CE219940 – PSoC 6 MCU Multiple DMA Concatenation

Objective

This example demonstrates how to use DMA to combine data from multiple inputs and store it in a memory array on a PSoC® 6 MCU device.

Overview

This example demonstrates the use of multiple concatenated DMA channels to manipulate memory, with no CPU usage. The incoming data from the UART is packed into 5-byte packets and stored in a memory array, along with a timestamp from the RTC. When four packets of data are stored, they are echoed back to the UART using DMA.

Requirements

Tool: PSoC Creator™ 4.2

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

Associated Parts: All PSoC 6 parts

Related Hardware: CY8CKIT-062 BLE

Design

This code example uses three DMA channels, a UART, and an RTC to timestamp incoming data and store it in a memory array. Figure 1 shows the PSoC Creator schematic for this project.

The RTC is configured with a start date of March 30th 2017, at 12:00 pm, and will generate an interrupt every second to update the RTC buffer. As a result, the buffer always contains a timestamp string in the following format: "\r\nhh:mm:ss mm/dd/yy ".

The RxDma Component configures two descriptors to concatenate the data from the RTC and the UART. The first descriptor copies the RTC timestamp when the first byte in the UART arrives in the data buffer. Afterwards, the DMA channel switches to the second descriptor, which moves the incoming data from the UART into the same buffer. After five bytes of data, the descriptor chain resets and the MemoryDma descriptor is triggered.

To handle 2D arrays, the MemoryDma descriptor implements an X and Y loop. The X loop determines the number of bytes moved in each Y loop. The Y loop manages the destination address. An X loop is performed every time the MemoryDma is triggered, moving a complete data buffer (time stamp plus data) into the memory array. The Y loop is configured to be four X loops. When complete, the descriptor is reset and the TxDma descriptor is enabled.

The TxDma descriptor also implements an X and Y loop to read the memory array. It sends the entire memory array to the UART. The TxDma is triggered by the UART-TX-FIFO-is-empty signal until the Y loop has been completed. Afterwards, it is disabled and must be enabled again by the MemoryDma.

The UART is configured to trigger the RxDma process every time data is received, and to continuously trigger the TxDma when there is no data in the TX FIFO buffer. The interrupts are configured to show FIFO overflows, going into an infinite loop within the interrupt service routine.

Figure 1. Project Schematic

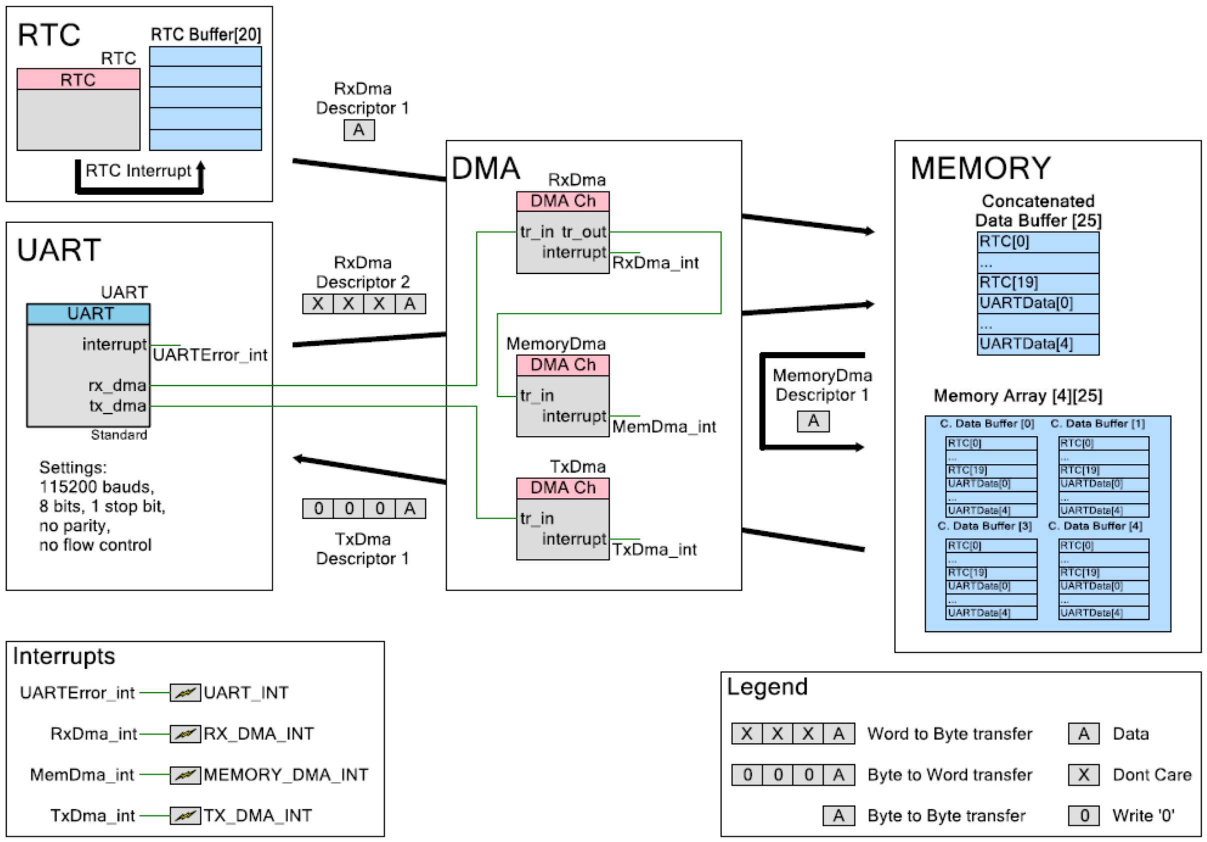
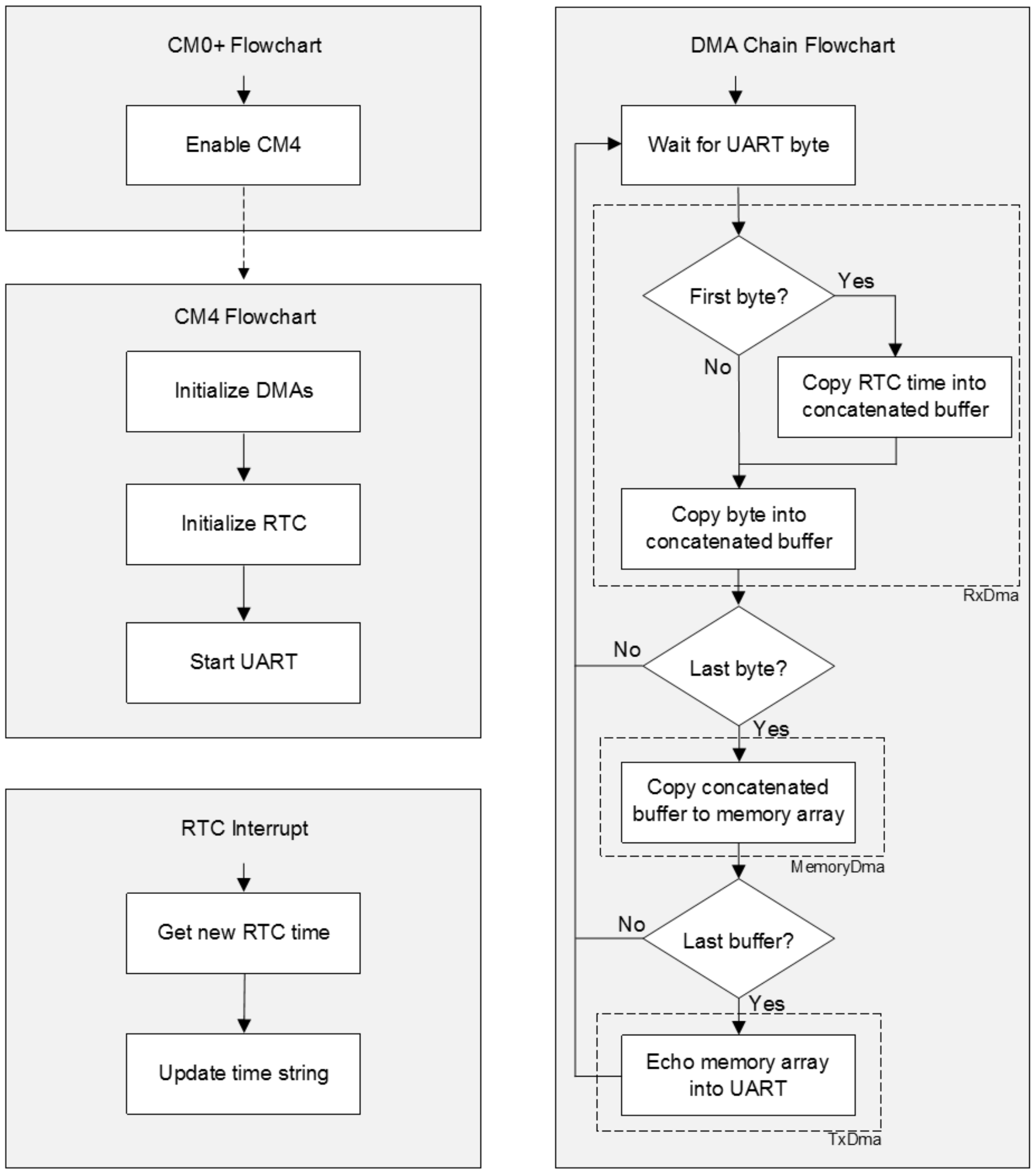


