CE218552 – PSoC 6 MCU: UART to Memory Buffer Using DMA

Objective

This PSoC 6 code example demonstrates a UART-to-memory buffer data transfer using DMA, with no CPU usage. DMA channels are used to implement data transfer both on received data and transmitted data of the UART.

Overview

This example demonstrates how a PSoC® 6 DMA channel transfers data received from the UART to a buffer in memory. When the buffer is filled, a second DMA channel drains the buffer to the UART, to be transmitted.

Requirements

Tool: PSoC Creator™ 4.2

Programming Language: C (ARM® GCC)

Associated Parts: All PSoC 6 parts

Related Hardware: CY8CKIT-062 PSoC 6 BLE Pioneer Kit

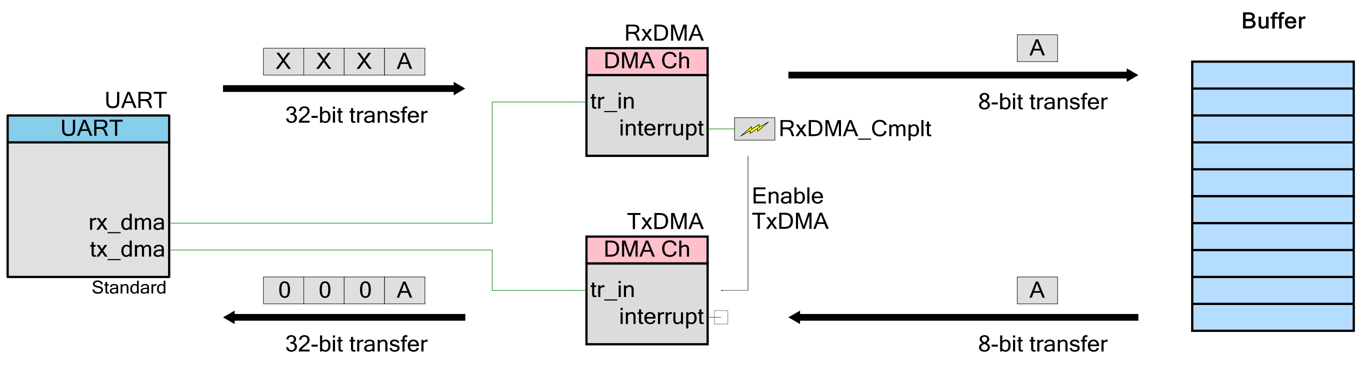
Design

The design implements a DMA interface between a UART and a buffer in RAM. There are two DMA channels, as Figure 1 shows. RxDMA handles the transfer of data from the UART's Rx FIFO to the buffer. TxDMA handles the transfer of data back from the buffer to the UART TxFIFO. The buffer size is 10 bytes.

The UART is configured to trigger the RxDMA channel when there is at least one byte of data present in the UART's RxFIFO. The RxDMA moves one byte of data from the UART to the buffer each time it is triggered. RxDMA's transfer descriptor is configured to transfer 10 bytes, which is the size of the buffer. After transferring 10 bytes, RxDMA generates an interrupt (RxDMA_Cmplt), which enables the TxDMA channel.

The TxDMA channel's transfer descriptor is configured to transfer 10 bytes from the buffer to the UART's TxFIFO. Each byte transfer of the TxDMA is triggered by the UART's TxFIFO status. The UART triggers TxDMA when there is at least one byte of space available in the TxFIFO. This mechanism ensures that TxDMA transfer to the UART does not result in overflow of TxFIFO..

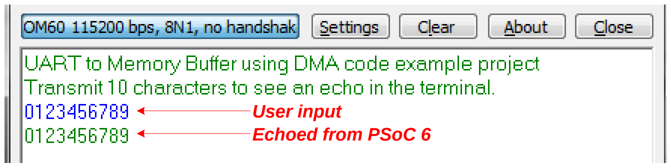
Figure 1. PSoC Creator Project Schematic



Operation

1. Program the project on to CY8CKIT-062. Rx and Tx lines of PSoC 6 are already connected to Tx and RX lines of the PSoC 5LP on the board. PSoC 5LP implements a UART-to-USB bridge, which enumerates as a COM port on the PC.
2. Set up a terminal program to transmit/receive the UART data. You also need to find and connect to the COM port that is enumerated. The UART settings for the terminal are as follows
   - Baud Rate: 115,200 bps
   - Data bits: 8 bits
   - Stop bits: 1 bit
   - Parity: None
   - Flow control: None
3. Once the terminal program is started, a welcome text is printed as shown in Figure 2.

