

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 1.1](#)

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 importing the application for that kit, refer to [reusing the example](#) section.

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 If you do not upgrade, you will see an error like "unable to find CMSIS-DAP device" or "KitProg firmware is out of date".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

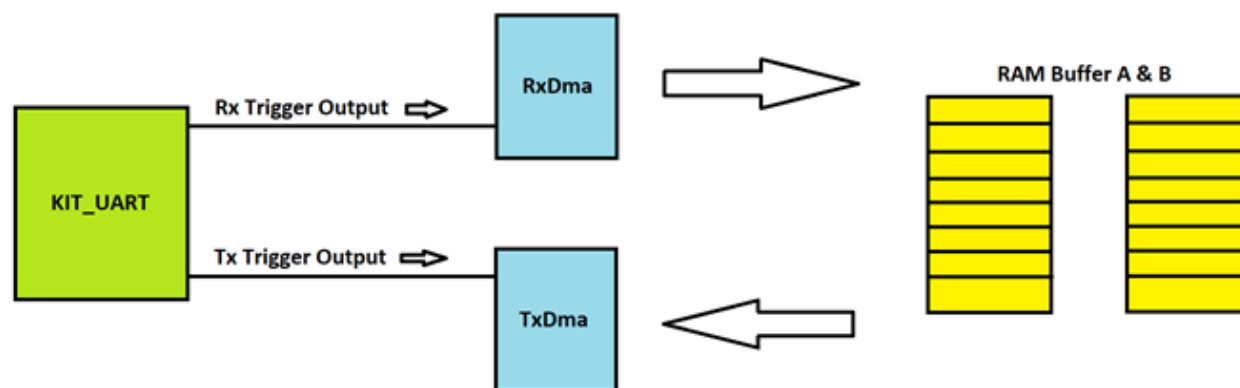
You can debug the example to step through the code. Use the Debug (KitProg3) configuration. See [KBA224621](#) to learn how to start a debug session with ModusToolbox IDE.

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 (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

In some cases, a resource used by a code example (for example, a peripheral) is not supported on another device. In that case, the example will not work. See the device datasheet for information on what resources the device support.

