CE219656 - PSoC 6 MCU UART Using Low Level APIs

Objective

This example demonstrates the UART transmit and receive operation in PSoC® 6 MCU using low-level APIs, using ModusToolbox™ IDE.

Requirements

Tool: ModusToolbox™ IDE 1.0

Programming Language: C

Associated Parts: All PSoC 6 MCU parts

Related Hardware: PSoC 6 BLE Pioneer Kit, PSoC 6 WiFi-BT Pioneer Kit, PSoC6 WiFi-Prototyping Kit

Overview

This example contains three applications that implement a UART transmit and receive operation. Each application uses low-level UART APIs and echoes what is received on the UART serial terminal. The *UART Low Level Polling* example uses the polling method to transfer the bytes. The *UART Low Level User ISR* example uses an interrupt. The *UART Low Level DMA* example uses Direct Memory Access (DMA) functions.

Hardware Setup

This example uses the PSoC 6 WiFi-BT Pioneer Kit's default configuration. Refer to the kit guide to ensure the kit is configured correctly. You can also use PSoC 6 BLE Pioneer Kit or PSoC 6 WiFi-BT Stamp Board Kit by modifying the application to use the corresponding device on the board.

Note: The PSoC 6 BLE Pioneer kit and the PSoC 6 WiFi-BT Pioneer kit ship with KitProg2. ModusToolbox only works 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 If you do not upgrade, you will see an error like "unable to find CMSIS-DAP device" or "KitProg firmware is out of date".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 program. Install one on your PC if you don't have one. The instructions use Tera Term.

Operation

1. Connect the Pioneer board to your PC using the provided USB cable through the USB connector.
1. Open your terminal program and select the KitProg COM port. Set the other serial port parameters as follows:
    a) Baud Rate: 115200bps
    b) Data: 8 bits
    c) Parity: None
    d) Stop: 1 bit
    e) Flow Control: None
2. Import the application into a new workspace. If you are unsure how to import an application, see KBA225201.
3. Build the application. Choose **Project** > **Build All**.
4. Program the PSoC 6 MCU device. Select the **mainapp** project. In the **Quick Panel**, scroll down and click the **Program Kitprog3** item.

5. Observe the UART example header message printed in the terminal window.
6. Type any character in the terminal window, and make sure that the same character is displayed in the terminal.

Figure 1 shows a snapshot of a sample UART terminal output.

Figure 1. UART Terminal Output

Debugging

You can debug the example to step through the code. Use the Debug (KitProg3) configuration. See KBA224621 to learn how to start a debug session with ModusToolbox IDE.

Design and Implementation

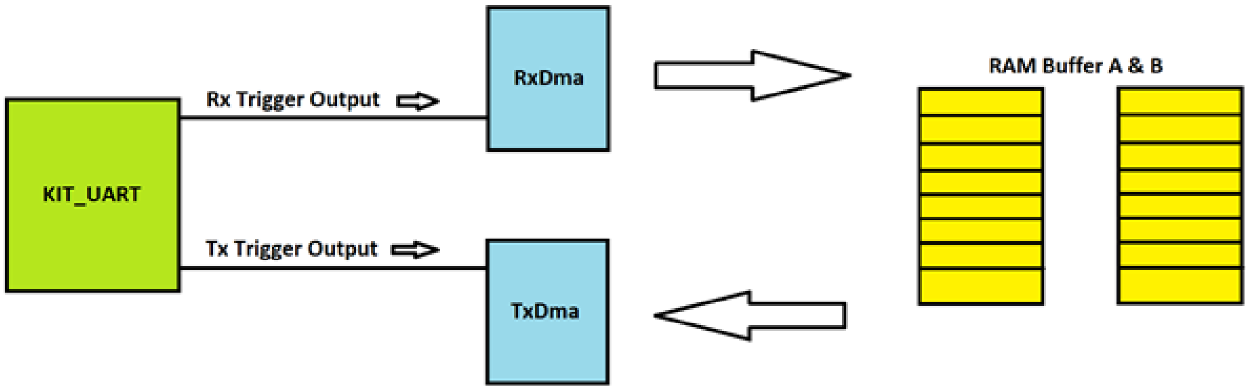
The current project encloses three applications. Each explaining different method to manage the UART resource. In all the applications, user transmits some data through UART terminal and the same date is echoed back on the terminal.

1) **UART Low Level Polling:** This application uses polling method to check if some data is present in the UART RX FIFO. When the user transmits a data through UART terminal, the RX FIFO starts to fill. The data in the RX FIFO is polled and echoed back on the terminal.