Figure 2 shows the firmware flow of the code example. After a reset, CM0+ must activate the CM4 core. The CM4 core then initializes the peripherals and handles the interrupts. The code example requires almost no CPU usage.

Figure 2. Firmware Flowchart



Design Considerations

This code example runs on CY8CKIT-062-BLE, which has a PSoC 63 MCU device. To port the design to other PSoC 6 MCU devices and kits, change the target device using PSoC Creator **Project** > **Device Selector**, and pin assignments in the **Design Wide Resources** window.

The software configuration of the DMA and RTC Components is in the respective Configure___ functions. This code can be easily copied, with a few modifications, into other designs to get RTC and DMA functionality.

Hardware Setup

No special hardware setup is required for CY8CKIT-062-BLE.

Leave the switches and jumpers in their default positions. Default positions of relevant switches are as shown in Table 1.

Table 1. Switch Selection

Switch / Jumper	Position
SW5	3.3 V
SW6	PSoC 6 BLE
SW7	VDDD / KitProg2

Software Setup

Macro values can be changed to modify the starting date of the RTC and the size of the buffers. These macros are in the *main_cm4.c* file.

The RTC buffer stores a string with the current time using the following format: “\r\nhh:mm:ss mm/dd/yy ”. To change the size of the RTC buffer, change the RTC_BUFFER_SIZE macro. The value should reflect the size of the new format. If changed, the **Number of data elements to transfer** value within the X loop settings of the RxDma Descriptor 1, MemoryDma, and TxDma must be updated accordingly.

To change the packet size of the incoming UART data, change the value of the PACKET_SIZE macro. The **Number of data elements to transfer** value within the X loop settings of the RxDma Descriptor 2, MemoryDma, and TxDma must be updated as RTC_BUFFER_SIZE + PACKET_SIZE.

When changing the number of memory buffers in the array, the value of the **Number of data elements to transfer** within the Y loop settings of the TxDma and MemoryDma components must be updated accordingly.