Parameter Settings

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

Figure 3. KIT_UART Configuration

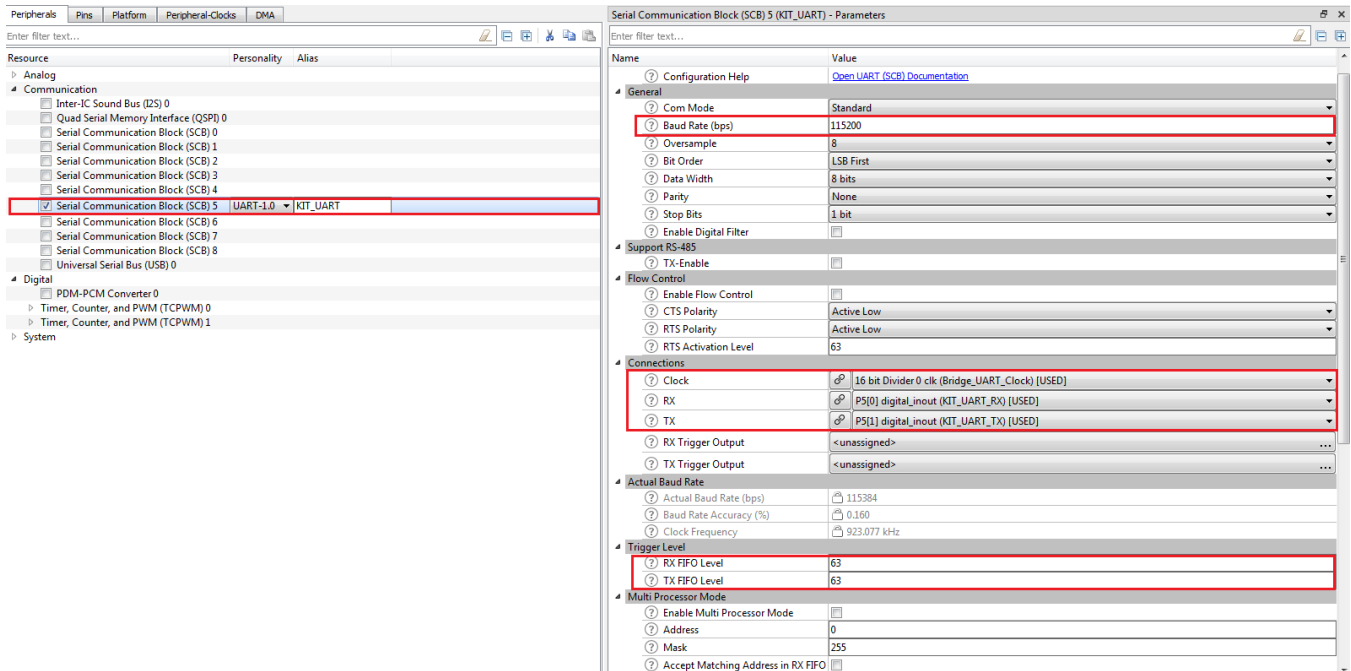


Figure 4. KIT_LED2 Configuration

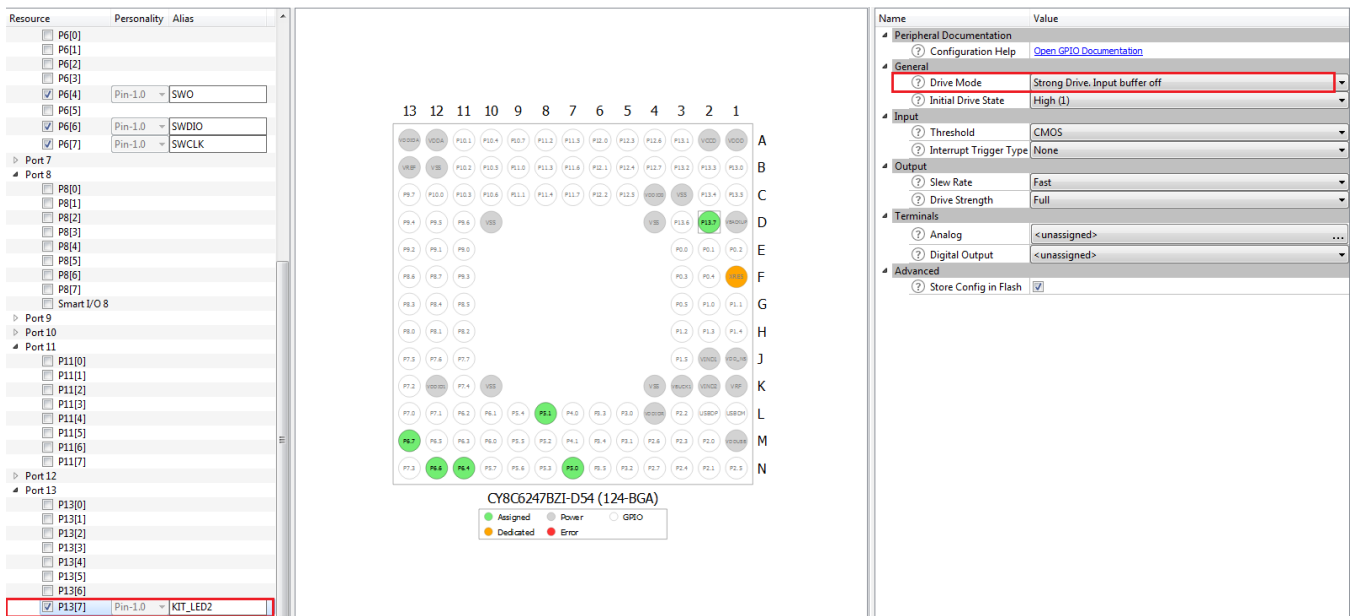
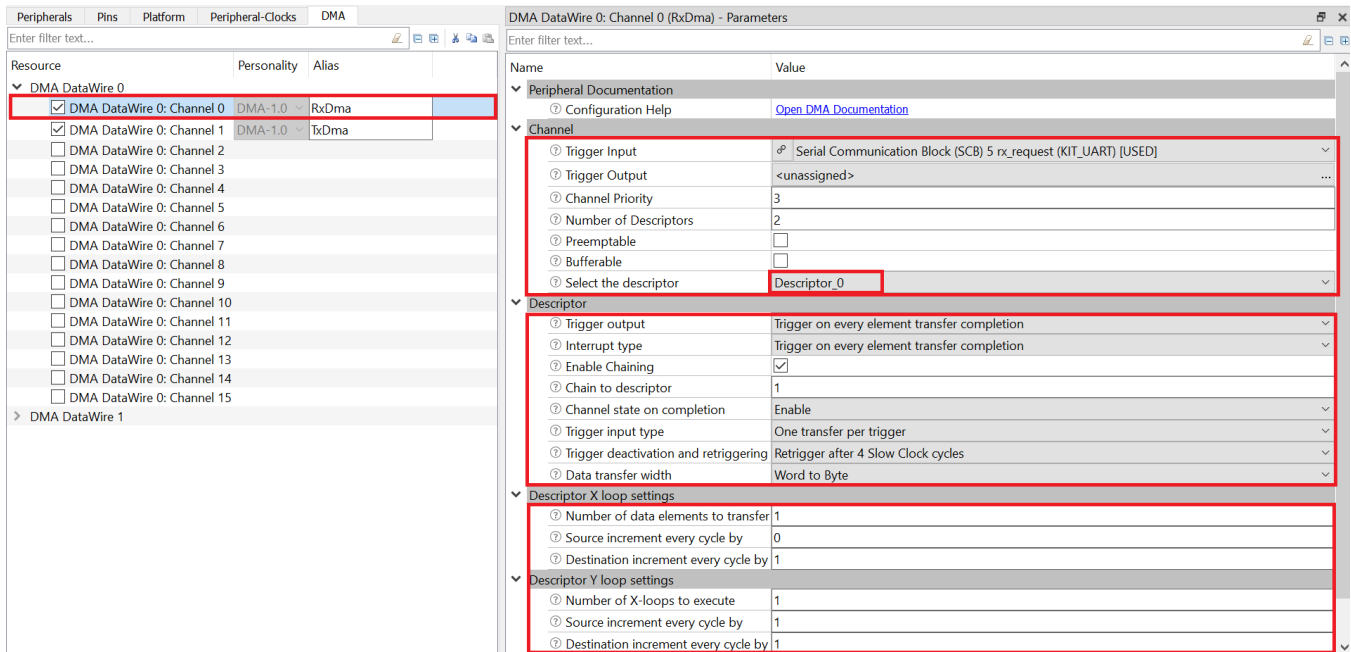