Figure 2. Terminal Output



```
UART to Memory Buffer using DMA code example project
Transmit 10 characters to see an echo in the terminal.
0123456789
0123456789
```

4. Now, you can enter bytes through the terminal input, which will be received by the PSoC 6 UART and stored in the buffer. When the number of characters sent is equal to 10, PSoC 6 echoes back all the 10 characters.

Components

Table 1 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each.

Table 1. List of Components

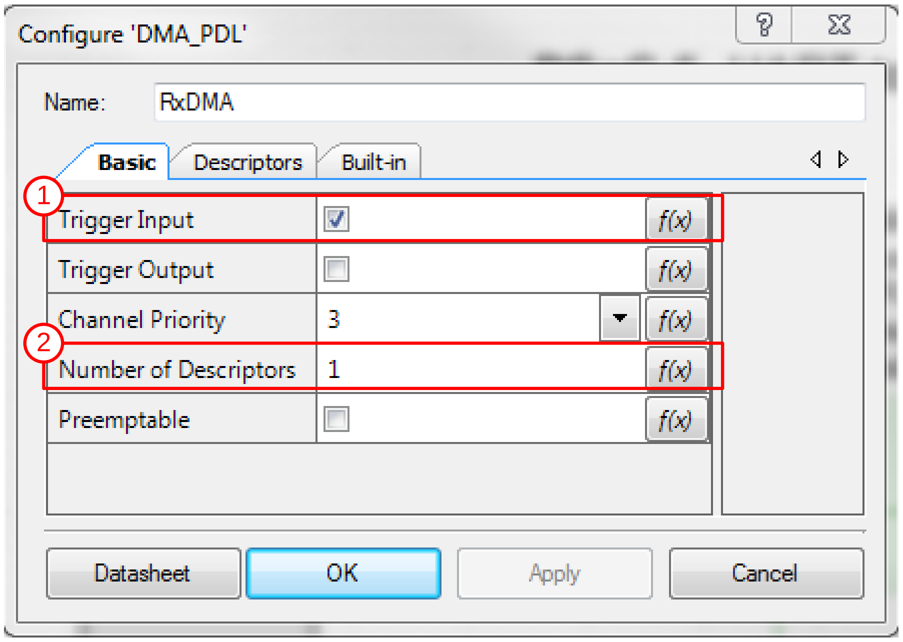
Component	Hardware Resources
RxDMA	1 DMA channel
TxDMA	1 DMA channel
UART	SCB

Parameter Settings

RxDMA Configuration

RxDMA transfers data from the UART to buffer. The basic configuration for the RxDMA Component is shown in Figure 3.

Figure 3. RxDMA Basic Configuration


Configure 'DMA_PDL'