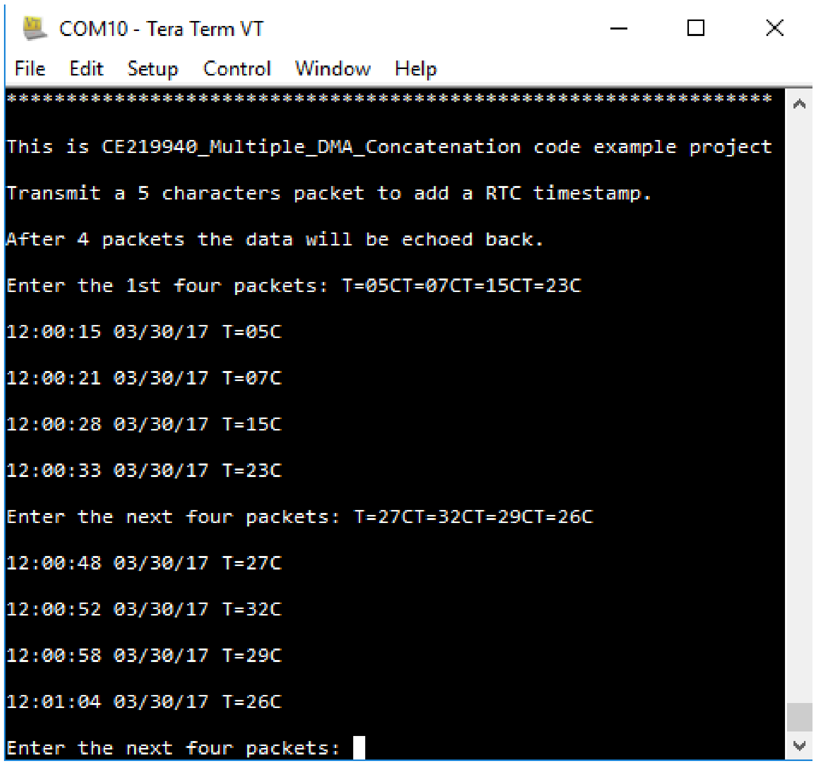
```
/* Defines for starting date */
#define START_SEC                (0u)    /* Value must be in range 0-59 */
#define START_MIN                (0u)    /* Value must be in range 0-59 */
#define START_HOUR               (12u)   /* Value must be in range 0-59 */
#define START_DAY                (30u)   /* Value must be in range 1-31 */
#define START_MONTH              (3u)    /* Value must be in range 1-12 */
#define START_YEAR               (17u)   /* Value must be in range 0-99 */

/* Defines for RTC Buffer Size */
#define RTC_BUFFER_SIZE          (20u)

/* Defines for Buffer Size */
#define PACKET_SIZE              (5u)
#define CON_BUFFER_SIZE          (PACKET_SIZE +  RTC_BUFFER_SIZE)
#define MAX_LOG                  (4u)    /* Number of memory buffers in the array */
```

Operation

1. Plug in the kit to your computer using the USB cable provided.
2. Build the project and program it into the PSoC 6 MCU with BLE connectivity device. For more information on device programming, see PSoC Creator Help. The flash memory of both cores is programmed in a single program operation.
3. Open a serial port terminal and connect to the PSoC 6 MCU with BLE connectivity port ('KitProg2 USB-UART') with a baud rate of 115200 bps, no parity, 1 stop bit, and 8 bits of data.
4. Press **SW1/RST** to reset the PSoC 6 MCU with BLE connectivity device. A message indicating the start of the code example with instructions should appear on the terminal. At reset, the RTC time is set to March 30th, 2017, at 12:00 pm.
5. Enter a packet of 5 characters, 4 times. With the first character of each package, the time is captured. After all packets (20 characters = 4 packets) are received, they are echoed to the serial port with the related timestamp.
6. Repeat the process. Sending four additional packets results in receiving four new strings with the new timestamped data.

Figure 3 shows the operation of the code example. Local echo is enabled in the terminal to visualize the input data. The timestamped data can be seen after inputting 4 packets.

Figure 3. Timestamped Data on the Terminal



```
*****************************************************************

This is CE219940_Multiple_DMA_Concatenation code example project

Transmit a 5 characters packet to add a RTC timestamp.

After 4 packets the data will be echoed back.

Enter the 1st four packets: T=05CT=07CT=15CT=23C

12:00:15 03/30/17 T=05C

12:00:21 03/30/17 T=07C

12:00:28 03/30/17 T=15C

12:00:33 03/30/17 T=23C

Enter the next four packets: T=27CT=32CT=29CT=26C

12:00:48 03/30/17 T=27C

12:00:52 03/30/17 T=32C

12:00:58 03/30/17 T=29C

12:01:04 03/30/17 T=26C

Enter the next four packets:
```

The sections that follow discuss the Components, parameter settings, and resources used to make the example.

Components

Table 2 lists the PSoC Creator Components used in this example, and the hardware resources used by each.

Table 2. List of PSoC Creator Components

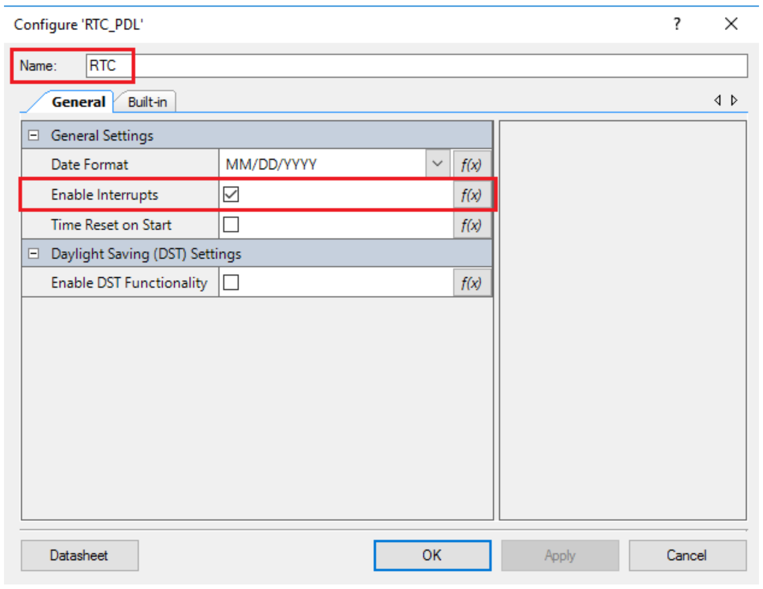
Component	Instance Name	Hardware Resources
RTC	RTC	1 Real Time Clock 1 Interrupt [CM4]
UART (SCB)	UART	1 SCB
DMA	RxDma	3 DMA Channels
	MemoryDma	
	TxDma	
SysInt	UART_INT	4 Interrupts [CM4]
	RX_DMA_INT	
	MEMORY_DMA_INT	
	TX_DMA_INT	

Parameter Settings

Figure 4 to Figure 12 highlight the non-default settings for each Component in this example. For more information on Component configuration, see the individual Component datasheets.