2) **UART Low Level User ISR:** This application uses interrupt method to check if some data is present in the UART RX FIFO . When the user transmits a data through UART terminal, “RX FIFO NOT EMPTY” interrupt is raised. The callback ISR handles this interrupt and echoes back the received data.

3) **UART Low Level DMA:** This application uses DMA to handle the data received in the UART RX FIFO. Two DMA channels are used to handle data in transmit and receive direction. Two SRAM buffers are alternatively used on the receive side to hold the data received from the UART terminal. These buffers are called ping-pong buffers and are mainly used to provide time for pulling the data out of either buffer. RxDma resource handles data transfer in the receive direction. RxDma has two descriptors in the chain, these two descriptors are configured such that the source alternates between the ping-pong buffers in the receive direction. TxDma is used to handle data in transmit direction. TxDma has only one descriptor configured to transfer a single element alternatively from the ping-pong buffers, this data is finally echoed back.

Figure 2. The UART Low Level DMA example schematics



Resources

Table 1 lists the resources used in this example, and how they are used in the design.

Table 1: Resource Table

Resource	Alias	Purpose	Non – Default Settings
SCB5	KIT_UART	Used for UART Transmit	Figure 3
General Purpose Input / Output (GPIO)	KIT_LED2	Provide visual feedback	Figure 4
DMA Channel	RxDma, TxDma	Used to R/W data from UART buffer	Figure 5, Figure 6, Figure 7

Reusing This Example

This example is configured for the PSoC 6 WiFi-BT Prototyping Kit. To port the design to a different PSoC 6 MCU device, create a new project selecting the required kit. If you want to manually edit the same application, right-click the application project and choose **Change Device** (you will need to reassign the DMA channels).

See the Parameter Settings for information on resource configuration.

Table 2: Device-Specific Resource Allocation

Kit name	Device Used	RxDma	TxDma
CY8CKIT-062-WiFi-BT	CY8C6247BZI-D54	DMA DataWire0: Channel 0	DMA DataWire0: Channel 1
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	DMA DataWire0: Channel 0	DMA DataWire0: Channel 1
CY8CPROTO-062-4343W	CY8C624ABZI-D44	DMA DataWire0: Channel 27	DMA DataWire0: Channel 26

Parameter Settings

Non-default settings for each Resource are outlined in red in the following figures.