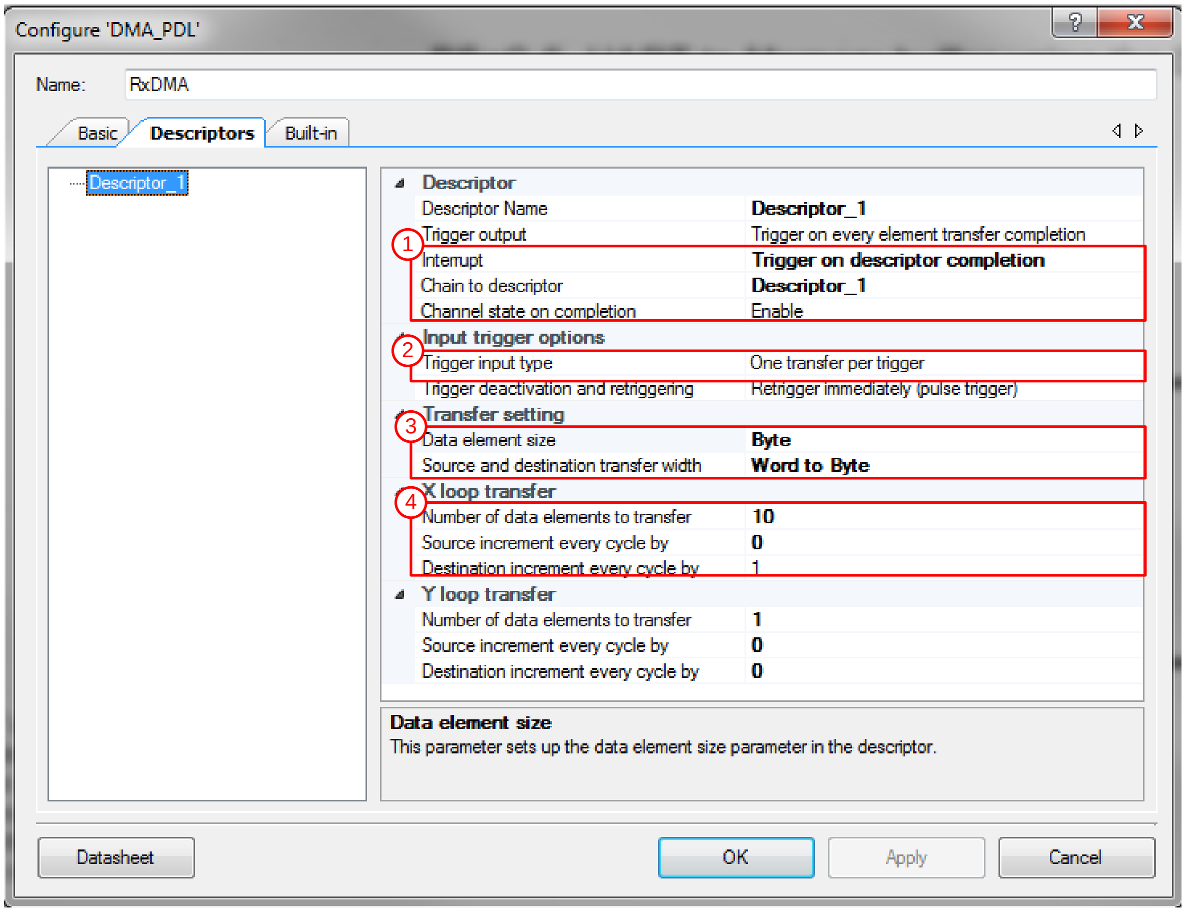
Name: RxDMA

Trigger Input	✓
Trigger Output	
Channel Priority	3
Number of Descriptors	1
Preemptable	

1. The **trigger input** to the DMA is enabled so that a trigger signal triggers the DMA channel. For the RxDMA channel, the trigger signal is routed from the UART's rx_dma output.
2. The DMA channel is configured to a simple UART register-to-buffer transfer; only one descriptor is needed. The **number of descriptors** is set to 1.

   The RxDMA descriptor is configured in the Descriptors tab, as Figure 4 shows.

Figure 4. RxDMA Descriptor Configuration



Name: RxDMA

Descriptor	
Descriptor Name	**Descriptor_1**
Trigger output	Trigger on every element transfer completion
Interrupt	**Trigger on descriptor completion**
Chain to descriptor	**Descriptor_1**
Channel state on completion	Enable
Input trigger options	
Trigger input type	One transfer per trigger
Trigger deactivation and retriggering	Retrigger immediately (pulse trigger)
Transfer setting	
Data element size	**Byte**
Source and destination transfer width	**Word to Byte**
X loop transfer	
Number of data elements to transfer	**10**
Source increment every cycle by	**0**
Destination increment every cycle by	1
Y loop transfer	
Number of data elements to transfer	**1**
Source increment every cycle by	**0**
Destination increment every cycle by	**0**

Data element size
This parameter sets up the data element size parameter in the descriptor.

The descriptor configuration determines the characteristics of the DMA transfer.

1. RxDMA implements an interrupt on completion of the descriptor. Therefore, the **Interrupt** is set as "Trigger on descriptor completion". Once the descriptor is complete, the next receive event on the UART should continue the DMA transfers to the buffer. This is achieved by setting the **chain to descriptor** as "Descriptor_1" which makes the DMA execute the same descriptor in a loop. In addition, the **Channel state on completion** is set to "Enable" so that the descriptor is running in a loop with no interruption.

2. The input trigger options set how the descriptor responds to the trigger inputs. In this code example, the **trigger input type** is set to one transfer per trigger because a single byte transfer from the UART to the buffer is needed on every trigger generated by the UART.