RTC Component Configuration

Figure 4. RTC Component Configuration



Configure 'RTC_PDL'

Name: RTC

General | Built-in

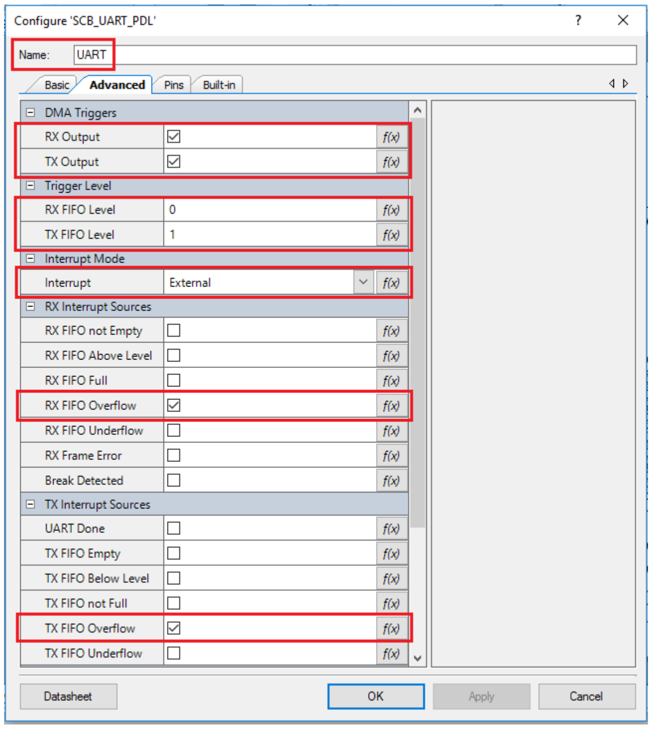
General Settings
- Date Format: MM/DD/YYYY
- Enable Interrupts: ☑
- Time Reset on Start: ☐

Daylight Saving (DST) Settings
- Enable DST Functionality: ☐

RTC interrupts are enabled to update the RTC Buffer every second. The configuration is done by the software using an RTC alarm for the interrupt trigger. The buffer update is done in the interrupt service routine Cy_RTC_Alarm1Interrupt.

UART Component Configuration

Figure 5. UART Component Configuration



Configure 'SCB_UART_PDL'

Name: UART

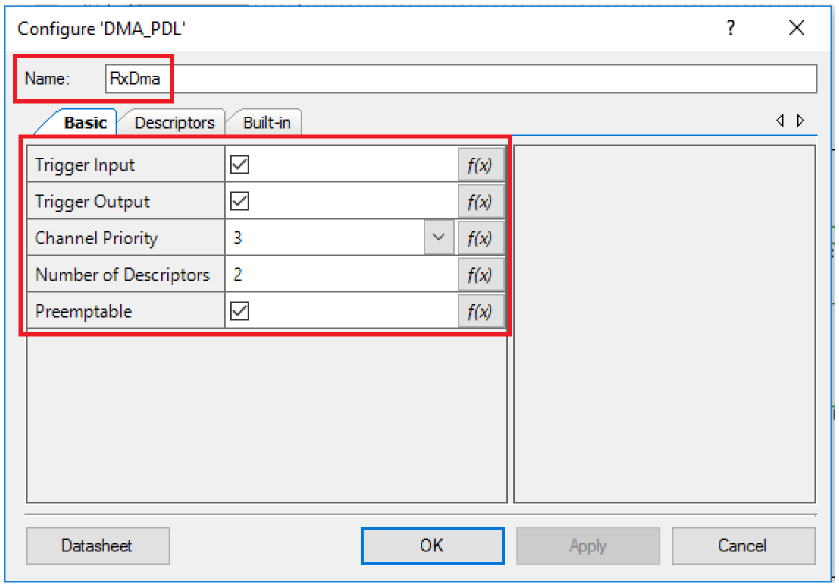
Basic | **Advanced** | Pins | Built-in

DMA Triggers	
RX Output	☑
TX Output	☑
Trigger Level	
RX FIFO Level	0
TX FIFO Level	1
Interrupt Mode	
Interrupt	External
RX Interrupt Sources	
RX FIFO not Empty	☐
RX FIFO Above Level	☐
RX FIFO Full	☐
RX FIFO Overflow	☑
RX FIFO Underflow	☐
RX Frame Error	☐
Break Detected	☐
TX Interrupt Sources	
UART Done	☐
TX FIFO Empty	☐
TX FIFO Below Level	☐
TX FIFO not Full	☐
TX FIFO Overflow	☑
TX FIFO Underflow	☐

The RX and TX DMA triggers are enabled. The TX FIFO level is set to 1 to act as a FIFO-is-empty signal for TxDma. The RX FIFO level is set to 0 to act as a FIFO-not-empty signal for RxDma. The overflows are selected as interrupt sources to indicate an error in the operation, going into an infinite loop within the interrupt service routine if they are triggered.

RxDma Component Configuration

Figure 6. RxDma Component Basic Configuration



Configure 'DMA_PDL'

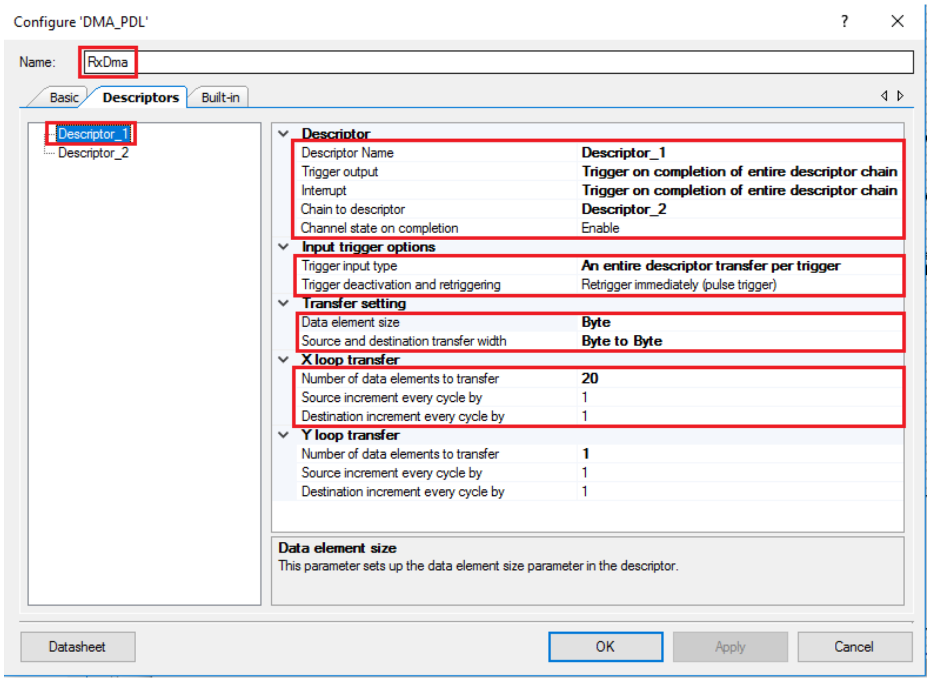
Name: RxDma

Basic | Descriptors | Built-in

Trigger Input	☑
Trigger Output	☑
Channel Priority	3
Number of Descriptors	2
Preemptable	☑

RxDma is triggered by UART RX DMA. After its descriptor chain is completed, it triggers MemoryDma. RxDma has two descriptors for moving data from the RTC and the UART to the concatenated data buffer. DMA can be preempted by other DMA channels. The configuration of the source/destination addresses of the descriptors and its other parameters are done in the `ConfigureRxDma` function of the software.