Figure 3. KIT_UART Configuration

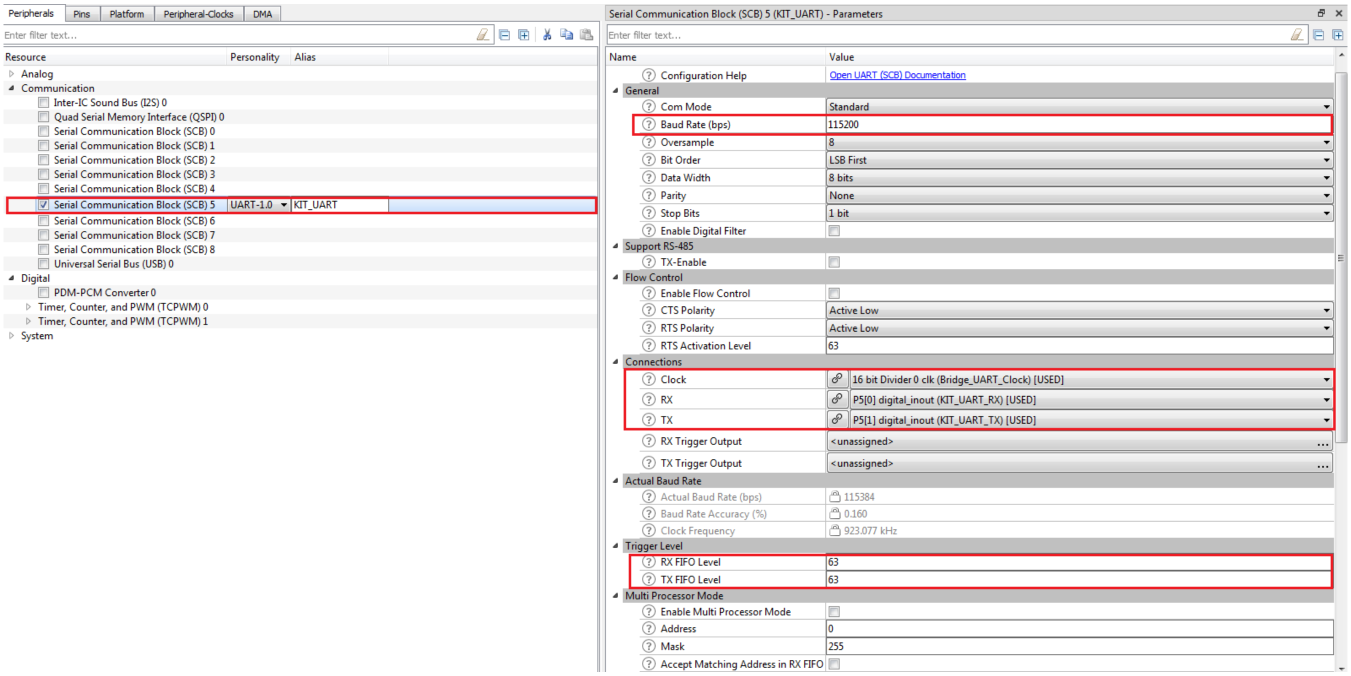


Figure 4. KIT_LED2 Configuration

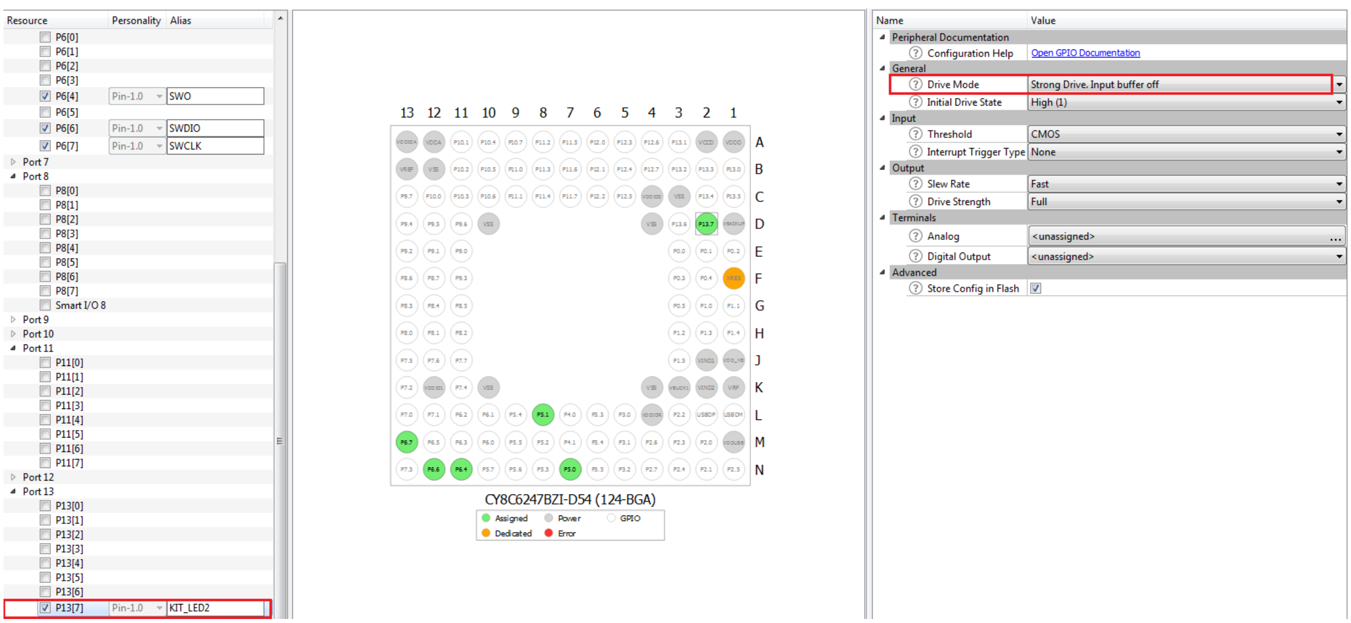


Figure 5. RxDma Descriptor 0 Configuration

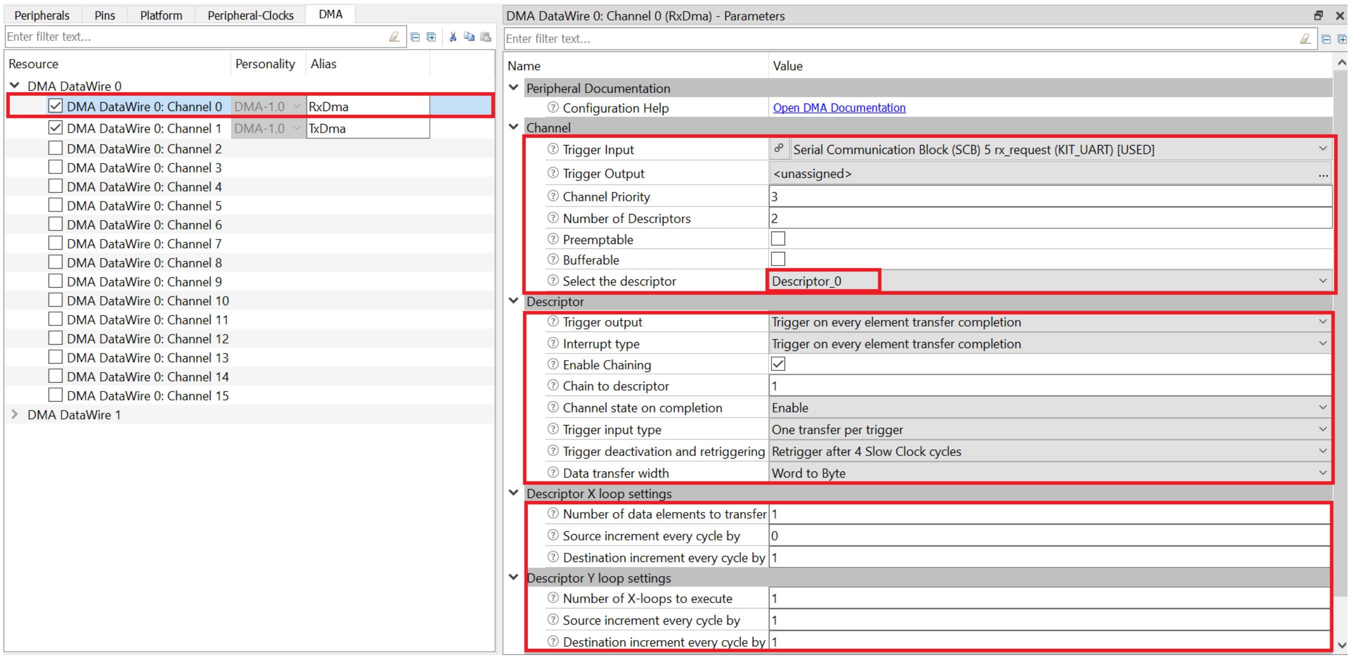


Figure 6. RxDma Descriptor 1 Configuration

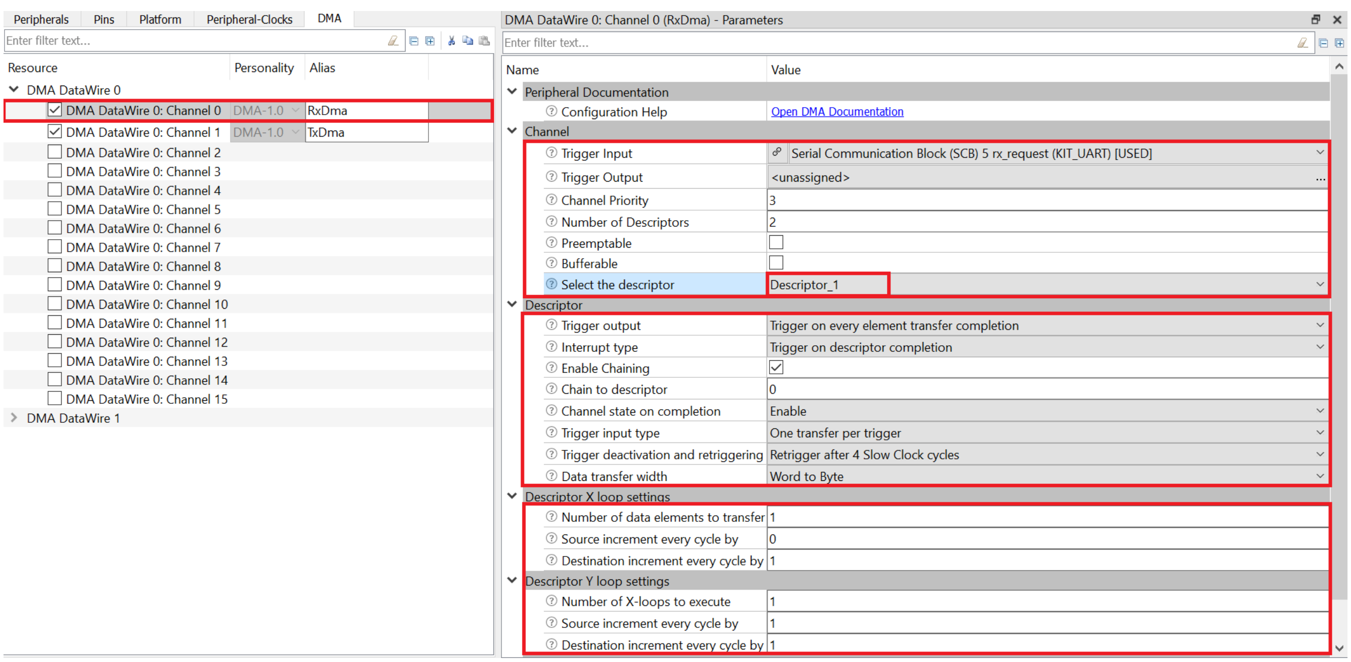
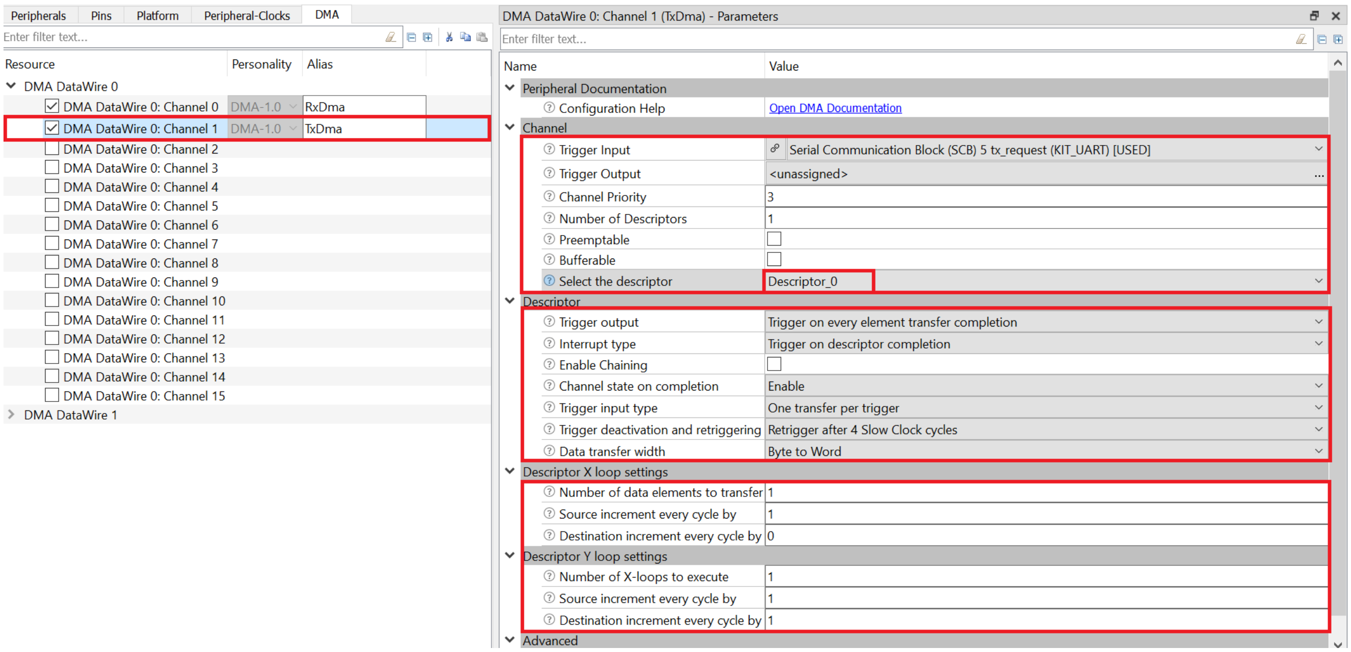


Figure 7. TxDma Descriptor 0 Configuration



Related Documents

For a comprehensive list of PSoC 6 MCU resources, see KBA223067 in the Cypress community.

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices.
AN221774 - Getting Started with PSoC 6 MCU	Describes PSoC 6 MCU devices and how to build your first ModusToolbox application and 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	
CE218472 - PSoC 6 MCU Comparing External Voltages Using a Low-Power Comparator	
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 Kit Documentation	
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	
Tool Documentation	
ModusToolbox	The Cypress IDE for IoT designers

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: CE219656 - PSoC 6 MCU UART Using Low Level APIs

Document Number: 002-25534

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6369721	YEKT	11/02/2018	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 | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709