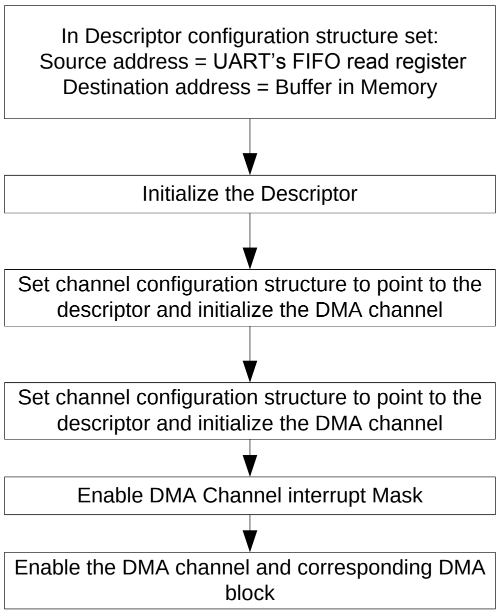
3. The data is transferred from a UART hardware register to a memory location. The data being transferred is a byte wide. Therefore, the **data size** is set to "Byte". Access to the UART register is 32-bit, while the memory access can be 8-, 16- or 32-bit. This is why **source and destination transfer widths** is set to "Word to byte".

4. The X loop transfer setting sets up the x loop for the descriptor. The DMA can set up two nested loops of transfer; x loop is the inner loop of transfer. Refer to the DMA Component datasheet for details. This descriptor transfers 10 bytes from the UART to buffer. Therefore, the **number of data elements to transfer** is set as 10. Because the data source is a UART register, **source increment every cycle by 0**. Because the destination is a buffer array, **destination increment every cycle by 1**. Thus, the data from the UART register is moved to sequential locations in the buffer.

5. The Y loop is not used in this code example. Therefore, the number of data elements to transfer is set as 1. Both source and destination increments are set to 0.

6. In the code section, the DMA requires some initialization code. Figure 5 shows the flow chart for the DMA initialization code for RxDMA channel.

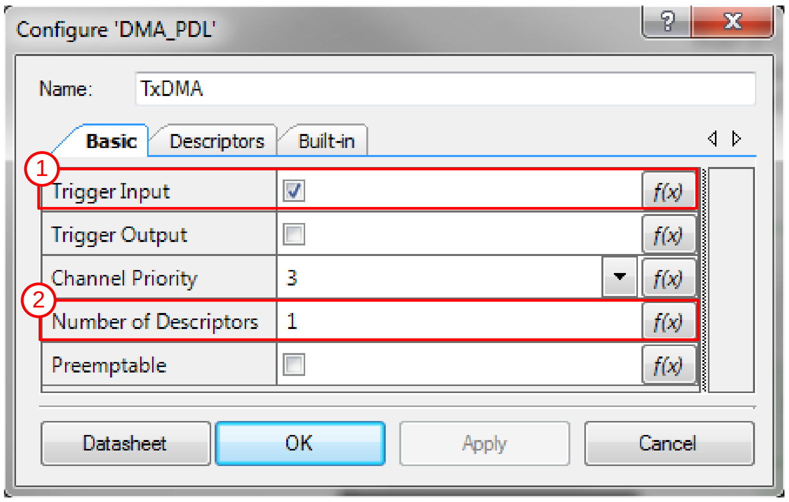
Figure 5. Flowchart for DMA Initialization Code


In Descriptor configuration structure set:
Source address = UART's FIFO read register
Destination address = Buffer in Memory
Initialize the Descriptor
Set channel configuration structure to point to the descriptor and initialize the DMA channel
Set channel configuration structure to point to the descriptor and initialize the DMA channel
Enable DMA Channel interrupt Mask
Enable the DMA channel and corresponding DMA block

TxDMA Configuration

TxDMA transfers data from the buffer to UART. The basic configuration for the TxDMA Component is shown in Figure 6.

