The example uses a UART to display information for debugging.

Requirements

Tool: PSoC Creator™ 4.2; Peripheral Driver Library (PDL) 3.0.1

Programming Language: C (Arm® GCC 5.4-2016-q2-update)

Associated Parts: All PSoC 6 MCU parts

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure that the kit is configured correctly.

Software Setup

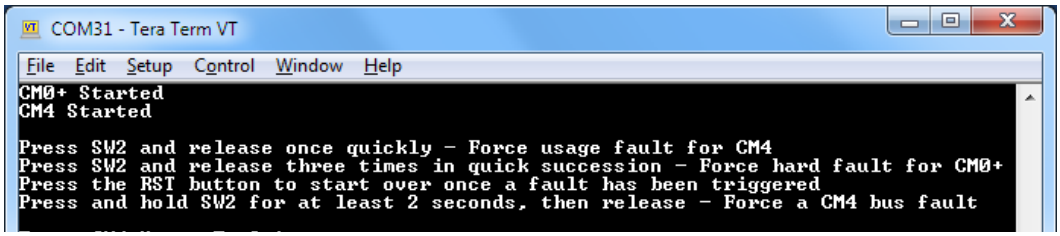
A terminal software is needed for this project.

Operation

1. Connect the CY8CKIT-062-BLE PSoC 63 with BLE Connectivity Pioneer Kit baseboard to your computer's USB port.
2. Open a PC terminal tool. Use a tool like Tera Term or PuTTY. Configure it for 115,200 baud at 8N1 to match the UART Component.
3. Build the project and program it into the PSoC 6 MCU device. For more information on device programming, see PSoC. Creator Help. The flash for both CPUs is programmed in a single program operation.

   Note: Do not delete or replace "*stdio_user.h*" file.
4. Confirm that the terminal program is working. It should show a message with some operating instructions, like Figure 1.

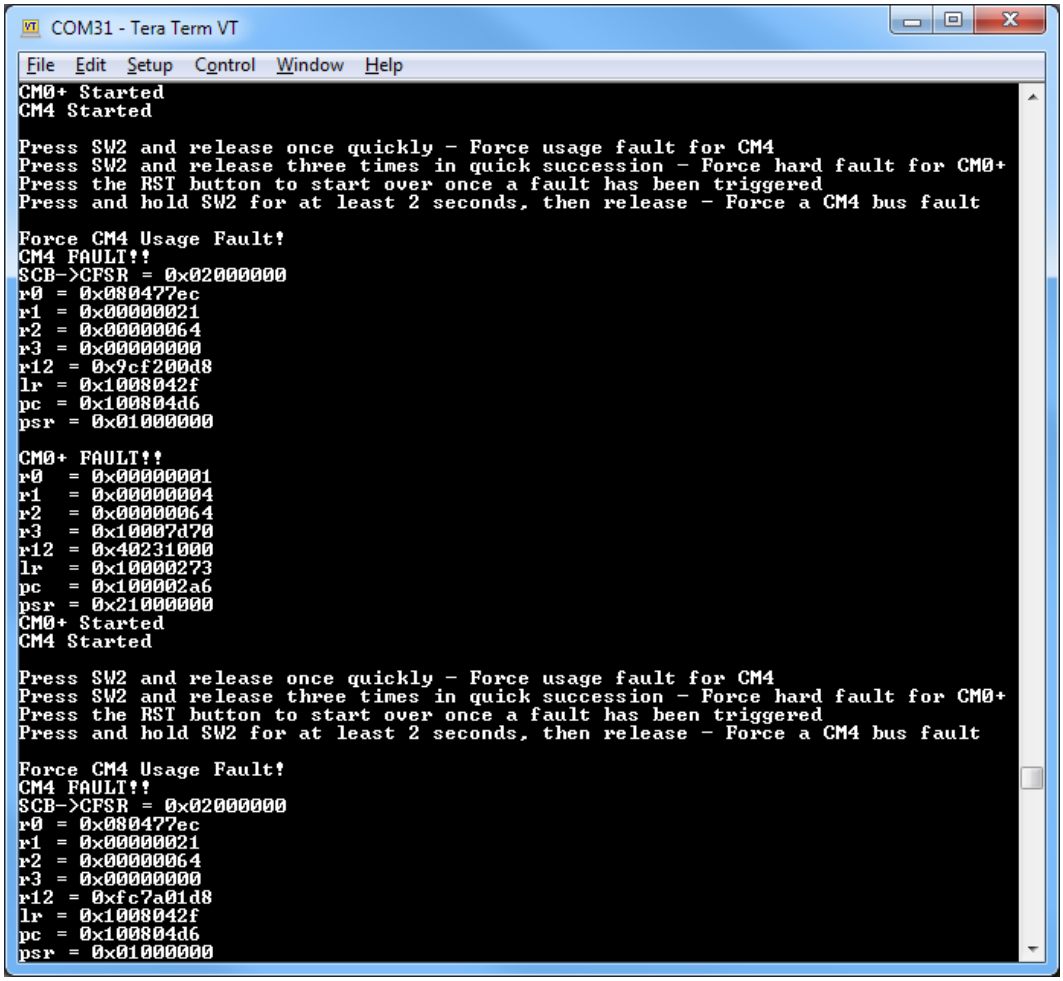
Figure 1. Operating Instructions in the Terminal Window