Figure 7. RxDma Component Descriptor 1 Configuration



Name: RxDma

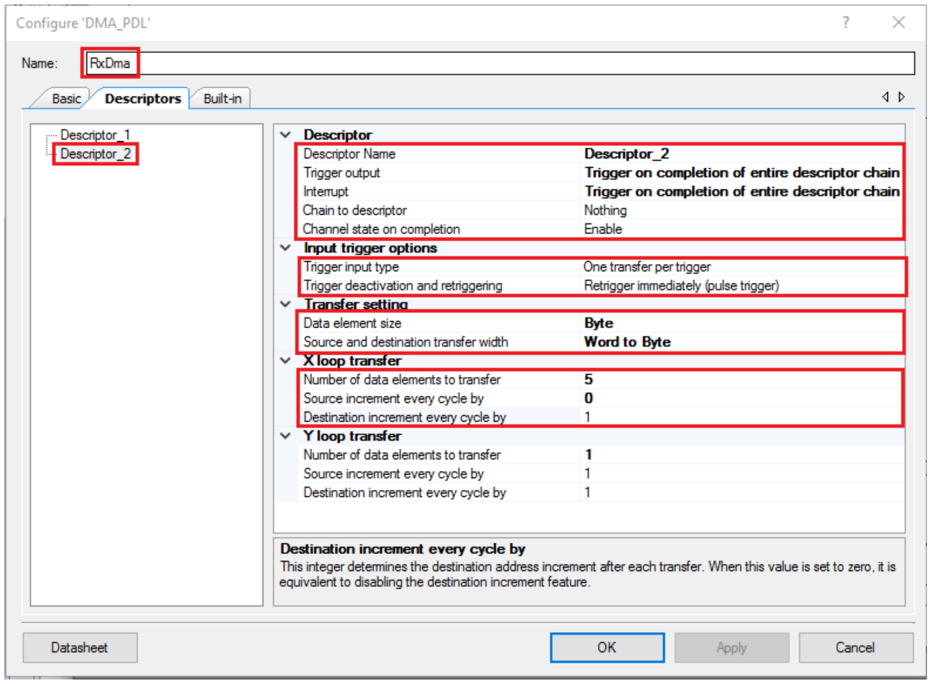
Descriptor_1
Descriptor_2

Descriptor	
Descriptor Name	Descriptor_1
Trigger output	Trigger on completion of entire descriptor chain
Interrupt	Trigger on completion of entire descriptor chain
Chain to descriptor	Descriptor_2
Channel state on completion	Enable
Input trigger options	
Trigger input type	An entire descriptor transfer per trigger
Trigger deactivation and retriggering	Retrigger immediately (pulse trigger)
Transfer setting	
Data element size	Byte
Source and destination transfer width	Byte to Byte
X loop transfer	
Number of data elements to transfer	20
Source increment every cycle by	1
Destination increment every cycle by	1
Y loop transfer	
Number of data elements to transfer	1
Source increment every cycle by	1
Destination increment every cycle by	1

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

The first descriptor moves the RTC buffer to the beginning of the concatenated data buffer. The RTC buffer has a non-null terminated 20-byte long string (“\r\nhh:mm:ss mm/dd/yy “). After the transfer is completed, descriptor 2 will be active.

Figure 8. RxDma Component Descriptor 2 Configuration



Name: RxDma

Descriptor_1
Descriptor_2

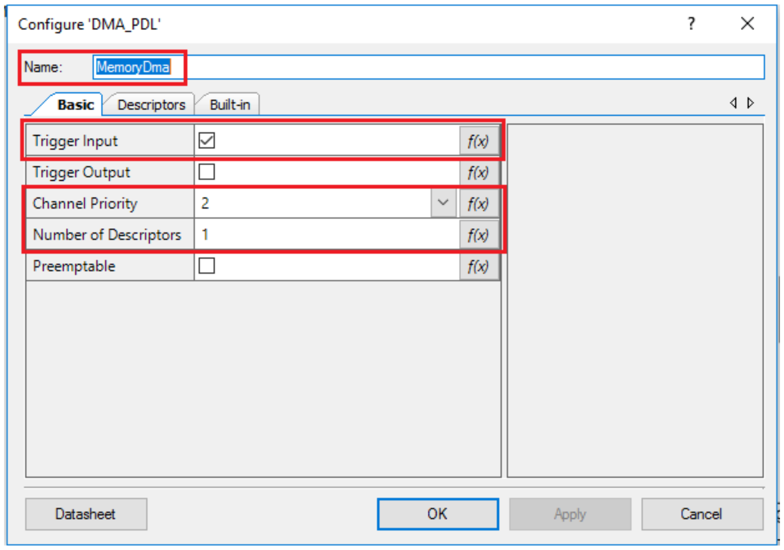
Descriptor	
Descriptor Name	Descriptor_2
Trigger output	Trigger on completion of entire descriptor chain
Interrupt	Trigger on completion of entire descriptor chain
Chain to descriptor	Nothing
Channel state on completion	Enable
Input trigger options	
Trigger input type	One transfer per trigger
Trigger 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	5
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	1
Destination increment every cycle by	1

Destination increment every cycle by
This integer determines the destination address increment after each transfer. When this value is set to zero, it is equivalent to disabling the destination increment feature.

The second descriptor moves the data from the UART to the concatenated data buffer. The source is not incremented as the FIFO address must not change. After the 5-byte transfer is complete, MemoryDma is triggered and an interrupt is fired. If there was an error during the DMA transfer, the program goes into an infinite loop. If not, the descriptor chain is reset. For the interrupt service routine, see the `RxDma_Complete` function.