Figure 6. TxDMA Basic Configuration



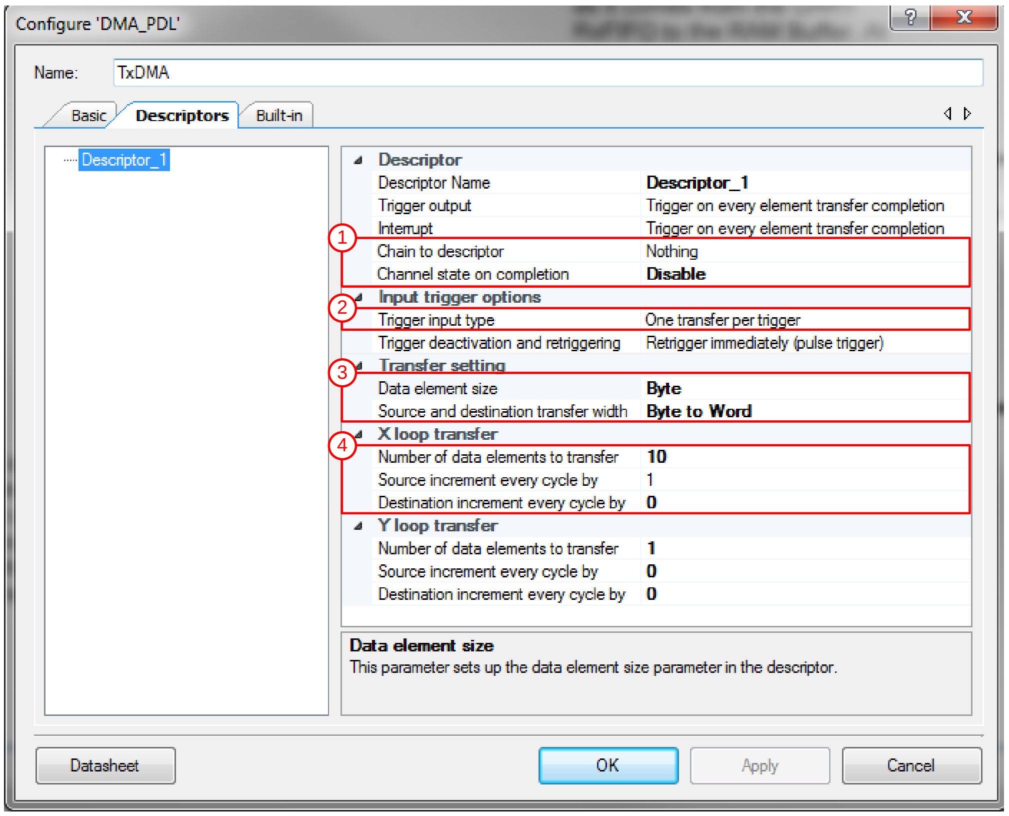
Name: TxDMA

Trigger Input	✓
Trigger Output	
Channel Priority	3
Number of Descriptors	1
Preemptable	

1. The **trigger input** to the DMA is enabled so that a trigger signal could be used to trigger the DMA channel. In case of the TxDMA, the trigger signal is routed from the UART's tx_dma trigger line.
2. The DMA is implementing a simple buffer to UART register transfer, which can be achieved by using a single descriptor. The **number of descriptors** is set to 1.

The Descriptor of the TxDMA is configured in the Descriptor configuration tab shown in Figure 7.

Figure 7. TxDMA Descriptor Configuration



Name: TxDMA

Descriptor	
Descriptor Name	**Descriptor_1**
Trigger output	Trigger on every element transfer completion
Interrupt	Trigger on every element transfer completion
Chain to descriptor	Nothing
Channel state on completion	**Disable**
Input trigger options	
Trigger input type	One transfer per trigger
Trigger deactivation and retriggering	Retrigger immediately (pulse trigger)
Transfer setting	
Data element size	**Byte**
Source and destination transfer width	**Byte to Word**
X loop transfer	
Number of data elements to transfer	**10**
Source increment every cycle by	1
Destination increment every cycle by	**0**
Y loop transfer	
Number of data elements to transfer	**1**
Source increment every cycle by	**0**
Destination increment every cycle by	**0**

Data element size
This parameter sets up the data element size parameter in the descriptor.