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

6. Press and release SW2 twice more, quickly, to cause the second fault. This makes a total of three button releases after boot up. The CM0+ exception occurs when you release SW2 for the third time. The CM0+ Hard Fault message appears in the terminal window.
7. Press the reset button (SW1). The opening message appears in the terminal again. Resetting the board is required 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. At this point, you have generated all three faults. Fault frames for each are in the terminal window. In the remaining steps, you compare some of the information in the fault frames with the disassembly output. Your terminal window looks something like Figure 2.

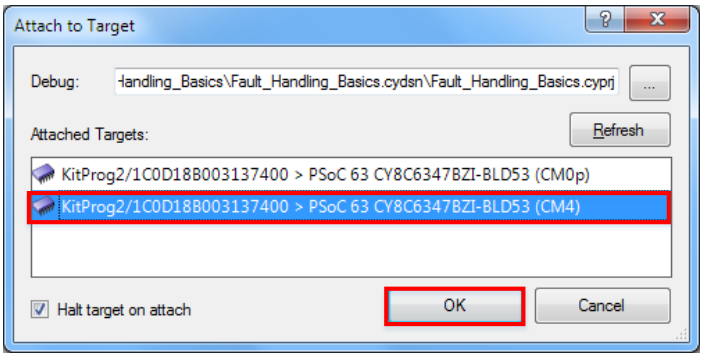
Figure 2. Fault Frame Information in the Terminal Window



9. Attach the debugger to the CM4 target. **Choose Debug** > **Attach to Running Process**. The Attach to Target dialog appears, as shown in Figure 3. Select the Arm CM4 target and click **OK**.

   The debugger launches, connects to the process, and halts execution.

Figure 3. Attach to the CM4 Target

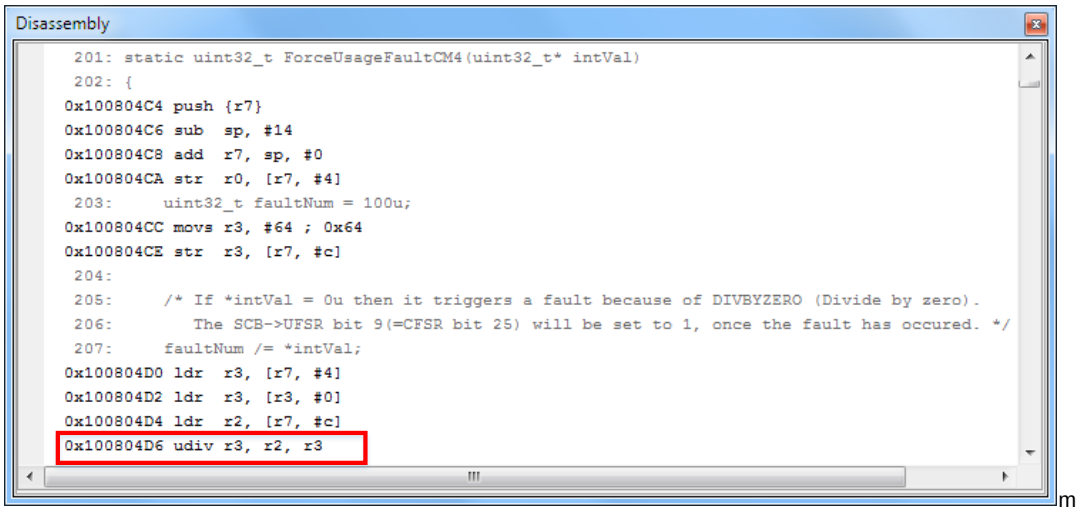


10. Open the CM4 disassembly window. **Choose Debug** > **Windows > Disassembly**. The disassembly window appears.

    Note: If you are working with custom hardware, you cannot attach the debugger.

11. Compare the CM4 Usage Fault PC with the disassembly code. In the terminal window, locate the fault frame for the CM4 usage fault, and note the PC address. In the disassembly window, scroll to that address. This is where the fault occurred. For example, if the fault frame indicates the fault occurred at PC 0x100804D6, then in the disassembly you see information like Figure 4.

Figure 4. CM4 Usage Fault Disassembly



You can perform similar operation to debug other faults.

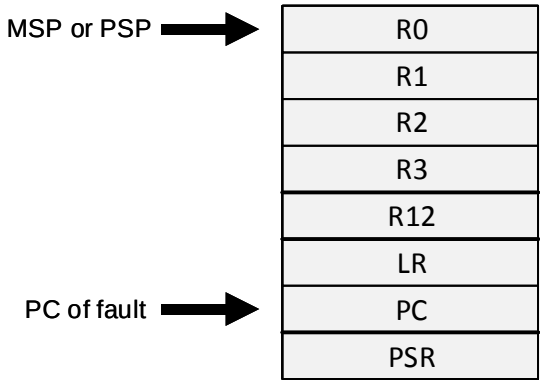
Design and Implementation

PSoC Creator does not provide any Component for Arm fault handling because Arm cores already provide the proper architecture and registers to track fault exceptions.