MemoryDma Component Configuration

Figure 9. MemoryDma Component Basic Configuration

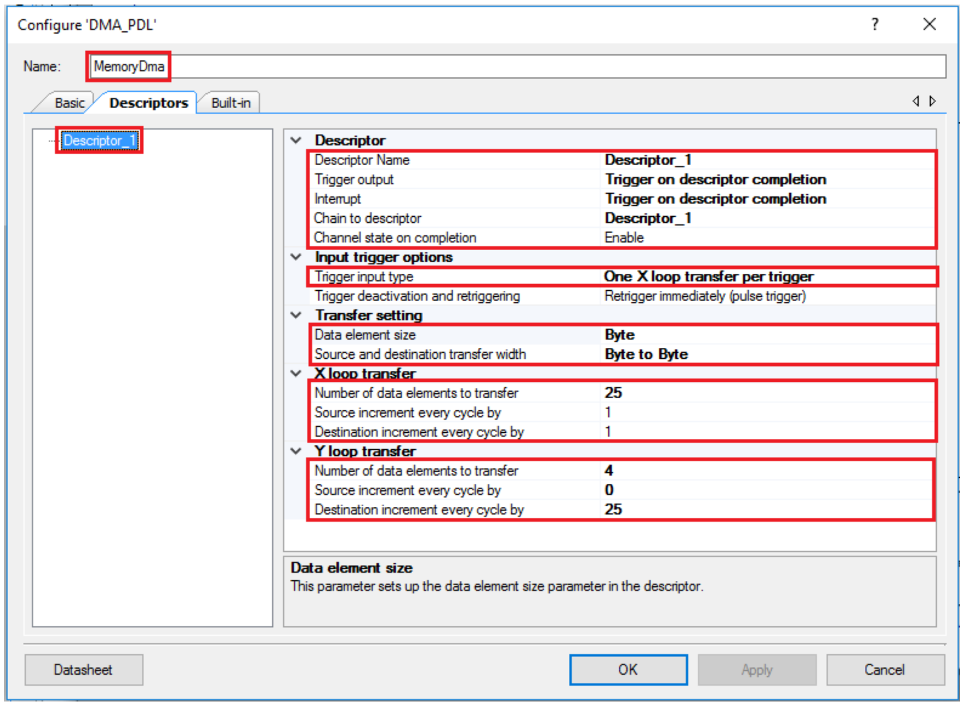


Name: MemoryDma

Trigger Input	☑	f(x)
Trigger Output	☐	f(x)
Channel Priority	2	f(x)
Number of Descriptors	1	f(x)
Preemptable	☐	f(x)

MemoryDma is triggered after RxDma finishes its descriptor chain, meaning that the concatenated data buffer is complete. It has a higher priority than the other DMA channels to ensure that the data in the concatenated buffer is not overwritten. The configuration of the source/destination addresses of the descriptor and its other parameters is done in the `ConfigureMemoryDma` function of the software.

Figure 10. MemoryDma Component Descriptor 1 Configuration



Name: MemoryDma

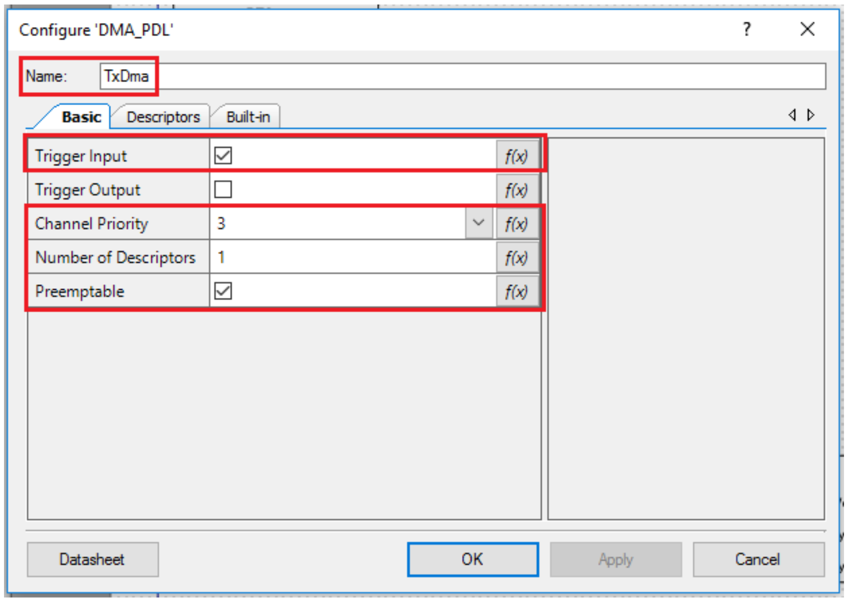
Descriptor	
Descriptor Name	Descriptor_1
Trigger output	Trigger on descriptor completion
Interrupt	Trigger on descriptor completion
Chain to descriptor	Descriptor_1
Channel state on completion	Enable
Input trigger options	
Trigger input type	One X loop transfer per trigger
Trigger deactivation and retriggering	Retrigger immediately (pulse trigger)
Transfer setting	
Data element size	Byte
Source and destination transfer width	Byte to Byte
X loop transfer	
Number of data elements to transfer	25
Source increment every cycle by	1
Destination increment every cycle by	1
Y loop transfer	
Number of data elements to transfer	4
Source increment every cycle by	0
Destination increment every cycle by	25

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

The descriptor implements a 2D transfer. The X loop transfers 25 bytes of data (RTC time and user input) to the memory array from the data buffer. Afterwards the Y loop increments the destination address by 25 bytes, allowing the next X loop to write the next data buffer to the correct destination in the memory array. After the Y loop is completed, the MemoryDma_Complete interrupt service routine fires, resetting the descriptor of the TxDma and enabling it. For the interrupt service routine, see the `MemoryDma_Complete` function.