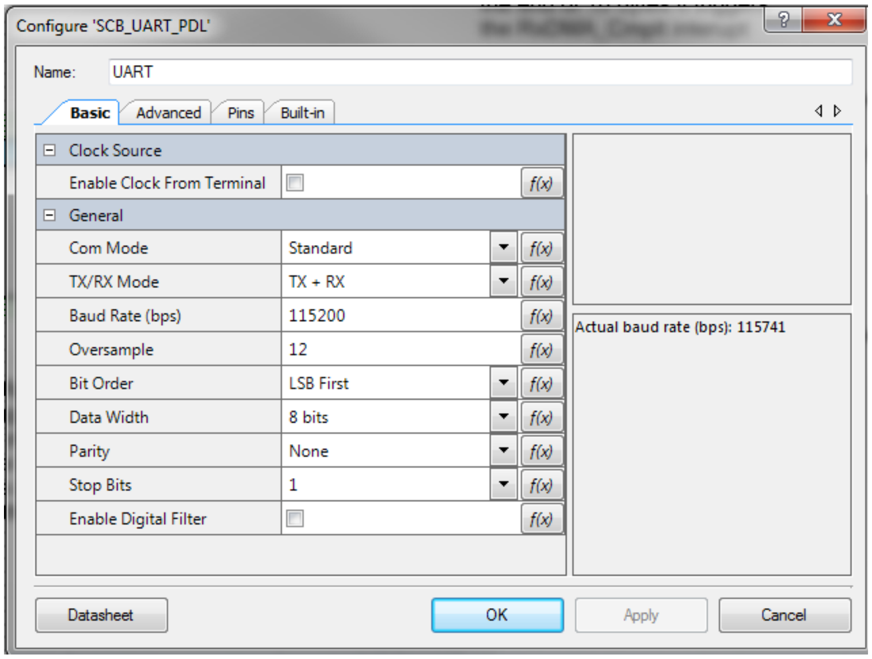
The descriptor configuration determines the characteristics of the DMA transfer.

1. The descriptor settings set up the name for the descriptor as "Descriptor_1". On completion of the descriptor, TxDMA should go to a disabled state with no active descriptor. This is achieved by setting **chain to descriptor** as “None”. Also, the **Channel state on completion** is set to “Disable” so that the descriptor is disabled at the end of the transfer. The TxDMA will be re-enabled in the RxDMA_Cmplt interrupt.
2. The input trigger options set how the descriptor has to behave to the trigger inputs. In this code example, the **trigger input type** is set to one transfer per trigger because a single byte transfer is needed from the Buffer to UART on every trigger generated by the UART.
3. The data is transferred from a memory location to the UART hardware register. The data being transferred is a byte wide. Therefore, the **data size** is set to “Byte”. Access to hardware registers like the UART register is 32-bit while the memory access can be 8-, 16-, or 32-bit. This is why **source and destination transfer widths** is set to “Byte to Word”.
4. The X loop transfer setting sets up the x loop for the descriptor. The DMA can set up two nested loops of transfer; x loop is the inner loop of transfer. Refer to the DMA Component datasheet for details. This descriptor transfers 10 bytes from the buffer to UART. Therefore, the **number of data elements to transfer** is set as 10. Because the data source is the buffer array, **source increment every cycle by 1**. Because the destination is a UART register, **destination increment every cycle by 0**. Thus, the data from the buffer array is moved sequentially to the UART.
5. The Y loop is not used in this code example. Therefore, the number of data elements to transfer is set as 1. Both source and destination increments are set to 0.
6. Similar to RxDMA, there is some amount of code needed to initialize TxDMA. The initialization code for TxDMA is very similar to the one for RxDMA, shown in Figure 5. The initialization code first configures the descriptor to source as the buffer and the destination as the UART’s TX_FIFO_WR register. After this, the initialization code simply initializes the descriptor and the channel. For TxDMA, there is no Interrupt enabled because a DMA channel interrupt is not used from TxDMA.

UART Configuration

The basic UART configuration is shown in Figure 8. The basic UART configurations are all left as default.