Figure 5. RxDma Descriptor 0 Configuration



Peripherals Pins Platform Peripheral-Clocks DMA

Enter filter text...

Resource Personality Alias

▼ DMA DataWire 0

☒ DMA DataWire 0: Channel 0 DMA-1.0 RxDma

☐ DMA DataWire 0: Channel 1 DMA-1.0 TxDma

☐ DMA DataWire 0: Channel 2

☐ DMA DataWire 0: Channel 3

☐ DMA DataWire 0: Channel 4

☐ DMA DataWire 0: Channel 5

☐ DMA DataWire 0: Channel 6

☐ DMA DataWire 0: Channel 7

☐ DMA DataWire 0: Channel 8

☐ DMA DataWire 0: Channel 9

☐ DMA DataWire 0: Channel 10

☐ DMA DataWire 0: Channel 11

☐ DMA DataWire 0: Channel 12

☐ DMA DataWire 0: Channel 13

☐ DMA DataWire 0: Channel 14

☐ DMA DataWire 0: Channel 15

► DMA DataWire 1

DMA DataWire 0: Channel 0 (RxDma) - Parameters

Enter filter text...

Name Value

▼ Peripheral Documentation

⑦ Configuration Help [Open DMA Documentation](#)

▼ Channel

⑦ Trigger Input ☒ Serial Communication Block (SCB) 5 rx_request (KIT_UART) [USED]