TxDma Component Configuration

Figure 11. TxDma Component Basic Configuration

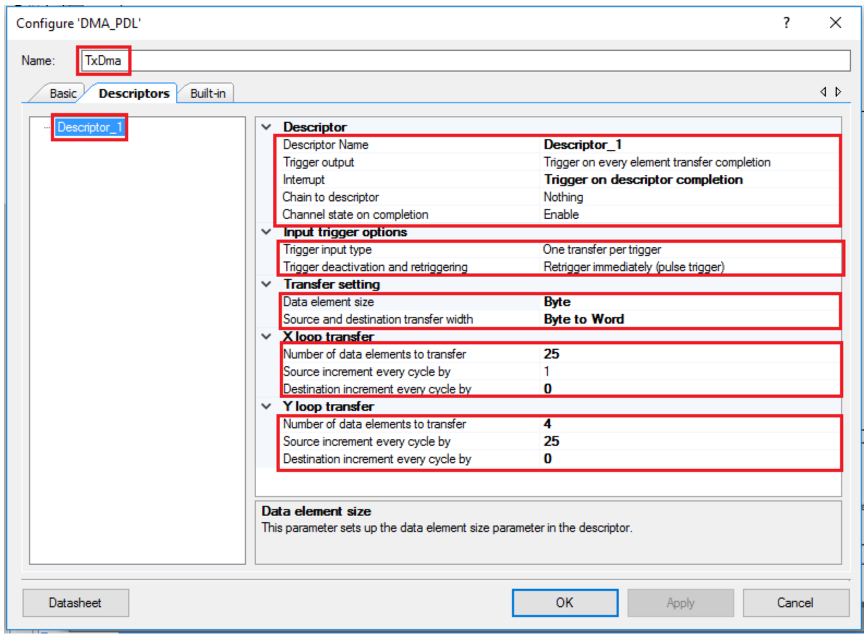


Name: TxDma

Trigger Input	☑
Trigger Output	☐
Channel Priority	3
Number of Descriptors	1
Preemptable	☑

TxDma is continuously triggered by the UART but remains disabled until activated by the MemoryDma interrupt. The configuration of the source/destination addresses of the descriptor and its other parameters is done in in the `ConfigureTxDma` function of the software.

Figure 12. TxDma Component Descriptor 1 Configuration



Name: TxDma

Descriptor	
Descriptor Name	**Descriptor_1**
Trigger output	Trigger on every element transfer completion
Interrupt	**Trigger on descriptor completion**
Chain to descriptor	Nothing
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	**Byte to Word**
X loop transfer	
Number of data elements to transfer	**25**
Source increment every cycle by	1
Destination increment every cycle by	**0**
Y loop transfer	
Number of data elements to transfer	**4**
Source increment every cycle by	**25**
Destination increment every cycle by	**0**

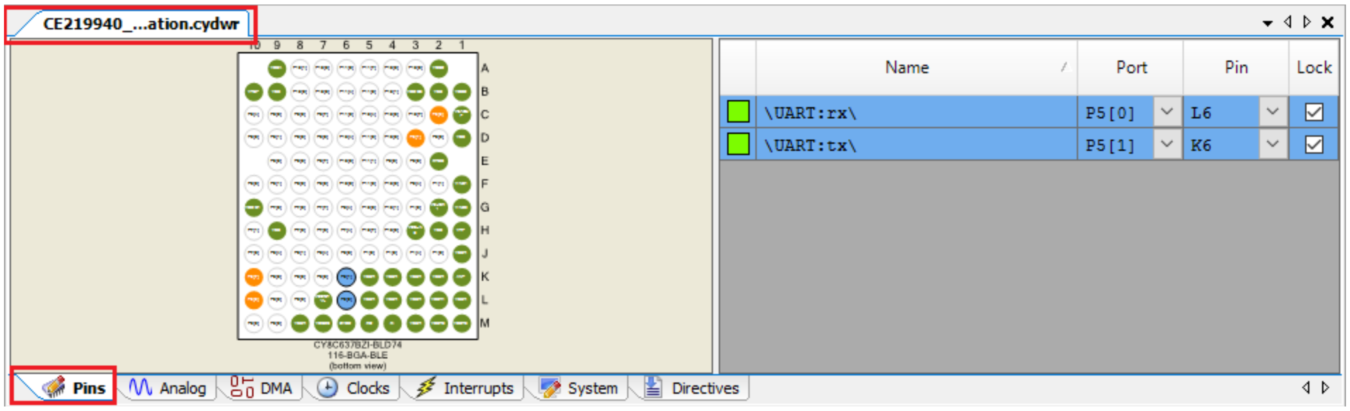
Data element size
This parameter sets up the data element size parameter in the descriptor.

The descriptor implements a 2D transfer with the X and Y loop. The X loop transfers 25 bytes of data to the UART (the time stamp plus the received UART data). Afterwards the Y loop increments the source address by 25 bytes, allowing the next X loop to read the next data buffer. The Y loop is configured for four X loops. The TxDma_Complete interrupt service routine prompts the user for more data.

Design-Wide / Global Resources

Figure 13 shows the pin assignments for the CY8CKIT-062-BLE.