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 Point (MSP) or Process Stack Point (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 Cortex M4.

Cortex M4 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 Cortex M4 System Control Block at the Arm Information center.

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



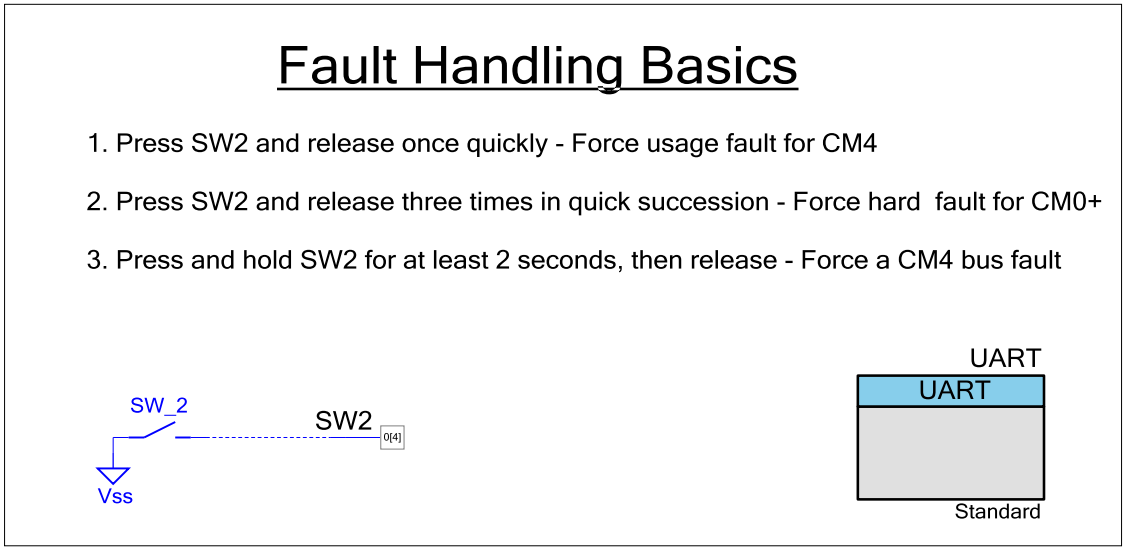
After storing the information, the handler calls `__WEAK void Cy_SysLib_ProcessingFault()`. The default implementation of this function is an infinite loop. Because of the weak linkage, you can override this function with a custom function.

This code example generates three different faults:

- Cortex M0+ Hard Fault exception
- Cortex M4 Bus Fault exception
- Cortex M4 Usage Fault exception

Figure 6 shows the PSoC Creator schematic for this code example. It uses a UART Component to display messages related to each exception. It also connects to the SW2 button on the PSoC 6 BLE Pioneer Kit.

Figure 6. Hard Fault Handling Basic Design



The code example overrides `Cy_SysLib_ProcessingFault()` to print out the exception frame via a UART Component.

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. List of PSoC Creator Components

Component	Instance name	Purpose	Parameter
UART (SCB)	UART	To display information for debugging	**[General Tab]:** TX/RX Mode: TX
Digital Input Pin	SW2	Provide human interaction	**[General Tab]:** Uncheck HW connection Drive mode: Resistive Pull Up

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

Table 2 shows the pin assignment for the project done through the **Pins** tab in the **Design Wide Resources** window. These assignments are compatible with CY8CKIT-062-BLE.

Table 2. Pin names and Locations

Pin Name	Location
SW2	P0[4]
UART:tx	P5[1]

Reusing This Example

This example is designed for the CY8CKIT-062-BLE Pioneer Kit. To port the design to a different PSoC 6 MCU device and/or kit, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed.

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with BLE Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project
AN215656 – PSoC 6 MCU Dual-Core CPU system Design	Describes the dual-core CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-core design
AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project	Describes how to import the code generated by PSoC Creator into your preferred IDE
PSoC Creator Component Datasheets	
UART	Provides asynchronous communication interface using SCB hardware
Pins	Supports connection of hardware resources to physical pins
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual PSoC 6 MCU: PSoC 63 with BLE Registers Technical Reference Manual
Development Kit (DVK) Documentation	
CY8CKIT-062-BLE Pioneer Kit	

Document History

Document Title: CE218541 - PSoC 6 MCU Fault-Handling Basics

Document Number: 002-18541

Revision	ECN	Orig. of Change	Submission Date	Description of Change
*A	5993916	AJYA	12/20/2017	Initial Public Release
*B	6079299	AJYA	03/07/2018	Updated to PSoC Creator 4.2

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 | 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, 2017-2018. 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.