⑦ Trigger Output <unassigned>

⑦ Channel Priority 3

⑦ Number of Descriptors 2

⑦ Preemptable ☐

⑦ Bufferable ☐

⑦ Select the descriptor **Descriptor_0**

▼ Descriptor

⑦ Trigger output Trigger on every element transfer completion

⑦ Interrupt type Trigger on every element transfer completion

⑦ Enable Chaining ☒

⑦ Chain to descriptor 1

⑦ Channel state on completion Enable

⑦ Trigger input type One transfer per trigger

⑦ Trigger deactivation and retriggering Retrigger after 4 Slow Clock cycles

⑦ Data transfer width Word to Byte

▼ Descriptor X loop settings

⑦ Number of data elements to transfer 1

⑦ Source increment every cycle by 0

⑦ Destination increment every cycle by 1

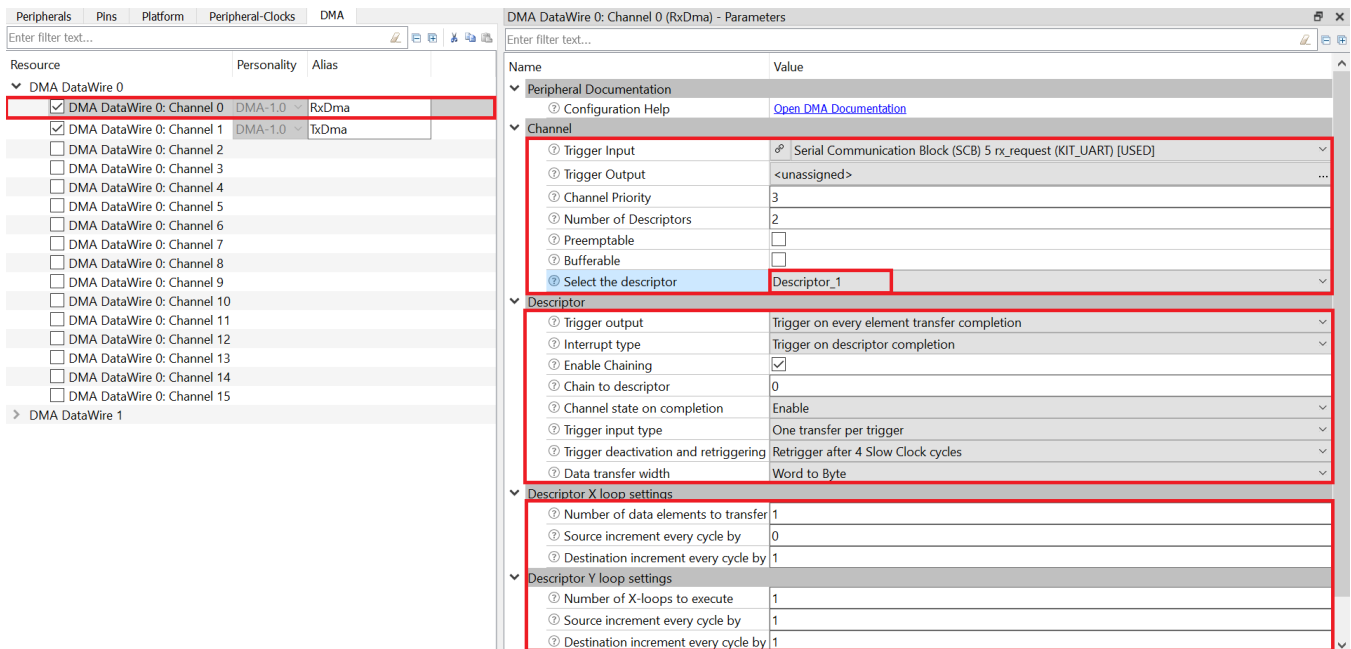
▼ Descriptor Y loop settings

⑦ Number of X-loops to execute 1

⑦ Source increment every cycle by 1

⑦ Destination increment every cycle by 1

Figure 6. RxDma Descriptor 1 Configuration



Peripherals Pins Platform Peripheral-Clocks DMA

Enter filter text...

Resource Personality Alias

▼ DMA DataWire 0

☒ DMA DataWire 0: Channel 0 DMA-1.0 RxDma

☐ DMA DataWire 0: Channel 1 DMA-1.0 TxDma

☐ DMA DataWire 0: Channel 2

☐ DMA DataWire 0: Channel 3

☐ DMA DataWire 0: Channel 4

☐ DMA DataWire 0: Channel 5

☐ DMA DataWire 0: Channel 6

☐ DMA DataWire 0: Channel 7

☐ DMA DataWire 0: Channel 8

☐ DMA DataWire 0: Channel 9

☐ DMA DataWire 0: Channel 10

☐ DMA DataWire 0: Channel 11

☐ DMA DataWire 0: Channel 12

☐ DMA DataWire 0: Channel 13

☐ DMA DataWire 0: Channel 14

☐ DMA DataWire 0: Channel 15

► DMA DataWire 1

DMA DataWire 0: Channel 0 (RxDma) - Parameters

Enter filter text...

Name Value

▼ Peripheral Documentation

⑦ Configuration Help [Open DMA Documentation](#)

▼ Channel

⑦ Trigger Input ☒ Serial Communication Block (SCB) 5 rx_request (KIT_UART) [USED]