Figure 8. UART Basic Configuration



Name: UART

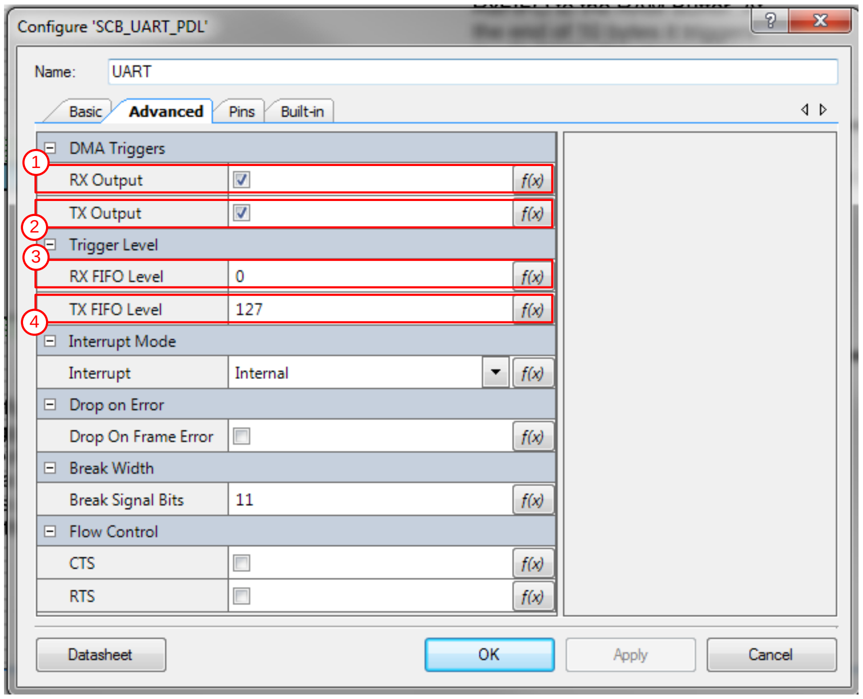
Clock Source	
Enable Clock From Terminal	☐
General	
Com Mode	Standard
TX/RX Mode	TX + RX
Baud Rate (bps)	115200
Oversample	12
Bit Order	LSB First
Data Width	8 bits
Parity	None
Stop Bits	1
Enable Digital Filter	☐

Actual baud rate (bps): 115741

The UART is set to have a standard Tx/Rx mode. The baud rate is set at 115,200 with oversample of 12. Data width is 8 bits with no parity and one stop bit.

Advanced configuration settings for the UART are shown in Figure 9.

Figure 9. UART Advanced Configuration


Configure 'SCB_UART_PDL'

Name: UART

Basic | **Advanced** | Pins | Built-in

DMA Triggers	
RX Output	✓
TX Output	✓
Trigger Level	
RX FIFO Level	0
TX FIFO Level	127
Interrupt Mode	
Interrupt	Internal
Drop on Error	
Drop On Frame Error	☐
Break Width	
Break Signal Bits	11
Flow Control	
CTS	☐
RTS	☐

1. **RX Output** is enabled. This will provide a DMA trigger line based on the RxFIFO level. This trigger is used to trigger RxDMA.
2. **TX Output** is enabled. This will provide a DMA trigger line based on the TxFIFO level. This trigger is used to trigger TxDMA.
3. **RX FIFO Level** is set to 0. This setting means that an RX DMA trigger is generated when the Rx FIFO level is above 0.
4. **TX FIFO Level** is set to 127. This setting means that a TX DMA trigger is generated when the Tx FIFO level is below 127.

Related Documents

Table 2 lists the relevant application notes, code examples, Component datasheets, and device and DVK documentation.

Table 2. Related Documents

Application Notes	
AN210781: Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	This application note introduces the PSoC 6 BLE device
Code Examples	
CE218553- PSoC 6 MCU- PWM triggering a DMA channel	This code example is a simple case of a PWM triggering a DMA channel.
PSoC Creator Component Datasheets	
UART	UART (SCB_UART_PDL)
DMA	DIRECT MEMORY ACCESS (DMA_PDL)
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual (TRM)	PSoC 6 MCU with BLE Architecture Technical Reference Manual
PSoC 6 MCU: PSoC 63 with BLE Register Technical Reference Manual	PSoC 6 MCU with BLE Register Technical Reference Manual
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 63 with BLE Datasheet
PSoC 6 MCU: PSoC 62 Datasheet	PSoC 62 Datasheet
Development Kit (DVK) Documentation	
CY8CKIT-062-BLE: PSoC 6 BLE Pioneer Kit	

Document History

Document Title: CE218552 - PSoC 6 MCU: UART to Memory Buffer Using DMA

Document Number: 002-18552

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	**5781252**	QVS	06/21/2017	New code example
*A	**5856035**	QVS	08/21/2017	Updated to latest version of PSoC Creator and updated the document template

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

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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