Figure 13. Pin Assignments for CY8CKIT-062-BLE

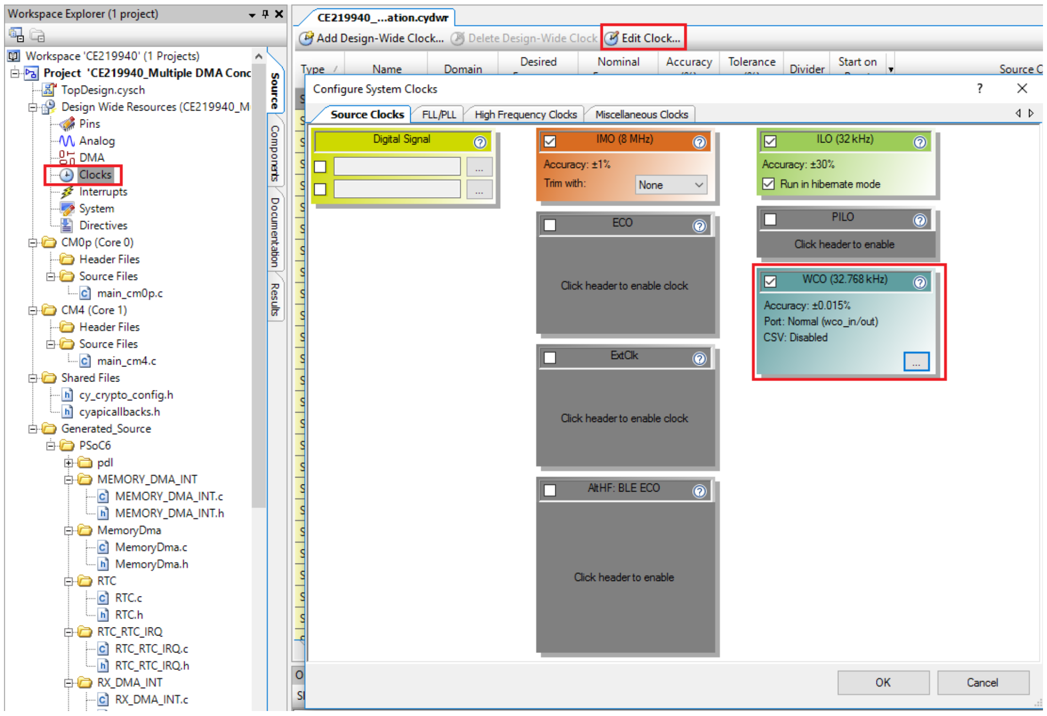


	Name	Port	Pin	Lock
🟩	\UART:rx\	P5[0]	L6	✓
🟩	\UART:tx\	P5[1]	K6	✓

P5[0] and **P5[1]** are the default UART pins used by the on-board KitProg2 USB-UART to communicate with the PC.

Figure 14 and Figure 15 show the system clocks configuration for using the RTC with the Watch-Crystal Oscillator.

Figure 14. Enabling of the Watch-Crystal Oscillator



IMO (8 MHz) ✓
Accuracy: ±1%
Trim with: None

ILO (32 kHz) ✓
Accuracy: ±30%
✓ Run in hibernate mode

ECO ☐
Click header to enable clock

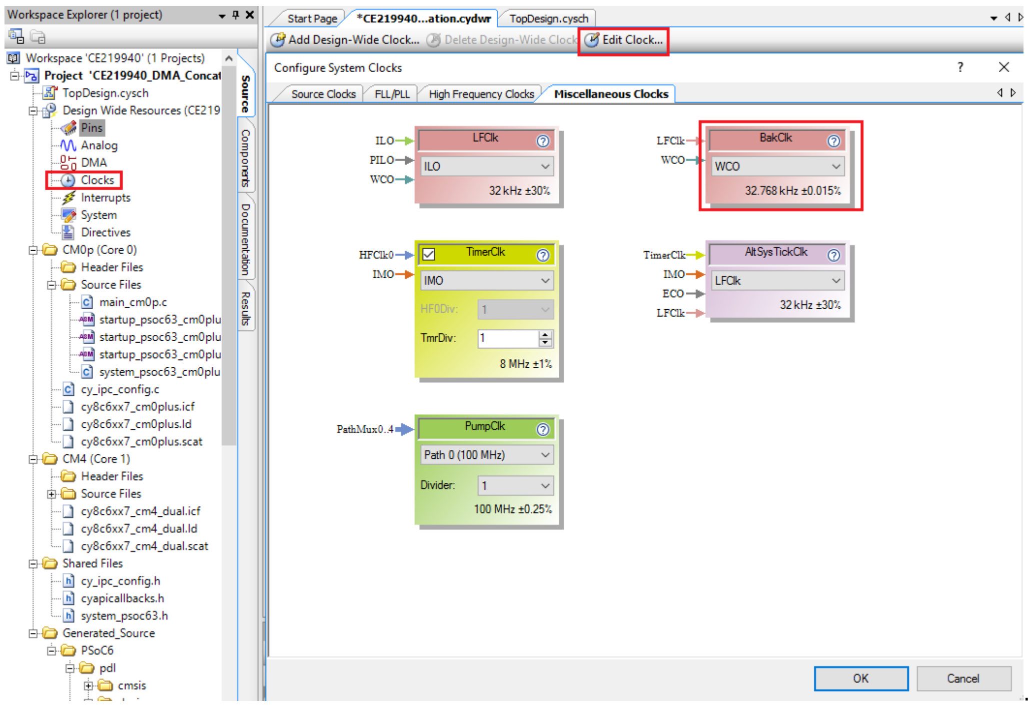
PILO ☐
Click header to enable

WCO (32.768 kHz) ✓
Accuracy: ±0.015%
Port: Normal (wco_in/out)
CSV: Disabled

ExtClk ☐
Click header to enable clock

AltHF: BLE ECO ☐
Click header to enable

Figure 15. Configuration of the RTC Clock



```
Workspace 'CE219940' (1 Projects)
  Project 'CE219940_DMA_Concat
    TopDesign.cysch
    Design Wide Resources (CE219
      Pins
      Analog
      DMA
      Clocks
      Interrupts
      System
      Directives
    CM0p (Core 0)
      Header Files
      Source Files
        main_cm0p.c
        startup_psoc63_cm0plu
        startup_psoc63_cm0plu
        startup_psoc63_cm0plu
        system_psoc63_cm0plu
      cy_ipc_config.c
      cy8c6xx7_cm0plus.icf
      cy8c6xx7_cm0plus.ld
      cy8c6xx7_cm0plus.scat
    CM4 (Core 1)
      Header Files
      Source Files
      cy8c6xx7_cm4_dual.icf
      cy8c6xx7_cm4_dual.ld
      cy8c6xx7_cm4_dual.scat
    Shared Files
      cy_ipc_config.h
      cyapicallbacks.h
      system_psoc63.h
    Generated_Source
      PSoC6
        pdl
          cmsis
```

LFClk: inputs ILO, PILO, WCO; source ILO; 32 kHz ±30%

BakClk: inputs LFClk, WCO; source WCO; 32.768 kHz ±0.015%

TimerClk: inputs HFClk0, IMO; source IMO; HFODiv: 1; TmrDiv: 1; 8 MHz ±1%

AltSysTickClk: inputs TimerClk, IMO, ECO, LFClk; source LFClk; 32 kHz ±30%

PumpClk: input PathMux0..4; Path 0 (100 MHz); Divider: 1; 100 MHz ±0.25%

Using the WCO for the RTC provides sufficient accuracy for this example.

Related Documents

Application Notes	
AN210781 Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes the PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity and how to build this code example.
PSoC Creator Component Datasheets	
DMA	Provides information on DMA settings and API.
RTC	Provides information on Real Time Clock settings and API.
UART	Provides information on UART settings and API.
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheets	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kit (DVK) Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	

Document History

Document Title: CE219940 - PSoC 6 MCU Multiple DMA Concatenation

Document Number: 002-19940

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5843249	CFMM	08/14/2017	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

Cypress Developer Community

Forums | WICED IOT Forums | 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