⑦ Trigger Output <unassigned>

⑦ Channel Priority 3

⑦ Number of Descriptors 2

⑦ Preemptable ☐

⑦ Bufferable ☐

⑦ Select the descriptor **Descriptor_1**

▼ Descriptor

⑦ Trigger output Trigger on every element transfer completion

⑦ Interrupt type Trigger on descriptor completion

⑦ Enable Chaining ☒

⑦ Chain to descriptor 0

⑦ Channel state on completion Enable

⑦ Trigger input type One transfer per trigger

⑦ Trigger deactivation and retriggering Retrigger after 4 Slow Clock cycles

⑦ Data transfer width Word to Byte

▼ Descriptor X loop settings

⑦ Number of data elements to transfer 1

⑦ Source increment every cycle by 0

⑦ Destination increment every cycle by 1

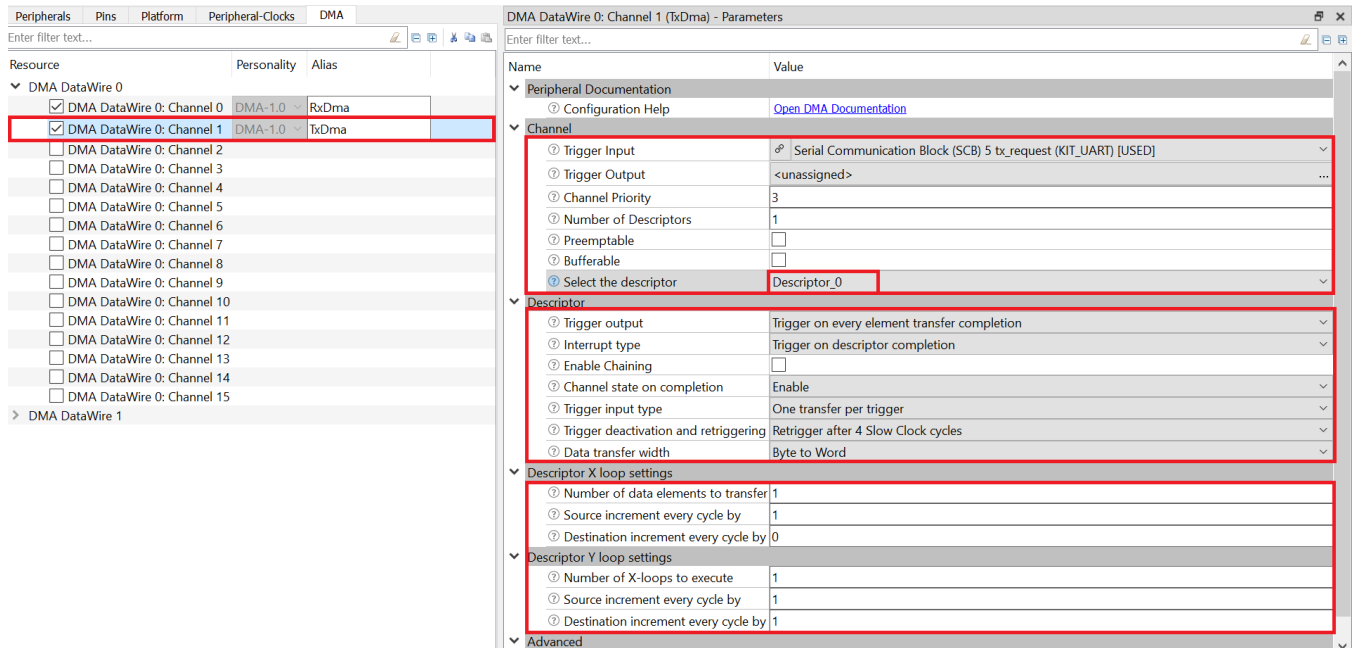
▼ Descriptor Y loop settings

⑦ Number of X-loops to execute 1

⑦ Source increment every cycle by 1

⑦ Destination increment every cycle by 1

Figure 7. TxDma Descriptor 0 Configuration



The screenshot displays the configuration interface for the DMA DataWire 0: Channel 1 (TxDma). The left pane shows the resource tree with 'DMA DataWire 0: Channel 1' selected. The right pane shows the configuration parameters for this channel.

Name	Value
Peripheral Documentation	
Configuration Help	Open DMA Documentation
Channel	
Trigger Input	Serial Communication Block (SCB) 5 tx_request (KIT_UART) [USED]
Trigger Output	<unassigned>
Channel Priority	3
Number of Descriptors	1
Preemptable	<input type="checkbox"/>
Bufferable	<input type="checkbox"/>
Select the descriptor	Descriptor_0
Descriptor	
Trigger output	Trigger on every element transfer completion
Interrupt type	Trigger on descriptor completion
Enable Chaining	<input type="checkbox"/>
Channel state on completion	Enable
Trigger input type	One transfer per trigger
Trigger deactivation and retriggering	Retrigger after 4 Slow Clock cycles
Data transfer width	Byte to Word
Descriptor X loop settings	
Number of data elements to transfer	1
Source increment every cycle by	1
Destination increment every cycle by	0
Descriptor Y loop settings	
Number of X-loops to execute	1
Source increment every cycle by	1
Destination increment every cycle by	1
Advanced	

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	11/2/2018	New code example
*A	6484272	YEKT	2/20/2019	Code example updated for 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](#) | [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, 2018-2019. 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. Other names and brands may be claimed as property of their respective owners.